WIDE – End User Panel Meeting

TOOLBOX for Matlab©

SEVENTH FRAMEWORK PROGRAMME

European Commission
Information Society and Media

WIDE

WIDE toolbox

- **Generality:**
  - **Networked Control System** (developed by TU/e);
  - **Large Scale Model Management** (developed by Honeywell);
  - **Decentralized and Hierarchical MPC** (developed by Unisi/Unitn)
  - Available for public download on **September 1st 2011**

    http://ist-wide.dii.unisi.it/
- **Documentation**
  - Automatically generated with Publish Matlab function, thus included in the download;
  - Living WEB wiki as part of Hycon2:

    http://cse.lab.imtlucca.it/HYCON2/index.php/Main_Page
- **Requirements**
  - Hybrid toolbox, Linsyskit, Mpt Toolbox, SeDuMi, TrueTime 1.5, Cplex

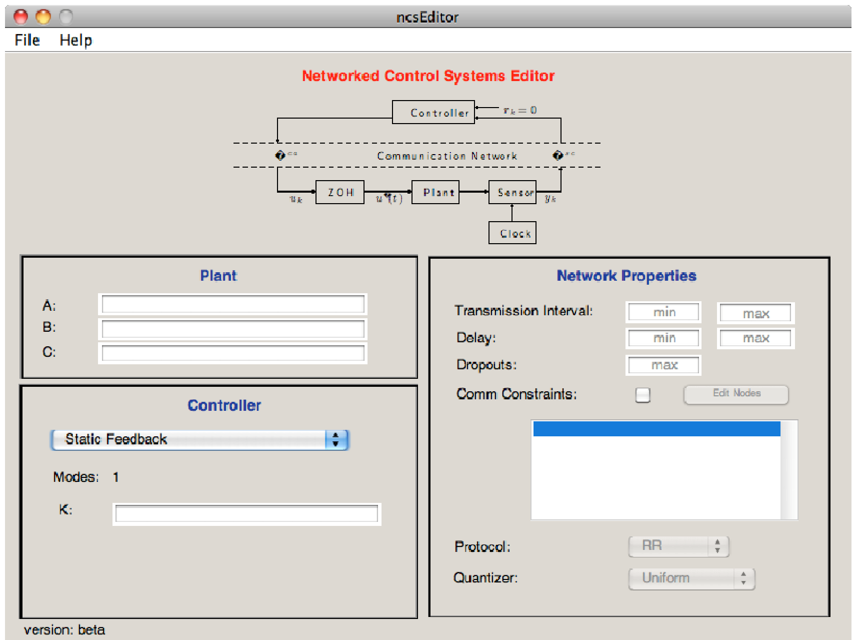
WIDE toolbox: Networked Control System

- **Purpose:**
  - **model**, **analyze** and **synthesize** control of a linear time invariant plant **over a network.**

- **Modeling approaches:**
  - **Discretized** NCS model
  - **Hybrid** NCS model.

- **Modeled effects**
  - varying transmission intervals
  - varying delays
  - communication constraints

NCS Editor



WIDE toolbox: Networked Control System

- **DNCS Functionalities:**
  - *isNcsStable(ovraprx,drpmdl,lyap,gamma):* verifies stability with
    - ovraprx: Jordan Normal Form or Caley-Hamilton overapproximations
    - *drpmdl:* prolonged transmission-interval' or 'explicit 'dropout model
    - *lyap: quadratic or parameter dependent* Lyapunov function
    - *gamma:* measure of the Lyapunov function decay
  - *stabilizeNcs(ovraprx,drpmdl,lyap,gamma):* computes a stabilizing gain accounting for the specified network;

- ***HNCS Functionalities:***
  - *findNcsStablilityTradeoff(Sy,Su):* finds the stability tradeoff between the maximally allowable transmission interval (MATI) and maximally allowable delay (MAD) with
    - *Sy: continuous faultless or sampled networked sensor-to-controller*
    - *Su: continuous faultless or sampled networked controller-to-actuator*

WIDE toolbox: Networked Control System

• Discretized example:

- Upper-bound on the convergence rate of a given NCS modeled as discrete-time linear parameter varying (DLPV) system.
- The system LTI model is

$$\dot{x} = \left(\underbrace{\begin{bmatrix} 0.6 & -4.2 & 0.1 & 2.1 \\ 0.1 & -2.1 & 0.01 & 0 \\ 0 & 0 & -3.2 & 0.2 \\ 0 & -0.03 & 5.3 & -0.2 \end{bmatrix}}_{\mathbf{A}} + \underbrace{\begin{bmatrix} 0.7 & 1.9 & -0.02 \\ 0 & 1 & -0.01 \\ 0 & 0 & 0.8 \\ 0 & 0 & -0.4 \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} 1.94 & -1.40 & 0 & 0 \\ -0.56 & -0.86 & 0 & 0 \\ 0 & 0 & -0.02 & -0.01 \end{bmatrix}}_{\mathbf{K}} \right) x$$

- Sampling time in [0.9;1.1], delay in [0;0.001] and no dropouts.
- Verify stability for values of γ in

$$||\bar{x}_k|| \leq c||\bar{x}_0||(1-\gamma)^k$$

- Method *isNcsStable('JNF','explicit','pardep',gamma):*

lower bound on gamma is 0.2
Stability: Guaranteed

---

lower bound on gamma is 0.3
Stability: Not Guaranteed

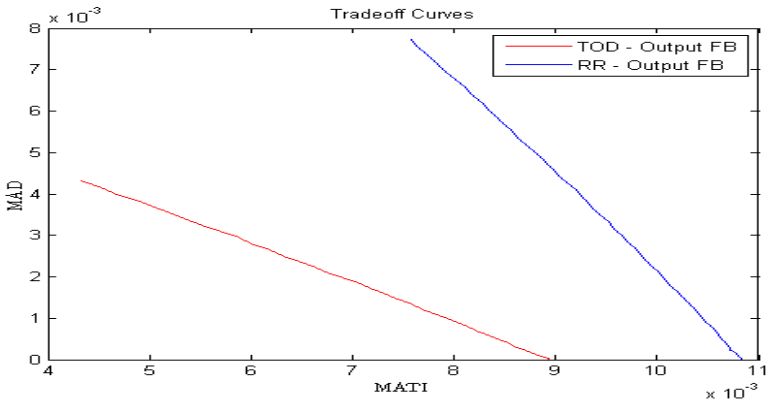
• Hybrid example:

- Compare the robustness of Try-Once-Discard (TOD) network protocol against Round Robin (RR) network protocol.

$$\begin{cases} \dot{x}_p = \begin{bmatrix} 1.38 & -0.2077 & 6.715 & -5.676 \\ -0.5814 & -4.29 & 0 & 0.675 \\ 1.067 & 4.273 & -6.654 & 5.893 \\ 0.048 & 4.273 & 1.343 & -2.104 \end{bmatrix} x_p + \begin{bmatrix} 0 & 0 \\ 5.679 & 0 \\ 1.136 & -3.146 \\ 1.136 & 0 \end{bmatrix} u \\ y = \begin{bmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \end{bmatrix} x_p \\ \dot{x}_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} x_c + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} y \\ u = \begin{bmatrix} -2 & 0 \\ 0 & 8 \end{bmatrix} x_c + \begin{bmatrix} 0 & -2 \\ 5 & 0 \end{bmatrix} y \end{cases}$$

Sensor-to-controller wired
Controller-to actuator via network

- Tradeoff plot between the maximally allowable transmission interval (MATI) and maximally allowable delay (MAD).
- *hncs_RR.findNcsStablilityTradeoff([0 0],[1 1]);*
- *hncs_TOD.findNcsStablilityTradeoff([0 0],[1 1]);*



WIDE toolbox: Large Scale Model Management

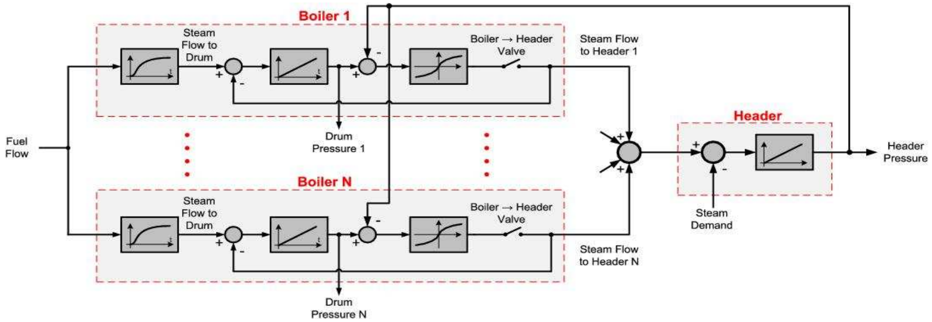
- **Large Scale:**
  - Represent and manage **Large Scale** models. LS model is described as a set of submodels together with description of mutual and external inputs/outputs interconnections.
  - Model creation from a set of submodels, a set of summators and string cell arrays defining external inputs and outputs.
  - Add/remove submodels and external inputs/outputs.
  - Handle **structured model order reduction**, **decomposition** of subsystems into **groups** for **distributed control/estimation** and **merging** of information from multiple models into single one.
  - Model plot and analysis via many standard functions.
- **Water Network Model**
  - Child of LS, extended to model water distribution networks;
  - Import scheme from text file, customized plot, customized Epsilon decomposition procedure.

WIDE toolbox: Large Scale Model Management

- **LS Functionalities:**
  - **Model Editing**
    - *add_mod; add_sum; add_ext_inp; add_ext_out; rem_mod; rem_sum; rem_ext_inp; rem_ext_out; select, group, squeeze* (remove unused)*;*
    - *set_sig_type* (Manipulated Variable, Measured Disturbances , Unmeasured Disturbances, Measured Outputs, Unmeasured Outputs, Internal Signal)*; set_sig_lim* (signal limits)*; set_sig_data;*
  - **Model Reduction**
    - struct_red, merge (ARX models of different structure), freq_uncert (frequency uncertainty for ARX model); eps (epsilon decomposition); bbd (Border Block Diagonal decomposition);
  - **Model Representation**
    - display, n (total order: sum of subsystems order), orders, plot (interactions);
  - **Overloaded functions**
    - dcgain, pole, zero, impulse, step, bode, nyquist, pzmap, iopzmap, ss;
- **WN Functionalities:**
  - *import_scheme* (imports water network model from file);
  - *plot, eps;*

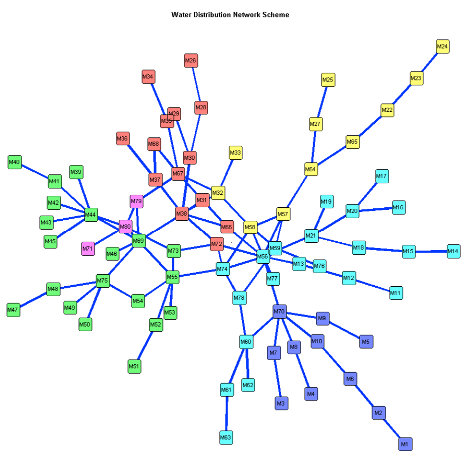
WIDE toolbox: Large Scale Model Management

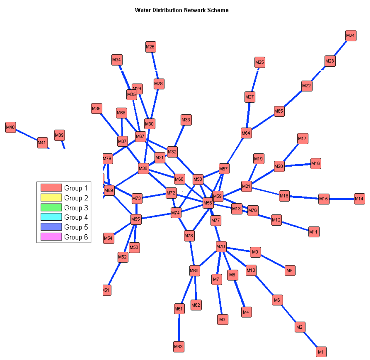
- **Large Scale example:**
  - Plant: N boiler to single heater
    - % M: boilers + heater state space models

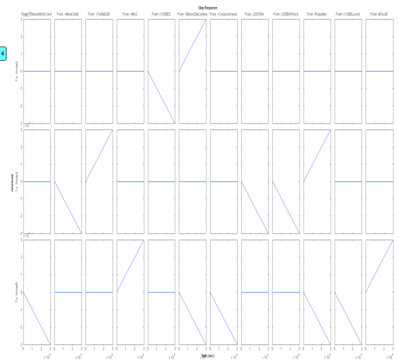


    - % sums steam flows from boilers to header
    - sum1 = sumblk('SF','SF1','SF2','SF3','SF4','SF5')
    - sum1.Name = 'SFsum';
  - *LSmodel(M,sum1,{'FF','SD'},{'ph','SF'}* (standard construction with cell array of ss model);
  - *connect(M{1}, M{2}, M{3}, M{4}, M{5}, M{6}, sum1, {'FF','SD'}, {'ph','SF'});*(connect function with individual models);
  - *LSmodel(M{1}, M{2}, M{3}, M{4}, M{5}, M{6}, Q, inputs, outputs)* (numeric indexing of inputs and outputs)*;*

- **Water Network example:**
  - Import file structure:
    - Tank##,<tank name> / Node##
    - d,<demand name>
    - s,<source name>
    - +,<outlet pump/valve name>,<destination tank name>
    - -,<inlet pump/valve name>,<source tank name>
  - mod = WNmodel('BCN_network')
  - plot(mod);
  - *mod.eps(6);*







  - *step(mod6_gr6);*

WIDE toolbox: Decentralized/Hierarchical MPC

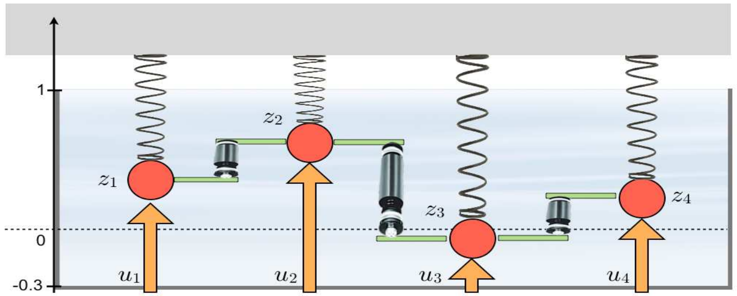
- **Purpose:**
  - Generate TrueTime code for quick NCS simulations.
  - Synthesize Robust/Stochastic decentralized linear regulator by solving LMI.
  - Use single command to compute control action of a set of decentralized MPC controllers and test 'a-posteriori' the closed loop stability with bounded measurement losses.
  - Explicit MPC controller with sensors measurements subject to an energy-aware policy intended to lower the number of transmissions and, ultimately, save sensor nodes battery.
  - In a upper layer decentralized hierarchical control structure, with linear regulators at lower level, compute reference restrictions so as to enforce plant constraints.
  - Connect to real devices to close the control loop with Matlab: currently supported devices are Telos Motes and Esenza Nodes.

WIDE toolbox: Decentralized/Hierarchical MPC

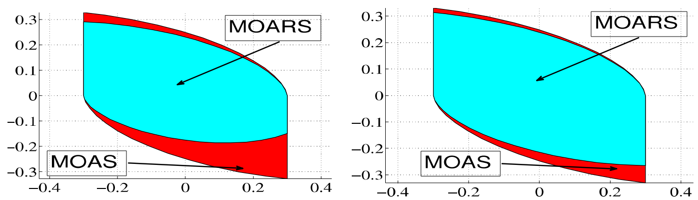
- **Functionalities:**
  - TrueTime code generation:
    - ACG (number of sensors/actuators) ; GenerateCode; RemoveOldCode;
  - Linear Regulator:
    - decLMI, solve_centralized_lm, solve_dec_ideal_lmi(), solve_dec_lossy_lmi(), solve_dec_stoch_lmi;
  - Decentralized MPC:
    - Dlincon, Deval, stability_test;
  - Energy Aware MPC:
    - eampc, init_sim, send_predictions, get_measurements, get_input, build_MPC;
  - Hierarchical MPC:
    - HiMPC, computeMOARS, plotMOARS, computeDeltaR, plotDeltaR;
  - Connect to device (yet to be completed):
    - Connect, send , receive;

WIDE toolbox: Decentralized/Hierarchical MPC

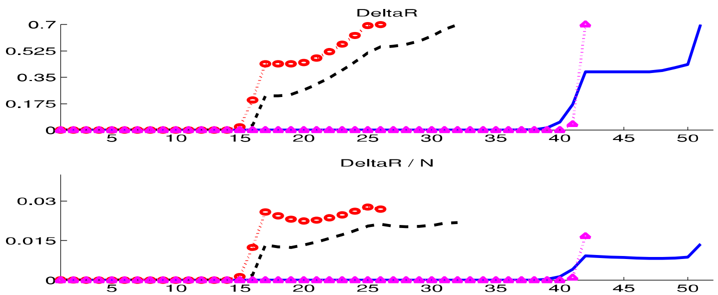
- **DHiMPC example:**
  - Plant model:



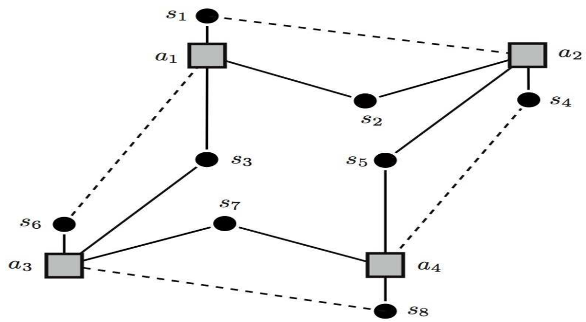
  - *HiMPC(sys, dec, Xcon, DeltaX, coupledCons);*
  - *computeMOARS(); plotMOARS();*



  - *computeDeltaR(50); plotDeltaR();*



- **decLMI example:**
  - Plant: randomly unstable with communication structure;



  - decLMI(Net, A, B, Qx, Qu, X0, xmax, umax, Mc);
  - solve_centralized_lmi()
  - solve_dec_ideal_lmi();
  - solve_dec_lossy_lmi();
  - solve_dec_lossy_lmi();
  - Results over 50 simulations:

	$\mu(J_i)$	$\sigma(J_i)$	CPU
Ideal network			(off-line time)
Centralized control	41.0	0	2.8 s
Decentralized control	45.1	0	1.2 s
Lossy network			(off-line time)
Dec. robust control	50.0	1.57	8.1 s
Dec. stochastic control	47.1	2.38	59.2 s