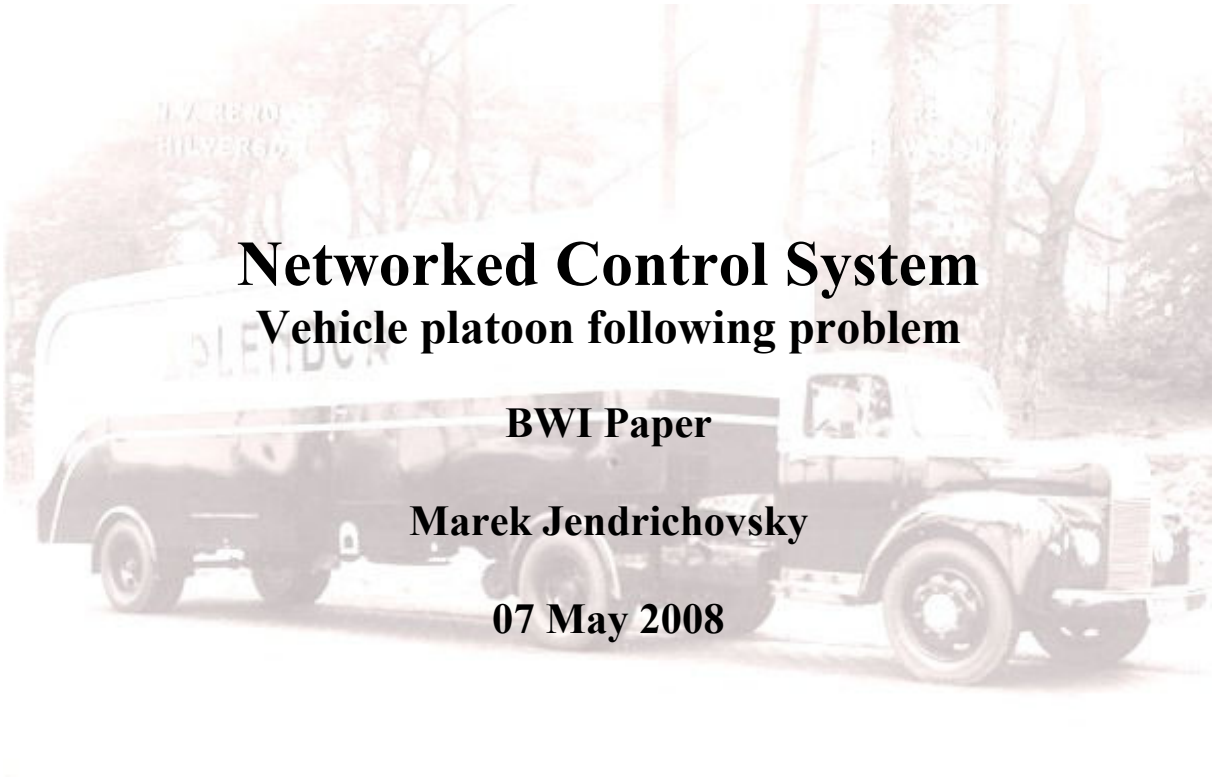


Networked Control System

Vehicle platoon following problem

BWI Paper

Marek Jendrichovsky

07 May 2008

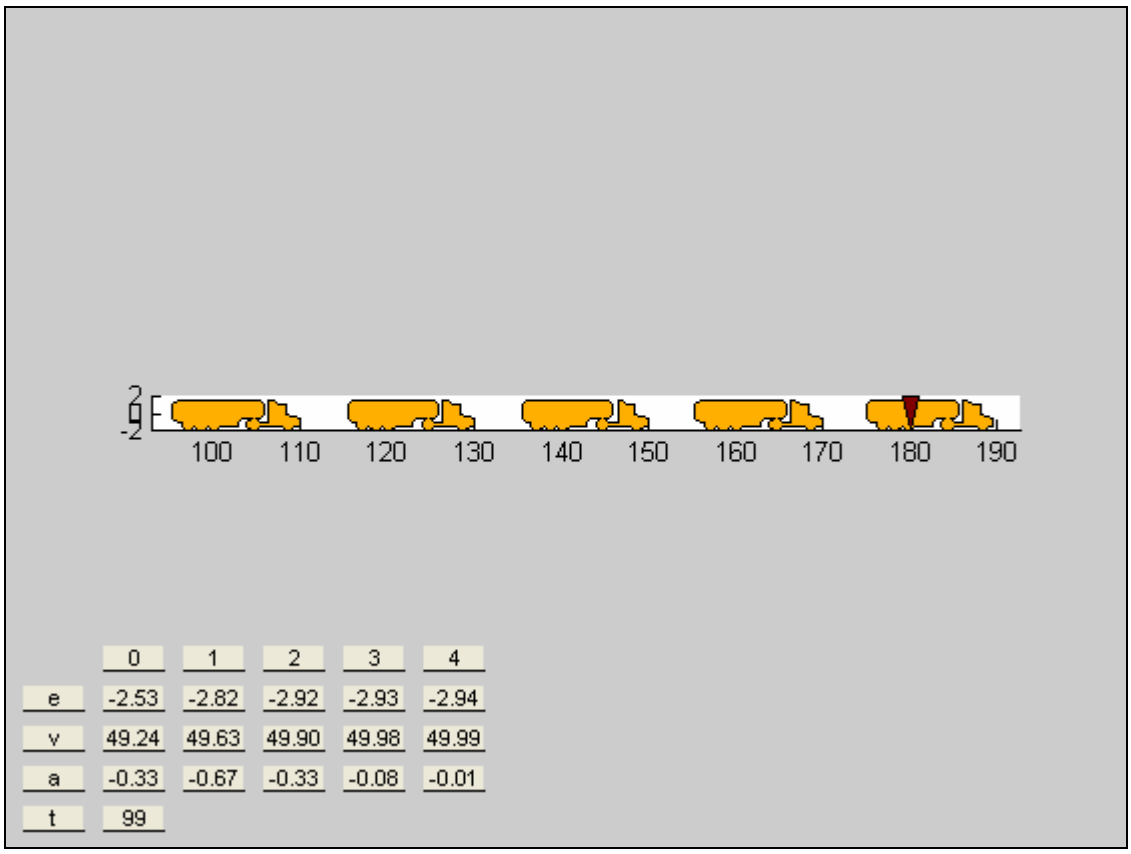
Table of contents:

1 Introduction

This paper presents analysis and simulation of the vehicle platoon following problem described in [1]. The problem the authors try to tackle there is to model the vehicle-platoon following problem via *Networked Control System* (NCS), concretely *Markovian Jump Linear System* (MJLS). The vehicle platoon consists of five trucks following the leader. The four followers are supposed to be fully "auto-piloted" with the help of:

- on-board radar to give the information about the predecessor vehicle,
- wireless network to broadcast the feedback information about the leading and other vehicles.

The leading vehicle is supposed to follow a so-called "reference-trajectory", actually a constant-speed virtual "position-mark" moving along the track, where the aim of the feedback control law is to minimize the tracking and spacing errors of all the vehicles.



This is a screenshot from the simulation of the vehicle platoon in MATLAB, we see (in the units based on seconds and meters):

- for each vehicle (0=leader down to 4):
  - tracking error **e**,
  - velocity **v,**
  - acceleration **a,**
- current time **t,**
- track position (leading vehicle is now at ~182),
- red marker for a reference track.

1.1 Sources/References

Several other sources have been used in the course of the study that led to this thesis. Background information concerning general systems theory was taken from [7] and [8], information on H_∞-control theory was taken from [2].
Information concerning linear matrix inequalities, which play an important role in solving H_∞-control problems was taken from [3]. The source for information on probability theory was [10]. The conversion of continuous time systems to discrete time systems using sampling, which we use later on in this thesis, can be found in many sources; we follow [9] here.

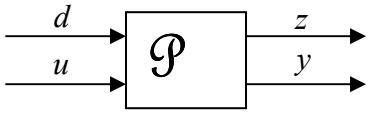
The area of Networked Control Systems is an area of much recent research in systems and control theory. Several issues concerning the use of a network to exchange information between the plant and controller have been studied in many papers, and are still being studied. See [4] for an overview of much of the literature up to 2004. We will focus on the facts of information-package loss, modeled by a Markovian jump system, and in doing so take the example from [1] as our lead. See also [6]. Other papers in this area focus on the effects of package delay [5]. As always in systems theory, stability analysis of the class of control systems under consideration plays an important role in the subject. See [9] for a good introduction to that particular problem.

1.2 Acknowledgements

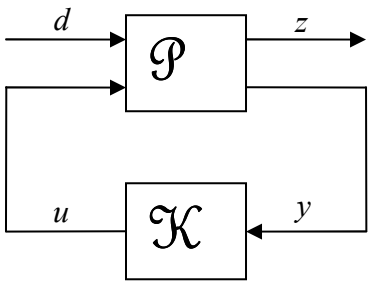
I would like to thank my advisor professor Andre Ran to help me “having fun all the way”.

2 Mathematical background

The main subject of study for the *Mathematical Control and System Theory* branch of mathematics is the dynamical system ($\mathcal{P}$), a "transformation" of input/disturbances (u / d) into output/error measurements (y / z):



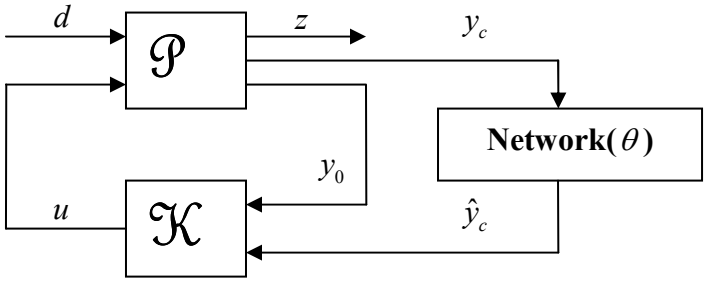
The goal of this analysis is often a synthesis of a feedback control law ($\mathcal{K}$), which is itself also a dynamical system, in order to stabilize the (unstable/marginally stable) dynamical system:



Analysis of dynamical systems and the synthesis of the feedback laws was long time dominated by the *Kalman filtering* method, also called H_2-control, originally developed in the 1960s for aerospace applications. The main objection to this approach was that it makes quite severe assumptions about the disturbance process statistics (zero-mean Gaussian). H_∞ analysis/synthesis is the answer to this problem as it tries to bring some robustness in the feedback, in order to make the closed loop system behave acceptably even in the face of unpredicted/misestimated noise. More information on this subject can be found in the reference [2].

2.1 H_∞ analysis/synthesis for Networked Control Systems

In the paper [1] the H_∞ analysis/synthesis is used in order to study the problem of the *Networked Control System*, a system where the feedback has to be communicated via the network, subjected to errors, delays or complete packet-losses:



The approach of the authors of the paper was to model such a system as a *Markovian Jump Linear System* (MJLS), meaning:

- The network stochastic process θ is a *Markov process*: the state of the network at the time t is independent of the history up to this time, it depends only on the current state of the network (here stated for a discrete case):
  $\mathbb{E}[\theta(k) \mid \theta(k-1), \theta(k-2), \theta(k-3), ..., \theta(0)] = \mathbb{E}[\theta(k) \mid \theta(k-1)]$,
- The physical continuous dynamical system is "*linearized*" into the discrete settings:

$$\begin{bmatrix} x(k+1) \\ z(k) \end{bmatrix} = \begin{bmatrix} A_{\theta(k)} & B_{\theta(k)} \\ C_{\theta(k)} & D_{\theta(k)} \end{bmatrix} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix}. \quad (1)$$

2.1.1 H_∞ analysis/synthesis "toolbox"

The presence of the network brings "stochasticity" into the system, therefore an additional (to the deterministic case) mathematical "toolbox" is used to analyze such systems.

We have several concepts of the (stochastic) stability:

- *Second-moment stability*: the MJLS is said to be second-moment stable (for initial state/network state x_0, θ_0) if it is (equivalently):
  - Mean-square stable: $\forall (x_0, \theta_0) : \lim_{k\to\infty} \mathbb{E}[\|x(k)\|^2 \mid x_0, \theta_0] = 0$,
  - Stochastically stable: $\forall (x_0, \theta_0) : \sum_{k=0}^{\infty} \lim_{k\to\infty} \mathbb{E}[\|x(k)\|^2 \mid x_0, \theta_0] < \infty$,
  - Exponentially mean-square stable:
    $\forall (x_0, \theta_0) \exists 0 < \alpha < 1, 0 < \beta : \forall k, \mathbb{E}[\|x(k)\|^2 \mid x_0, \theta_0] < \beta \alpha^k \|x_0\|^2$,
- *Almost-sure stability*: $\forall (x_0, \theta_0) : \mathbb{P}(\lim_{k\to\infty} \|x(k)\| = 0] = 1$.

To measure the sensitivity of the system to the input/disturbance, in other words: to measure system gain from input/disturbance to output/error-measurement, we use the following norm:

- H_∞ *norm* of an MJLS for a network with two states $\{0,1\}$ = network down/up:
  - Assume that system $\mathcal{P}$ is second-moment stable (SMS), let $x_0 = 0$,
  - Then the H_∞ norm is: $\|\mathcal{P}\|_\infty := \sup_{\theta(0)\in\{0,1\}} \sup_{0\neq d\in l_2^n} \frac{\|z\|_2}{\|d\|_2}$,
  - Where l_2^n is defined as:

$$l_2^n \equiv \left\{ \{x(k)\}_{k=0}^{\infty} : \forall k, x(k) \in \mathbb{R}^n \text{ is a random variable dependent on } \{\theta(k)\}_{k=0}^{\infty} \text{ and } \|x\|_2 < \infty \right\},$$

  - And the $\|\cdot\|_2$ norm there is given by: $\|x\|_2^2 \equiv \sum_{k=0}^{\infty} \mathbb{E}[x(k)^T x(k)]$.

The actual synthesis of the feedback law with gain less then one is based on:

- *Bounded real lemma*:
  - Assume system $\mathcal{P}$ given by (1) is weakly-controllable,
  - Then: $\mathcal{P}$ is SMS with $\|\mathcal{P}\|_\infty < 1$ if and only if

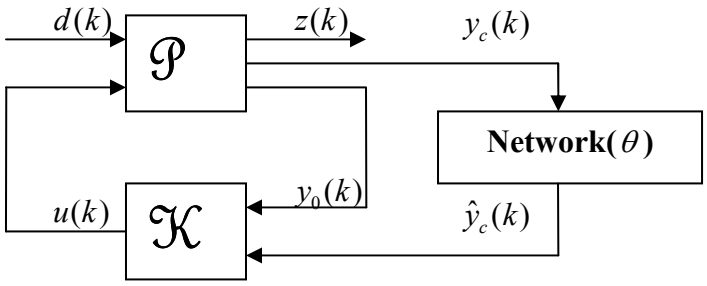
$$\exists \{G_i\} > 0 : \begin{bmatrix} A_i & B_i \\ C_i & D_i \end{bmatrix}^T \begin{bmatrix} \sum_{j=1}^{2} p_{ij} G_j & 0 \\ 0 & I \end{bmatrix} - \begin{bmatrix} G_i & 0 \\ 0 & I \end{bmatrix} < 0, \quad i \in \{0,1\}.$$

For practical use this lemma is cast into a more familiar/standard settings of:

- *Linear matrix inequalities* (LMI):
  - The synthesis of the feedback law can be reduced to a more standard *semi-definite programming problem*:
    - minimize γ and find $\Sigma = \{A_{cl}, B_{cl}, C_{cl}, D_{cl}\}$,
    - Subject to $\begin{bmatrix} \begin{bmatrix} Z & 0 \\ 0 & \gamma^2 I \end{bmatrix} & [*]^T & [*]^T \\ \sqrt{p_1}\begin{bmatrix} A_{cl,1}Z & B_{cl,1} \\ C_{cl,1}Z & D_{cl,1} \end{bmatrix} & \begin{bmatrix} Z & 0 \\ 0 & I \end{bmatrix} & [*]^T \\ \sqrt{p_2}\begin{bmatrix} A_{cl,2}Z & B_{cl,2} \\ C_{cl,2}Z & D_{cl,2} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} Z & 0 \\ 0 & I \end{bmatrix} \end{bmatrix} > 0, Z > 0$,
    - Where "cl" stands for closed-loop, "[*]" stands for symmetric terms,
  - LMIs are mostly solved via the *interior-point methods* using freely-available software.

3 The model

The model, discrete-time linear system with Markovian jumping parameters, stabilized via (network transmission dependent) output feedback is shown here:



Where:

- $\mathcal{P}$ is the (open-loop) plant, actually the dynamics of the 5-vehicles-platoon,
- $\mathcal{K}$ is the feedback controller to stabilize the platoon via internal $y_0(k)$ and external (network-communicated) $\hat{y}_c(k)$ plant output,
- **Network(θ):** $\theta(k)$ is the Markov (Bernoulli) process simulating the network packet loss, it has two states: packet lost/received: $\forall k : \theta(k) \in \{0,1\}$,
- $d(k)$ is the general system-disturbance, further on defined as $d(k) \equiv r_0(k)$, the reference path of the leading vehicle,
- $z(k)$ is the error vector: $z(k) = \begin{bmatrix} e_0(k) & t_1(k) & \cdots & t_4(k) \end{bmatrix}^T$, where $e_0 = r_0 - x_0$ is the tracking error of the first vehicle, $t_i = r_0 - x_i - i.\delta_i, i \in \{1,2,3,4\}$ are the tracking errors of the rest of the platoon vehicles, where δ_i are the desired spacing between the vehicles, x_i is the horizontal position of i-th vehicle, we also assume $\delta_i \equiv \delta, \forall i$,
- $u(k)$ is the system input: applied throttle/brake,
- $y_c(k)$ is the (network-) communicated vehicle-platoon status, $y_c(k) = \begin{bmatrix} v_0 & a_0 & a_1 & a_2 & a_3 \end{bmatrix}^T$, where v_i, a_i stand for i-th vehicle speed resp. acceleration,
- $\hat{y}_c(k)$ is the platoon status as received by the individual vehicles, thus this is actually $y_c(k)$ "disturbed" by the network: $\hat{y}_c(k) = \begin{cases} y_c(k), & \theta(k) = 1 \\ \varnothing, & \theta(k) = 0 \end{cases}$,
- $y_0(k)$ is the internal vehicle status as read form the on-board sensors, thus this is kind of a fall-back-output-state when the network packet gets lost, $y_0(k) = \begin{bmatrix} e_0 \cdots & e_4 & \dot{e}_0 & \cdots & \dot{e}_4 & v_1 & \cdots & v_4 \end{bmatrix}^T$, where $e_i = x_{i-1} - x_i - \delta_i, i \in \{1,2,3,4\}$ are spacing errors of the other vehicles.

3.1 Model derivation

3.1.1 Packet loss simulation: Markov (Bernoulli) process

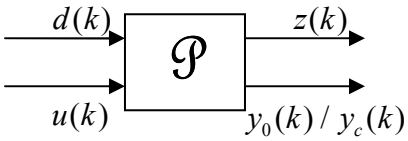
The feedback information (network packet) loss is modelled via a *Markov process*, actually a *Bernoulli process*, so we have:

- jumping probabilities: $p_{ij} = p_j, \forall i, j \in \{0,1\}$,
- Markov process transition matrix: $A = \begin{pmatrix} p & 1-p \\ p & 1-p \end{pmatrix}$.

where the process has two states:

- 0 = packet lost, with probability p,
- 1 = packet received, probability $1-p$.

3.1.2 Derivation of the open-loop plant



Individual vehicle dynamics model (derived from the "first principles") are:

$$\begin{pmatrix} \dot{x}(t) \\ \dot{v}(t) \\ \dot{a}(t) \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\frac{1}{\tau} \end{pmatrix}}_{A_v} \begin{pmatrix} x(t) \\ v(t) \\ a(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{\tau} \end{pmatrix} u(t)$$

Where:

- x, v, a stand for horizontal position, speed and acceleration,
- u is the input (throttle/brake).

The throttle/brake dynamics are:
$\tau \dot{a}(t) + a(t) = u(t)$,
so we see that the applied throttle/brake is actually a first order approximation of the actual acceleration with the actuator delay τ, and also that this brings the system to the stable condition if u matches a:
$\tau \dot{a}(t) = u(t) - a(t) = 0$

3.1.3 Discretization

Now we are going to *discretize* the (continuous) vehicle platoon dynamics system:
$\dot{x}(t) = Ax(t) + B_1 d(t) + B_2 u(t)$.

Where x now stands for the whole 5-vehiscles open-loop system state:
$x = \begin{pmatrix} x_0 & v_0 & a_0 & x_1 & v_1 & a_1 & \cdots & x_4 & v_4 & a_4 \end{pmatrix}^T$.

And the whole 5-vehicles platoon (continuous) dynamics are captured by:

$$\Sigma_C \begin{cases} A = \begin{pmatrix} A_v & 0 & \cdots & 0 \\ 0 & A_v & & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & A_v \end{pmatrix} \\ B_1 = \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix} \\ B_2 = \begin{pmatrix} b & 0 & \cdots & 0 \\ 0 & b & & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & b \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{\tau} \end{pmatrix} \end{cases}.$$

Discretization uses two facts (and the reference [9]):

- For the input u a zero-order-hold is used: $u(t) = u_{ZOH}(t_k), t \in [t_k, t_{k+1}], t_{k+1} - t_k = h, \forall k$,
- And the reference trajectory $r_0(t)$ is sampled each ZOH period h too:
  $r_0(t) = r_{sampled}(t_k) \quad , t \in [t_k, t_{k+1}]$.

Therefore the system could be rewritten as:
$\dot{x}(t) = Ax(t) + B_1 d_k + B_2 u_k, t \in [t_k, t_{k+1}], t_{k+1} - t_k = h, \forall k$.

By integration we arrive to:

$$x(k+1) = \begin{bmatrix} e^{Ah} & -\int_0^h e^{A(h-x)} dx B_1 & -\int_0^h e^{A(h-x)} dx B_2 \end{bmatrix} \begin{bmatrix} x(k) \\ d(k) \\ u(k) \end{bmatrix}.$$

And using:
$A_{OL} = e^{Ah}, \quad B_{OL,1} = -\int_0^h e^{A(h-x)} dx B_1, \quad B_{OL,2} = -\int_0^h e^{A(h-x)} dx B_2$.

Where OL stands for open-loop plant, we can rewrite the state-update part of the discretized system $\mathcal{P}$ as:

$x_{OL}(k+1) = A_{OL} x_{OL}(k) + B_{OL,1} d(k) + B_{OL,2} u(k)$.

We have discretized the continuous system Σ_C using the sampling frequency $h = 20ms$ and actuator delay $\tau = 100ms$. For discretisation we have used MATLAB which resulted in the following discrete system Σ_D for the platoon dynamics (open-loop-plant):

$$\Sigma_D \begin{cases} A_{OL} = \begin{pmatrix} 1 & 0.02 & 0.0002 & \cdots & 0 \\ 0 & 1 & 0.0181 & \cdots & 0 \\ 0 & 0 & 0.8187 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0.8187 \end{pmatrix} \\ B_{OL,1} = \begin{pmatrix} 0.02 \\ 0 \\ \vdots \end{pmatrix} \\ B_{OL,2} = \begin{pmatrix} 0 & \cdots & 0 \\ 0.0019 & & 0 \\ 0.1813 & & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0.1813 \end{pmatrix} \end{cases}.$$

With the corresponding state vector:
$x = \begin{pmatrix} x_0 & v_0 & a_0 & x_1 & v_1 & a_1 & \cdots & x_4 & v_4 & a_4 \end{pmatrix}^T$.

Notice that there is no equilibrium point at 0 for the state vector, and also that the disturbance r_0 has no zero mean but a “drift”.

3.1.4 The equivalent (zero-state-equilibrium, zero-mean disturbance) system derivation

Now we derive the equivalent open-loop system, but this time with the different state vector:

$x_{OL} = \begin{pmatrix} e_0 & \dot{e}_0 & \ddot{e}_0 & \cdots & e_4 & \dot{e}_4 & \ddot{e}_4 \end{pmatrix}^T$,

and different disturbance r_0, which is now a deviation of the leading vehicle from the reference (desired) trajectory, zero mean indeed, and also notice that our new state vector is zero in the equilibrium/steady state.

We would actually like to look at the vehicle platoon as it is already moving with the constant speed and correct spacing (tracking/spacing errors are zero), being disturbed by the deviations of the leading vehicle from its reference trajectory, so we will disturb the position of the leading vehicle with r_0.

This all is actually needed in order to be able to perform the correct H_∞ analysis.

We will define new open-loop system matrices for this "zero-equilibrium" model from:

- $x_i(k+1) = x_i(k) + 0,2.v_i(k) + 0,0002.a_i(k)$,
- $v_i(k+1) = v_i(k) + 0,0181.a_i(k) + 0,0019.u_i(k)$,
- $a_i(k+1) = 0,8187.a_i(k) + 0,1813.u_i(k)$.

So let's see about the evolution of $e_0 = r_0 - x_0$ and its derivatives:

$e_0(k+1) = r_0(k+1) - x_0(k+1) =$

$= r_0(k) + 0,2.\dot{r}_0(k) + 0,0002.\ddot{r}_0(k) - x_0(k) - 0,2.v_0(k) - 0,0002.a_0(k) =$

$= [r_0(k) - x_0(k)] + 0,2.[\dot{r}_0(k) - v_0(k)] + 0,0002.[\ddot{r}_0(k) - a_0(k)] =$

$= e_0(k) + 0,2.\dot{e}_0(k) + 0,0002.\ddot{e}_0(k)$

$\dot{e}_0(k+1) = \dot{r}_0(k+1) - \dot{x}_0(k+1) = \dot{r}_0(k+1) - v_0(k+1) =$

$= \dot{r}_0(k) + 0,0181.\ddot{r}_0(k) - v_0(k) - 0,0181.a_0(k) - 0,0019.u_0(k) =$

$= \dot{e}_i(k) + 0,0181.\ddot{e}_i(k) - 0,0019.u_0(k)$

$\ddot{e}_0(k+1) = 0,8187.\ddot{e}_0(k) - 0,1813.u_0(k)$

For $e_i = x_{i-1} - x_i - \delta$:

$e_i(k+1) = x_{i-1}(k+1) - x_i(k+1) - \delta =$

$= x_{i-1}(k) + 0,2.v_{i-1}(k) + 0,0002.a_{i-1}(k) - x_i(k) - 0,2.v_i(k) - 0,0002.a_i(k) - \delta =$

$= x_{i-1}(k) - x_i(k) - \delta + 0,2.[v_{i-1}(k) - v_i(k)] + 0,0002.[a_{i-1}(k) - a_i(k)] =$

$= e_i(k) + 0,2.\dot{e}_i(k) + 0,0002.\ddot{e}_i(k)$

$$\dot{e}_i(k+1) = \dot{x}_{i-1}(k+1) - \dot{x}_i(k+1) = v_{i-1}(k+1) - v_i(k+1)$$
$$= v_{i-1}(k) + 0,0181.a_{i-1}(k) + 0,0019.u_{i-1}(k) - v_i(k) + 0,0181.a_i(k) - 0,0019.u_i(k) =$$
$$= \dot{e}_i(k) + 0,0181.\ddot{e}_i(k) + 0,0019.[u_{i-1}(k) - u_i(k)]$$

$$\ddot{e}_i(k+1) = 0,8187.\ddot{e}_i(k) + 0,1813.[u_{i-1}(k) - u_i(k)]$$

Getting the system matrices A_{OL}, $B_{OL,1}$ and $B_{OL,2}$:

$$\begin{pmatrix} e_0 \\ \dot{e}_0 \\ \ddot{e}_0 \\ \vdots \\ e_4 \\ \dot{e}_4 \\ \ddot{e}_4 \end{pmatrix} \leftarrow \underbrace{\begin{pmatrix} 1 & 0,02 & 0,0002 & \cdots & 0 \\ 0 & 1 & 0,0181 & & 0 \\ 0 & 0 & 0,8187 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0,8187 \end{pmatrix}}_{A_{OL}} \begin{pmatrix} e_0 \\ \dot{e}_0 \\ \ddot{e}_0 \\ \vdots \\ e_4 \\ \dot{e}_4 \\ \ddot{e}_4 \end{pmatrix} + \underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}}_{B_{OL,1}} r_0 + \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -0,0019 & 0 & 0 & 0 & 0 \\ -0,1813 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0,0019 & -0,0019 & 0 & 0 & 0 \\ 0,1813 & -0,1813 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0,0019 & -0,0019 & 0 & 0 \\ 0 & 0,1813 & -0,1813 & 0 & 0 \\ \vdots & & & & \vdots \end{pmatrix}}_{B_{OL,2}} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_4 \end{pmatrix}$$

Now for the error vector z:
We have $t_i = r_0 - x_i - i.\delta, i \in \{1,2,3,4\}$, this we can further simplify to:

- for $i = 1 : t_1 = r_0 - x_1 - \delta = r_0 - x_0 + x_0 - x_1 - \delta = r_0 - x_0 + x_0 - x_1 - \delta = e_0 + e_1$,
- for $i = 2$:
  $t_2 = r_0 - x_2 - 2\delta = r_0 - x_0 + x_0 - x_2 - 2\delta = e_0 + x_0 - x_1 - \delta + x_1 - x_2 - \delta = e_0 + e_1 + e_2$,
- therefore: $t_i = \sum_{j=0}^{i} e_j$.

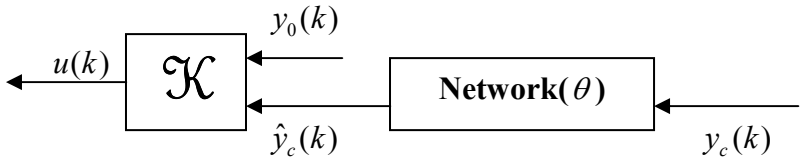
$$\begin{pmatrix} e_0 \\ t_1 \\ t_2 \\ \vdots \\ t_4 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & & \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \ddots & \vdots \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}}_{C_{OL}} \begin{pmatrix} e_0 \\ \dot{e}_0 \\ \ddot{e}_0 \\ \vdots \\ e_4 \\ \dot{e}_4 \\ \ddot{e}_4 \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}}_{D_{OL,1}} r_0 + \underbrace{\begin{pmatrix} 0 & & 0 \\ & \ddots & \\ 0 & & 0 \end{pmatrix}}_{D_{OL,2}} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_4 \end{pmatrix}$$

Thus the complete open-loop plant $\mathcal{P}$ becomes:

$$\begin{pmatrix} x_{OL}(k+1) \\ z(k) \end{pmatrix} = \begin{pmatrix} A_{OL} & B_{OL,1} & B_{OL,2} \\ C_{OL} & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{OL}(k) \\ d(k) \\ u(k) \end{pmatrix}.$$

3.1.5 "Closing the loop"

We will stabilize the system with the output-feedback controller, as we see the network is drawn as a part of the feedback controller, because the controller must have knowledge of the success/failure of the feedback-information-network-packet delivery.



Define the output network-state dependent feedback as:
$u(k) = K_i x_{OL}(k), i \in \{0,1\}$

Where

$$u_i(k) = \begin{cases} \lambda a_0(k) + (1-\lambda)a_{i-1}(k) + k_0(v_0(k) - v_i(k)) + k_d \dot{e}_i(k) + k_p e_i(k), & \theta(k) = 1, i \neq 0 \\ k_d \dot{e}_i(k) + k_p e_i(k), & \theta(k) = 0, i = 1 \text{ or } i = 0 \end{cases}$$

Feedback analysis (see [8] for more detail):

- in the case when the network packet is lost (or feedback for the platoon leader), we see a "classical" proportional $k_p e_i(k)$ and derivative $k_d \dot{e}_i(k)$ feedback,
- when a vehicle receives the network information, this feedback is augmented with the part proportional to the difference of the speeds between this vehicle and the platoon leader $k_0(v_0(k) - v_i(k))$, and also with kind of the "interpolation" between the acceleration of the vehicle ahead and the leader $\lambda a_0(k) + (1-\lambda)a_{i-1}(k)$.

Using the facts that:

- $v_0 - v_1 = \dot{e}_1, v_0 - v_2 = v_0 - v_1 + v_1 - v_2 = \dot{e}_1 + \dot{e}_2$, therefore: $v_0 - v_i = \sum_{j=1}^{i} \dot{e}_j$,
- we are studying the system in equilibrium, which is only being disturbed by the external factors, thus no acceleration/braking is applied to the reference trajectory, therefore we assume $\ddot{r}_0 \equiv 0$, and then we see $a_0 = \ddot{r}_0 - \ddot{e}_0 = -\ddot{e}_0$,
  $\ddot{e}_1 = a_0 - a_1 \Rightarrow a_1 = a_0 - \ddot{e}_1 = -\ddot{e}_0 - \ddot{e}_1$, $\ddot{e}_2 = a_1 - a_2 \Rightarrow a_2 = a_1 - \ddot{e}_2 = -\ddot{e}_0 - \ddot{e}_1 - \ddot{e}_2$,
  therefore $a_i = -\sum_{j=0}^{i} \ddot{e}_j$.

We rewrite the feedback laws:

- for $\theta(k) = 0, i \in \{0,1\}$ nothing changes,
- for $\theta(k) = 1, i = 1$:
  $u_1 = \lambda a_0 + (1-\lambda)a_0 + k_0(v_0 - v_1) + k_d\dot{e}_1 + k_p e_1 = a_0 + k_0(v_0 - v_1) + k_d\dot{e}_1 + k_p e_1 =$
  $= a_0 + k_0\dot{e}_1 + k_d\dot{e}_1 + k_p e_1 = -\ddot{e}_0 + [k_0 + k_d].\dot{e}_1 + k_p e_1$,
- for $\theta(k) = 1, i \in \{2,3,4\}$:
  $u_i = \lambda a_0 + (1-\lambda)a_{i-1} + k_0(v_0 - v_i) + k_d\dot{e}_i + k_p e_i =$
  $= -\lambda\ddot{e}_0 - (1-\lambda).\sum_{j=0}^{i-1} \ddot{e}_j + k_0 \sum_{j=1}^{i} \dot{e}_j + k_d\dot{e}_i + k_p e_i =$
  $= -\lambda\ddot{e}_0 + (\lambda - 1)(\ddot{e}_0 + \ddot{e}_1 + ... + \ddot{e}_{i-1}) + k_0(\dot{e}_1 + ... + \dot{e}_i) + k_d\dot{e}_i + k_p e_i =$
  $= -\ddot{e}_0 + (\lambda - 1)\sum_{j=1}^{i-1} \ddot{e}_j + k_0 \sum_{j=1}^{i-1} \dot{e}_j + [k_0 + k_d]\dot{e}_i + k_p e_i$

Getting:

$$u_i(k) = \begin{cases} -\ddot{e}_0(k) + [k_0 + k_d].\dot{e}_1(k) + k_p e_1(k), & \theta(k) = 1, i = 1 \\ -\ddot{e}_0(k) + (\lambda - 1)\sum_{j=1}^{i-1} \ddot{e}_j(k) + k_0 \sum_{j=1}^{i-1} \dot{e}_j(k) + [k_0 + k_d]\dot{e}_i(k) + k_p e_i(k), & \theta(k) = 1, i \in \{2,3,4\} \\ k_d\dot{e}_i(k) + k_p e_i(k), & \theta(k) = 0 \text{ or } \theta(k) = 1, i = 0 \end{cases}$$

And let us derive the feedback matrices:

- *Packet received* $\theta(k)=1$, *feedback matrix* K_1

$$\underbrace{\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}}_{u} = \underbrace{\begin{pmatrix} k_p & k_d & 0 & 0 & 0 & 0 & 0 & \cdots & & & & & \cdots & 0 \\ 0 & 0 & -1 & k_p & k_d+k_0 & 0 & 0 & 0 & \cdots & & & & \cdots & 0 \\ 0 & 0 & -1 & 0 & k_0 & \lambda-1 & k_p & k_d+k_0 & 0 & \cdots & & & \cdots & 0 \\ 0 & 0 & -1 & 0 & k_0 & \lambda-1 & 0 & k_0 & \lambda-1 & k_p & k_d+k_0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & k_0 & \lambda-1 & 0 & k_0 & \lambda-1 & 0 & k_0 & \lambda-1 & k_p & k_d+k_0 \end{pmatrix}}_{K_1} \underbrace{\begin{pmatrix} e_0 \\ \dot{e}_0 \\ \ddot{e}_0 \\ \vdots \\ e_4 \\ \dot{e}_4 \\ \ddot{e}_4 \end{pmatrix}}_{x_{OL}}$$

- *Packet lost* $\theta(k)=0$, *feedback matrix* K_0

$$\underbrace{\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}}_{u} = \underbrace{\begin{pmatrix} k_p & k_d & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & k_p & k_d & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}}_{K_0} \underbrace{\begin{pmatrix} e_0 \\ \dot{e}_0 \\ \ddot{e}_0 \\ \vdots \\ e_4 \\ \dot{e}_4 \\ \ddot{e}_4 \end{pmatrix}}_{x_{OL}}$$

Let's derive the closed-loop system matrices now, we have:
$x_{OL}(k+1) = A_{OL}x_{OL}(k) + B_{OL,1}d(k) + B_{OL,2}u(k)$,
and
$u(k) = K_i x_{OL}(k)$.

Therefore "closing the loop" results in:
$$x(k+1) = A_{OL}x_{OL}(k) + B_{OL,1}d(k) + B_{OL,2}K_i x_{OL}(k) =$$
$$= \left[A_{OL} + B_{OL,2}K_i \right] x_{OL}(k) + B_{OL,1}d(k) =$$
$$= \begin{bmatrix} \underbrace{A_{OL} + B_{OL,2}K_i}_{A_{CL,i}} & \underbrace{B_{OL,1}}_{B_{CL}} \end{bmatrix} \begin{pmatrix} x_{OL}(k) \\ d(k) \end{pmatrix}$$
where subscripts CL, i stand for closed-loop, resp. packet lost/received

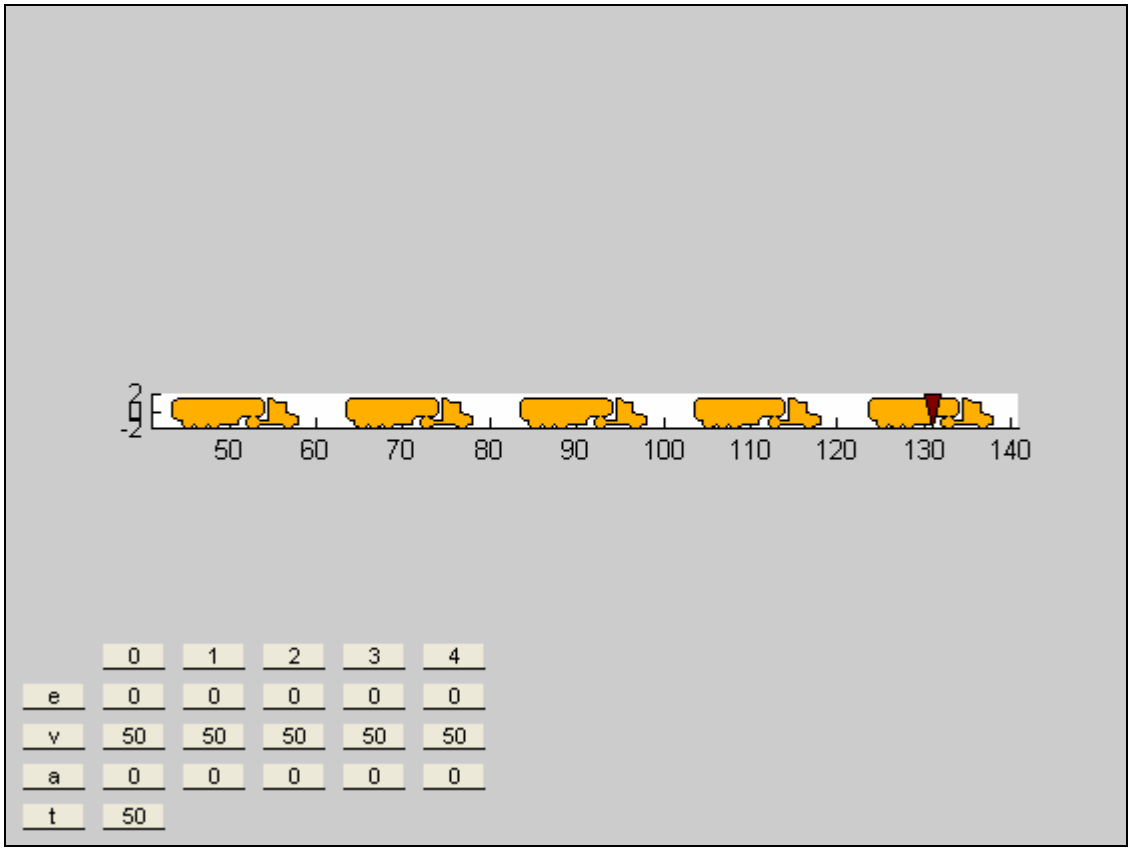
Setting $C_{CL} \equiv C_{OL}$ and $x_{CL}(k) \equiv x_{OL}(k)$ we arrive at the full closed-loop system matrix equation (notice that C_{CL} and B_{CL} are independent of $\theta(k)$, the packet delivery status):

$$\begin{bmatrix} x_{CL}(k+1) \\ z(k) \end{bmatrix} = \begin{bmatrix} A_{CL,i} & B_{CL} \\ C_{CL} & 0 \end{bmatrix} \begin{bmatrix} x_{CL}(k) \\ d(k) \end{bmatrix}.$$

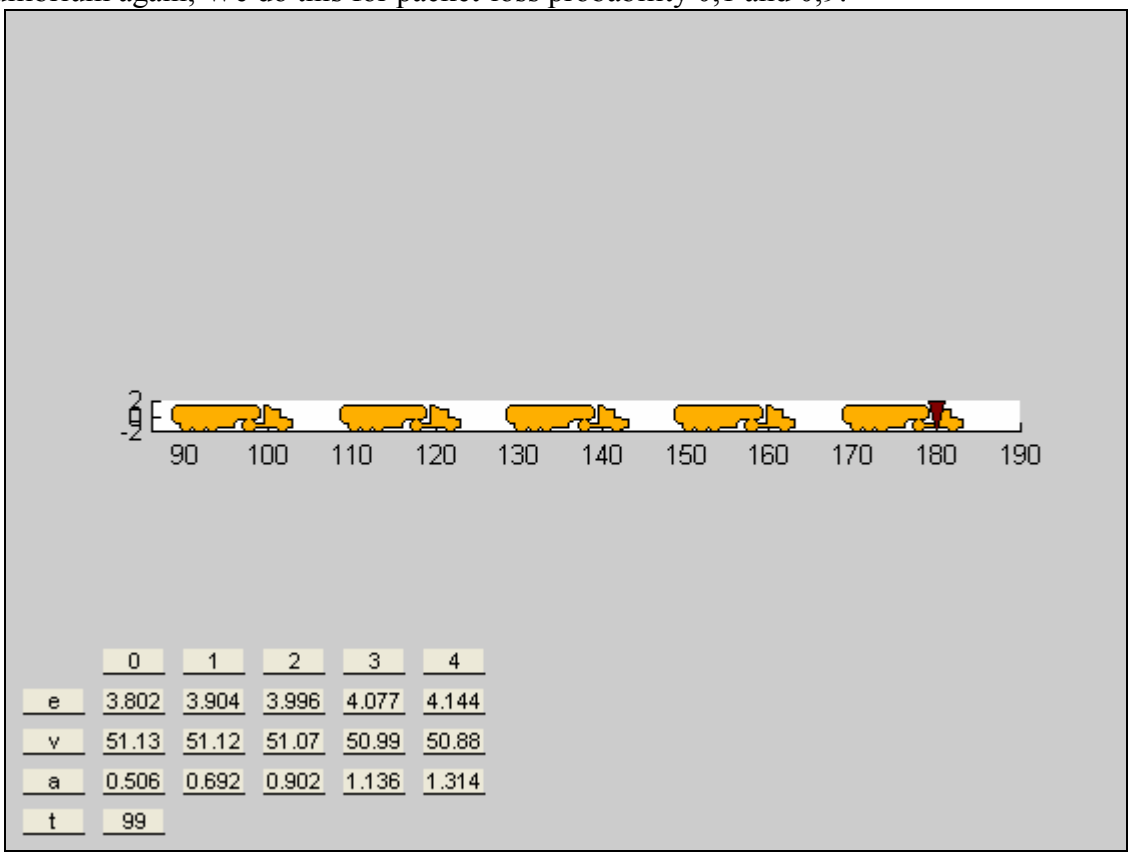
4 Simulations

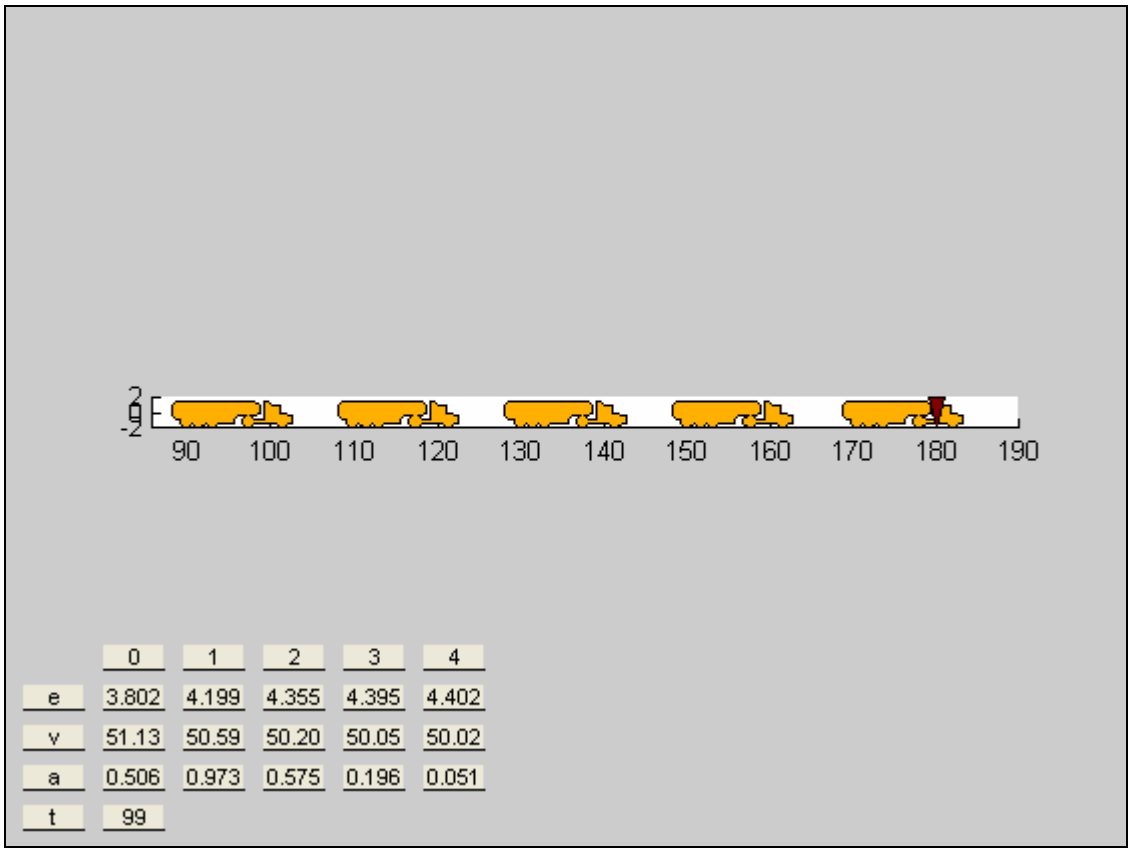
In this section we present a couple of animations/scenario's generated using MATLAB.

A. System at equilibrium: moving with constant speed and spacing:

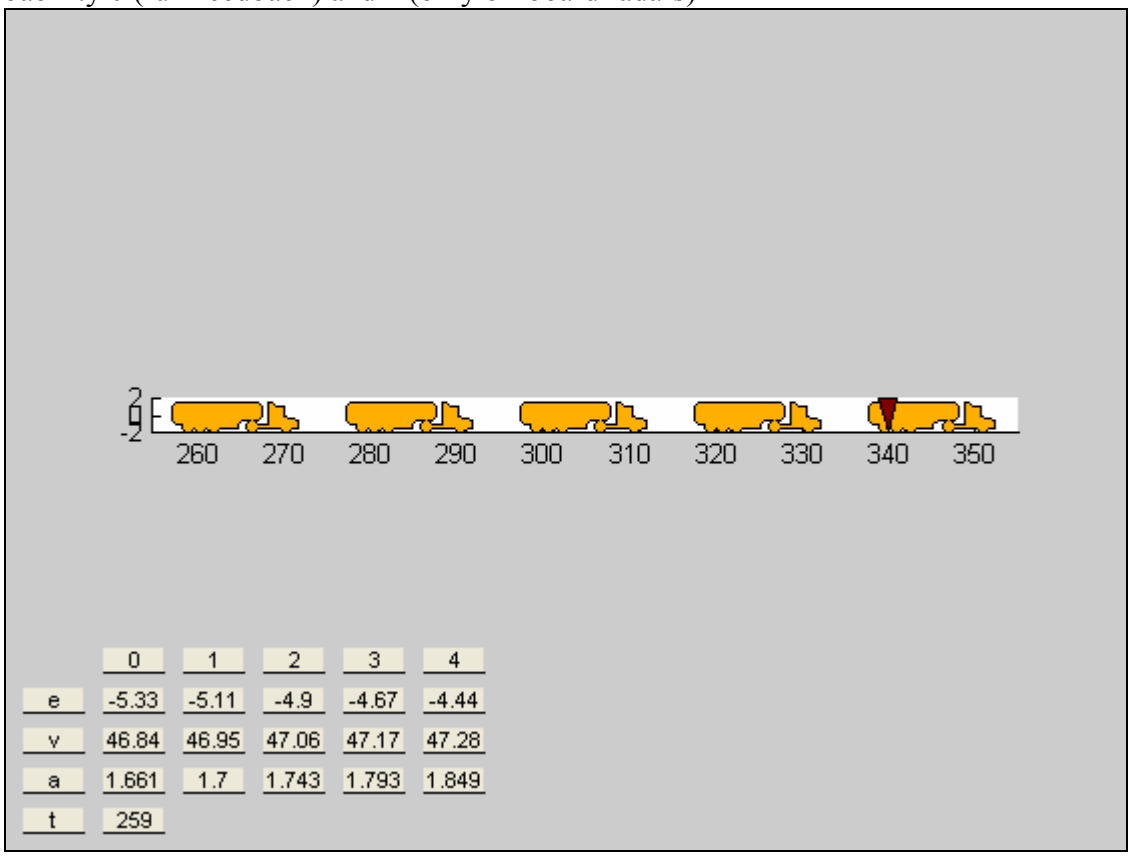


B. In this simulation we test the impulse-response of the system to the sudden “forward” jump of the reference trajectory, thus the platoon has to accelerate in order to reach equilibrium again; We do this for packet-loss probability 0,1 and 0,9:

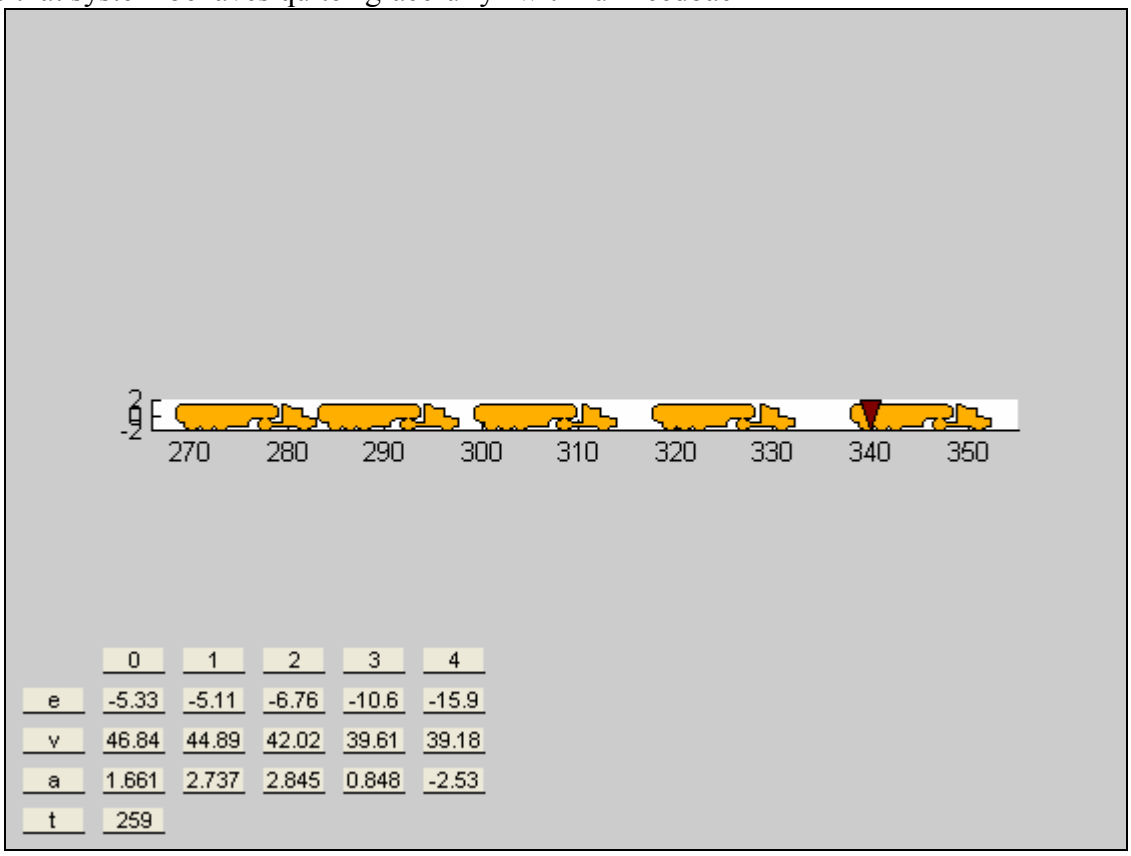




C. This simulation is similar, only we do some braking now for the extreme case of probability 0 (full feedback) and 1 (only on-board radars)

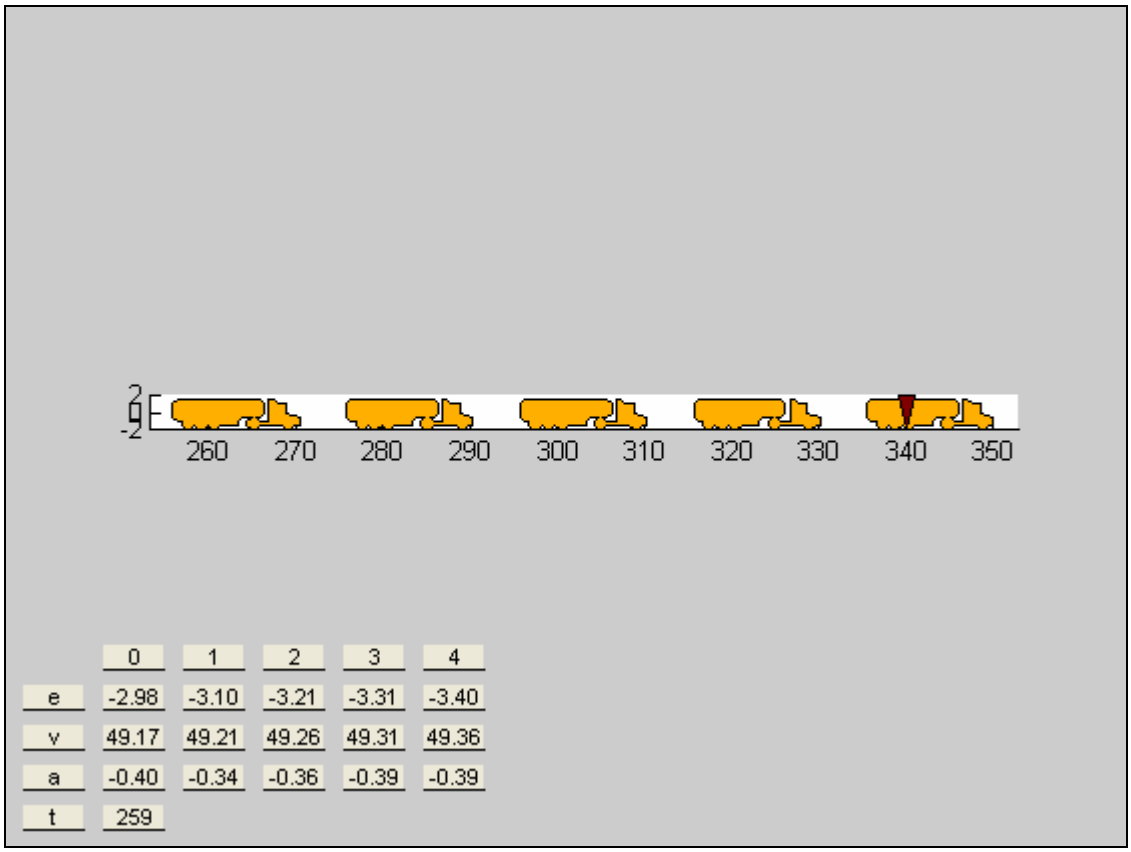


See that system behaves quite “gracefully” with full feedback



”Bumper kissing” was unavoidable with such low network quality.

D. and finally somehow realistic situation, uniform random disturbance, suppose it's quite windy this time, probability of the packet loss is 0.5



5 (Stochastic) Frequency domain analysis

In this section we would like to analyze the system in its frequency domain (see [7] and [8]). This is especially "challenging" because of the randomness in the system. The main "quantity of interest" in this domain (just like in the deterministic case) is the gain of the system for the frequency values lying on the unit disk in the complex plane:

- For $z \in \mathbb{C} : z = e^{i\omega}, \omega \in [0, 2\pi], |z| = 1$,
- Find $\max_{|z|=1} \|G_z\|$,
- Where G_z is actually the amplification of the $\frac{2\pi}{\omega}$ frequency in the disturbance/input.

We actually have "stochastic" transfer function: suppose we have some realisation of the Markov network-noise process $\theta(k) : k = 1, 2, 3....$, then the transfer function of some z would be:

$$G(z) = D + \tfrac{1}{z} CB_{\theta(1)} + \tfrac{1}{z^2} CA_{\theta(1)} B_{\theta(2)} + \tfrac{1}{z^3} CA_{\theta(2)} A_{\theta(1)} B_{\theta(3)} + ...$$

Therefore:

$$G(z) = D + \sum_{k=1}^{\infty} \left[\tfrac{1}{z^k} C \left(\prod_{j=1}^{k-1} A_{\theta(k-j)} \right) B_{\theta(k)} \right].$$

Thus if we would simulate multiple realisation of the transfer function (for a fixed z), "the goddess Fortuna" would always choose different realisation of the Markov process $\theta(k)$, nevertheless nothing can stop us from taking the expectation of this stochastic transfer function.

So we have a stochastic variable G_z, defined on the space $(\Omega, \Sigma, \mathbb{P})$, where:

- Sample space Ω is the space of all infinite Markov processes: $\Omega = \{\{0,1,1,0,0,1.....\}, \{1,0,1,0,0,1.....\}, \{0,1,0,0,1,1.....\}, ...\}$, sometimes noted as $\Omega = \{0,1\}^{\infty}$,
- With its corresponding sigma-algebra Σ, generated as usually by the "cylinder sets", actually a π-system consisting of all infinite sequences of $\{0,1\}$ with fixed first n terms:
  $\Sigma = \sigma(I) : I$ is a π-system such that:
  $I = \{|\theta(1), \theta(2), ..., \theta(n)\rangle_n : \theta(1) = \alpha_1, \theta(2) = \alpha_2, ..., \theta(n) = \alpha_n, \alpha_i \in \{0,1\}, n \in \mathbb{N}\}$
- Probability measure $\mathbb{P}$, actually a law of a sequence of independent and identically distributed Bernoulli random variables (with "chance of success" p), as our Markov process with $p_{ij} = p_j, \forall i, j \in \{0,1\}$ is actually a Bernoulli process, thus:
  $\mathbb{P}(\theta(1) = \alpha_1, \theta(2) = \alpha_2, ..., \theta(n) = \alpha_n) = p^k (1-p)^{n-k}$, where $k = \#\alpha_i : \alpha_i = 1$.

We start by stating that the problem of calculating $\mathbb{E}G_z$ is well-posed/consistent/meaningful, because the set of θ's where the system is "less stable than specified p" has measure/probability zero:

$$\mathbb{P}\left(\left\{\lim_{k\to\infty}\frac{\sum_{j=1}^{k}\theta(j)}{k}\neq p\right\}\right)=0\,.$$

In other words: the set of possible realizations/trajectories where the probability of packet-loss is different than pre-specified p is a $\mathbb{P}$-negligible/null set.

Now we calculate the expectation of G_z:

$$\mathbb{E}G_z=\mathbb{E}\left(D+\sum_{k=1}^{\infty}\left[\tfrac{1}{z^k}C\left(\prod_{j=1}^{k-1}A_{\theta(k-j)}\right)B_{\theta(k)}\right]\right)=\int\limits_{\substack{\theta=\{\{0,0,\ldots\},\\ \{0,1,\ldots\},\ldots\}}}\left(D+\sum_{k=1}^{\infty}\left[\tfrac{1}{z^k}C\left(\prod_{j=1}^{k-1}A_{\theta(k-j)}\right)B_{\theta(k)}\right]\right)d\mu_\theta\,,$$

where μ_θ is a measure/law of the Markov process θ.

Because the system is SMS (stable), the expectation is finite and then we can use the **Fubini theorem** (see [10]) to switch the integration order (the infinite sum is actually a discrete integral), thus expectation becomes:

$$D+\sum_{k=1}^{\infty}\left[\tfrac{1}{z^k}C\int\limits_{\substack{\theta=\{\{0,0,\ldots\},\\ \{0,1,\ldots\},\ldots\}}}\left(\left[\prod_{j=1}^{k-1}A_{\theta(k-j)}\right]B_{\theta(k)}\right)d\mu_\theta\right]=D+\sum_{k=1}^{\infty}\left(\tfrac{1}{z^k}C\mathbb{E}\left[\left(\prod_{j=1}^{k-1}A_{\theta(k-j)}\right)B_{\theta(k)}\right]\right).$$

Now we use the fact that the individual "steps" in the Markov process are independent, getting:

$$D+\sum_{k=1}^{\infty}\left(\tfrac{1}{z^k}C\mathbb{E}\left(\prod_{j=1}^{k-1}A_{\theta(k-j)}\right)\mathbb{E}B_{\theta(k)}\right)=D+\sum_{k=1}^{\infty}\left(\tfrac{1}{z^k}C\left(\prod_{j=1}^{k-1}\mathbb{E}A_{\theta(k-j)}\right)\mathbb{E}B_{\theta(k)}\right).$$

But moreover they are also identically distributed, thus this becomes:

$$D+\sum_{k=1}^{\infty}\left(\tfrac{1}{z^k}C\left(\mathbb{E}A_{\theta(1)}\right)^{k-1}\mathbb{E}B_{\theta(1)}\right).$$

Now we see that because:
$\mathbb{E}A_{\theta(1)}=pA_0+(1-p)A_1$ and $\mathbb{E}B_{\theta(1)}=pB_0+(1-p)B_1$, where indexes by the matrices means network packet received/lost, we can simplify this transfer function like in the deterministic case, and finally arriving at:

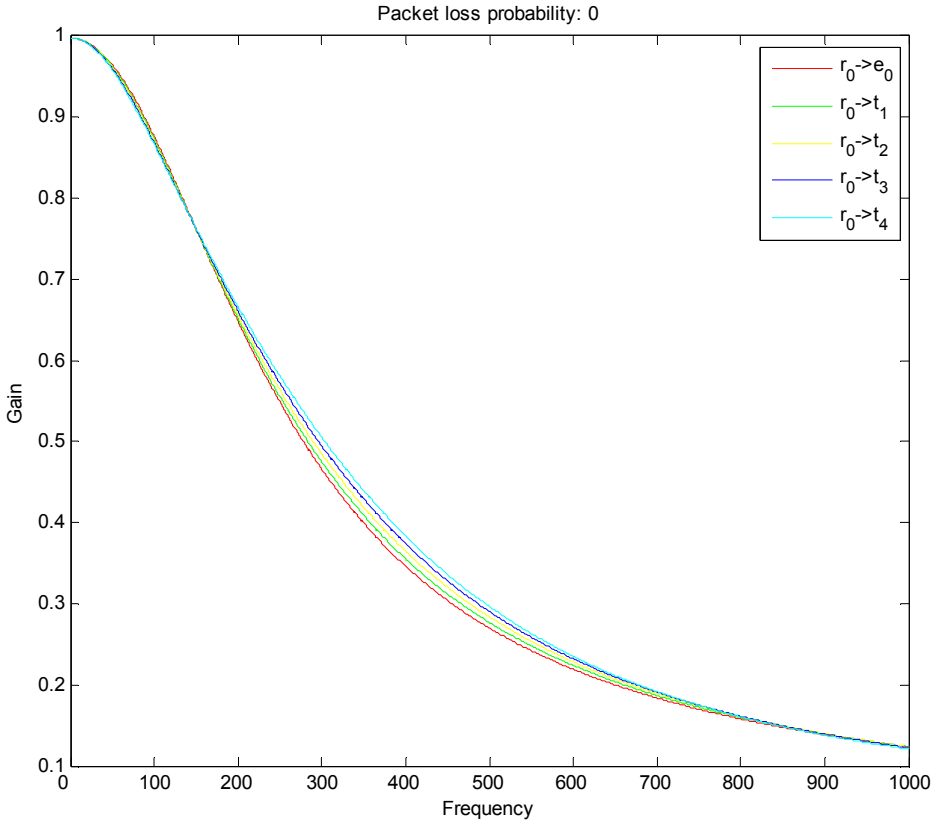
$$\mathbb{E}G_z=D+C\left(zI-\left[pA_0+(1-p)A_1\right]\right)^{-1}\left(pB_0+(1-p)B_1\right).$$

With this formula we can perform "standard" frequency domain analysis, thus searching for: $\max_{\|z\|=1}\mathbb{E}G_z$, for given packet-loss probability p.

5.1 Frequency domain simulations

Here we present a couple of $\mathbb{E}G_z$ gain graphs, with varying packet-loss probability, to see if the theoretical expectation of the transfer function is consistent with the simulations. We will display only the "left-most" part of the frequency spectrum, as the maximum is always there.

We begin "on the safe side", there is full feedback information, the packet-loss probability is zero:



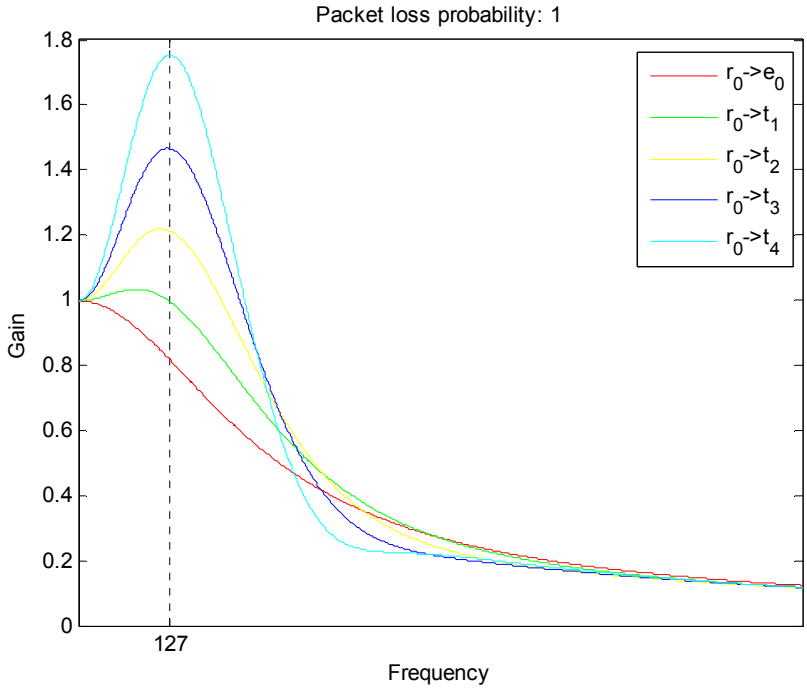
With this type of the graph the *x-axis* should be interpreted in the following way:

$0 \leftarrow$ frequency $\rightarrow \infty$ (actually equal to the sampling period),

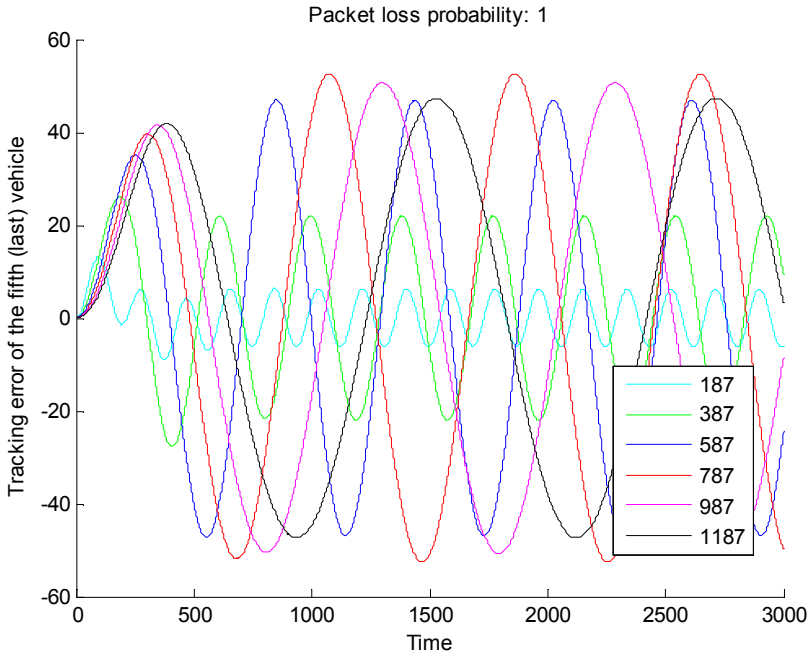
or equivalently:

$\infty \leftarrow$ period of frequency $\rightarrow 1$ (in units of the sampling period).

Now we jump right into a “disaster zone”, there is no network feedback available, only the information form the on-board radars:

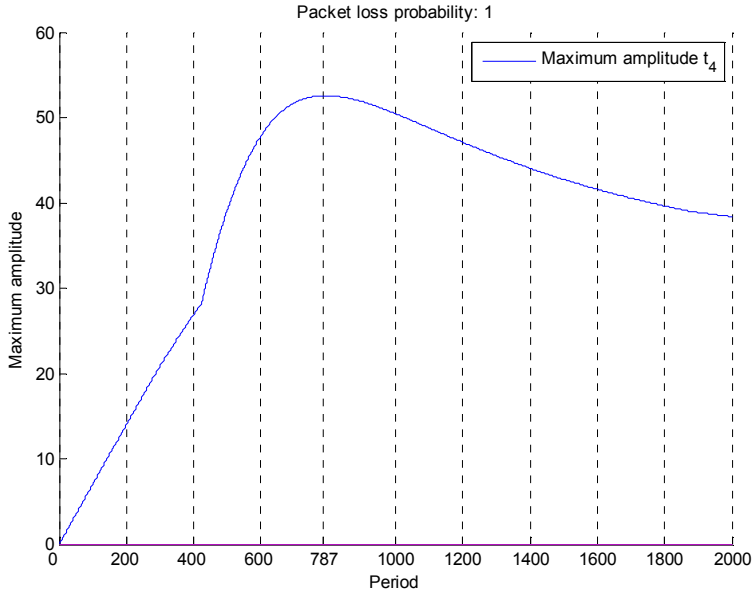


We see a “smooth” maximum of 1.7525 at a frequency of $k = 127$ (for the fifth vehicle), and because we are using the discretization step of $N = 100000$, we get the “dangerous” period of the disturbance frequency: $\frac{N}{k} \approx 787$, so let’s disturb the system with the periodic disturbance (sinusoid) with this period and amplitude of 30:



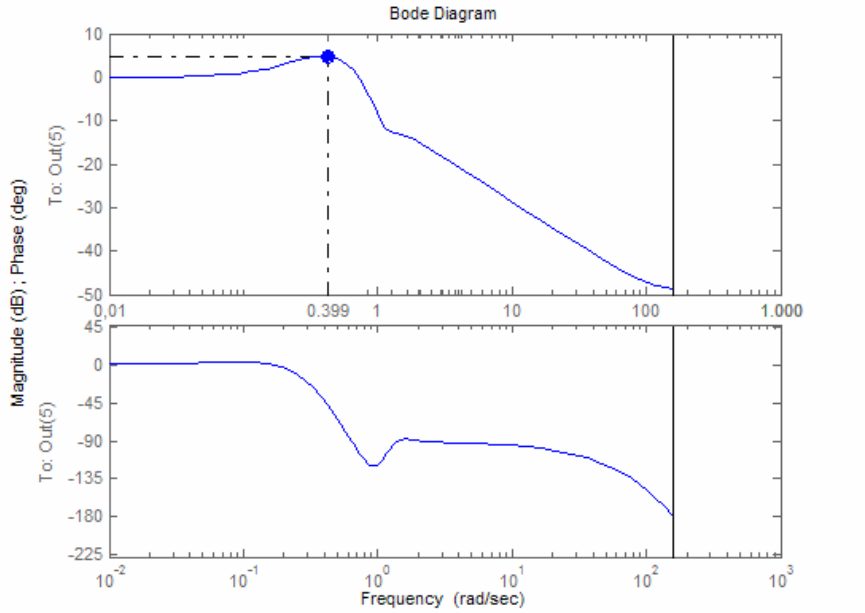
And indeed that frequency with the period of 787 causes the largest error in the tracking error of the last vehicle: $\max_t(t_4) = 1,7525.30 = 52,57$, as expected by $\mathbb{E}G_z$ calculation.

The next diagram shows the result of the simulation, where we disturb the system with 2000 different frequencies and look for the maximum amplitude in the tracking error of the last vehicle:



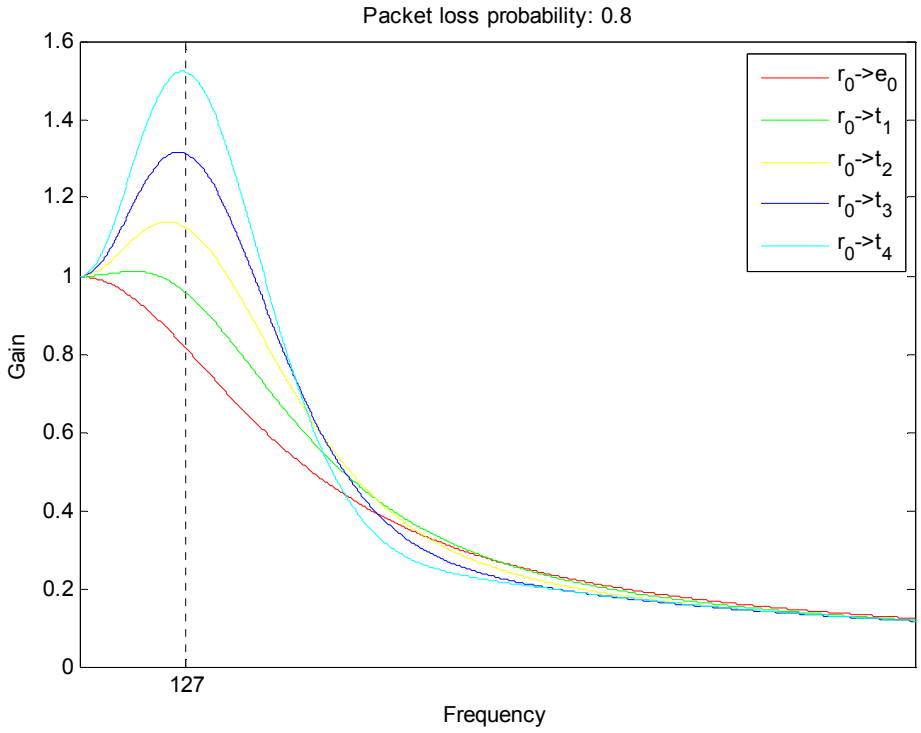
Notice again “smooth” maximum at 787, the results are consistent.

To make the picture complete, we also used the build-in MATLAB function to draw a *Bode plot* for the tracking error of the last vehicle for the same probability 1:

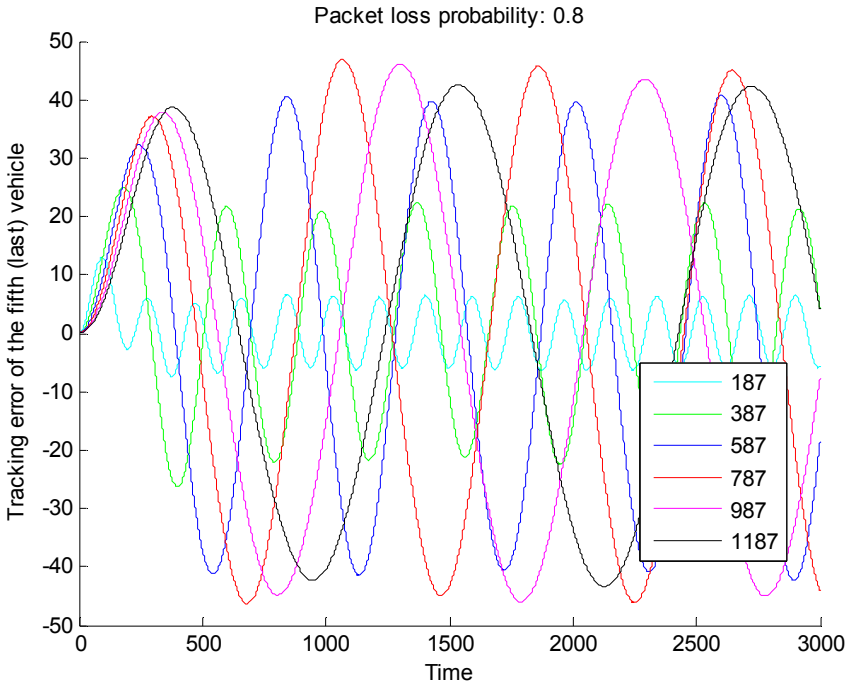


We see that the maximum is attained at ≈ 0.399, interpretation is that the maximum resonance frequency period for this input-output combination is 787 (from $\max_{\|z\|=1} \mathbb{E}G_z = \frac{N}{787}$), but expressed in 20ms units (sampling frequency period), therefore we recalculate the period: $T = 787.0,02 = 15,74$ seconds, and this gives frequency $f = \frac{1}{T} = 0,0635$ Hz, and finally in radians per seconds: $f_{rs} = 0,0635.2\pi \approx 0,399$.

Let's also try the other packet-loss probability:

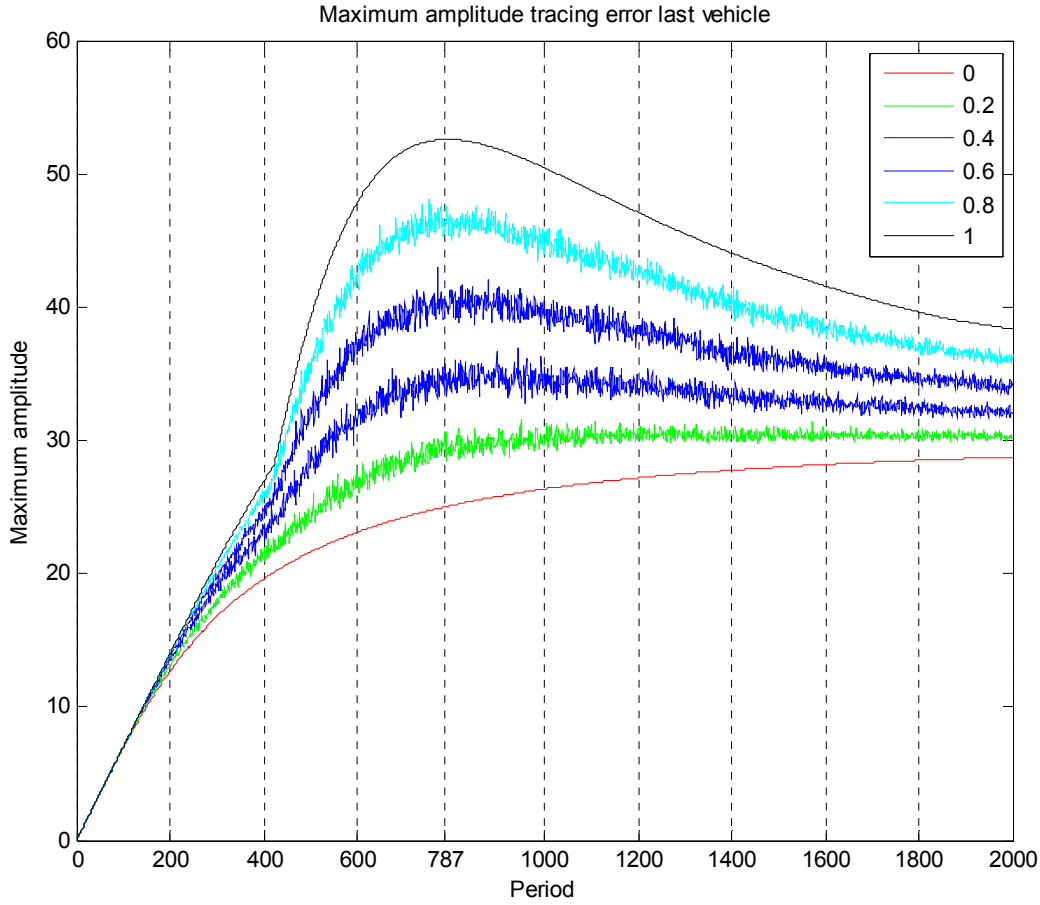


Again the maximum of 1.5213 at the same spot, and the second diagram shows the consistency as: $\max_t(t_4) = 1,5213.30 = 45,63$.



Notice also that the maximum tracking error amplitude for the last vehicle gets smaller with decreased packet-loss probability, exactly as expected.

Next diagram again shows the result of the simulation, where we disturb the system with 2000 different frequencies and look for the maximum amplitude in the tracking error of the last vehicle, now for different packet-loss probabilities:



The maximal amplitude gets smaller with decreasing probability, also notice how the graph gets "stochastic" for $p \notin \{0,1\}$.

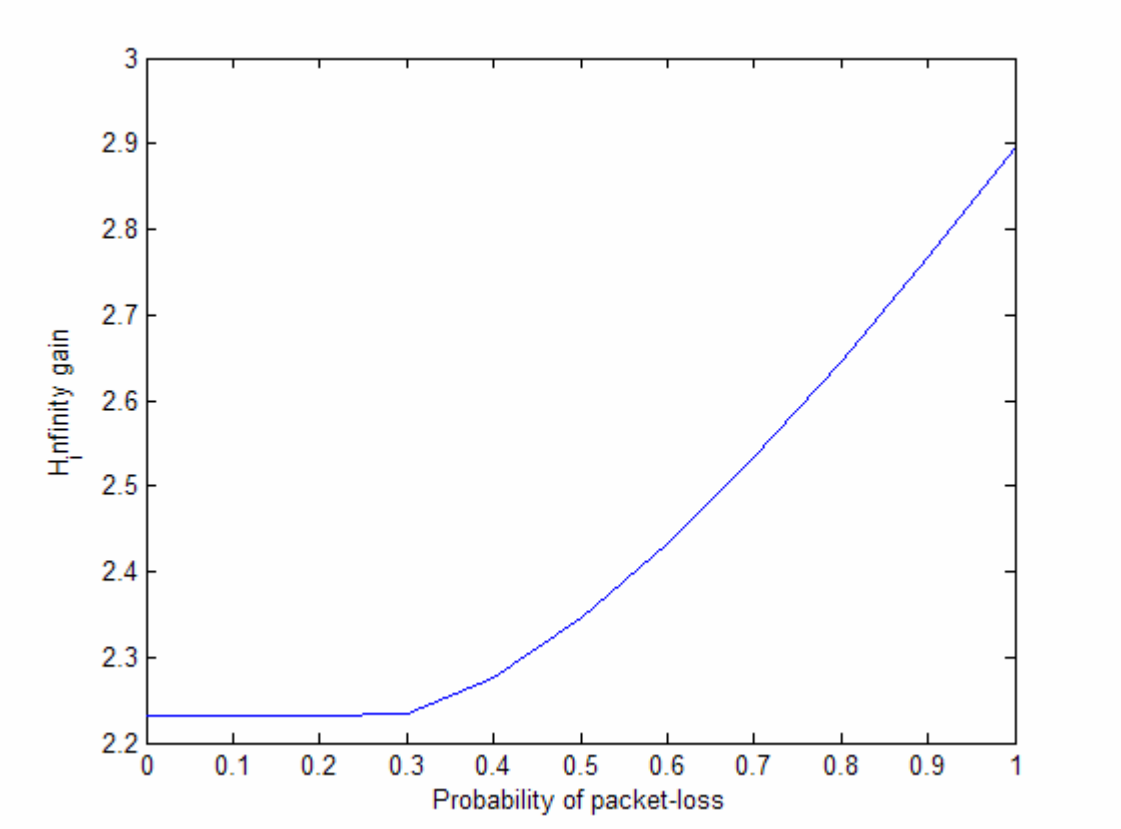
5.2 Linear matrix inequality analysis

Here we reproduce the LMI analysis of the smallest achievable H_∞ gain for a specific packet-loss probability.

This actually means that for different values of the packet loss probability $p \in [0,1]$ we let MATLAB (see [3]) solve the LMI defined by:

- *objective* of minimizing γ, that is: H_∞ gain of the system,
- and *constraints* $\begin{bmatrix} \begin{bmatrix} Z & 0 \\ 0 & \gamma^2 I \end{bmatrix} & [*]^T & [*]^T \\ \sqrt{p_1}\begin{bmatrix} A_{cl,1}Z & B_{cl,1} \\ C_{cl,1}Z & D_{cl,1} \end{bmatrix} & \begin{bmatrix} Z & 0 \\ 0 & I \end{bmatrix} & [*]^T \\ \sqrt{p_2}\begin{bmatrix} A_{cl,2}Z & B_{cl,2} \\ C_{cl,2}Z & D_{cl,2} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} Z & 0 \\ 0 & I \end{bmatrix} \end{bmatrix} > 0, Z > 0$.

The resulting minimal gain is plotted against its packet-loss probability:



The diagram resembles the one in the paper [1], see figure 5, the scale and the "knee" position are different, nevertheless the curve has the same general shape. The differences are to be blamed on the modelling differences, as the authors of the paper apparently did not reveal everything about exact form of their LMI analysis model.

6 Conclusion/Discussion:

We could rebuild the model based on the paper [1], the qualitative results obtained are also consistent (degradation of stability/frequency domain analysis/LMI analysis), nevertheless there were (small) quantitative differences. These are due to the ambiguity of authors in [1] to describe their used model. Often these ambiguities were very small details, in one particularly crucial case literally one word has made the whole difference: this was the case when we originally set-up the system based on the incorrect control objective: trying to force all *spacing* errors to zero, exactly as suggested by the corresponding formulas. Nevertheless these were inconsistent with the textual description which mentioned *tracking* errors, and voila! - all worked like a charm.

References:

[1] P. Seiler, Raja Sengupta, "An H_∞ Approach to Networked Control", IEEE Transactions on Automatic Control, Vol. 50, No. 3, March 2005.
[2] D. Simon, "From Here to Infinity", Embedded Systems Programming, July 2000.
[3] MATLAB Help, "Introduction to Linear Matrix Inequalities".
[4] P. Hokayem, Ch.T. Abdallah, "Inherent Issues in Networked Control Systems: A Survey", Proceeding of the 2004 American Control Conference, Boston, Massachusetts, June-July 2004.
[5] X. Liu, A. Goldsmith, S.S. Mahal, J.K. Hedrick, "Effects of Communication Delay on String Stability in Vehicle Platoons", IEEE Intelligent Transportation Systems Conference Proceedings, August 2001.
[6] P. Seiler, R. Sengupta, "Analysis of Communication Losses in Vehicle Control Problems", June 2001.
[7] Ch. Heij, A, Ran, F. van Schagen, "Introduction to Mathematical Systems Theory", 2007.
[8] G.F Franklin, J.D. Powell, A. Emami-Naeini, "Feedback Control of Dynamical Systems", 5th Edition, 2006.
[9] W. Zhang, "Stability Analysis of Networked Control Systems", PhD Thesis, August 2001.
[10] D. Williams, "Probability with Martingales", Cambridge University Press, October 1990.