IEEE RTC 2016, Padova

EUROfusion

Plasma current and shape control for ITER using fast online MPC

Samo Gerkšič, Boštjan Pregelj, Matija Perne, Matic Knap
Jožef Stefan Institute, Ljubljana, Slovenia
Gianmaria De Tommasi, Marco Ariola, Alfredo Pironti
Consorzio CREATE, Napoli, Italy

CREATE PPCC team

COSYLAB CONTROL SYSTEM LABORATORY

SLOVENIAN RESEARCH AGENCY

Eurofusion AWP15-ENR-01/JSI-02 "FMPCFMPC"

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

Overview

- **Model Predictive Control (MPC) introduction**
- **ITER plasma magnetic control** cascade scheme:
  Inner loop : Vertical Stabilisation (VS)
  Outer loop: Current and Shape Control
- **Model Predictive Control (MPC) for CSC**
  Prototype MPC control scheme for the flat-top phase
- **Fast quadratic programming (QP) methods for MPC**
  First-order methods

Model(-based) Predictive Control

- A control methodology in which future control actions are determined by optimisation of a performance criterion defined over a future horizon in which control signals are predicted using a dynamic process model

- System $\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k), \;\; \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k)$
- Cost function $J = \sum_{j=0}^{N-1} (\mathbf{x}_{k+j|k}^T \mathbf{Q}_x \mathbf{x}_{k+j|k} + \mathbf{u}_{k+j|k}^T \mathbf{R}_u \mathbf{u}_{k+j|k}) + \mathbf{x}_{k+N|k}^T \mathbf{Q}_{xN} \mathbf{x}_{k+N|k}$
- subject to constraints $\mathbf{u}_{\min} \le \mathbf{u} \le \mathbf{u}_{\max}, \quad \mathbf{x}_{\min} \le \mathbf{x} \le \mathbf{x}_{\max}$
- Receding-horizon implementation

Model(-based) Predictive Control

- Well established in the process industry (oil etc)
  Qin Badgwell 2003 A survey of industrial MPC technology, Con.Eng.Pra.
  for instance: **aspen**tech DMCplus / aspenONE APC DMC3
  - Systematic approach to control of large-scale multivariable systems
  - Efficient handling of constraints on process variables
  - Enables plant optimisation
- ... expected to be beneficial for plasma current & shape control
- Problem:
  On-line optimization with sub-second sampling time

ITER Tokamak

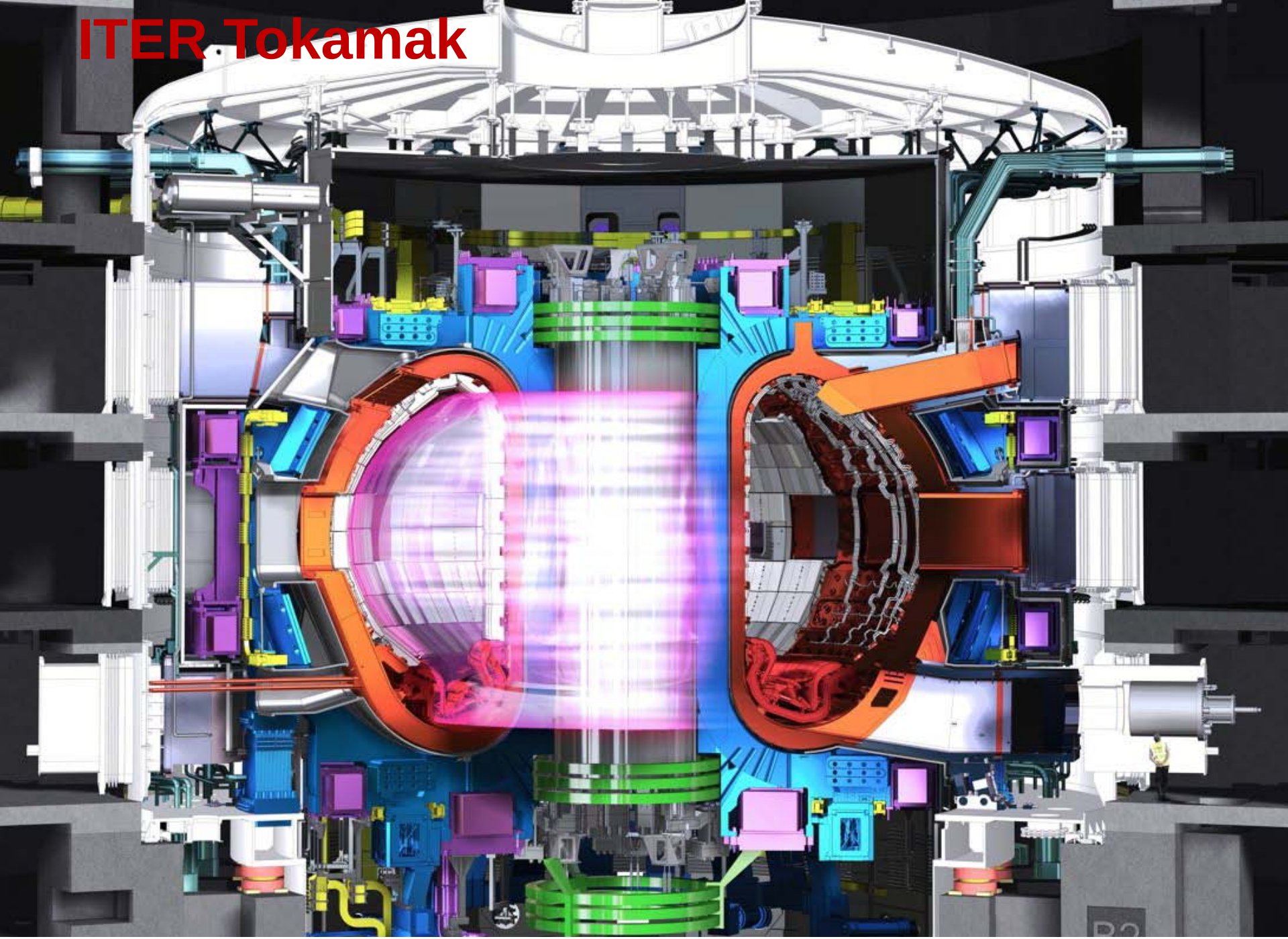


Image: www.iter.org

Cadarache FR, March 2016

Image: www.iter.org

Plasma simulation models (CREATE-L/-NL)

High-ordel local linear models from first principles

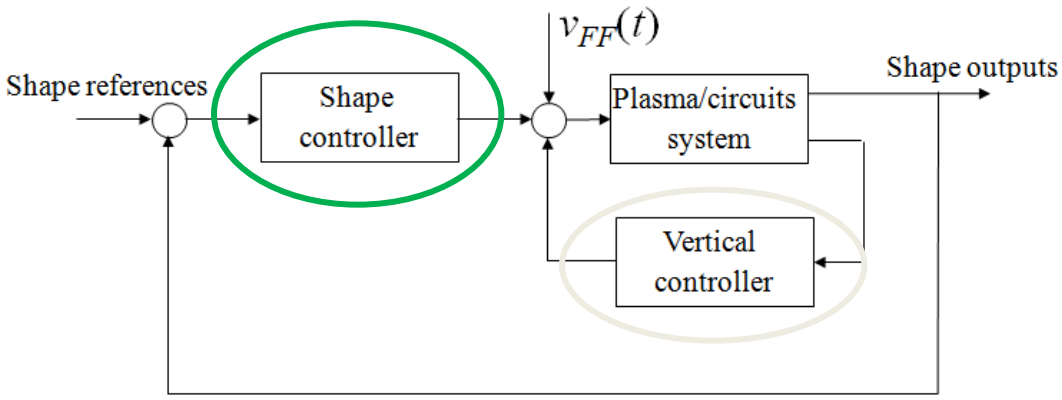
5 models in different equilibrium points of ITER scenarios,
defined by the nominal I_p, poloidal beta β_p and internal inductance l_i

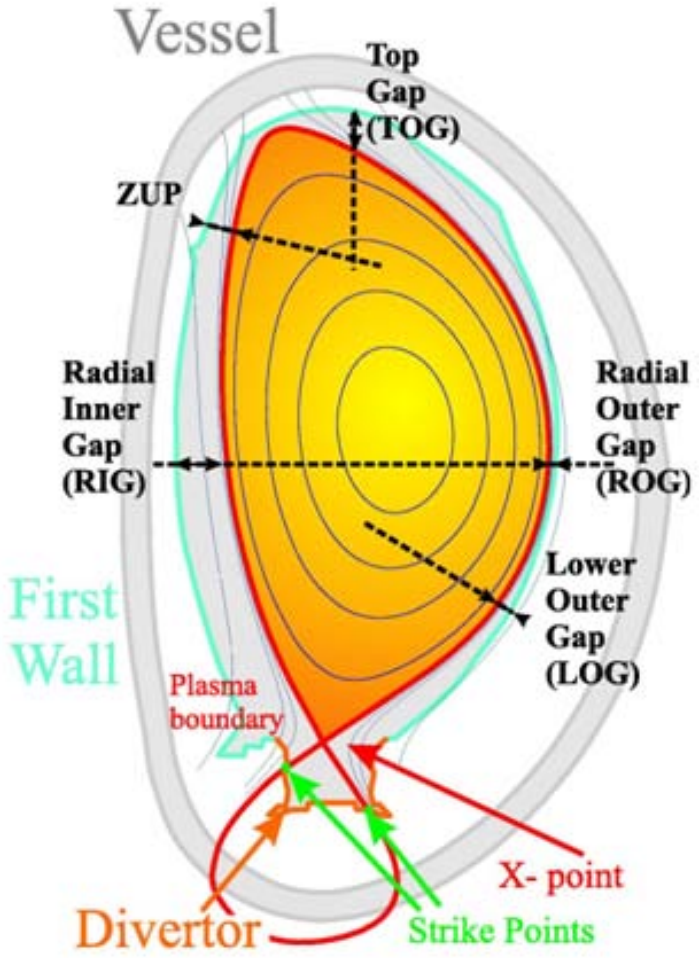
Simulation of disturbances:

- Vertical displacement event (VDE): via the initial state of the plasma model
- H-L transition, L-H transition, Minor disruption, Uncontrolled ELM: by profiles of β_p and l_i

Model code	I_p (MA)	β_p	l_i	Number of states
LMNE	15.0	0.10	1.21	120
LM52	15.0	0.10	0.80	123
LM53	15.0	0.10	1.00	123
LM59	15.0	0.60	0.60	123
LM60	15.0	0.60	0.80	123

Plasma magnetic control cascade





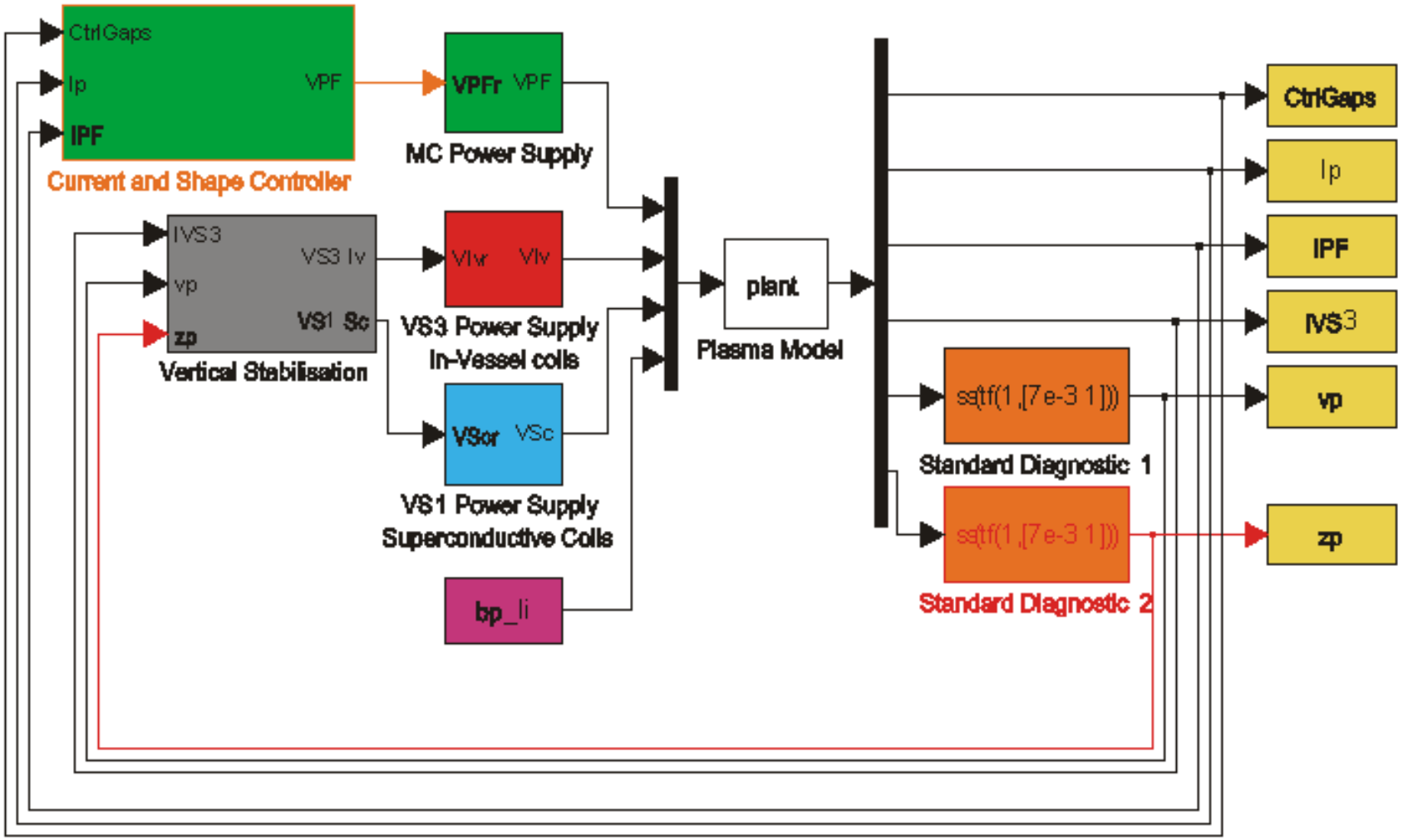
- Inner loop VS:
  fast stabilization of vertical position
- Outer loop CSC:
  plasma current and shape control
- Regulation of specific disturbances:
  VDE, ELM, H-L & L-H transitions
- Robustness to varying dynamics

Prototype ITER plasma magnetic control with MPC CSC

- Benchmark scheme for testing fast QP algorithms
  (SOFT 2014 - not the most recent one)
- Vertical Stabilization: **ctLQGz**
  continuous-time LQG (linear-quadratic gaussian) optimal controller
- Current and Shape Control:
  Model Predictive Control (MPC)

Prototype ITER plasma magnetic control with MPC CSC

Simplified Matlab/Simulink scheme



Outer loop:
CSC Plasma Current and Shape Control

Actuators:

- 11 main power supply voltages $\mathbf{V}_{\mathrm{PF}}$

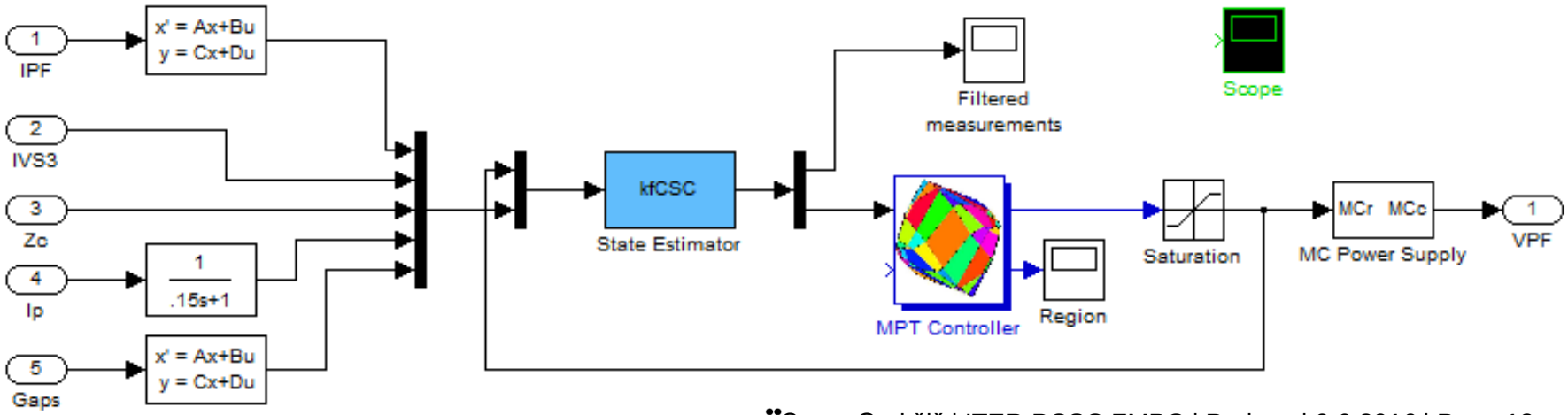
Controlled outputs:

- Plasma current I_{p}
- 6 controlled geometrical descriptors $\mathbf{g}$
  (2 strike points and 4 gaps)
- 11 superconductive coil currents $\mathbf{I}_{\mathrm{PF}}$

CSC: Model Predictive Control

MPC setup:

- Choose nominal model, define signal constraints
- Prepare nominal model for MPC
  (append actuator and diagnostics models; closed loop with VS; model reduction; discretization; model augmentation for offset-free regulation)
- Tuning: MPC costs and parameters + KF covariance matrices
- MPC is used in an LQG-like scheme where a Kalman filter estimates the states of the disturbance-augmented model.



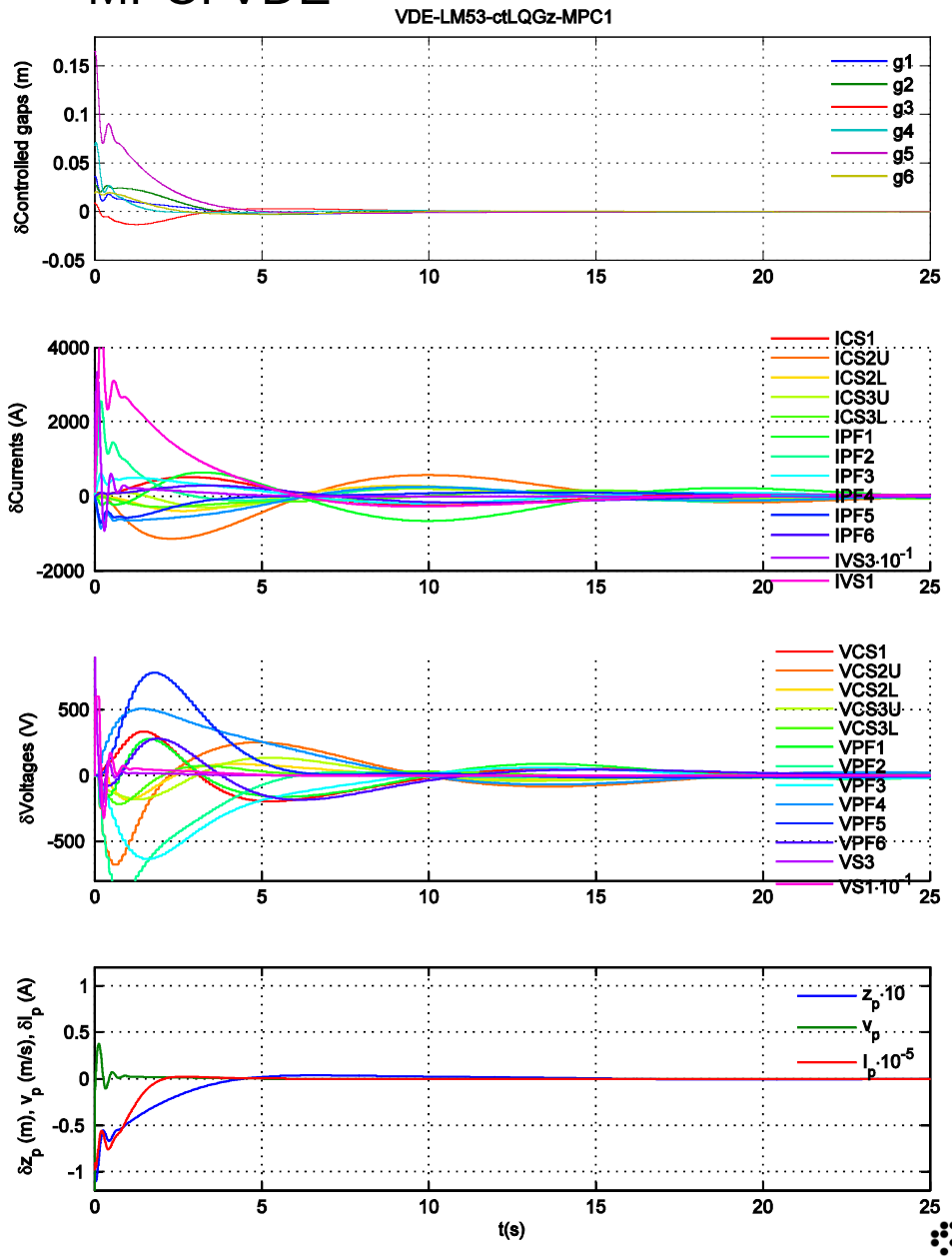
MPC Implementation

A Quadratic Programming solver is used in each step...
How to form the QP?

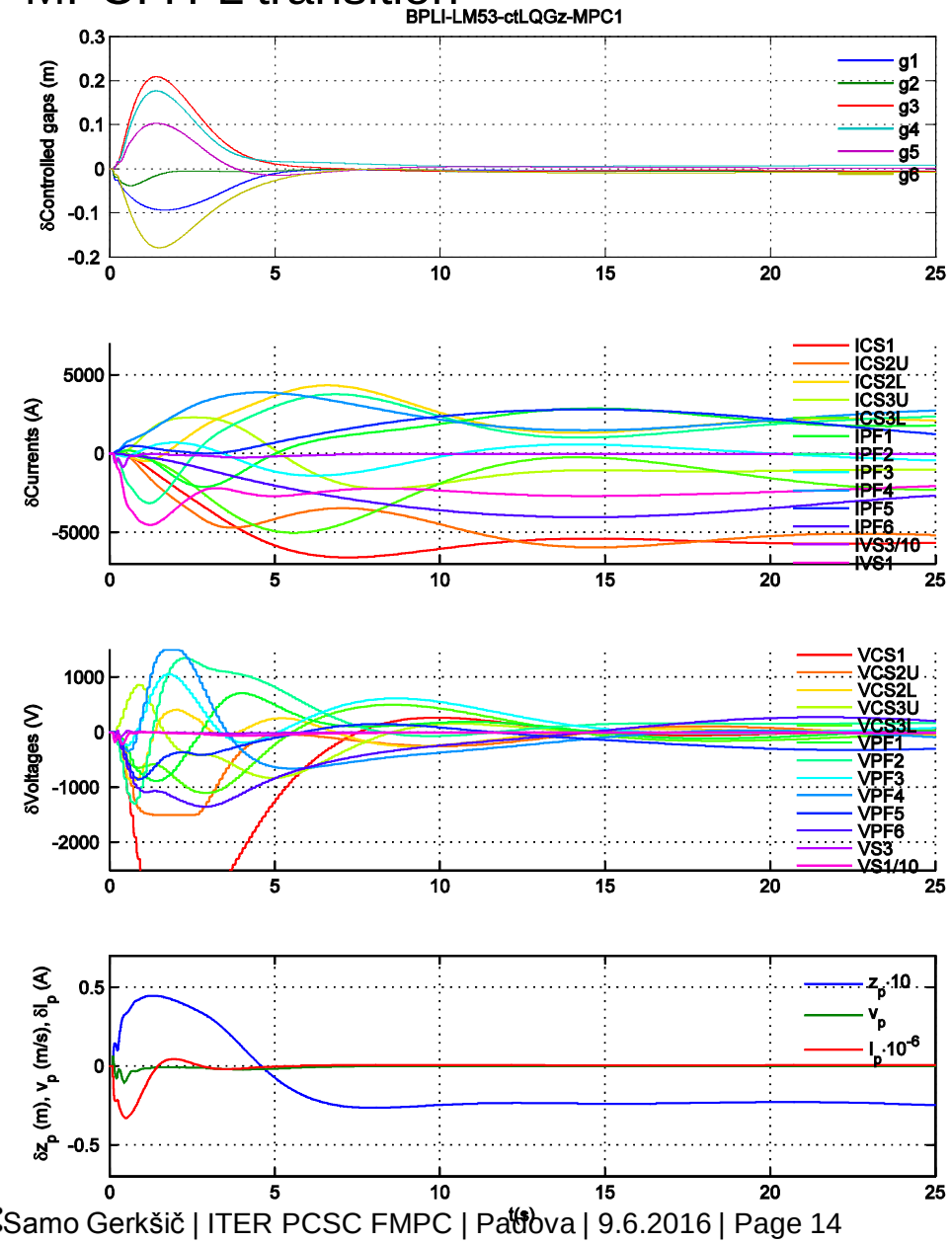
- Write down the sequence of predictions over the horizon, form the cost, build QP matrices in standard QP form
- May be done "manually"
- Matlab MPC Toolbox: configure via menus
  simple and flexible, if everything you need is supported
- Equation parser to build the QP from a problem description
  modified Multi-Parametric Toolbox + YALMIP

Simulation: model LM53, no y constraints

MPC: VDE

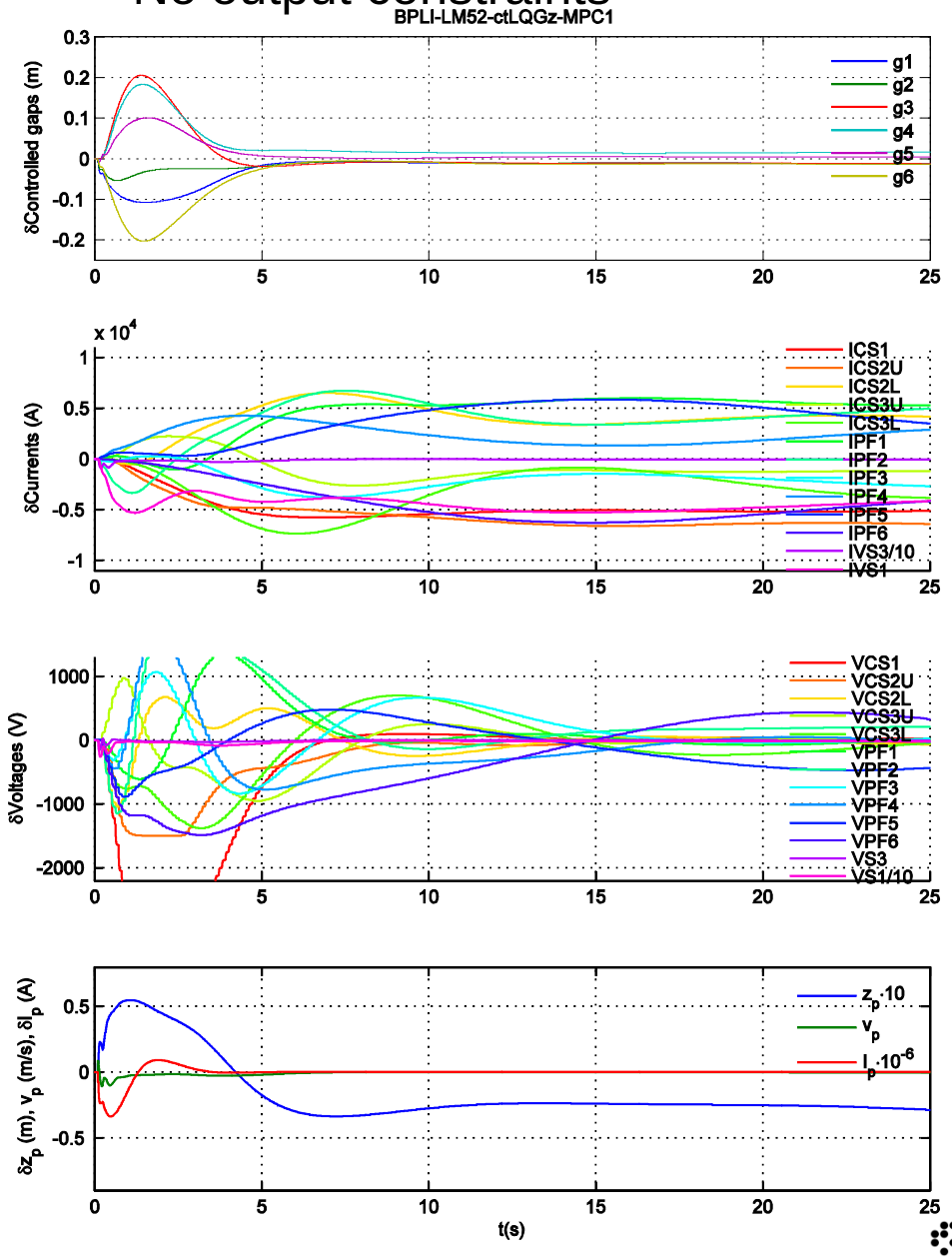


MPC: H-L transition

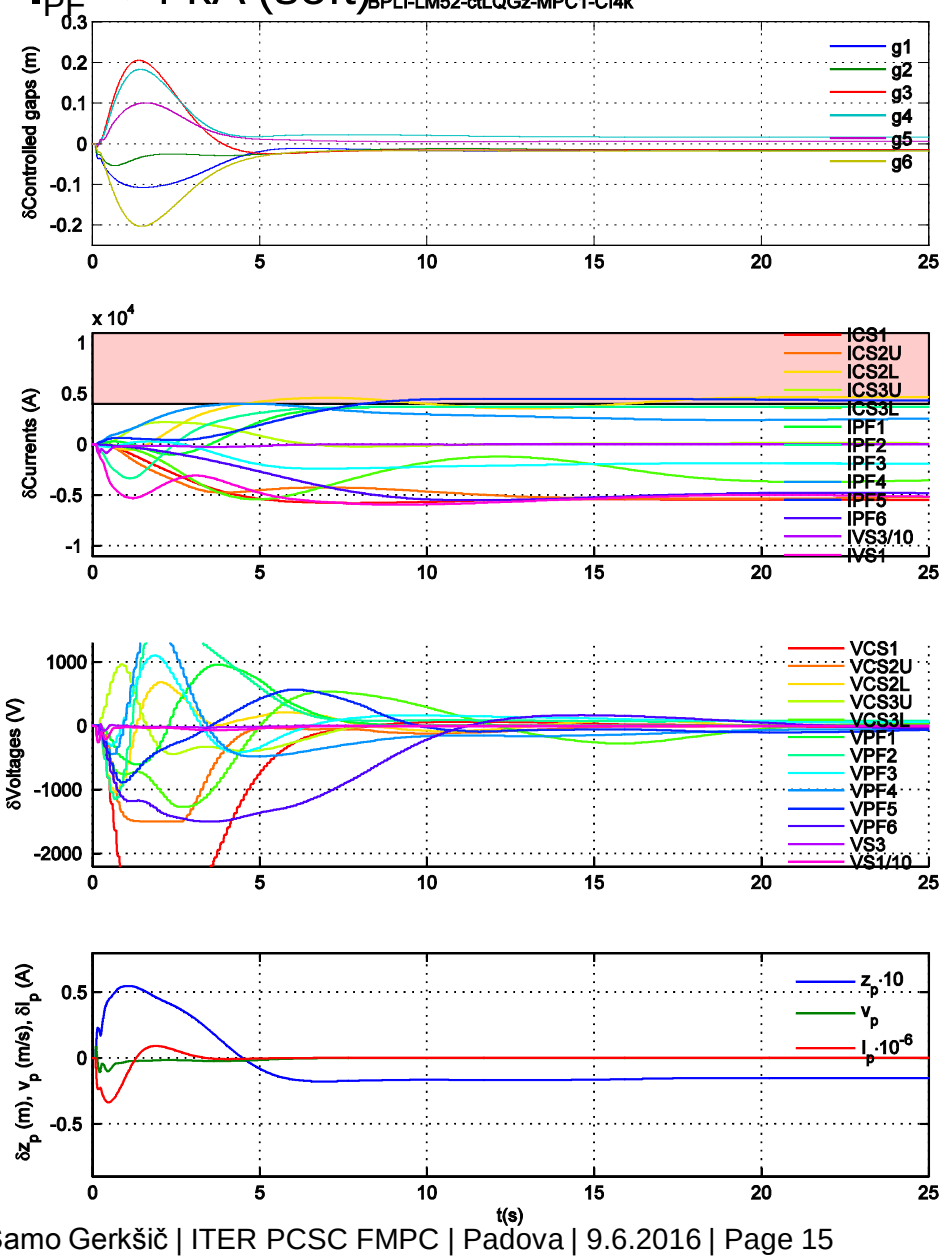


Simulation: model LM52, H-L transition

No output constraints



I_{PF} < 4 kA (soft)



MPC Implementation

Peak computation time using off-the-shelf QP solvers (with some complexity reduction):

- Mathworks Optimization Toolbox: 500 ms
- ILOG (IBM) CPLEX: 50 ms
- for sample time 100 ms (required for good disturbance rejection), under 10 ms is desired

... How to speed up QP computation?

Quadratic Programming

MPC based on linear models translates to Quadratic Programming:
quadratic cost function with linear inequality constraints

QP: min 0.5 z'Qz + q'z

s.t. inequality constraints Gz ≤ g

equality constraints Fz = f

MPC cost

actuator & state constraints

process dynamics

(eq.c. sometimes eliminated... structured vs condensed QP)

Typically: **on-line optimization**, iterative (time-consuming) methods
Active set / Interior point / First order methods

(Exception: multi-parametric QP (explicit MPC) – small problems only!)

"Traditional" QP solvers: CPLEX, NAG, XPRESS...

- General-purpose, designed for large-scale QP problems
- Mostly commercial, closed-source
- Problem analysis & code optimization may be included
  but is "bundled" in each call, possibly with license check

Fast Online Quadratic Programming

MPC: repetitive solution of similar S-to-M-sized QP problems

Algorithms best suited to small-medium scale problems, specific QP problem structures appearing in MPC

QP analysis & code-optimisation phase **decoupled** from the online solver

Code generator: produce fast and compact online solver code, optimized for the specific QP type and target hardware

Fast online QP solvers
(recent developments in all solver approaches)

- Active set methods
  Ferreau & al. 2014: **qpOASES**, a parametric active-set algorithm for quadratic programming, Math. Prog. Comp.
- Interior point methods
  Wang Boyd 2010: Fast MPC Using Online Optimization, IEEE TCST 18
  Mattingley Boyd 2012: **CVXGEN**, a code generator for embedded convex optimization, Optim Eng 13
  Domahidi & al. 2012: Efficient interior-point methods for multistage problems arising in RHC, CDC (**FORCES**)
- First order methods

First order fast online QP methods

- Approach the solution of KKT optimality conditions by succesive **gradient descent** steps
- Nesterov 1983 Fast Gradient Method... not much attention:
  Slow convergence... many iterations needed for high precision
- Interesting for MPC:
  Simple iterations, fits within restricted hardware (ARM, FPGA...)
  Efficient in quickly achieving medium precision
  Convergence certification (in certain cases)
  Soft y constraints *without* expanding the QP with slack variables
- The algorithm involves a projection, generally a QP (!)
  ...Simple with simple bounds on control inputs
  ...Not as simple with state constraints: via *dual* FGM

First order fast online QP methods

Several recent related methods

- Jerez & al 2014: Embedded Online Optimization for MPC at Megahertz Rates, IEEE TAC 59 (EmboTech – FORCES PRO – on FPGA)
- Boyd & al 2010: Distributed Optimization and Statistical Learning via the **Alternating Direction Method of Multipliers**, FTML 3(1) 1-122
- Peyrl & al 2014: Parallel implementations of the fast gradient method for high-speed MPC, CEP 33 (ABB)
- Patrinos Bemporad 2014: An **Accelerated Dual Gradient-Projection** Algorithm for Embedded Linear MPC, IEEE TAC 59
- Giselsson 2015: Improving Fast Dual Ascent for MPC - Part II The Embedded Case, arxiv.org 1312.3013 v2 (Automatica) (**QPgen**)
- Stathopoulos & al 2014: Splitting methods in control, ECC 2014
- Kufoalor & al 2014, Embedded MPC on a PLC Using a Primal-Dual First-Order Method for a Subsea Separation Process, MED 2014
- Hartley & al 2014: Predictive Control Using an FPGA With Application to Aircraft Control, IEEET CST 22(3)

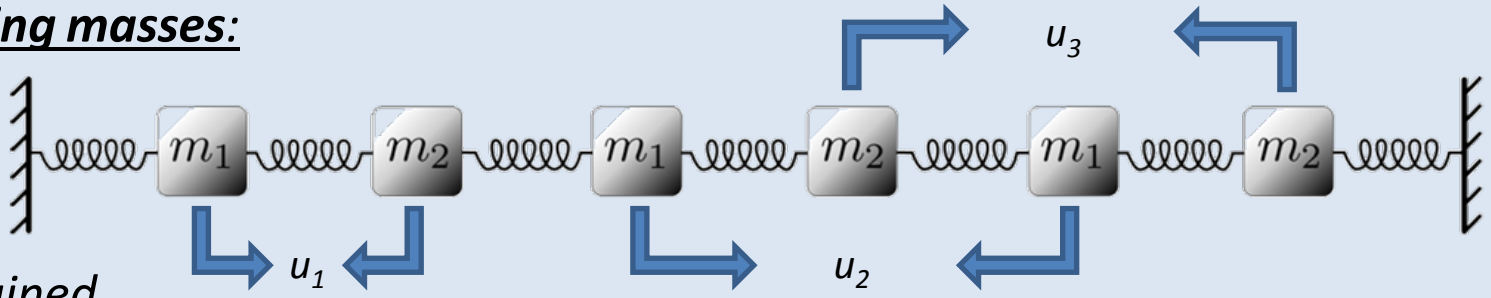
Most studies on academic examples of small dimensions or not considering soft state constraints (unlike industrial MPC problems)

Fast QP solvers – benchmark problems

6 oscillating masses:

Nx = 12
Nu = 3
Ny = 6
N = 10

unconstrained



Benchmark problem \ solver		QPgen	HP-MPC	FiOrdOs
Oscillating Masses	avg/max t_sol [ms]	0.08 / 4.0 ms		4.5 / 8.1 ms
	avg/max iterations	25 / 40		168 / 212

Linearized F16 aircraft model AFTI-16:

Nx = 4
Nu = 2
Ny = 2
N = 10

constrained, unstable and poorly conditioned

Benchmark problem \ solver		QPgen	HP-MPC	FiOrdOs
AFTI-16	avg/max t_sol [ms]	1.5 / 7.3 ms	6.7 / 21 ms	7.3 / 24.5 ms
	avg/max iterations	115 / 300	2660 / 8200	11k / 36k

Comparison on 3.4 GHz Core i7 Inter processor, 8GB RAM

Plasma CSC – MPC complexity reduction

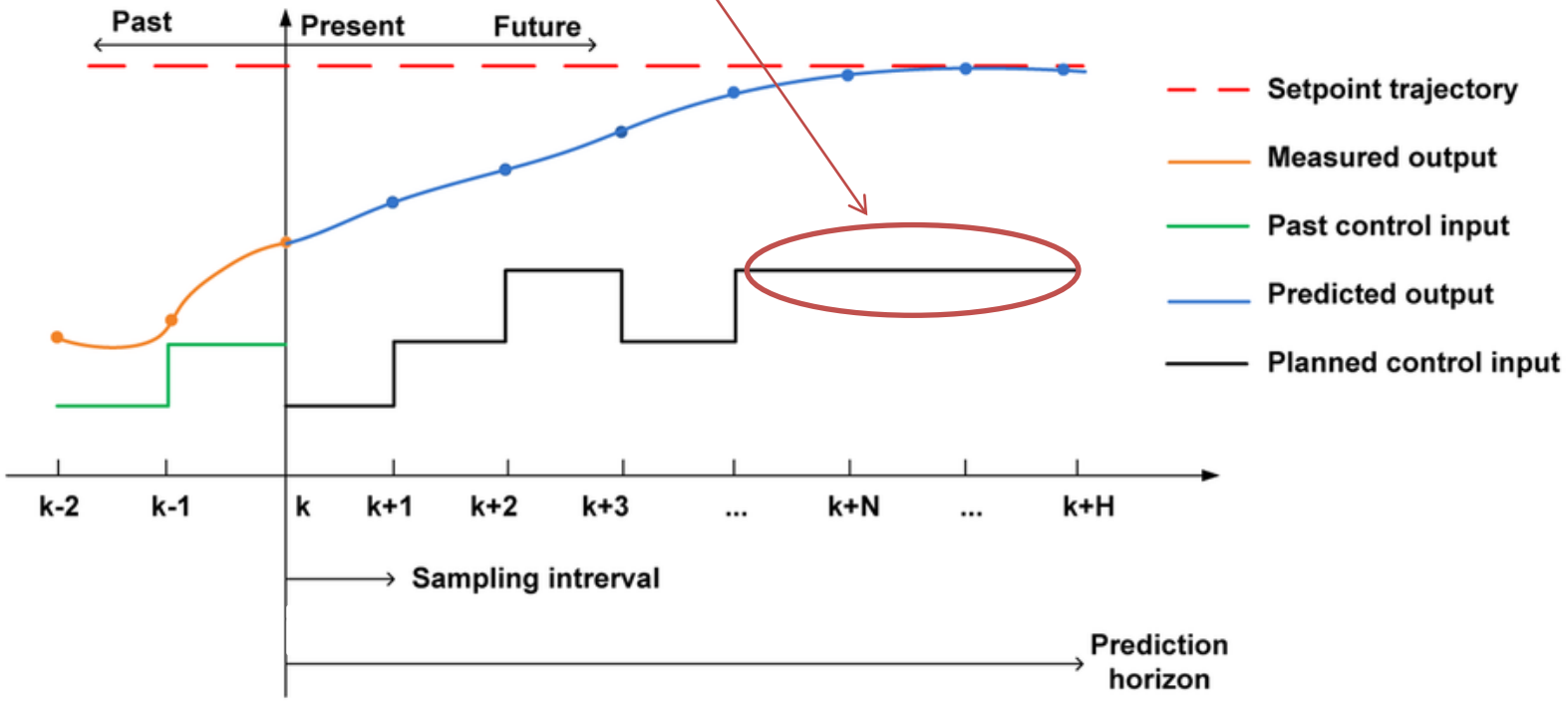
Nx = 62 (after model reduction), Nu = 11, Ny = 20,
N = 30, soft output constraints...
much larger problem, computation is too slow!

...Complexity reduction techniques

- Control move blocking
- Sparse placement of output constraints
- Elimination of redundant constraints
- Efficient handling of soft constraints (no slack variables)
- QP null-space reduction (structured to condensed QP form)

Plasma CSC – MPC complexity reduction

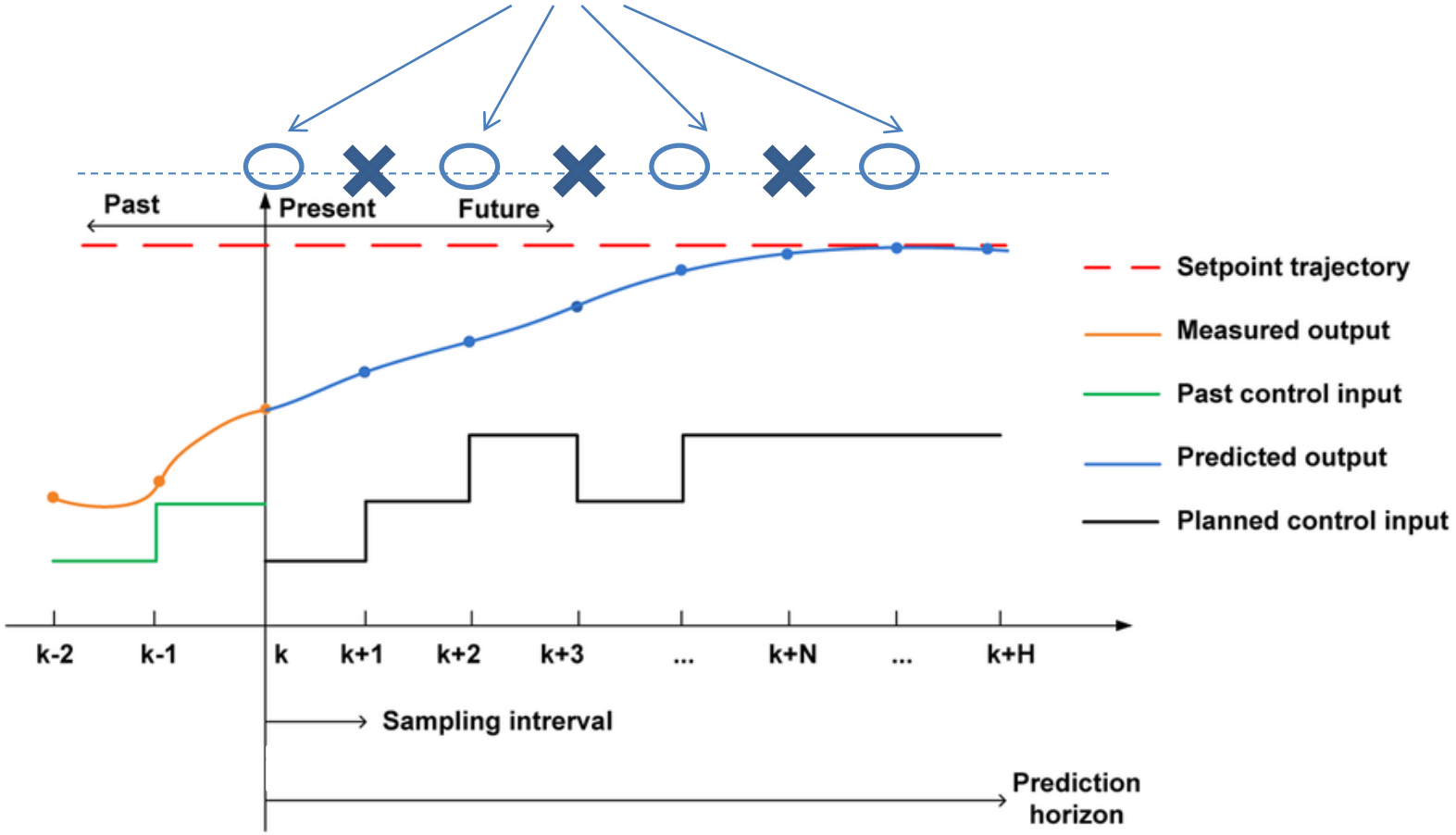
- Control move blocking
  - *Clustering of consequent control moves*
    - *Reduces the number of free optimization variables → H size*
    - *Reduces the number of inequality constraint equations → A_{ineq} rows*



Plasma CSC – MPC complexity reduction

- Sparse output constraints ("coincidence points")
  - *Constraints placed each k-th sample → less rows of A_{ineq}*

Fast sampling needed for disturbance rejection –
constraints need not be so dense, violations limited by process dynamics



Plasma control MPC simplification

Process: Nx = 62, Nu = 11, Ny = 20
MPC: horizon 30
- sparse constraints each 3rd sample (10 coincidence points)
- input blocking [1 2 27]… 3*11 = 33 free moves
- Only 11 output constraints effective

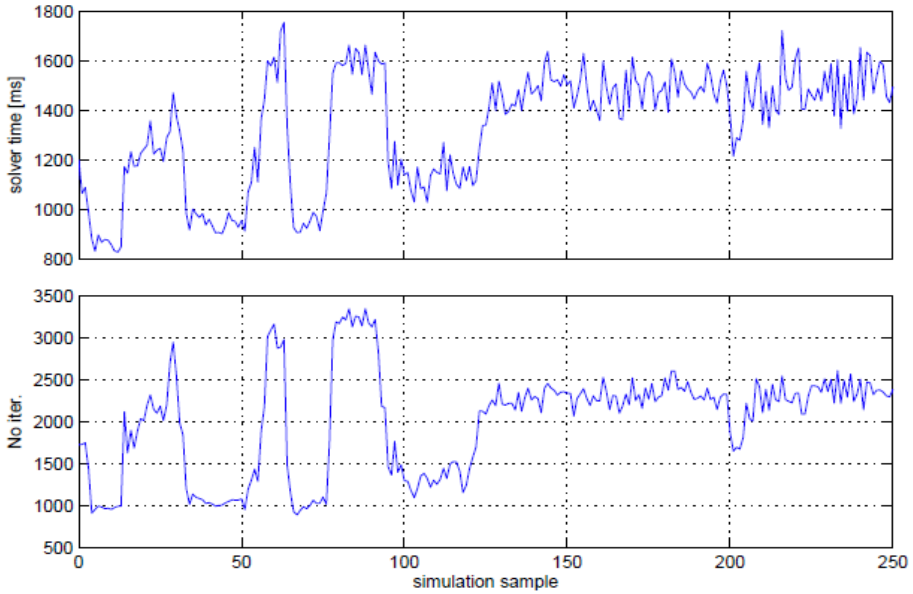
Significant QP size reduction !

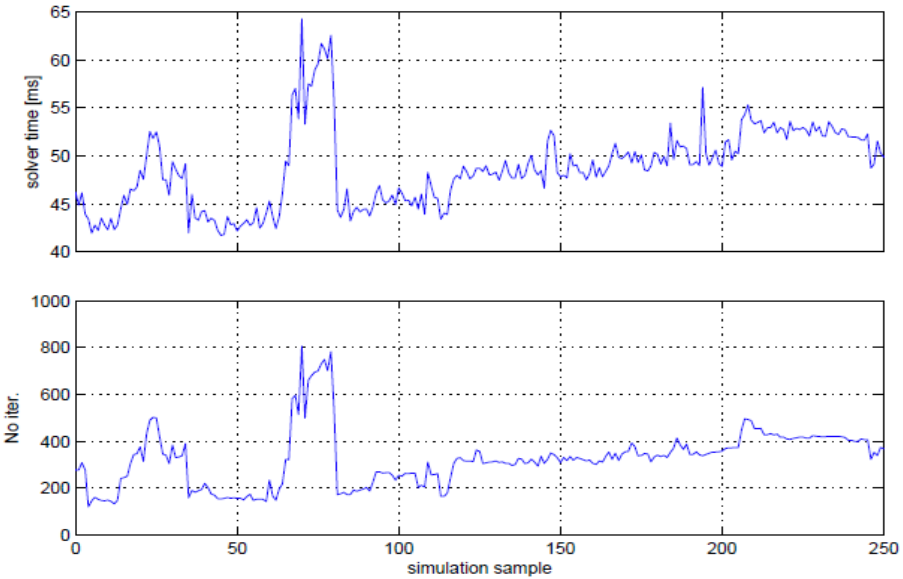
	Inputs	States	Outputs	Slacks	Ctrl. moves	Constraints				
Parameter size	**u**	**x**	**y**	**s**	**N_z**	**N_{cstr}**	**A_{ineq}** (y+u+s)*N×	**z** (x+u+s)*N	**H**	**Memory**
Original	11	62	20	600	30	30	1530×2790	2790	2790×2790	*19 MB**
Soft constraints without slacks	11	62	20	**0**	30	30	**930×2190**	**2190**	**2190×2190**	9.7 MB
Redundant constraints rem.	11	62	20	0	30	30	**660**×2190	2190	2190×2190	7.7MB
Move blocking	11	62	20	0	**3**	30	660×**219**	**219**	**219×219**	2.3MB
Sparse output constraints	11	62	20	0	3	**10**	220×219	219	219×219	1.6MB

[1] N_z – optimization-vector size affecting horizon length
[2] N_{cstr} – constraint equation number affecting horizon length
[3] For information only, not feasible for use due to slow convergence

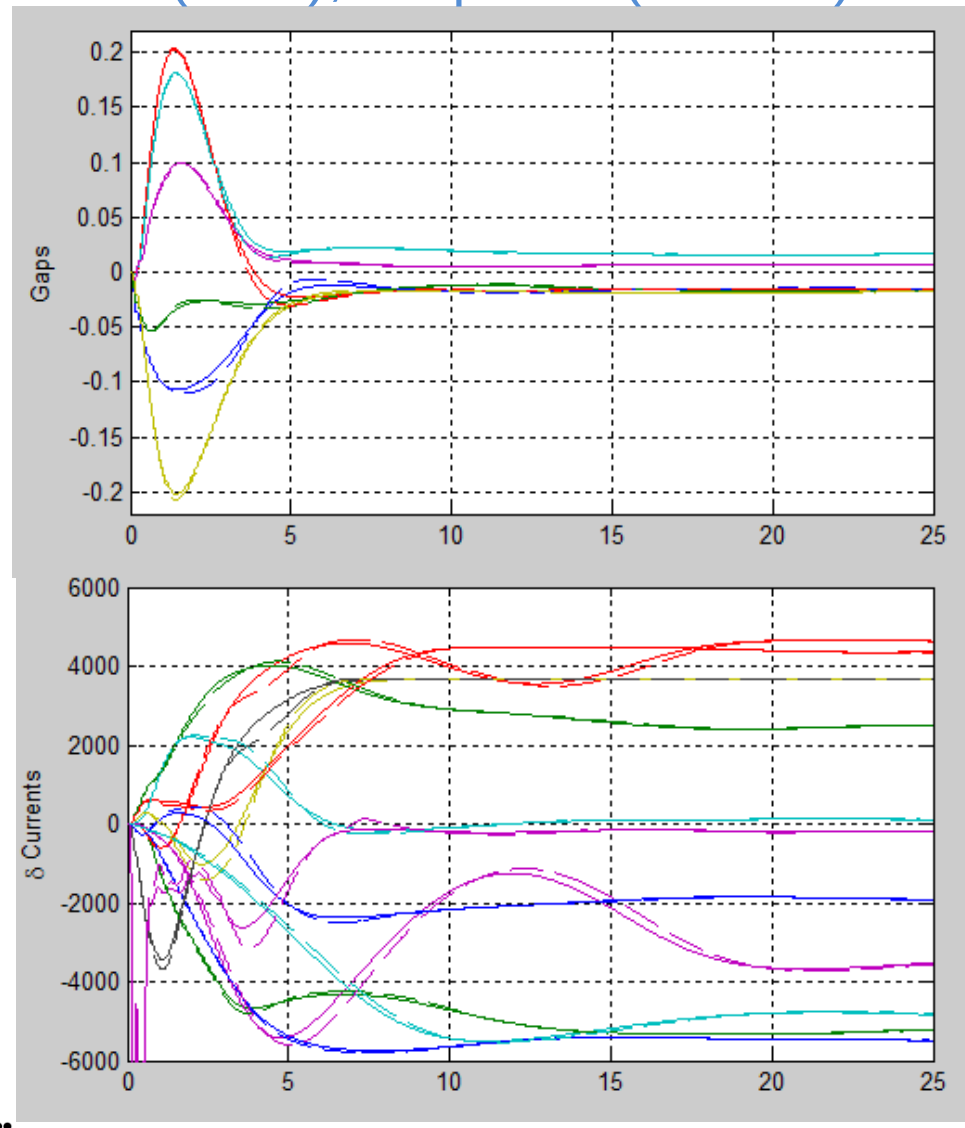
QP computational demand, CPLEX

CPLEX: full (top), simplified (bottom)





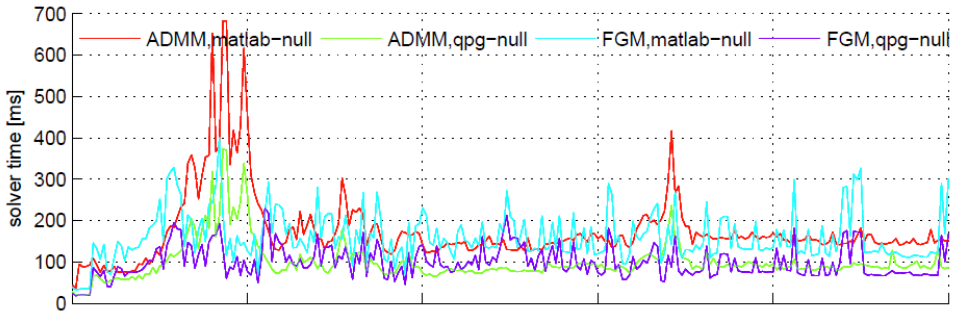
Simulated outputs:
full (solid), simplified (dashed)

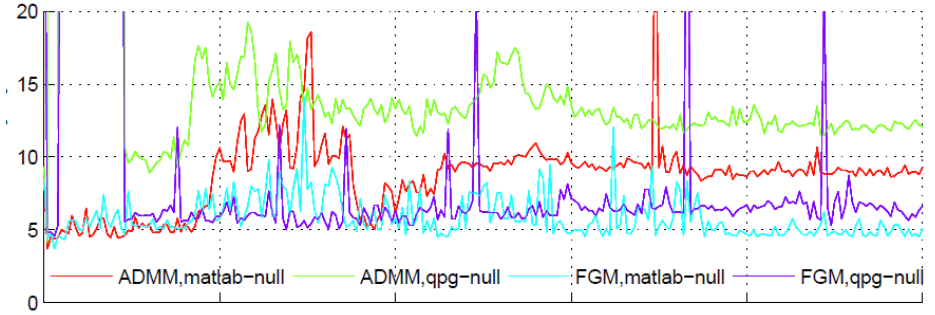


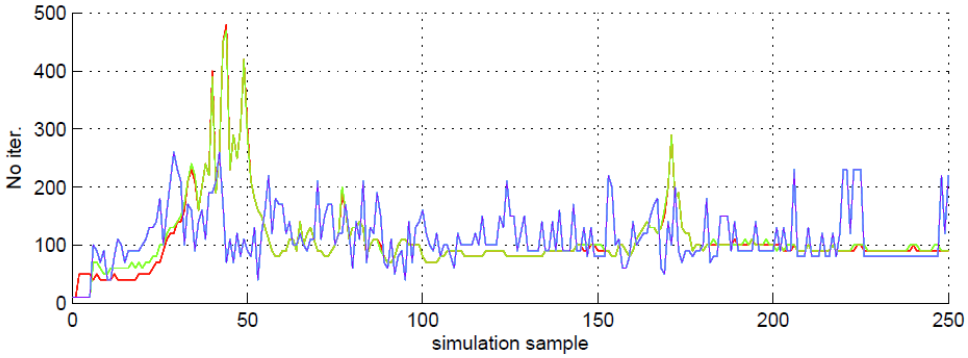
Fast QP computational demand, QPgen

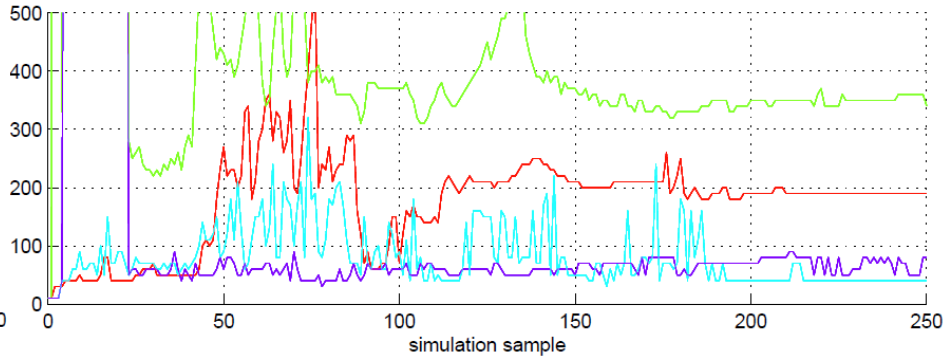
QPgen, original (left) vs simplified (right) PCSC MPC problem

- Comparison of FGM / ADMM method
- Comparison of SVD / Dynamical System (controllability matrix) nullspace









Fast QP algorithm selection - QPgen

- FGM faster than ADMM for the simplified problem
- DS-based null-space better for the original problem
- SVD null-space better for the simplified problem

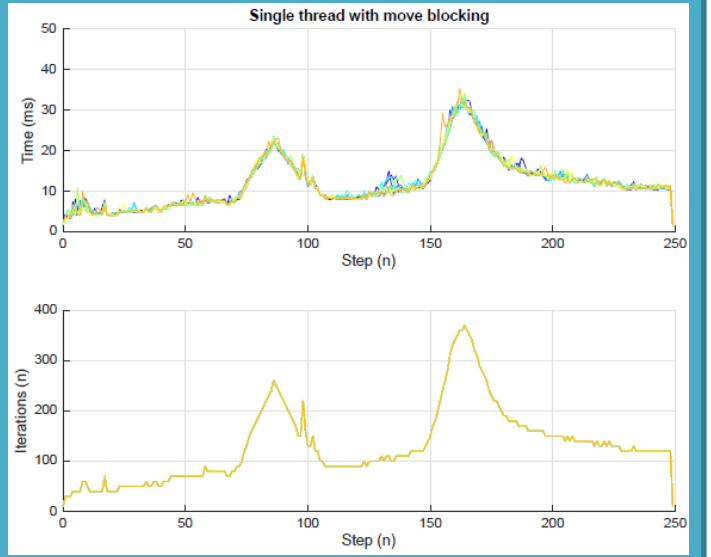
Parameter \ solver		Original problem		Simplified problem	
	tol.	*avg iter. per solution*	*avg time per solution [ms]*	*avg iter. per solution*	*avg time per solution [ms]*
CPLEX	10^{-3}	2000	1332	326	48.7
QPgen, ADMM, SVD null-space		108	171	177	8.7
QPgen ADMM, DS null-space		110	100	1000	30.4
QPgen, FGM dual, SVD null-space		113	160	80	5.9
QPgen FGM dual, DS null-space		113	100	200	10.1
QPgen FGM dual, SVD null-space	10^{-4}	/	/	286	9.7
QPgen FGM dual, DS null-space		/	/	385	13.1

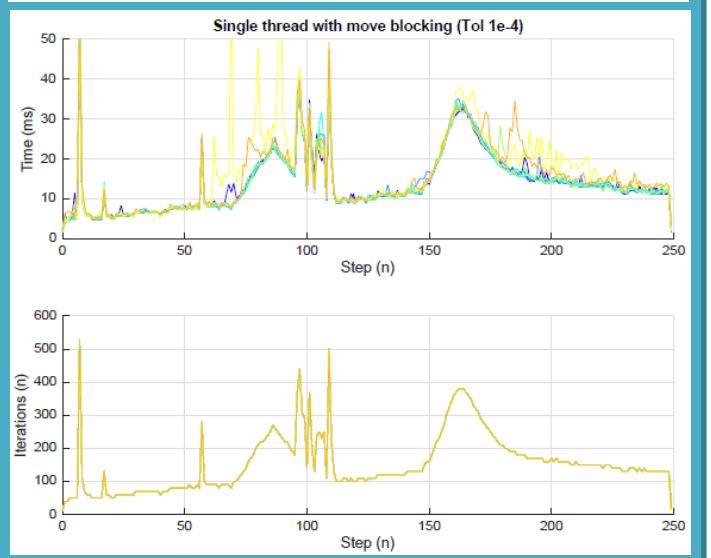
Fast QP computational demand - parallel

Parallelization and code optimization using *Intel compiler*

- QPgen simplified, single VS four cores, tol: 10^{-3} , 10^{-4}

Single-thread





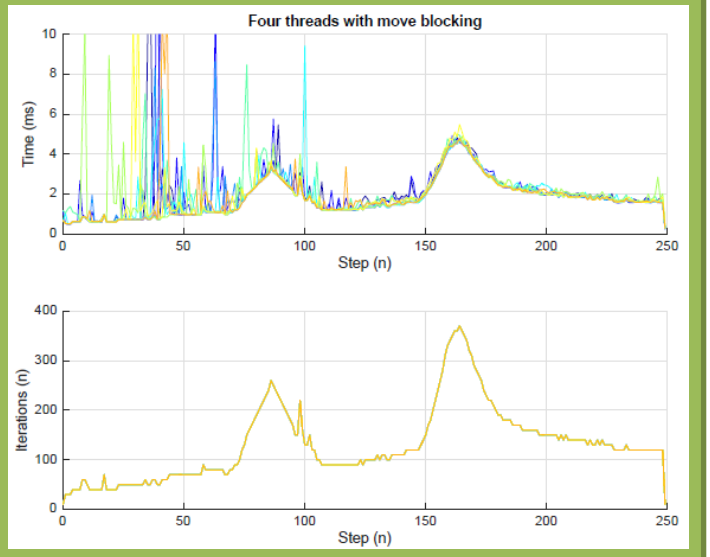
tol 10^{-3}

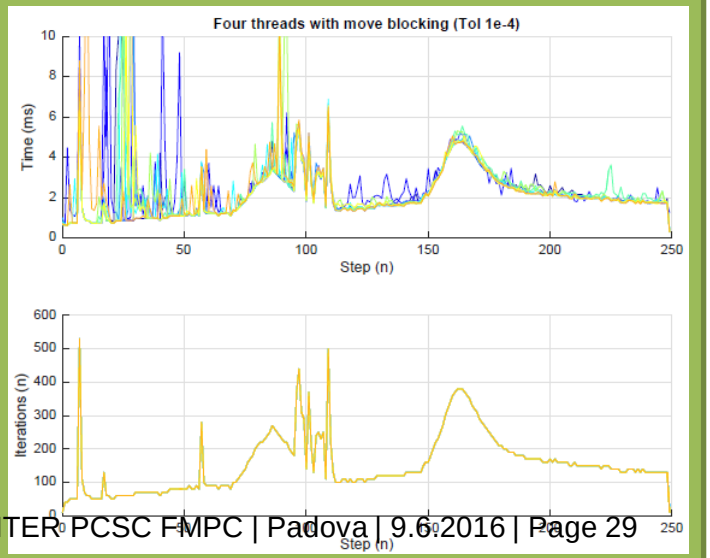
4x speed-up

tol 10^{-4}

5x speed-up

Parallel 4-threads (cores)





Fast QP computational demand - parallel

QPgen, parallelization, code optimization using *Intel compiler*

- Parallel implementation achieves the expected speed-up,
- Peak comp. time about 10 ms after removal of random spikes,
- *Random spikes are due to other running processes in a PC*

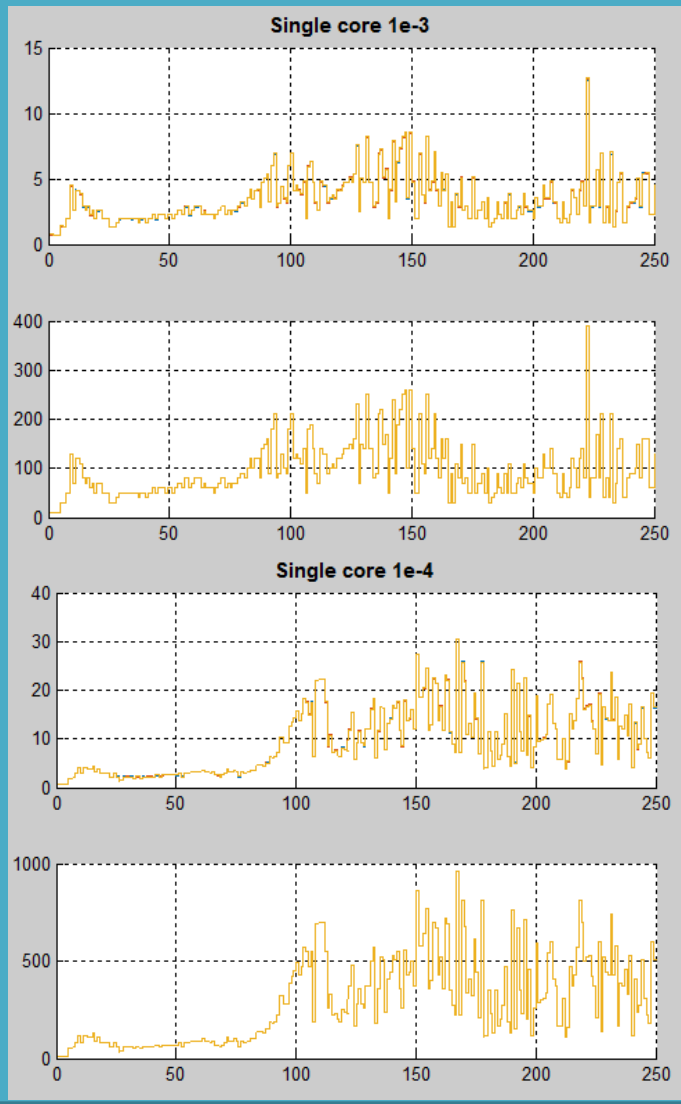
Parameter \ solver		Original problem		Simplified problem	
	tolerance	avg iterations per solution	avg / peak time per solution	avg iterations per solution	avg / peak time per solution
Single-core	10^{-3}	123.3	127.6 / 426.96	130.6	11.94 / 31.61
Parallel	10^{-3}	/	/	130.6	1.94 / 8.47
Single-core	10^{-4}	/	/	154.1	14.40 / 49.51
Parallel	10^{-4}	/	/	154.1	2.29 / 10.47

QP computational demand - parallel

- ITER-specified RT machine (**Xeon X3470 @ 2.93GHz**)
  QPgen simplified, single vs multi-core, tol: 10^{-3} , 10^{-4}

COSYLAB
CONTROL SYSTEM LABORATORY

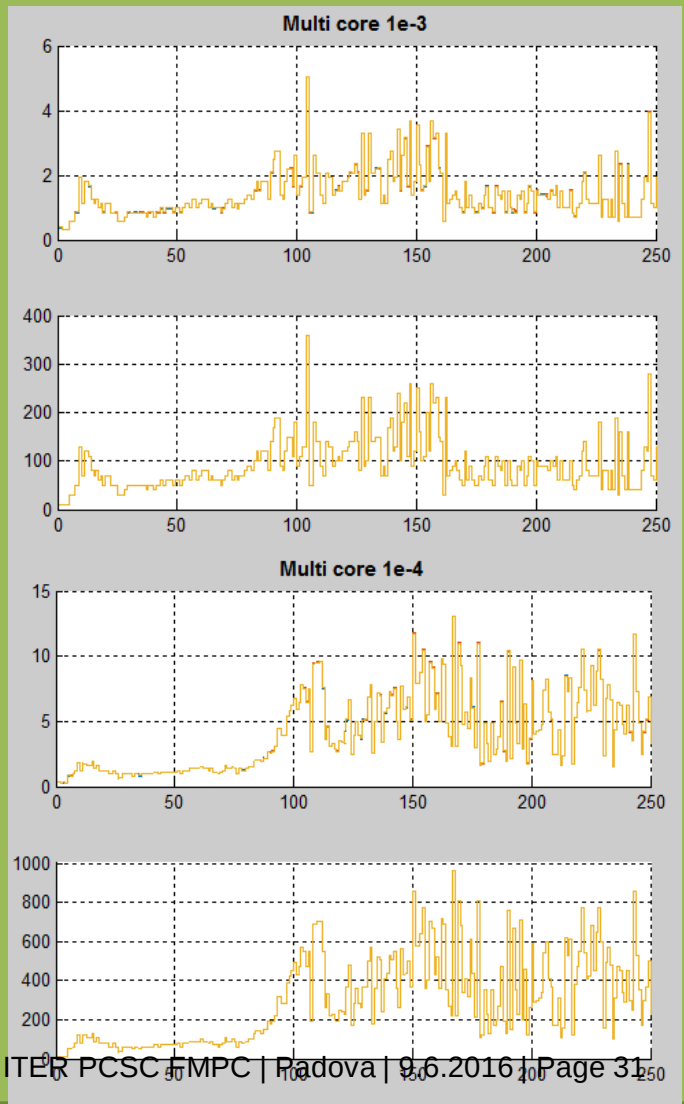
Single-thread



tol 10^{-3} → *2.5x speed-up*

tol 10^{-4} → *2.5x speed-up*

Parallel 3-threads (cores)



QP computational demand - parallel

RT system: low variance among repeated runs, spikes removed

- ITER reference system @Cosylab (Intel Xeon X3470 2.93GHz)
  (RedHat Linux 6.1, vmlinuz-3.0.9-rt26.46.el6rt.x86_64)
- RT Linux PC @JSI (Intel Core i7-2600K 3.4 GHz)
  (Ubuntu Linux 14.04, 4.4.9-rt17 SMP PREEMPT RT kernel)

Settings		ITER reference system		RT Linux PC	
Compiler (Intel C++ Compiler 16.0.3)	Tolerance	Iterations per solution avg / peak	Time per solution avg / peak	Iterations per solution avg / peak	Time per solution avg / peak
Single-core	10^{-3}	96.10 / 390	3.39 / 12.74	96.09 / 390	7.68 / 28.53
3-core parallel	10^{-3}	93.27 / 360	1.44 / 5.06	97.25 / 410	0.83 / 3.16
Single-core	10^{-4}	292.03 / 960	9.54 / 30.45	292.04 / 960	22.20 / 70.71
3-core parallel	10^{-4}	293.07 / 960	4.11 / 13.09	294.62 / 960	2.19 / 6.91

Summary

- Fast online QP methods and solver packages reviewed
- Solvers QPgen, FiOrdOs, HPMPC benchmark-compared
- ITER plasma MPC CSC prototype implemented with QPgen
  - Problem setup redesigned (without MPT/YALMIP)
  - Complexity-reduction techniques
  - Soft output constraints with quadratic cost without slacks
  - Various code adaptations needed, parallel implementation
- Peak computation time 7 ms (tol. 10^{-4}, regular 4-core PC)