Plasma MPC control implementation using fast QP solver

Boštjan Pregelj
Matic Knap
Matija Perne
Samo Gerkšič

Jožef Stefan Institute

FMPCFMPC
project meeting
25.2.2016
JSI, Ljubljana, Slovenia

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.

Outline

- Fast QP solvers
  - Introduction
  - packages - feature comparison
  - benchmark problems
- Plasma control
- QPgen implementation
  - MPC simplification
- Performance evaluation
- Summary & future work

Fast QP solvers – introduction

- Objective: efficient solver for fast MPC implementation
  - Requirements: high speed, low precision, code flexibility
- Many packages available, *open-source favored*
  - Interior point: *Forces, hp-mpc, CVXGEN*
  - Active set: *QPoases*
  - First order: *QPgen, FiOrdOs, hp-mpc, FORCES pro*
- Selected for analysis, *first order algorithms favored*
  - *QPgen, HP-MPC, FiOrdOs*
  - Benchmark problems:
    - Oscillating masses,
    - AFTI16 plane model,
    - Prototype ITER plasma control
- No package comes with all features → further modifications required

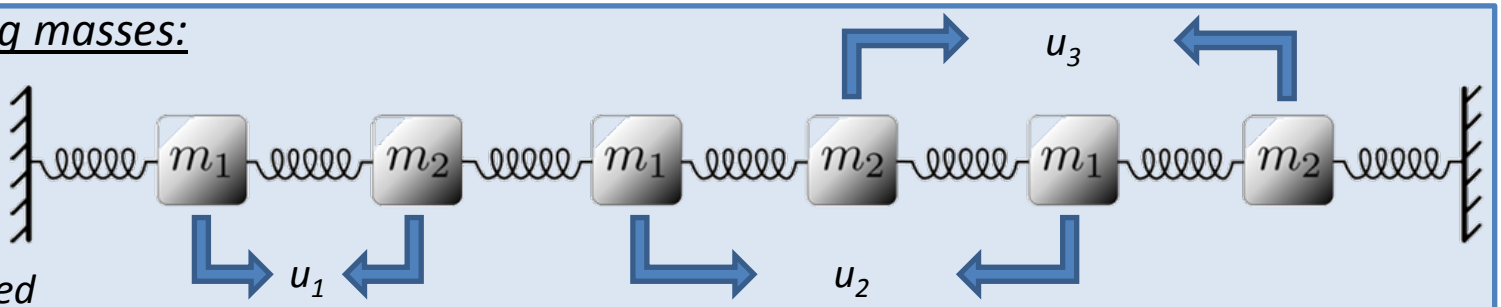
Fast QP solvers – feature comparison

	Feature\solver
flexibility & control suitability	**Matlab based**
	Soft constr. (lin/quad)
	Output feedback (bare MPC – state feedback controller, less useful, output feedback extension prefered

Fast QP solvers – benchmark problems

6 oscillating masses:

Nu = 3
Nx = 12
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:

Nu = 2
Nx = 4
Ny = 2
N = 10
constraint, unstable system, reference tracking

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

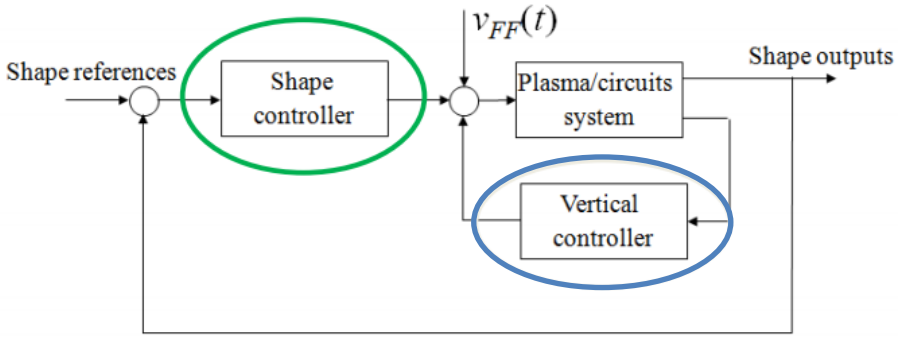
Outline

- Fast QP solvers
  - Introduction
  - packages - feature comparison
  - benchmark problems
- Plasma control
- QPgen implementation
  - MPC simplification
- Performance evaluation
- Summary & future work

Prototype plasma controller

Cascade of 2 control loops

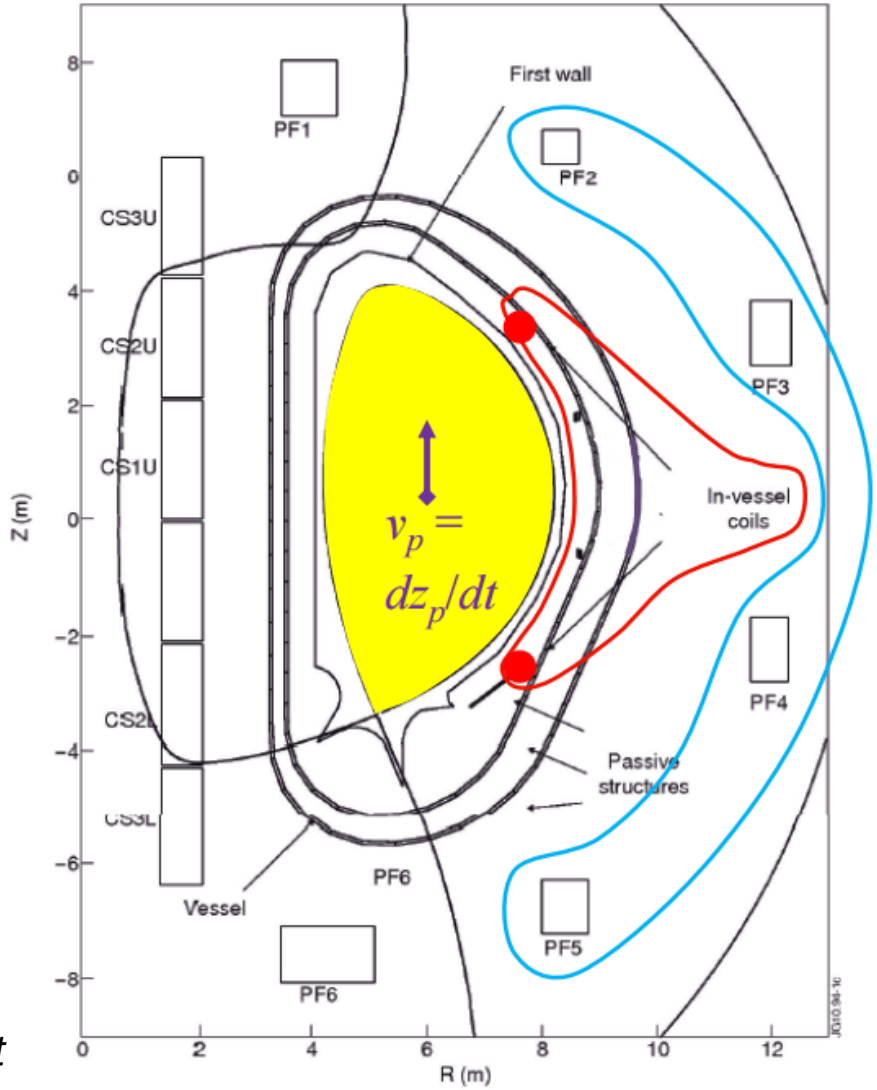
- Faster inner loop:
  - Vertical stabilization **(VS)**
- Slower outer loop:
  - Current & Shape control **(CSC)**



Considered disturbances

- Vertical Displacement Events
- H-L transition

more in FMPCFMPC D2 report



Prototype plasma controller

Inner loop: Vertical stabilization (VS) … *ctLQGz controller*

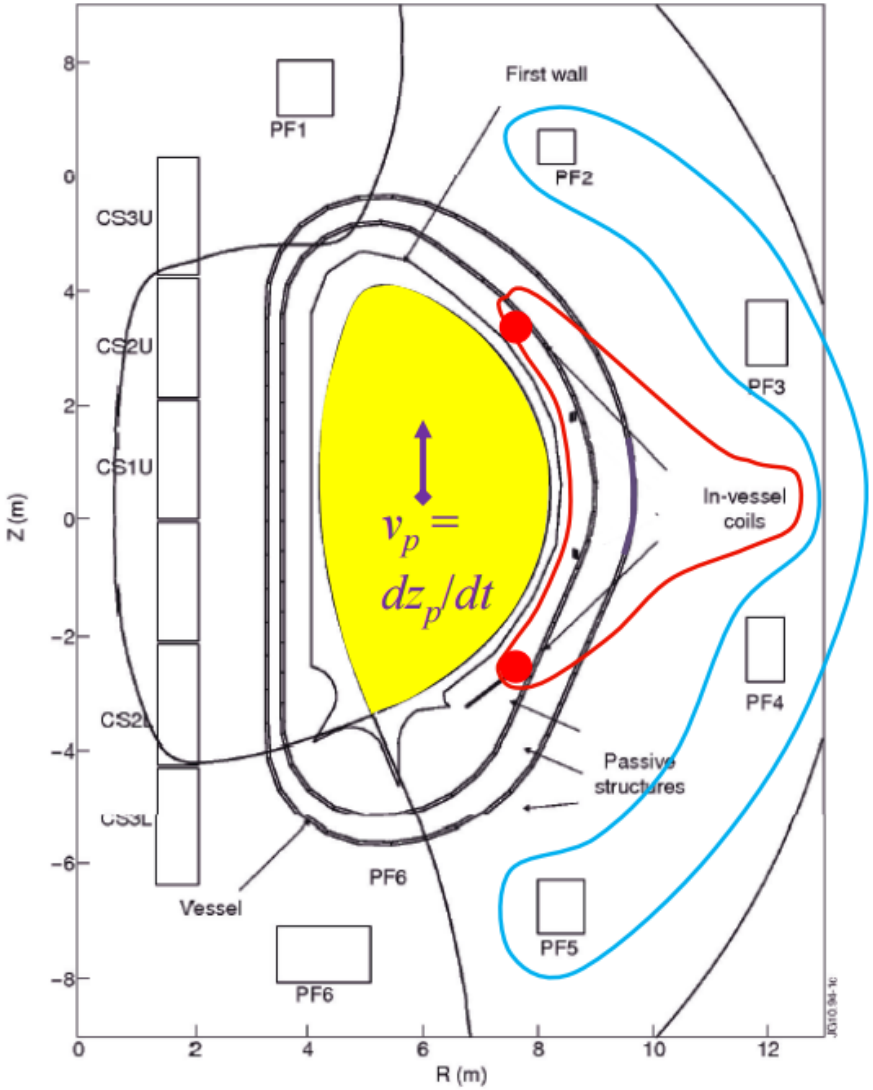
- Actuators:
  - In-vessel coils **(Ic)** VS3

    $u_1 = u_{ic}$
  - Superconductive circuit **(Sc)**

    VS1 (PF2-5) $u_2 = u_{VS1}$
- Controlled process outputs:
  - Plasma vertical velocity

    $y_2 = v_p$
  - Ic coils current

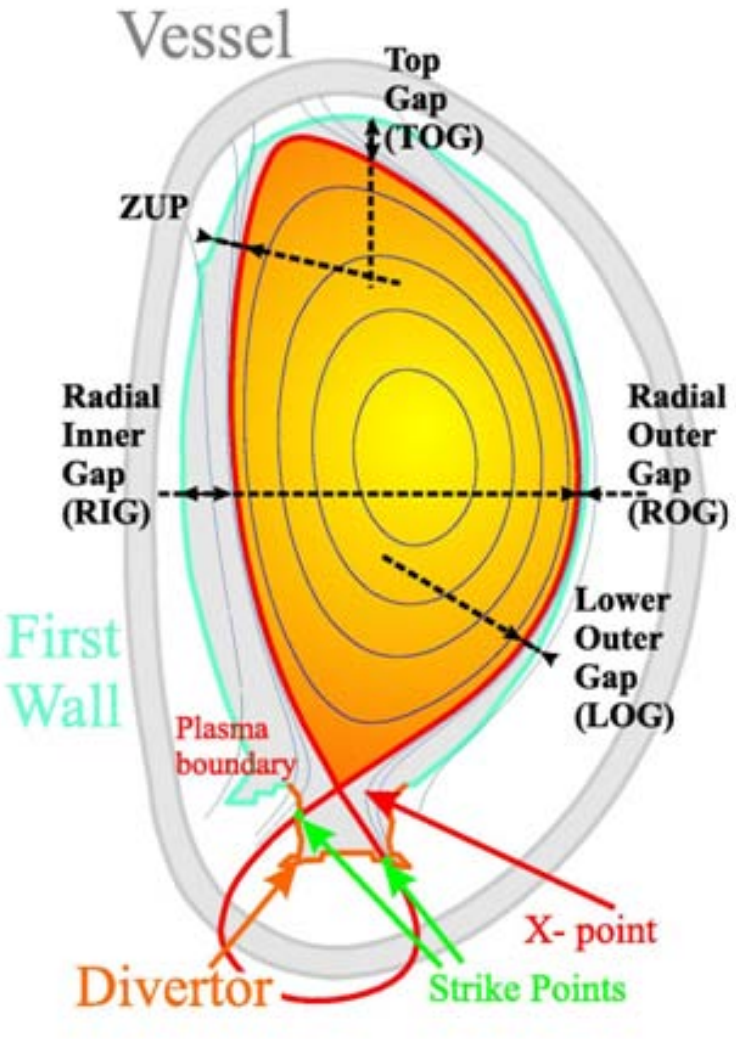
    $y_1 = i_{ic}$ ***! Thermal constraint***
- Additional controlled outputs:
  - Plasma vertical position z_p
  - Sc circuit current i_{VS1}



Prototype plasma controller

Outer loop: Current & Shape controller (CSC) ***… MPC***

- Actuators:
  - 11 main power supply voltages $\mathbf{V}_{PF}$
- Controlled process outputs:
  - Plasma current I_p
  - 6 controlled gaps g
    (2 strike points, 4 gaps)
- Additional measured outputs:
  - 11 superconductive coil currents $\mathbf{I}_{PF}$
  - Sc circuit current i_{VS1}



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: 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

Outline

- Fast QP solvers
  - Introduction
  - packages - feature comparison
  - benchmark problems
- Plasma control
- QPgen implementation
  - MPC problem simplification
- Performance evaluation
- Summary & future work

Plasma control – MPC simplification

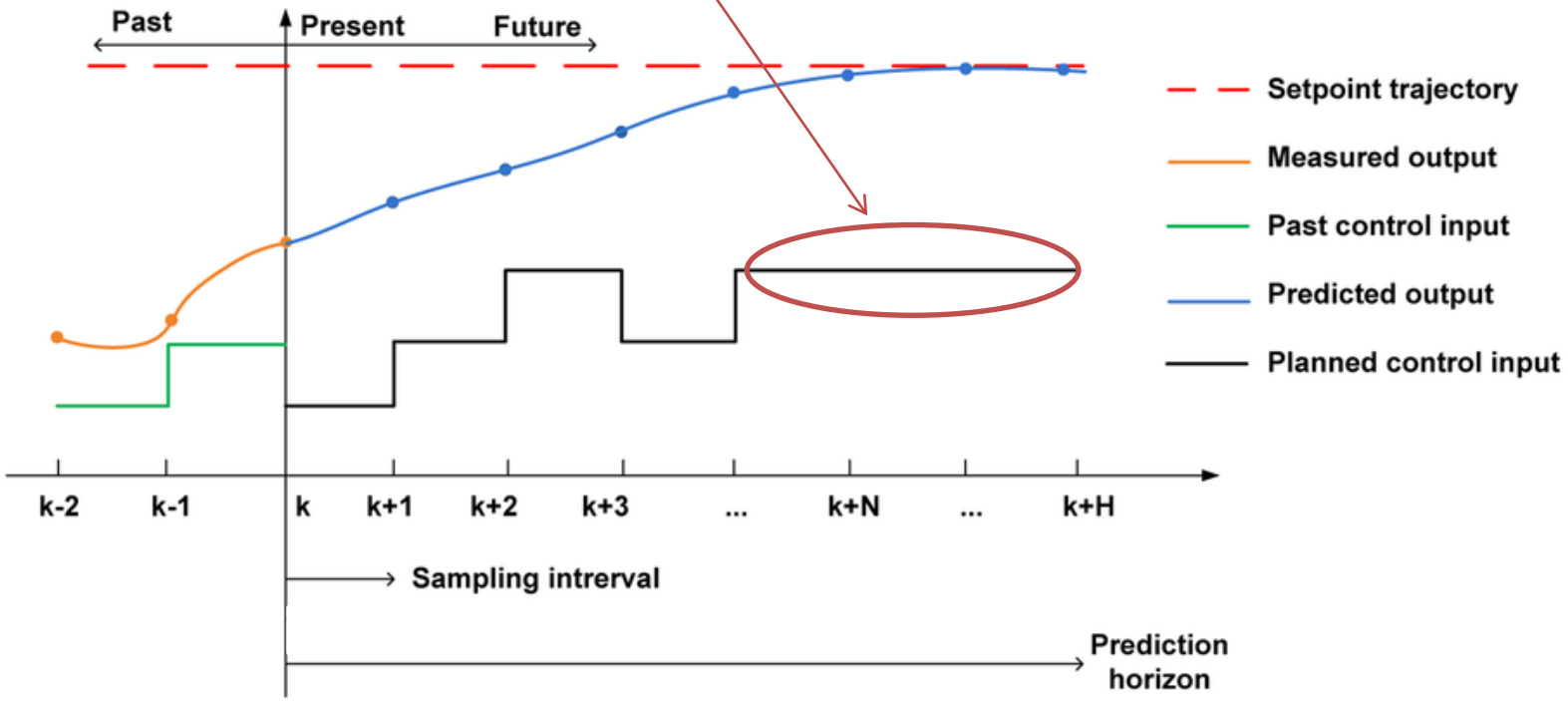
Means to achieve:

- Control Move-blocking
  - *Fixation/clustering of control moves*
- Sparse placement of constraints
  - *Constraints placed each k-th sample*
- Elimination of redundant constraints
  - *Infinity-bounded constraints*
- Efficient handling of soft constraints (no slack variables)
  - *Soft constraints handeled optimization method (Kouzoupius 2014)*
- *Nullspace/preconditioning – optimal calculation method*
  - *SVD based for simplified systems*
  - *DS based for original systems*

Plasma control – MPC simplification

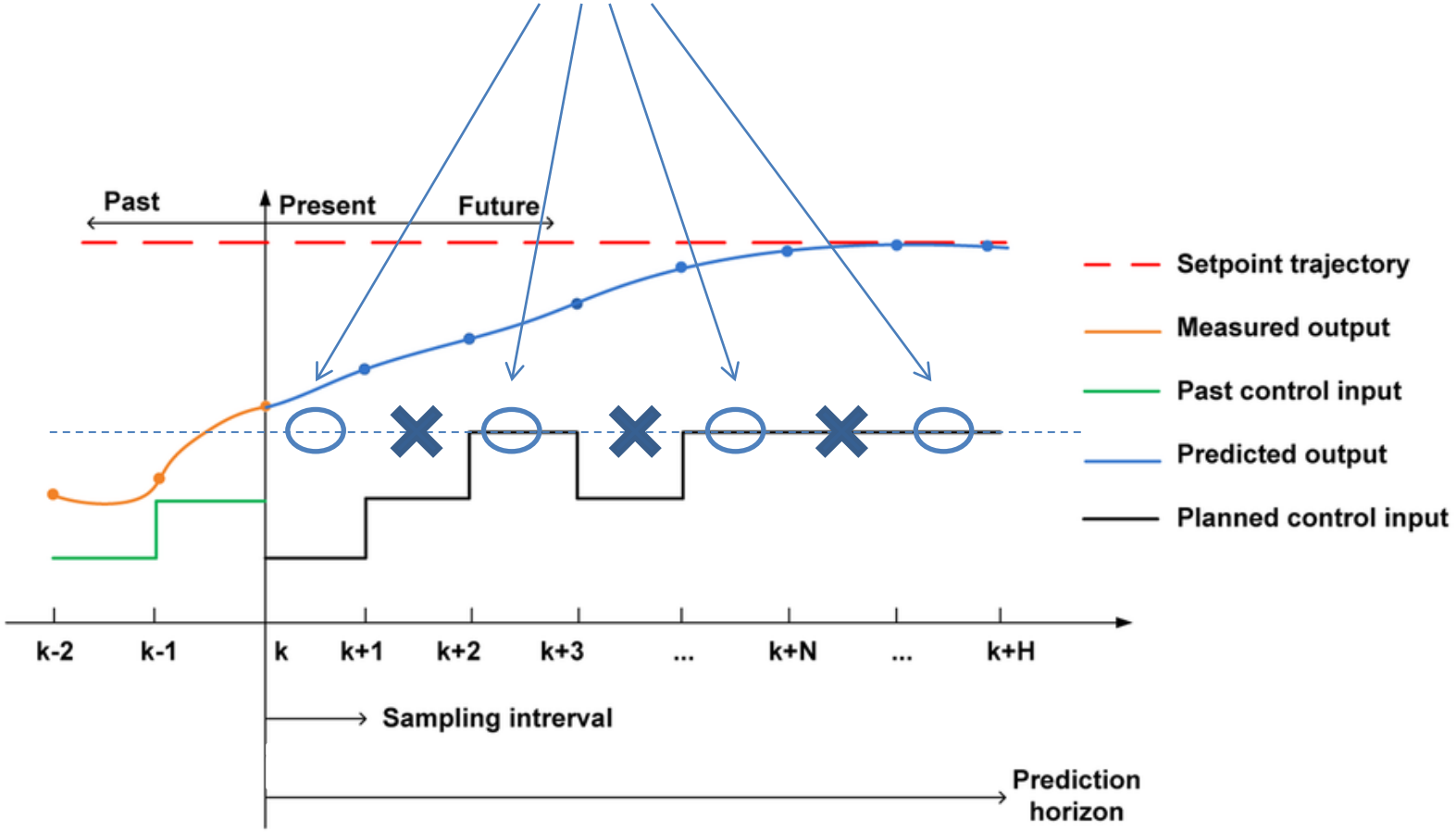
- Control Move-blocking
  - *Fixation/clustering of consequent control moves*
    - *Reduces number of input samples along prediction horizon → H size*
    - *Reduces number of inequality constraint equations → A_{ineq} rows*

    → Significantly affects QP complexity



Plasma control – MPC simplification

- Sparse placement of constraints
  - *Constraints placed each k-th sample*
    - *k-times reduces number of inequality constraints → rows of A_{ineq}*



Plasma control – MPC simplification

Means to achieve:

- Control Move-blocking
  - *Fixation/clustering of control moves*
- Sparse placement of constraints
  - *Constraints placed each k-th sample*

Implemented using MPT toolbox during previous work (Gerkšič)

- Elimination of redundant constraints
  - *Problem reduction:* $r_{cstr} \times N$ *less equations (inequality constr.)*
- Efficient handling of soft constraints (no slack variables)
  - *Soft constraints handeled by optimization method (Kouzoupius 2014)*
- *Nullspace/preconditioning – optimal method choice*
  - *SVD based for simplified systems*
  - *DS based for original systems* →→

$$\begin{bmatrix} B & & & 0 \\ AB & B & & \\ A^2B & AB & B & \\ | & & & \backslash \\ A^{N-1}B & --- & AB & B \end{bmatrix}$$

All functions implemented in QPgen during FMPC FMPC project

Plasma control MPC simplification

Process: Nu = 11, Bx = 62, Ny = 20
MPC: horizon 30
- sparse constraints each 3rd sample (10 coin.)
- 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	20	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	**11**	0	30	30	660×2190	2190	2190×2190	7.7MB
Move blocking	11	62	11	0	**3**	30	660×219	219	219×219	2.3MB
Constr. sample reduction	11	62	11	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

Outline

- Fast QP solvers
  - Introduction
  - packages - feature comparison
  - benchmark problems
- Plasma control
- QPgen implentation
  - MPC simplification
- Performance evaluation
- Summary & future work

Plasma control simulations (BPLI-LM52)

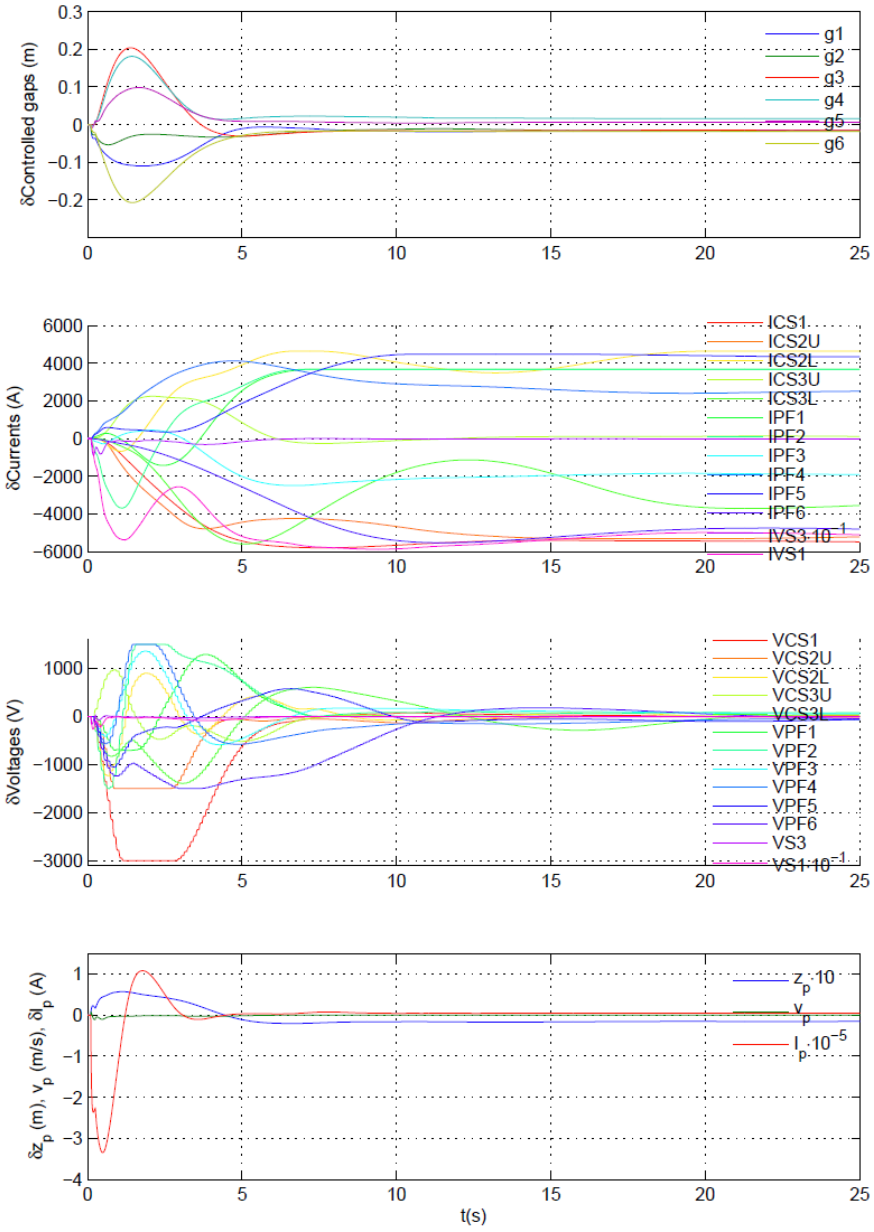


Fig. 29. BPLI simulation: ctLQGz-MPC, model LM52, using original model and CPLEX solver.

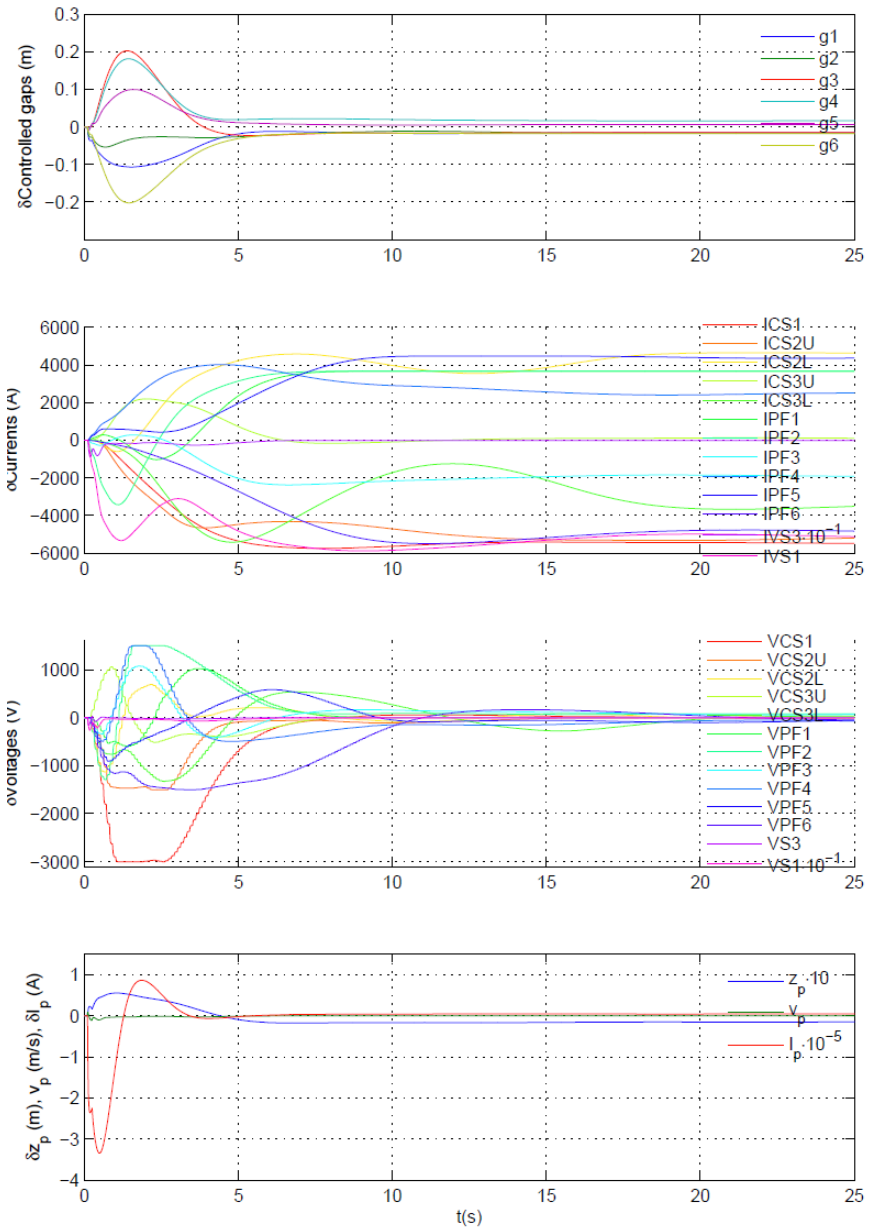
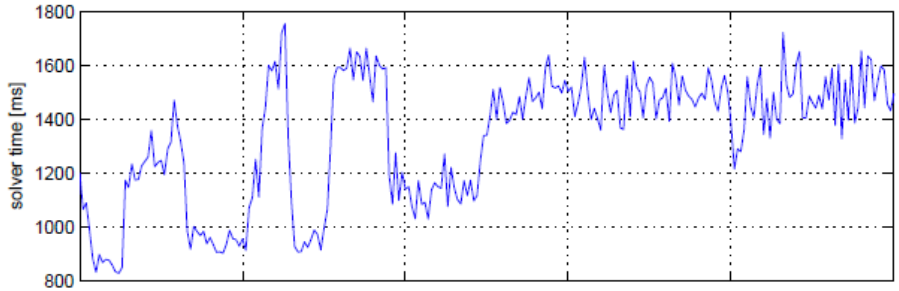
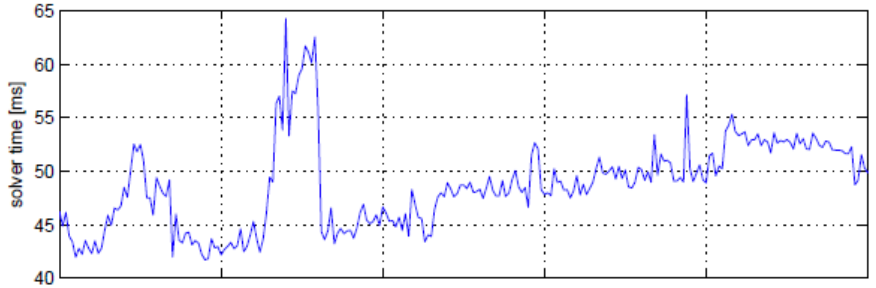


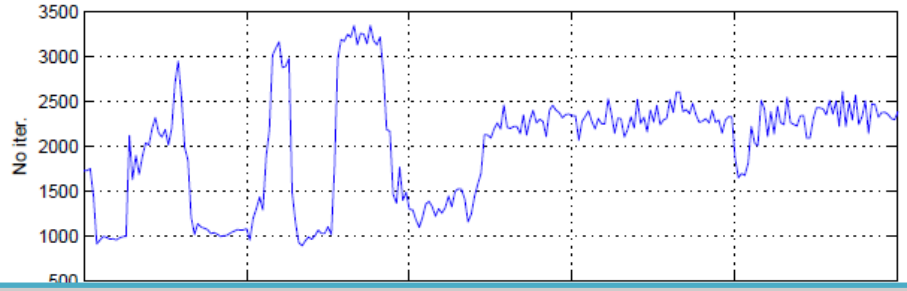
Fig. 30. BPLI simulation: ctLQGz-MPC, model LM52, using simplified model and CPLEX solver.

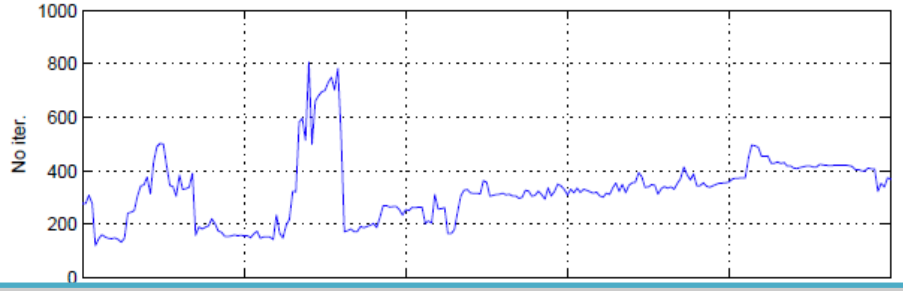
MPC computational efficiency

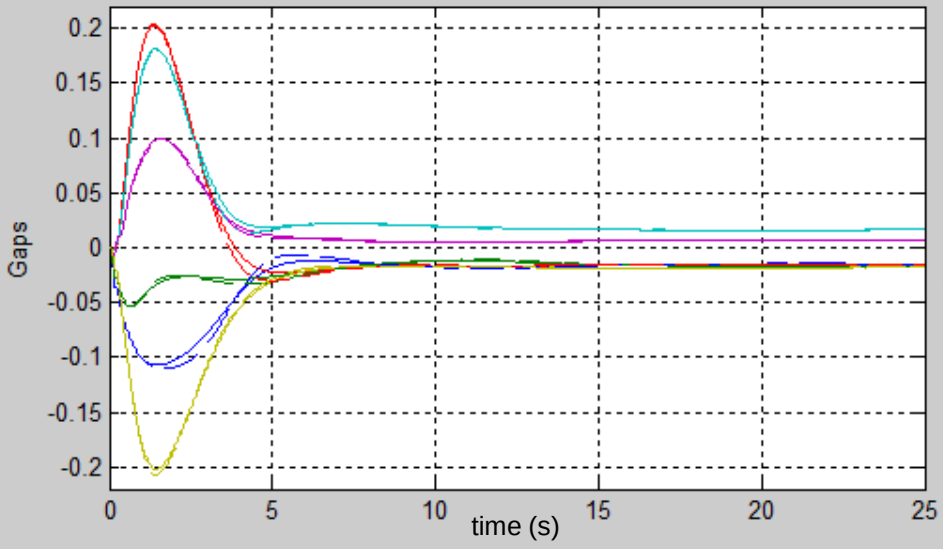
- CPLEX original (left) VS simplified MPC problem (right)

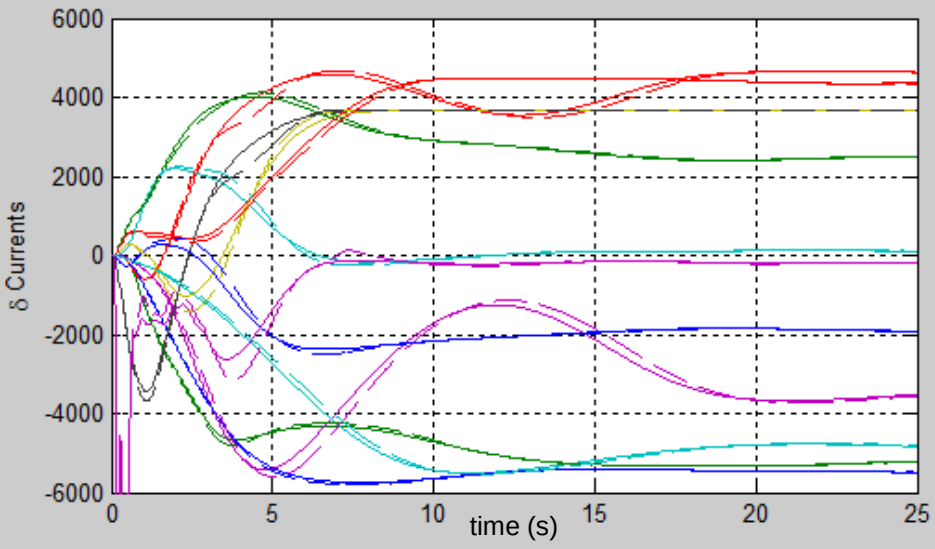






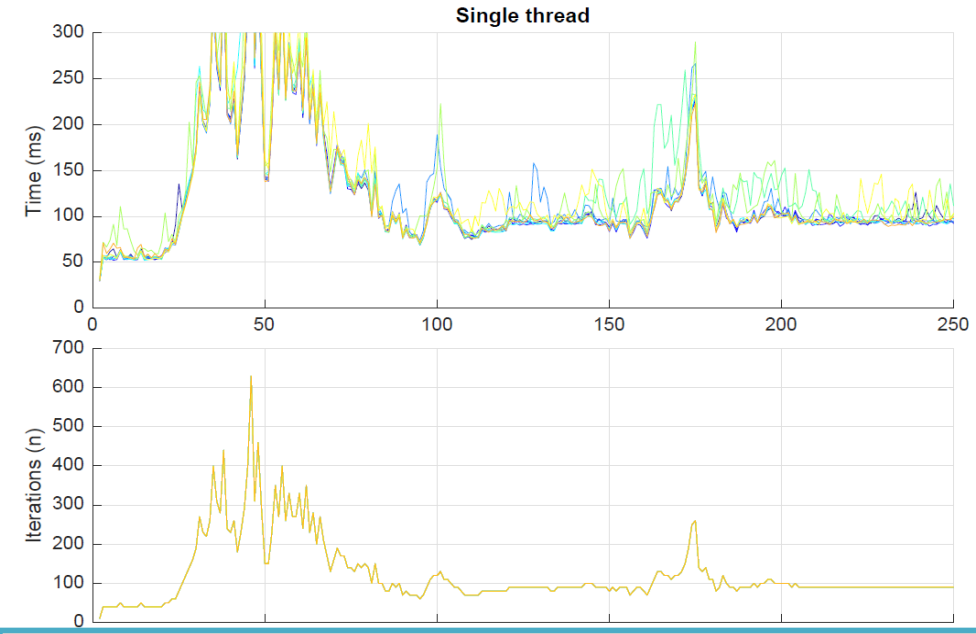


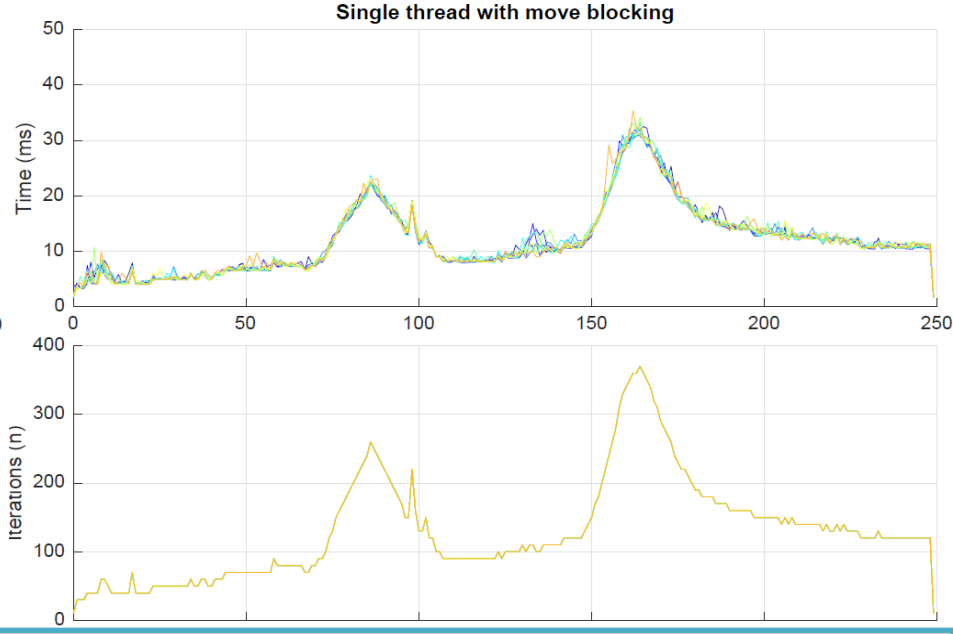


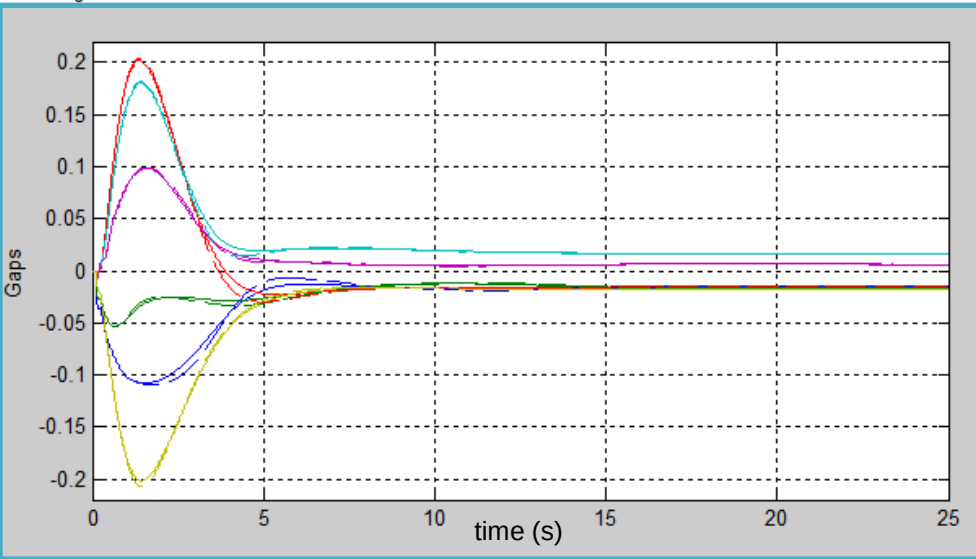


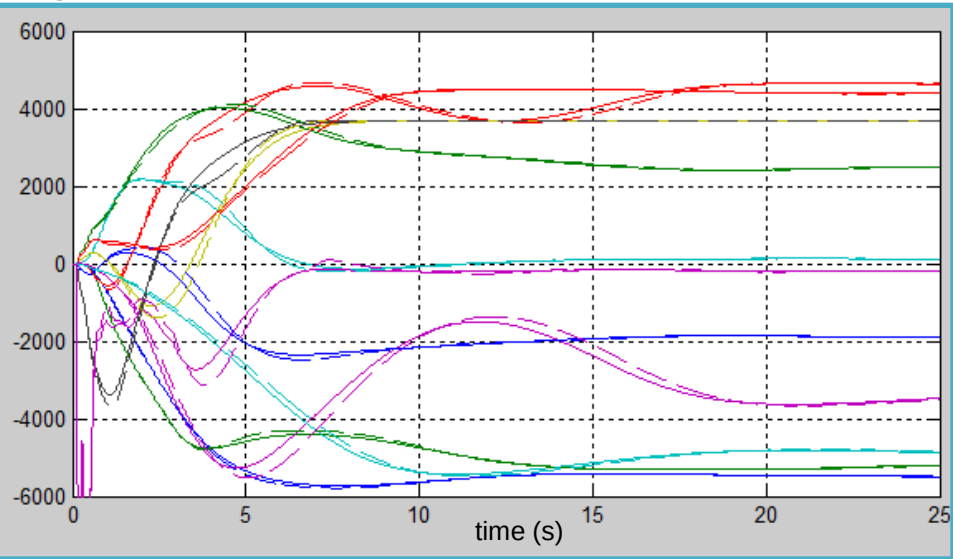
MPC computational efficiency

- QPgen original (left) VS simplified (right) MPC problem



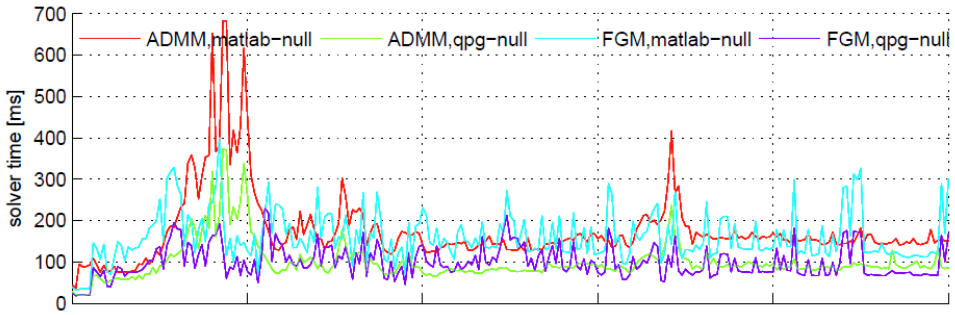


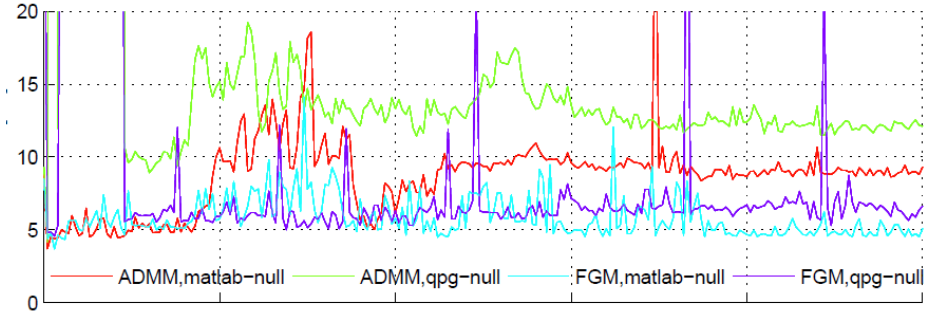


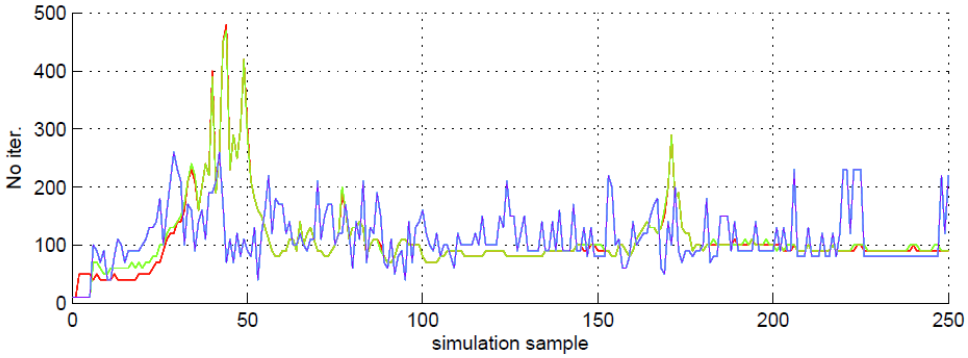


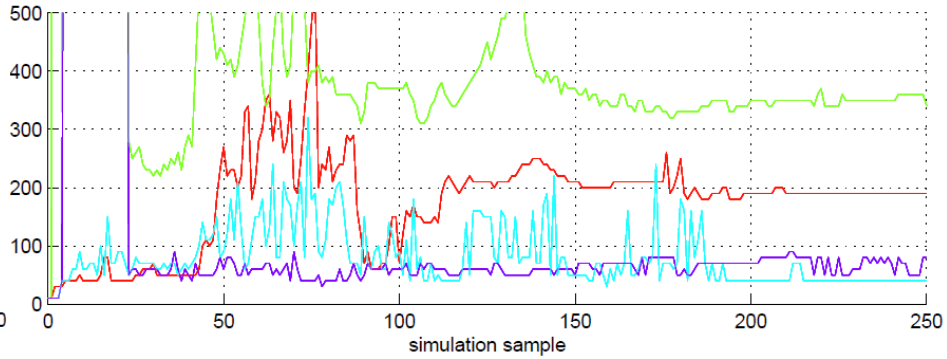
MPC computational efficiency

- Analisys of optimal method and algorithm selection
- Qpgen original (left) VS simplified (right) MPC problem
  - Comparison of FGM / ADMM method
  - Comparison of SVD / DS nullspace









MPC computational efficiency

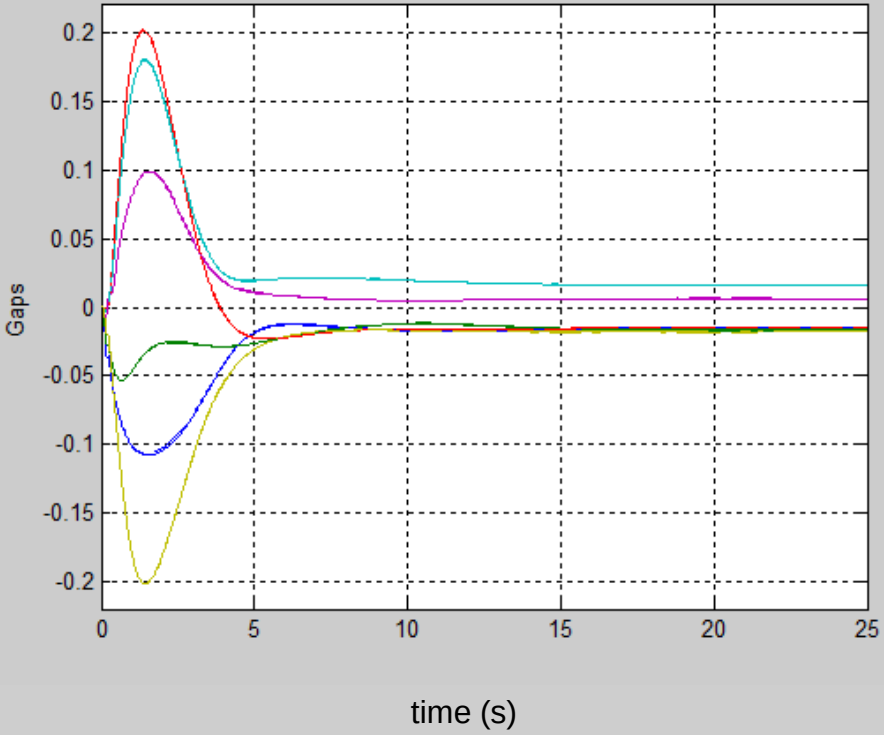
- FGM faster for simplified problem
- DS-based null-space better for original MPC problems
- SVD null-space better for simplified MPC problems
- 1-order precision increase → ~1.5x computation time increase

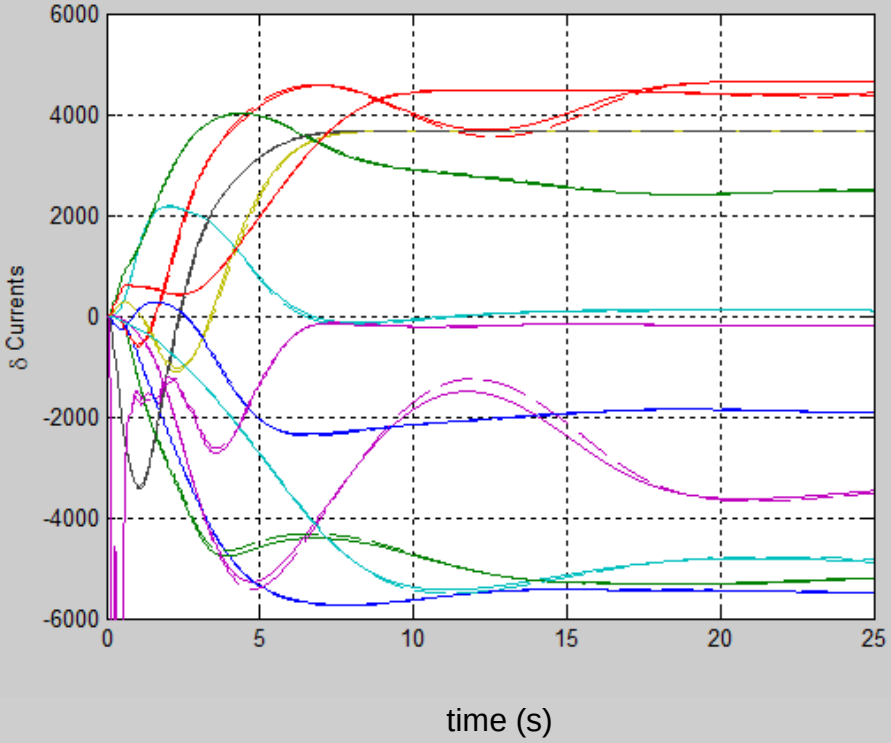
Parameter \ solver		Original problem		Simplified problem	
	tol.	*avg iter. per solution*	*avg time per solution [ms]*	*avg iter. per solution*	*avg time per solution [ms]*
CPLEX		2000	1332	326	48.7
QPgen, ADMM, SVD null-space		108	171	177	8.7
QPgen ADMM, DS null-space	***10^{-3}***	110	100	1000	30.4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

Control quality

CPLEX VS Qpgen, simplified MPC

- Small differences, no significant effect on control quality

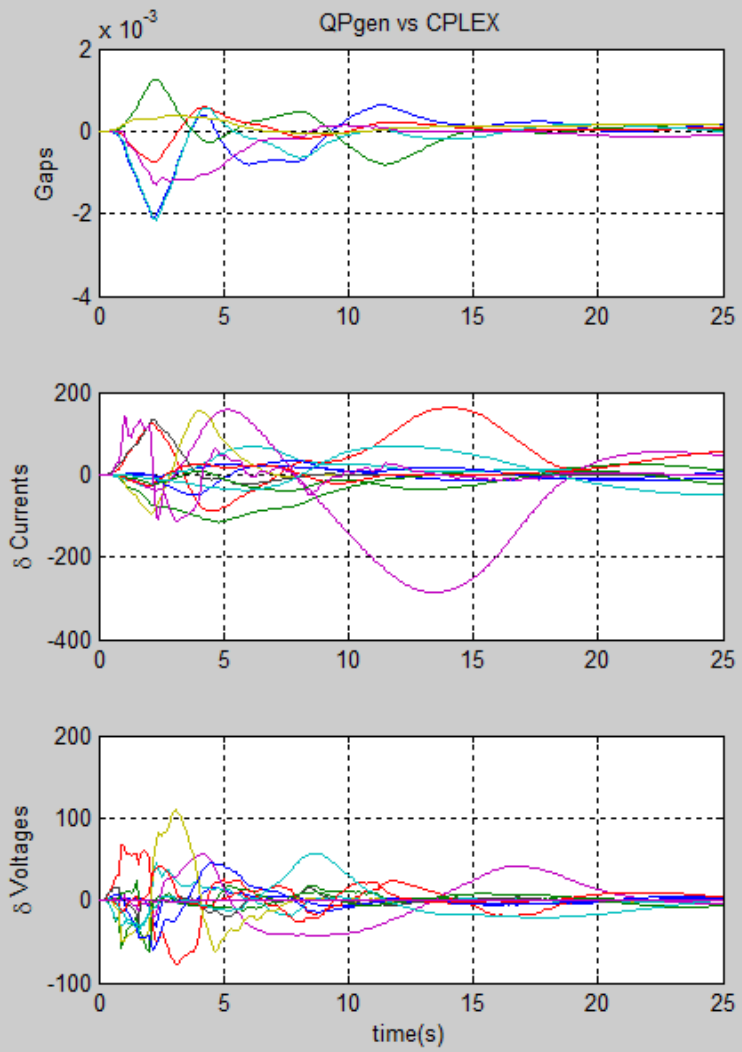


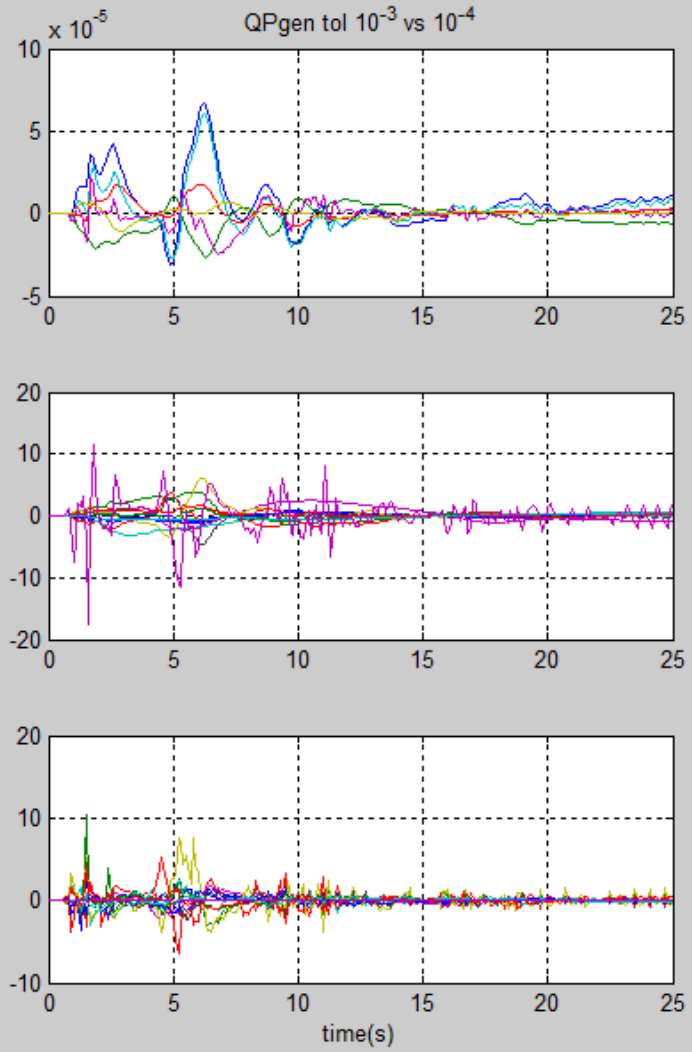


Control quality

CPLEX VS QPgen with tol: e=10^{-3} and 10^{-4}

- Almost no difference between QPgen tolerance variations

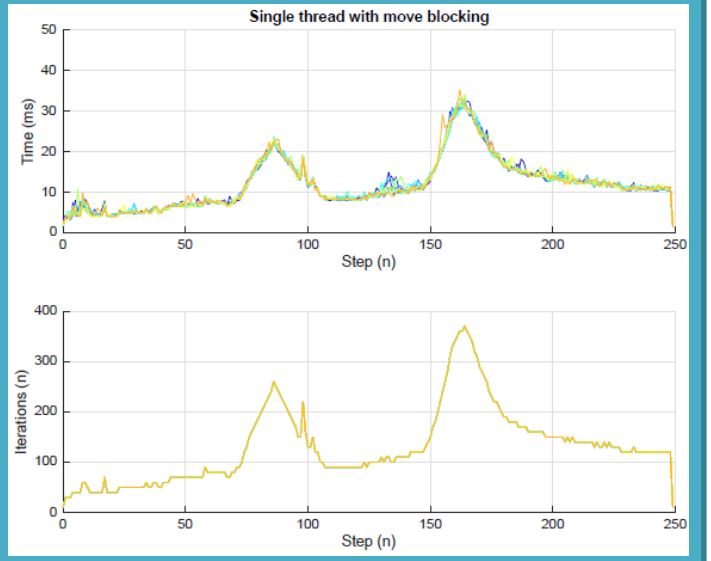


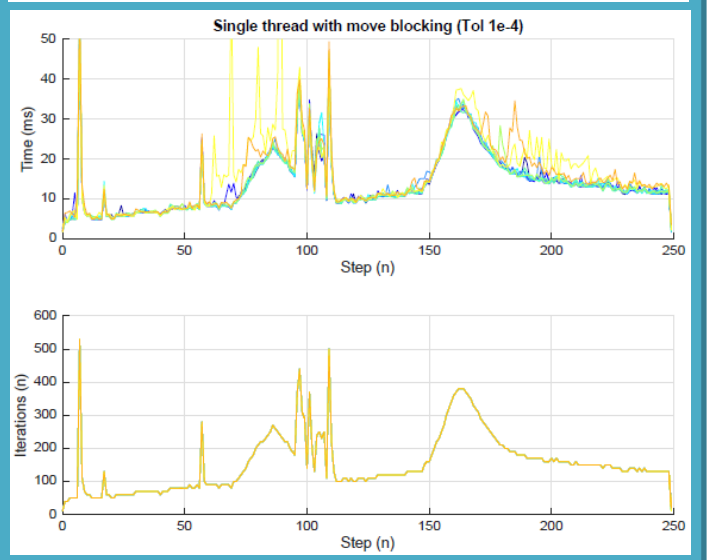


MPC computational efficiency

- Parallelization and code optimization using *Intel compiler*
  - Qpgen simplified, single VS multi thread, tol: 10^{-3} , 10^{-4}

Single-thread





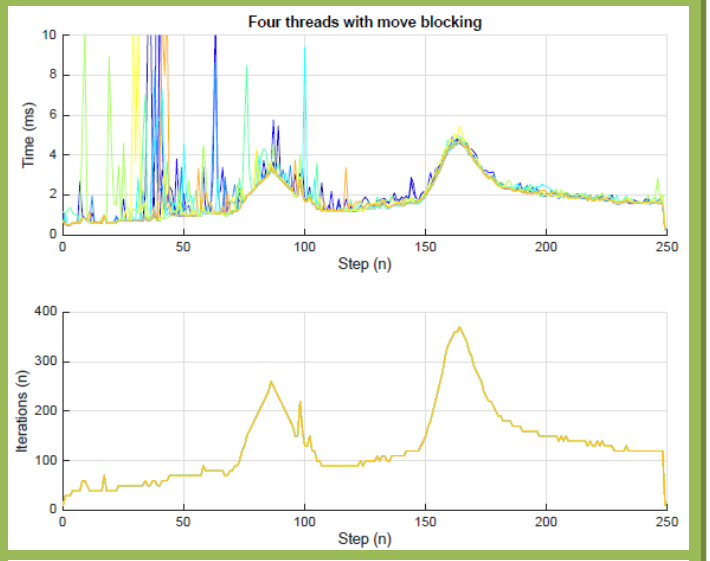
tol 10^{-3}

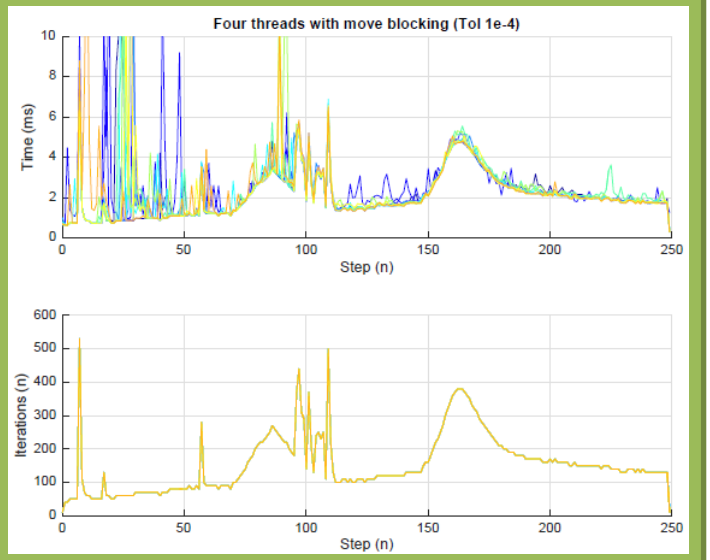
4x speed-up

tol 10^{-3}

5x speed-up

Parallel 4-threads (cores)





MPC computational efficiency

- Parallelization and code optimization using *Intel compiler*
  - Qpgen simplified, single VS multi thread, tol: 10^{-3} , 10^{-4}

Parameter \ solver	tolerance	Original problem		Simplified problem	
		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		11.94 / 31.61
Intel-compiler parallelization	10^{-3}	/	/	130.6	1.94 / 8.47
Single-core	10^{-4}	/	/	154.1	14.40 / 49.51
Intel-compiler parallelization	10^{-4}	/	/	154.1	2.29 / 10.47

HPMPC v Qpgen – algorithm comparison

- Algorithms for solving inner problem in dual methods:
  - HPMPC → Riccati factorization
  - QPgen → direct approach via matrix inverse (condensed system)
    → LDL factorization (sparse system)

- Profiling the solver functions showed Riccati to be faster for large systems with long horizon
  - > Future goal: try to use Riccati factorization algorithm within Qpgen for large problems

Summary

- Several QP solver packages and methods reviewed
- Few first order QP solvers benchmark-compared
- QPgen selected and further optimized
  - Quadratic slacks
  - Move-blocking
  - Sparse constraints
  - Optimized Nullspace method choice
- QPgen implementation for ITER plasma control
- Performance evaluation compared to universal solvers (CPLEX, quadprog)

Future work

- Verification of operation using hard-limited iterration number/solver time (when solution quality is not reached within allowed time)
- Verification of benefit and Implementation of HP-MPC Riccati factorization to Qpgen
- Validation of output tracking for variable reference
- Implementation of the final control scheme
- Code optimization using profiler, AVX-512
- Graphic processor Cuda

Plasma MPC control implementation using fast QP solver

Boštjan Pregelj
Matic Knap
Matija Perne
Samo Gerkšič

Jožef Stefan Institute

FMPCFMPC
project meeting
25.2.2016
JSI, Ljubljana, Slovenia

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.