Modelling Urban Transport Networks

Jan Přikryl, Pavla Pecherková, Jindřich Duník, Miroslav Flídr

Department of Adaptive Systems
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
and
Department of Cybernetics
University of West Bohemia

2. společná konference Výzkumného centra DAR,
ÚTIA AV ČR, Praha, 12. října 2005

Outline

Outline

Urban Traffic Networks

Current situation in many cities

- Amount of vehicles rises rapidly
- Existing infrastructures cannot accommodate current traffic
- Regular congestions

Possible solutions

- Rebuild parts of the network
- Reroute transit traffic
- Make drivers pay for entering central zones

$\Rightarrow$ we need efficient traffic control mechanisms to increase throughput on existing traffic network

Software Framework for Optimal Traffic Control

Current situation

ÚTIA hosts two R&D projects targeted at design of efficient adaptive urban traffic control mechanisms.

Simple "proof of concept" works $\Rightarrow$
$\Rightarrow$ rapid prototyping $\Rightarrow$
$\Rightarrow$ a lot of single-purpose code written in MATLAB.

Large MATLAB-based code libraries (e.g. Mixtools) are being developed for DAR $\Rightarrow$ traffic control tests

DAR team needs a software framework for design and testing of hierarchical traffic controller.

Outline

Software Framework for Optimal Traffic Control
Requirements

Final requirements

- Use MATLAB so that we can benefit from the existing large code-base
- Mimic the object-oriented approach in MATLAB and rely on our users (!!!) to not touch things they are not allowed to touch
- Separate model and controller interface
- Describe the network using structured MATLAB batch files that can be easily transformed into XML in the future

This allows also for interconnection to new generation of Mixtools.

Software Framework for Optimal Traffic Control

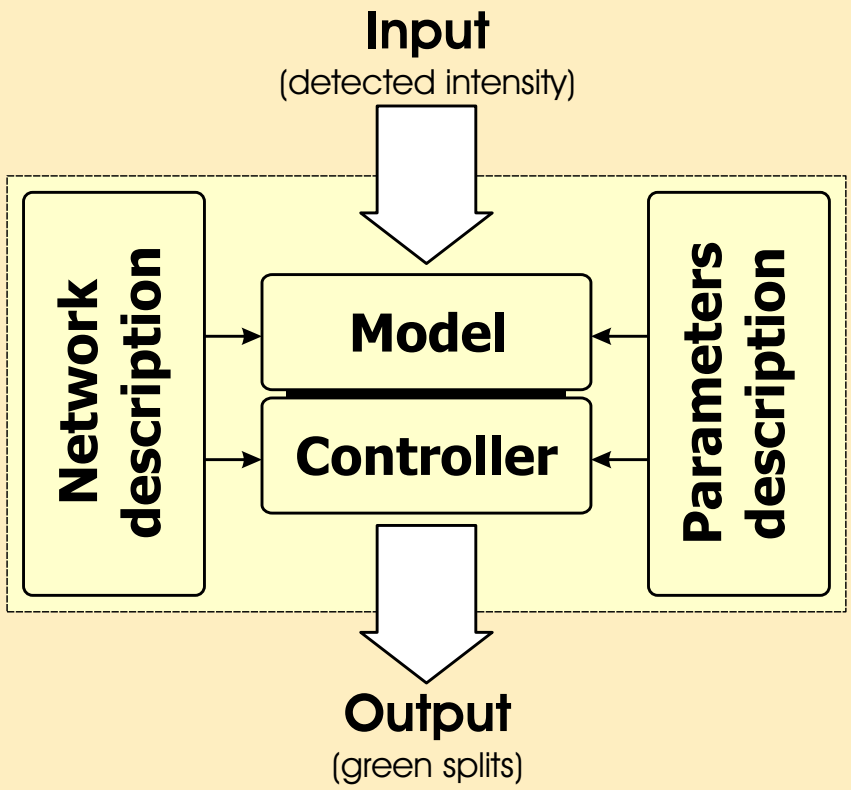
Software Model

Decomposition

Four basic building blocks:

- **Network description** [junction and microregion interconnection, junction parameters, signal plans]
- **Parameters description** input data specification, model and controller parameters
- **Model** implementation
- **Controller** implementation

Scheme



Urban Network Description

Description of

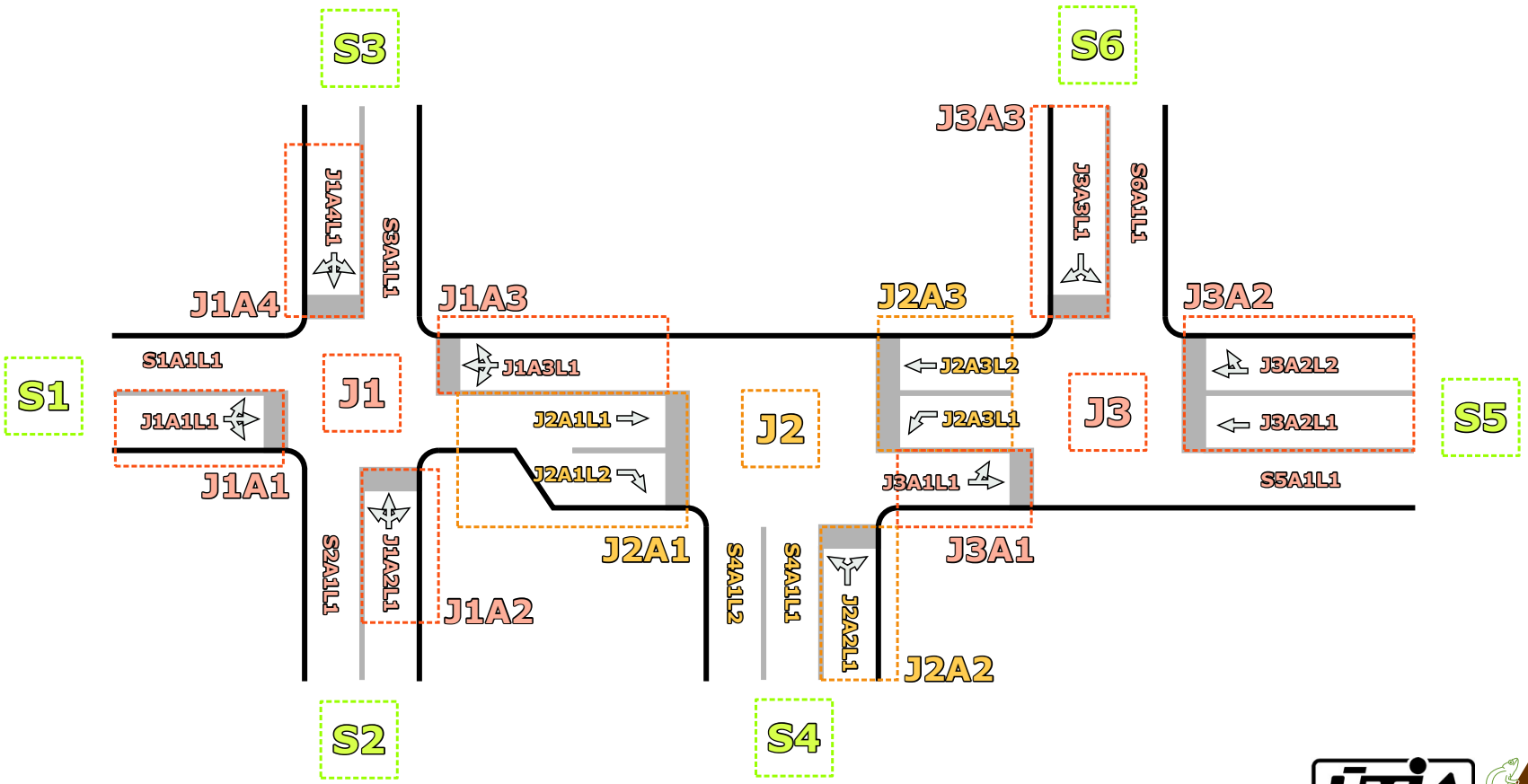
- Junctions, arms, and lanes [structure, input and output detectors]
- Turning rates
- Signal plans [green splits, saturation flows]
- Microregion structure [interconnection of junctions]
- Network structure [interconnection of microregions]

Example

```
%% Intersection of Zborovská - V Botanice
BeginJunction('J01');
  %% Zborovská input
  BeginArm('A1');
    CreateLane('L1','FullLane',196,35);
  EndArm();
  ...
  %% Fixed turning rates
  BeginTurningRates ();
    % From Zborovská straight ahead
    TurningRate('A1','A3',0.79);
    ...
  %% Fixed signal plan
  BeginSignalPlan();
    BeginPhase('P1');
      GreenTime(20);
      ClearingTime(4);
      % Saturation flow 3800 vehcls/hr.
      GreenForLane('A1','L1',3800.0);
      ...
```

Urban Network Description

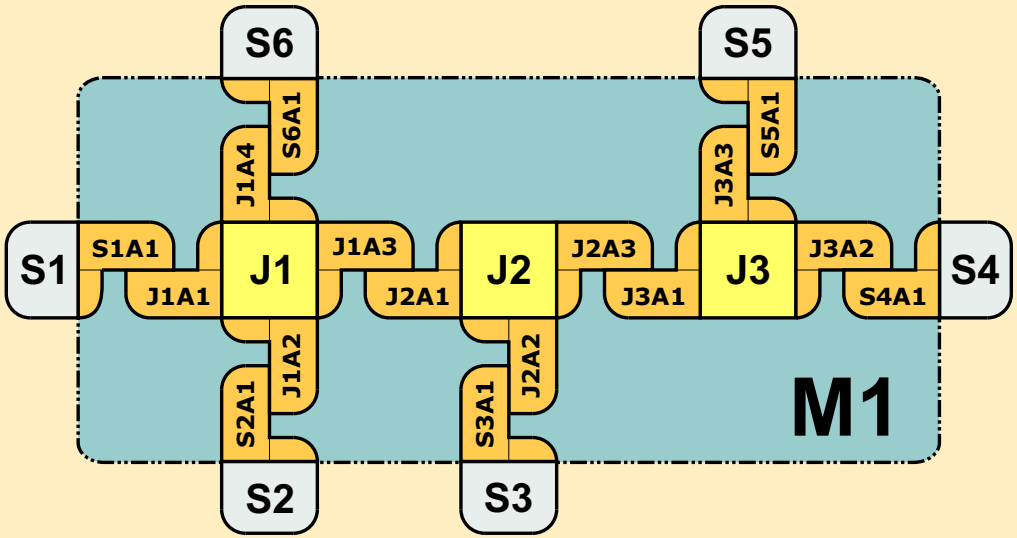
Microregion Example



Urban Network Description

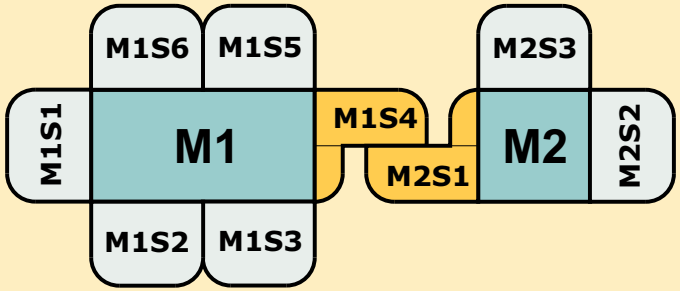
Hierarchical Network Decomposition

Microregion



Simple network

Second hierarchy level
Two microregions only



Model and Controller Implementation

- Framework
  - defines API
  - reads in global data – network structure and parameters
- Model
  - responsible for transforming global data into own internal data
  - predicts queue lengths using measured junction input and output intensities for given signal settings
- Controller
  - optimises the signal plan targeting at minimum queue lengths
  - defines green splits plan based on predicted queue lengths and output intensities

Outline

Model
Short Overview

In development at IITA CAS

- Basic assumption: delay $\sim$ queue length
- Internal states: $x_t = \mathbf{A}x_{t-1} + \mathbf{B}z_{t-1} + \mathbf{F} + e_t$
- Output: $y_t = \mathbf{C}x_t + \mathbf{G} + \epsilon_t$
- e_t, ϵ_t is noise (accounts for unobservable data)
- z_t are traffic control variables
- x_t holds queue lengths, traffic input and output intensities
- Matrices contain coefficients describing parameters of the modelled network (turning rates, saturation flows, queue relationships)

Model
Queue Length Estimation

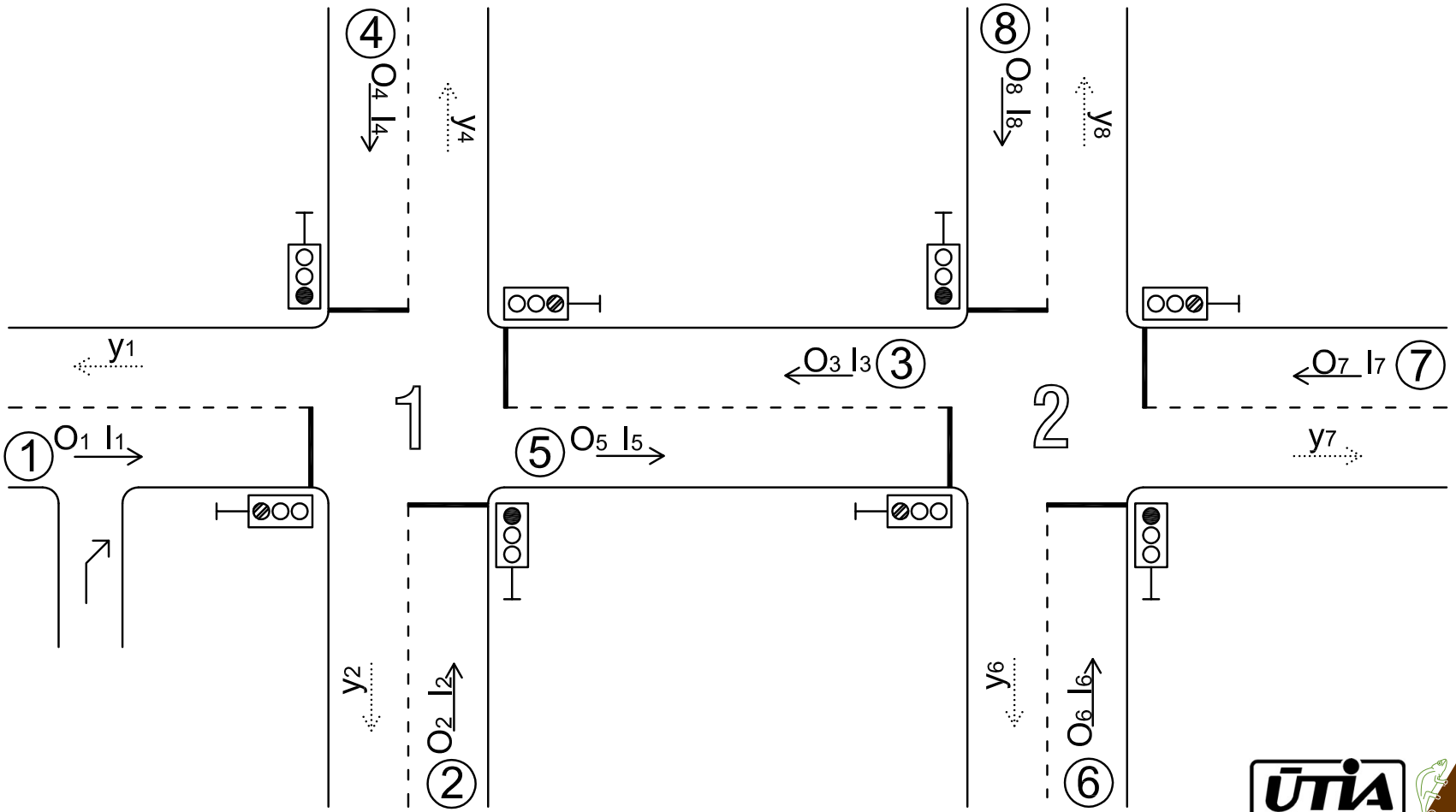
Queue length has to be estimated.

1. We know all parameters
   - Artificial case
   - Model is linear
   - Standard Kalman filter
2. Some parameters are unknown
   - This is reality
   - Nonlinearity in parameters (estimated *and* used for estimation)
   - Non-linear Kalman filter – DD1, DD2 variants
   - Some reservation about performance

How does the non-linear estimation work?

Kalman Filtering

Testing Microregion



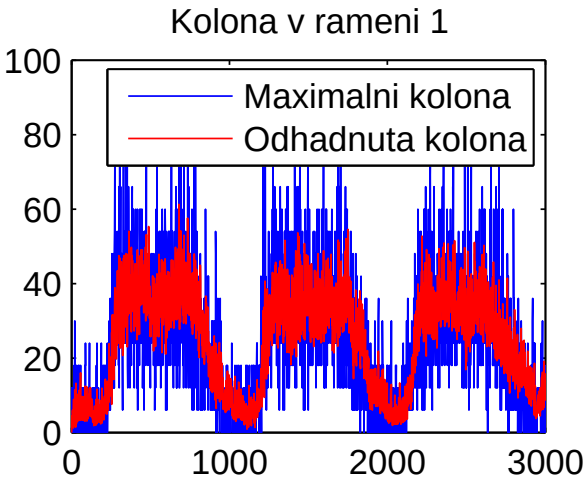
Kalman Filtering
Influence of Non-linearity

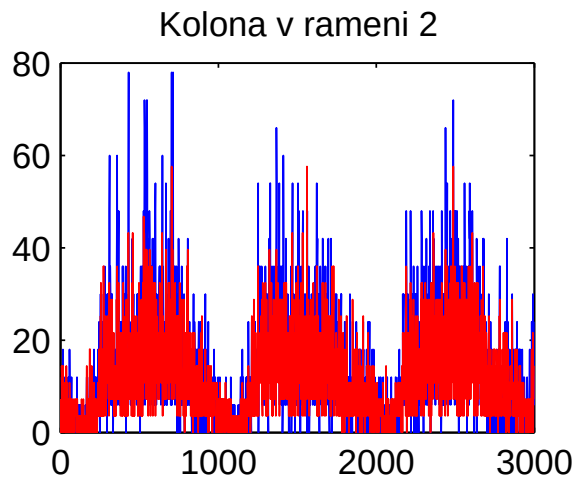
Classical Kalman should not perform well for non-linear models. And it surely doesn't.

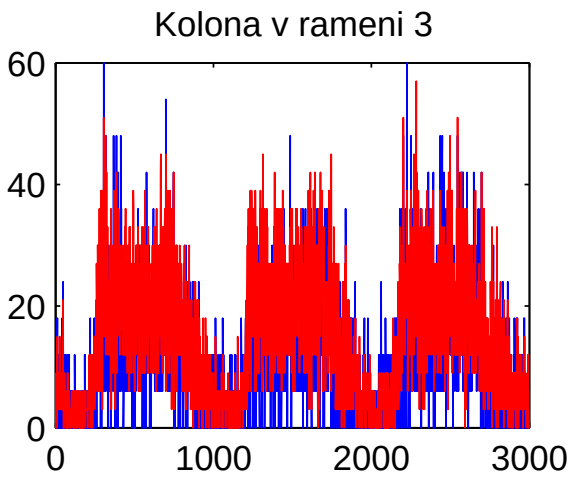
	KF	DD1	KF (param)	DD1 (param)
$\Delta_{\min}$	$3.4385 \cdot 10^5$	$3.3959 \cdot 10^5$	$4.3468 \cdot 10^5$	$3.4742 \cdot 10^5$
MSE	$5.6758 \cdot 10^5$	$5.5643 \cdot 10^5$	$9.2531 \cdot 10^5$	$5.6967 \cdot 10^5$

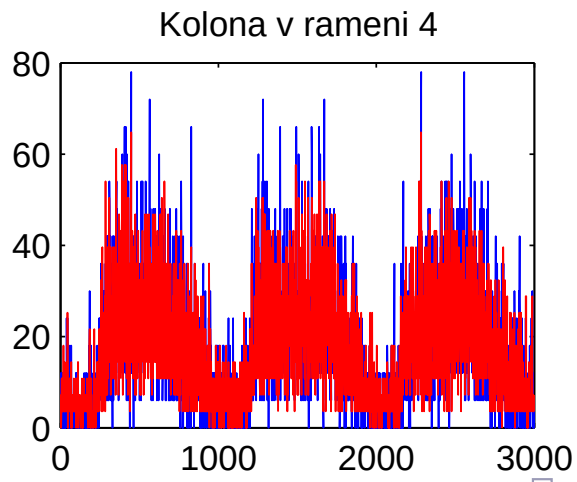
Results

Queue Length Estimation with DD1



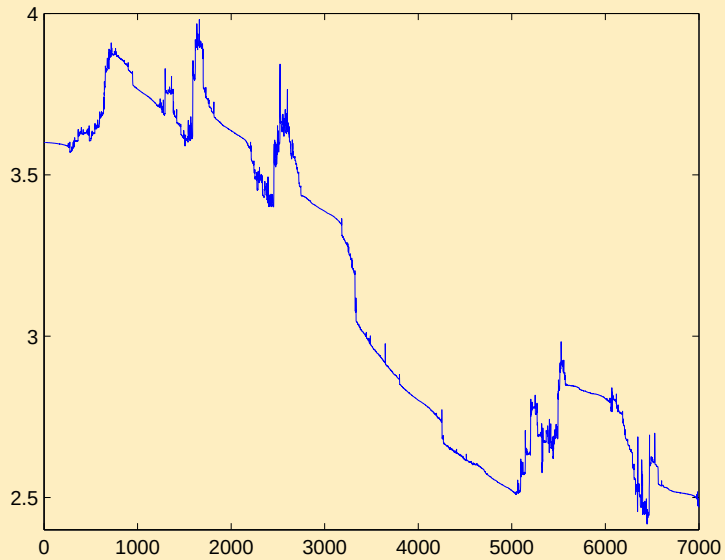






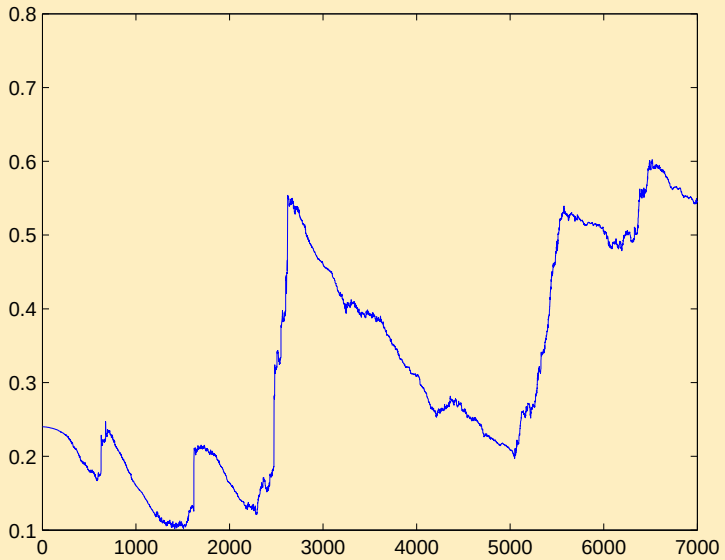
Results
Estimated Internal Parameters

Model parameter κ



[queue length $\rightarrow$ occupancy]

Model parameter β



[occupancy $\rightarrow$ occupancy]

Summary

- Software framework being implemented in MATLAB
  - Pseudo-class mechanism
  - Easy replacement of model and controller implementation – API
  - Separation of model data and parameters from the implementation
- Queue length estimation experiments
  - Kalman DD1 gives acceptable results
  - DD2 not worth the effort