FMPCFMPC

EUROfusion / Work Programme 2015 Enabling Research

Project: **Fast Model Predictive Control for Magnetic Plasma Control**

EUROfusion

Deliverable D2:

Conceptual design of fast MPC for ITER PCSC

Date due: 31.12.2015

Samo Gerkšič, Boštjan Pregelj, Andrej Debenjak, Matija Perne, Matic Knap [a]

Gianmaria De Tommasi, Marco Ariola, Alfredo Pironti, Massimiliano Mattei [b]

[a]*Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia*

[b]*Associazione EURATOM-ENEA-CREATE, Univ. di Napoli Federico II, Via Claudio 21,80125, Napoli, Italy*

CREATE PPCC team

cosylab CONTROL SYSTEM LABORATORY

SLOVENIAN RESEARCH AGENCY

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.

Contents

Summary

Plasma magnetic control schemes typically consist of a cascade control scheme with a plasma current and shape controller (CSC) in the outer loop and a vertical stabilization (VS) controller in the inner loop. The main challenges to tackle are: suppression of plasma shape transients after disturbances specific to tokamak reactors; robustness to changes in the plant dynamics; best possible use of the available chamber volume, so that the plasma is placed as close as possible to the plasma facing components; and tight control complying with the power supply and gap constraints. Model predictive control (MPC) is an advanced process control approach for dealing with constraints, already established in a range of industries involving multivariable processes with slower dynamics.

The aim of the FMPCFMPC project is to design a practically feasible MPC controller for the ITER CSC, which will be able to improve control in the presence of constraints on process signals.

Part I of this report expands the first deliverable D1 "A set of reduced-order models and a state-estimation scheme for ITER plasma shape and current control", where the simulation setup was defined, a set of models for control design and analysis was prepared, and a state-estimation scheme for the MPC CSC control was made. The section on the implementation of several MPC controller variants is added. In comparison to our previous prototype versions of ITER CSC, the described controller includes set-point tracking and is able to consider a larger number of gaps describing the plasma boundary. Tuning of the controller is provisional only, because the final structure should be chosen considering the abilities of the optimization algorithms, which are still under investigation.

Part II describes a parallel effort of examining efficient fast quadratic programming (QP) methods suitable for the on-line solution of MPC optimization problems which are being repeatedly solved at each sample time of the CSC control loop. A survey of the available QP methods is given, with emphasis on first-order methods, which have been recently considered as prime candidates for fast online MPC control. Although they do not posses excellent theoretical convergence properties, they may be very fast at finding medium-accuracy solutions, which are sufficiently accurate for closed-loop control. In addition, they are relatively simple, and their convergence properties may be certified. A description of three available open-source fast QP solvers is made. A case study of a prototype version of ITER CSC control based on the QP solver QPgen is presented, which has shown a five-fold speed-up compared to the state-of-the-art commercial solver CPLEX, with peak computation times around 10 ms, which is already considered sufficiently fast for the 100 ms sample time estimated to be suitable for the ITER CSC control loop.

Part I: Development of the MPC platform

1 Introduction

In a magnetically confined fusion reactor, the plasma current and shape controller (CSC) is the component of plasma magnetic control (PMC) that determines the voltages applied to the poloidal field coils, to control the coil currents and the plasma parameters, such as the plasma shape, current, and position. In case of elongated, and hence vertically unstable, plasmas, the CSC acts on the system already stabilized by the inner vertical stabilization (VS) controller. The task of PMC is to maintain the prescribed plasma shape and plasma-wall distances (gaps), in presence of disturbances, such as vertical displacement events (VDE), H-L transitions or edge-localized modes (ELM), and to changes of local dynamics in different operating points [1, 2]. In order to achieve high performance, control methods that would improve the performance near the vessel boundaries and the actuator constraints are desired.

Model Predictive Control (MPC) is an established advanced process control approach in the process industry. It has gained wide industrial acceptance by facilitating a systematic approach to control of large-scale multivariable systems, with efficient handling of constraints on process variables and enabling plant optimization [3]. These advantages are considered beneficial for CSC, and potentially also for other control systems of a tokamak. The main obstacle to using MPC for control of such processes is the restriction of the most relevant MPC methods to processes with relatively slow dynamics due to the long achievable sampling rates, typically needed for the on-line optimization. However, speeding up MPC has been a topic of intensive research recently [4, 5, 6].

Modern MPC methods are based on state-space models, where model order is an important consideration. Plasma modelling procedures based on first principles result in models of very high orders [9], which are not convenient for control, and model reduction techniques are required to obtain models used for MPC design, in order to reach a manageable computational demand and numerical conditioning. The model order cannot be reduced arbitrarily, because it is important that the models retain a sufficiently accurate description of model dynamics; on the other hand, a high model order may also result in over-fitting to specific local dynamics that may actually result in poor robustness to changes of operating conditions. A set of models for different operating points of the ITER scenario is required for assessment of robustness of the controller to model inaccuracy.

An MPC scheme involving a steady-state target calculator (TC) and a dynamic controller (DC) for transient dynamics is being considered for implementation. For such MPC schemes it is particularly important that the model is also accurate in the low-frequency region.

Plasma models involve a large number of states that are not measurable. In state-space MPC approaches it is commonplace to use a linear observer or state estimator (Kalman filter) to estimate model states from the applied inputs and the measured outputs. Conceptually it would be appropriate to use a moving-horizon state estimator, which may also consider constraints just like the MPC controller, however the constrained state-estimation task is computationally even more challenging than constrained control.

Due to the superconductive actuators, the plasma models for CSC control contain integrators (i.e., poles at the origin), which require special attention in the estimator implementation. Simple disturbance modelling using an open-loop observer with an output step disturbance model, which is frequently employed in simple industrial MPC methods, results in internal instability of the system, where internal signal values in the model may slowly diverge and eventually reach overflow. This problem is avoided by using a closed-loop estimator and by a proper choice of a disturbance model.

This report presents the concept of the plasma control simulation setup for the ITER CSC control problem, including a set of plasma models used for both plasma linear simulations and controller design, a set of disturbances expected to affect CSC control, a reference "conventional" control scheme including VS scheme based on [8], also intended to be used with MPC, and an alternative CSC based on singular perturbation decomposition (SPD) [7], and a state estimation scheme required for MPC CSC implementation.

2 Magnetic plasma control simulation setup

2.1 Simulation models

The simulations and controller design methods are based on high-order local linear dynamical models of the tokamak plasma from CREATE-L or CREATE-NL [9, 10] nonlinear equilibrium codes, at several different operating points for ITER plasma Scenario 1. The models are listed in Table 1. The model codes coincide with the time into the scenario. Initially, three different models were considered. Additional models were added in order to examine the effects of rapid changes of dynamics during L-H and H-L transitions, where closed-loop control with the reference control scheme is used. They may be used to assess the validity of simulation of such disturbances using a single linear model. The control design may be eventually validated with nonlinear simulations with the CREATE-NL model, but linear models are required for nominal control design and for fast performance assessment (nonlinear simulations are time consuming and may be affected by numerical issues).

Table 1. Local linear dynamic models.

Model code	Disturbance	Growth rate (s^{-1})
t80	No disturbance	9.1
t520	No disturbance	2.9
t079d50	L-H transition	9.3
t080d50	L-H transition	8.6
t081d50	L-H transition	7.8
t083d00	L-H transition	5.5
t085d00	No disturbance	5.0
t090d00	No disturbance	3.6
t529d00	H-L transition	3.5
t530d50	H-L transition	6.5
t531d00	H-L transition	10.1
t531d50	H-L transition	10.1
t532d00	H-L transition	6.8
t532d50	H-L transition	6.5

The models may be used to simulate disturbances specific to tokamak reactors. A Vertical displacement event (VDE) of given amplitude may be simulated by initializing the simulation with an appropriate non-zero initial state of the plasma model. Several other types of disturbances may be simulated by injecting pre-computed trajectories of β_p and l_i exogenous inputs to the plasma model: Minor disruption, Uncontrolled ELM, L-H transition, H-L transition. Some of these disturbances are relevant only at specific time into the pulse, therefore with specific local linear models.

2.2 Reference control scheme

As a reference control scheme, the Matlab/Simulink simulation scheme "v2d0" of CREATE is used. The scheme is displayed in Fig. 1, and the Plasma Model block is expanded in Fig. 2. The shown scheme is slightly modified to use the "absolute" values of the control signals rather than the "delta" differential values, relative to the operating point of the local linear model.

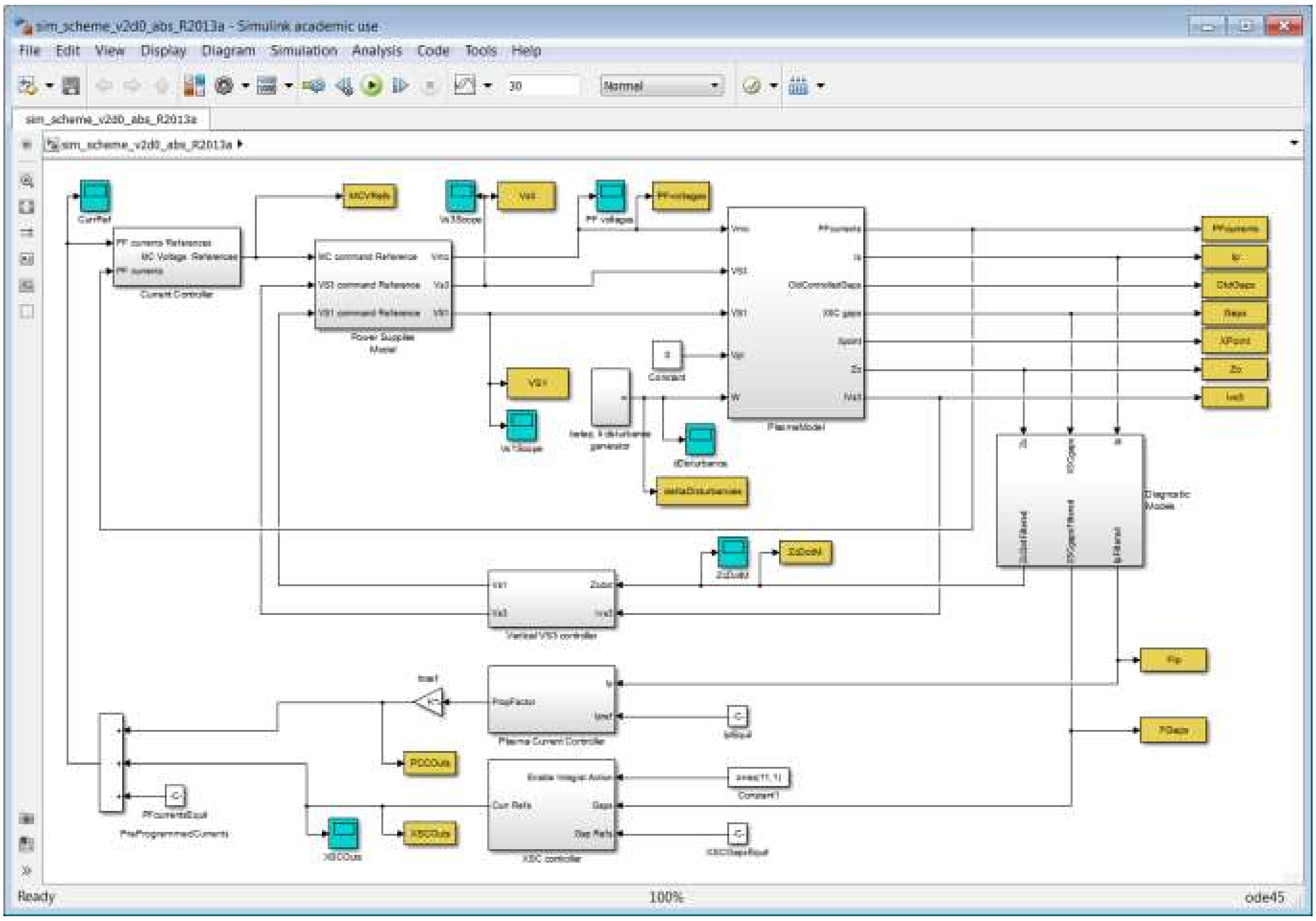


Fig. 1. Reference simulation control scheme "v2d0".

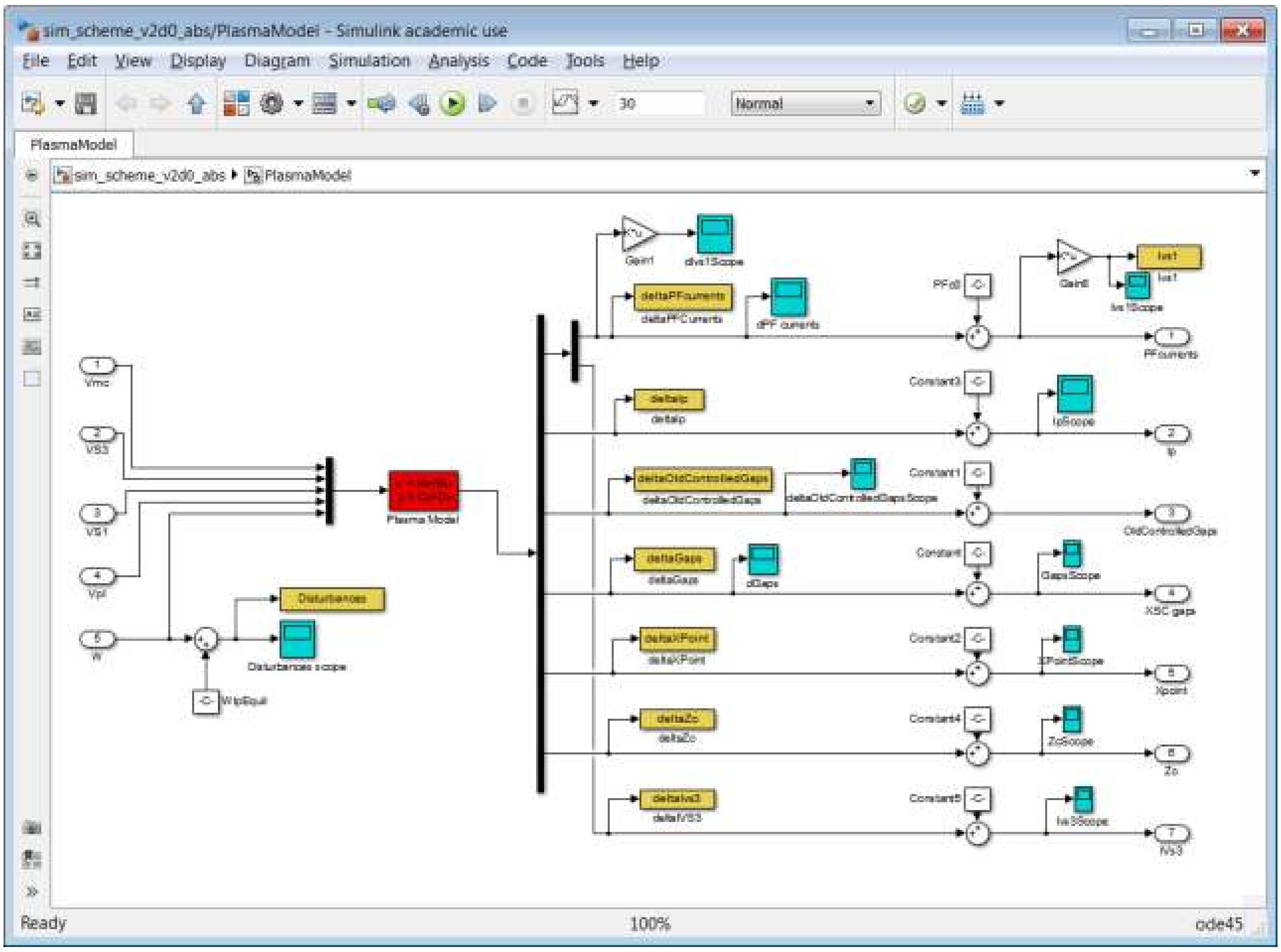


Fig. 2. Block PlasmaModel of the scheme "v2d0".

The scheme includes:

- the plasma/circuits linearized model (the state-space block "Plasma Model"),
- several sum nodes to append the operating-point offset to the outputs of the plasma linear model, in order to generate "absolute" signal values,
- a simplified model of plasma diagnostics for the plasma vertical velocity v_p and position z_p (a first-order dynamic lag filter with the time constant equal to $7{\cdot}10^{-3}$ s is considered),
- simplified models of the power supplies for the superconductive (SC) coils VS1 and for the in-vessel (IV) ohmic coils VS3 in the block "Power Supplies Model" (a first-order dynamic lag with the time constant equal to $7.5{\cdot}10^{-3}$ s; a delay equal to $2.5{\cdot}10^{-3}$ s; saturations ±6 kV and ±1.5 kV for VS1 and VS3, respectively),
- simplified models of the main power supplies in the block "Power Supplies Model" (saturations ±1.5 kV, except for VCS1 ±3 kV, and first-order dynamic lag with the time constant equal to 0.015 s and a delay equal to 0.015 s),
- the inner cascade control loop of the VS system in the block "Vertical VS3 controller", which aims at vertically stabilizing the plasma column,

- the outer cascade control loops of plasma current and shape control, comprising the blocks "Current Controller", "Plasma Current Controller" and "XSC Controller",
- blocks enabling the simulation of vertical displacement events (VDE), using a corresponding plasma model initial state, and H-L transitions, by injecting recorded profiles of β_p and l_i (BPLI) [11].

The simulation solver ode23tb is used, with relative tolerance 10^{-5}.

3 Vertical stabilization

For vertical stabilization, an approach similar to the one proposed in [14] is used both in the reference scheme and with the MPC controller.

The VS controller acts on the control variable $\mathbf{u}_{\mathrm{VS}} = [u_{\mathrm{VS},1}\ u_{\mathrm{VS},2}]^{\mathrm{T}}$, where:

- $u_{\mathrm{VS},1}$ is the voltage applied to the IV coils VS3,
- $u_{\mathrm{VS},2}$ is the voltage applied to the SC circuit VS1,

while it attempts to drive to zero the controlled inputs $\mathbf{y}_{\mathrm{VS}} = [y_{\mathrm{VS},1}\ y_{\mathrm{VS},2}]^{\mathrm{T}}$, where

- $y_{\mathrm{VS},1}$ is the VS3 power supply current,
- $y_{\mathrm{VS},2}$ is the plasma vertical velocity v_p.

The following feedback transfer function matrix is used

$$\mathbf{u}_{\mathrm{VS}}(t) = \mathbf{K}_{\mathrm{SOF}}\mathbf{y}_{\mathrm{VS}}(t), \quad \mathbf{K}_{\mathrm{SOF}} = \begin{bmatrix} \dfrac{0.00175s + 0.07}{(1/6)s + 1} & -\dfrac{175s + 7000}{(1/6)s + 1} \\ 0.1 & 0 \end{bmatrix} \tag{1}$$

4 MPC Plasma current and shape controller (CSC)

The CSC output $\mathbf{V}_{\mathrm{PF}}$ is the vector of the 11 voltage requests to the main power supply (main converters).

The CSC inputs include:

- the currents in the 11 superconductive coils $\mathbf{I}_{\mathrm{PF}}$.
- the plasma current I_p,
- the vector of controlled geometrical descriptors (gaps) $\mathbf{g}$.

Fig. 3 displays the simulation scheme with the MPC CSC. The MPC CSC is contained in a single block "predictiveCSC", which is shown expanded in Fig. 4. The MPC controller in the block "MPT Controller" is a tracking controller that drives the controlled outputs (CSC inputs $\mathbf{I}_{\mathrm{PF}}$, I_p, and $\mathbf{g}$) to their set-point values. It is based on a reduced-order discrete-time state-space model. The states of such models do not have physical meanings, and are therefore estimated using the Kalman filter kfCSC[1], which computes state estimates (xhat) based on output measurements and past controller outputs. Such a scheme may also be used with a LQ optimal controller.

[1] The kalman filter kfCSC is split into two blocks, kfCSC_ABC and kfCSC_D, to avoid a false algebraic loop warning of Simulink.

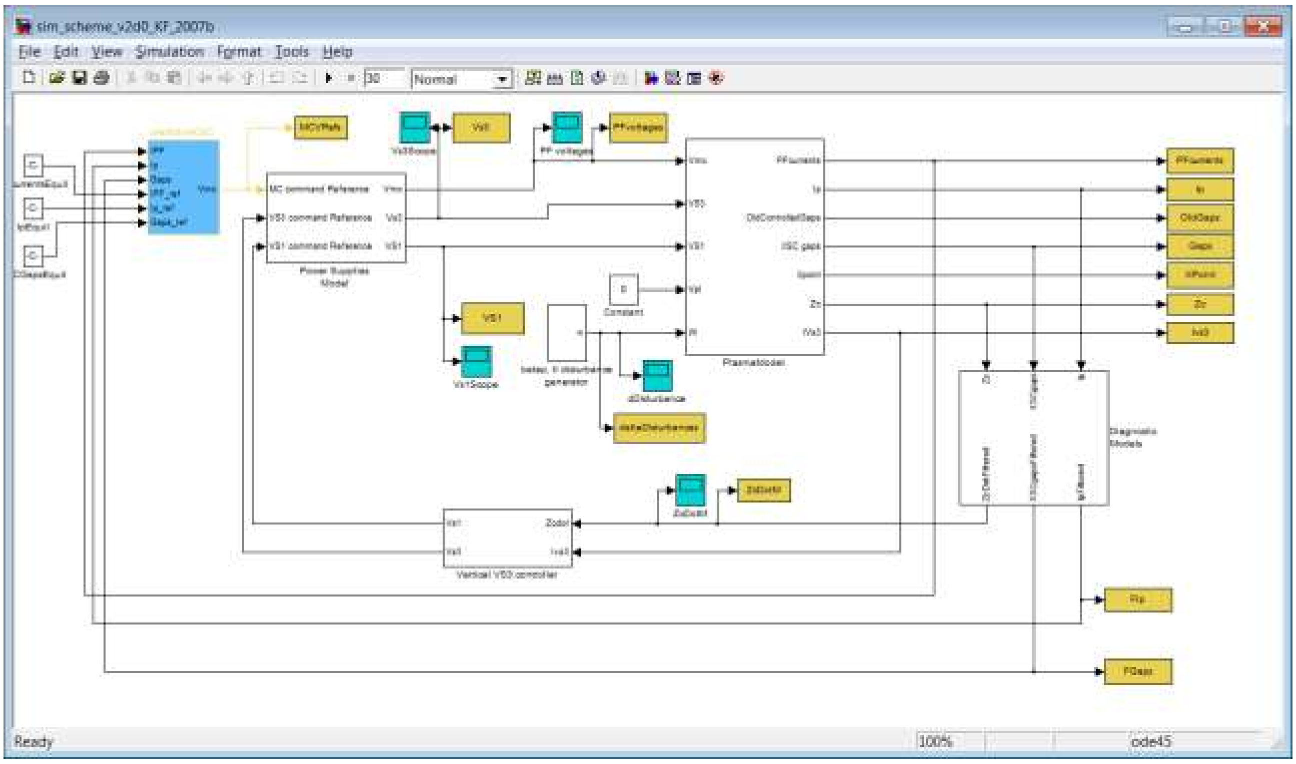


Fig. 3. Simulation control scheme with MPC CSC.

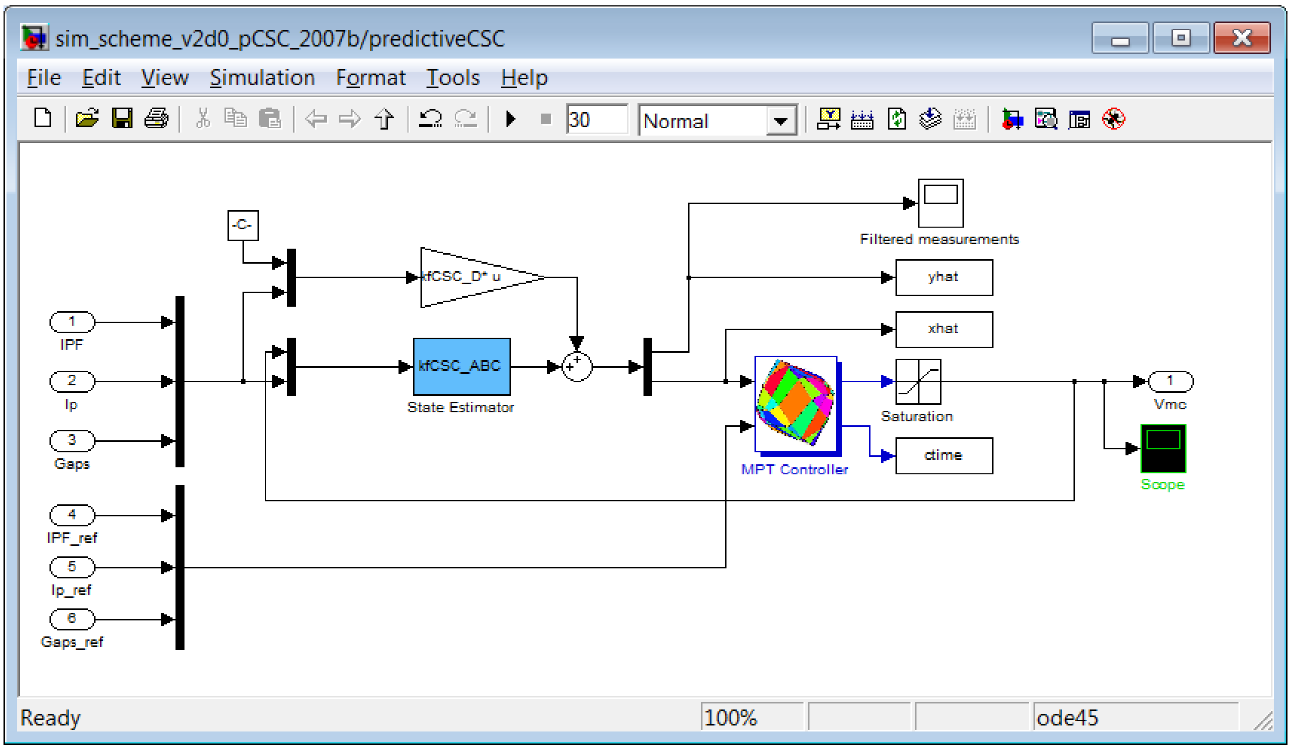


Fig. 4. Block "predictiveCSC".

4.1 Model preparation for KF and MPC

The nominal design of MPC and state estimation is based on the nominal plasma model t90 in the continuous-time form. A sequence of steps is required to prepare the model in a suitable form for MPC:

- A state-space model with the appropriate inputs and outputs for the simulation scheme is formed.

- For controller design, plasma resistance is set to zero, to avoid issues with model reduction.
- The vertical velocity output required for VS is appended.
- The simplified models of the power supplies and sensors (diagnostics) are appended.
- The VS feedback is added to stabilize the unstable pole. The computed closed-loop dynamics is taken as open-loop dynamics for the outer CSC control.
- The subsystem from the process inputs $\mathbf{u}_{\mathrm{CSC}} = \mathbf{V}_{\mathrm{PF}}$ to the outputs $\mathbf{y}_{\mathrm{CSC}} = [\mathbf{I}_{\mathrm{PF}}\ I_p\ \mathbf{g}]^{\mathrm{T}}$ for CSC is extracted.
- Any possible dynamic artefacts at very low frequencies due to numerical noise in computations must be removed using the `stabsep` Matlab function, because they have adverse effects on model reduction and cause problems with the Target Calculator design. Clean integrating dynamics due to the superconducting coils are expected from all 11 process inputs $\mathbf{u}_{\mathrm{CSC}}$.
- Model reduction is used to decrease the model order as much as possible without affecting the performance. Unfortunately, it is difficult to choose the appropriate order in advance, because the choice may depend on KF and MPC tuning - a better model may be required for a higher bandwidth. The effect of order reduction is illustrated with the Bode diagram in Fig. 5. Model order 60 is chosen for the moment.

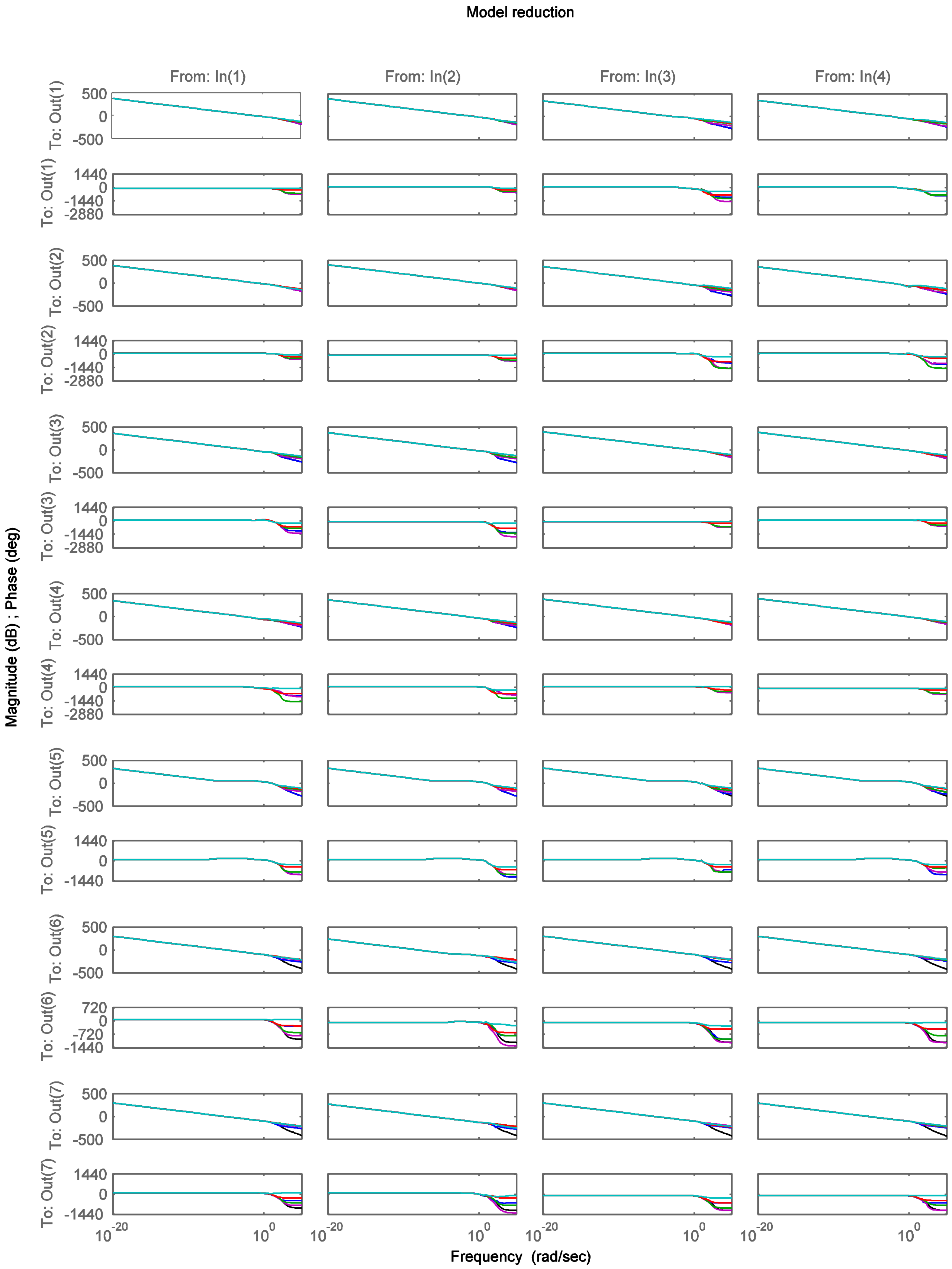


Fig. 5. Bode diagram of a subsystem of the unreduced model (black) and the reduced-order models with 129 (blue), 80 (magenta), 60 (green), 40 (red), and 20 states (cyan). A subsystem from inputs 1, 2, 10, and 11 to outputs 1, 2, 10, 11, 12, 13, and 14 is shown.

- The base model for the MPC CSC $\{\mathbf{A}_{CSC},\mathbf{B}_{CSC},\mathbf{C}_{CSC},\mathbf{0}\}$ is finally obtained with model conversion to discrete time with the sampling time $T_s = 0.1$ s, assuming zero-order hold.

- The CSC should facilitate offset-free control of I_p and $\mathbf{g}$ to zero with integral action, set-point tracking. In our implementation, integral action is based on the *disturbance estimation* (DE) concept [13]. For the estimation of asymptotically non-zero disturbances, the base model is augmented with DE integrators at the outputs which require offset-free control. Consider the discrete-time state-space model

$$\begin{aligned}&\mathbf{x}(k+1)=\mathbf{A}\mathbf{x}(k)+\mathbf{B}\mathbf{u}(k)+\mathbf{w}(k),\\&\mathbf{y}(k)=\mathbf{C}\mathbf{x}(k)+\mathbf{v}(k)\end{aligned} \tag{2}$$

where $\mathbf{w}$ and $\mathbf{v}$ are white noise signals to the state and output, respectively. DE integrator states $\mathbf{d}$ are appended to the state $\mathbf{x}$, so that the augmented state is $\mathbf{x}_a = [\mathbf{x}^T\ \mathbf{d}^T]^T$. A white-noise disturbance signal $\mathbf{w}_d$ acts on the state $\mathbf{d}$, so that the augmented state disturbance signal is $\mathbf{w}_a = [\mathbf{w}^T\ \mathbf{w}_d^T]^T$. Hence, the augmented system is

$$\begin{aligned}&\begin{bmatrix}\mathbf{x}(k+1)\\ \mathbf{d}(k+1)\end{bmatrix}=\begin{bmatrix}\mathbf{A}&\mathbf{0}\\ \mathbf{0}&\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ \mathbf{d}(k)\end{bmatrix}+\begin{bmatrix}\mathbf{B}\\ \mathbf{0}\end{bmatrix}u(k)+\begin{bmatrix}\mathbf{I}&\mathbf{0}\\ \mathbf{0}&\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{w}(k)\\ \mathbf{w}_d(k)\end{bmatrix}\\&\qquad\mathbf{y}(k)=\begin{bmatrix}\mathbf{C}&\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ \mathbf{d}(k)\end{bmatrix}+\mathbf{v}(k)\end{aligned} \tag{3}$$

and is rewritten as

$$\begin{aligned}&\mathbf{x}_a(k+1)=\mathbf{A}_a\mathbf{x}_a(k)+\mathbf{B}_a\mathbf{u}_a(k)+\mathbf{w}_a(k),\\&\mathbf{y}(k)=\mathbf{C}_a\mathbf{x}_a(k)+\mathbf{v}(k)\end{aligned} \tag{4}$$

The steady-state Kalman filter (KF)

$$\begin{aligned}&\mathbf{x}_a(k|k-1)=\mathbf{A}_a\mathbf{x}_a(k-1|k-1)+\mathbf{B}_a\mathbf{u}(k-1)\\&\mathbf{x}_a(k|k)=\mathbf{x}_a(k|k-1)+\mathbf{M}_\mathbf{K}\left[\mathbf{y}(k)-\mathbf{C}_a\mathbf{x}_a(k|k-1)\right]\end{aligned} \tag{5}$$

is used for state estimation with the disturbance-augmented model, where $\mathbf{M}_K$ is computed via the steady-state solution of the Riccati equation from the covariance matrices $\mathbf{Q}_K = \mathrm{E}\{\mathbf{w}_a\mathbf{w}_a^T\}$ and $\mathbf{R}_K = \mathrm{E}\{\mathbf{v}\mathbf{v}^T\}$. The KF is used in the sense of an observer, where the diagonal elements of $\mathbf{Q}_K$ and $\mathbf{R}_K$ are used as tuning parameters to achieve desired dynamics.

- Another form of model augmentation is the *velocity form* (possibly combined with tracking), which is used to prevent offset due to the control cost when the control signal is non-zero at the steady-state.
  For the velocity form, the disturbance-augmented system $\{\mathbf{A}_a,\mathbf{B}_a,\mathbf{C}_a,\mathbf{0}\}$ is further augmented. In the velocity-augmentation, the input signal variation δ**u** becomes the new input; therfore the state expands to $\mathbf{x}_{av} = [\mathbf{x}_a^T\ \mathbf{u}(k-1)^T]^T$, and the new output is $\mathbf{y}_{av} = [\mathbf{y}_a^T\ \mathbf{u}(k-1)^T]^T$

$$\begin{aligned}&\begin{bmatrix}\mathbf{x}_a(k+1)\\ \mathbf{u}(k)\end{bmatrix}=\begin{bmatrix}\mathbf{A}_a&\mathbf{B}_a\\ \mathbf{0}&\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{x}_a(k)\\ \mathbf{u}(k-1)\end{bmatrix}+\begin{bmatrix}\mathbf{B}_a\\ \mathbf{0}\end{bmatrix}\delta\mathbf{u}(k)\\&\begin{bmatrix}\mathbf{y}_a(k)\\ \mathbf{u}(k-1)\end{bmatrix}=\begin{bmatrix}\mathbf{C}_a&\mathbf{D}_a\\ \mathbf{0}&\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{x}_a(k)\\ \mathbf{u}(k-1)\end{bmatrix}+\begin{bmatrix}\mathbf{D}_a\\ \mathbf{0}\end{bmatrix}\delta\mathbf{u}(k)\end{aligned} \tag{6}$$

  where $\mathbf{D}_a = \mathbf{0}$. The system (6) can be rewritten as

$$\begin{aligned}&\mathbf{x}_{av}(k+1)=\mathbf{A}_{av}\mathbf{x}_{av}(k)+\mathbf{B}_{av}\delta\mathbf{u}(k)\\&\mathbf{y}_{av}(k)=\mathbf{C}_{av}\mathbf{x}_{av}(k)\end{aligned} \tag{7}$$

  Then, the MPC controller is built using the Multi-Parametric Toobox (MPT) [4] in the output-cost formulation, with the cost function

$$J(\delta\tilde{\mathbf{u}}) = \sum_{k=0}^{N-1} \mathbf{y}_{av,k}^{T} \begin{bmatrix} \mathbf{Q}_{C;y} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{y}_{av,k} + \delta\mathbf{u}_k^T \mathbf{R}_{C,\delta u} \delta\mathbf{u}_k \tag{8}$$

where the diagonal elements of the cost matrices for the outputs $\mathbf{Q}_{C,y}$ and the control moves $\mathbf{R}_{C,\delta u}$ are used as tuning parameters, and $N = 30$ is the prediction horizon length. The control law is obtained by minimising J with respect to the vector of the future control moves $\delta\tilde{\mathbf{u}}$ subject to constraints (currently, only control amplitude constraints $\mathbf{u}_{\min} \le \mathbf{u} \le \mathbf{u}_{\max}$ are used). To reduce the computational demand, the number of free control moves is reduced from 30 to 3 using the "move blocking" technique to three intervals of lengths [2 2 26]. Due to the finite horizon length, the control law is computed as a least-squares problem in the unconstrained case, or as a quadratic programming problem in the constrained case [13].

4.2 Sample time

For discrete-time control algorithms, including the majority of practical forms of MPC, a suitable sample time T_s must be chosen. Controllers may be designed with a relatively wide range of T_s, and different rules of thumb are practiced.

For instance, one may choose T_s to be around a tenth of the settling time of the open-loop system, so that combined with the choice $N = 10$ the prediction horizon roughly covers the open-loop settling time. Such choice tends to result in relatively long sampling times; nevertheless, the resulting nominal closed-loop response to set-point changes may be quite comparable with conventional controllers based on continuous-time design (e.g., PID).

In practical control systems, the controllers are often expected to suppress fast-acting disturbances. The plasma CSC may be designed with $T_s = 1$ s or more, however its response to disturbances will appear sluggish in case when disturbances are not well aligned with sample hits. By using a shorter sample time, the disturbance response generally improves, but eventually a plateau is reached due to system dynamics. For the plasma system it was observed that T_s around 0.1 s is needed to avoid a negative impact of sampling on disturbance response.

Generally it is possible to implement discrete-time controllers with faster sampling times. At much faster sampling times, numerical issues due to finite-accuracy mathematic operations may be encountered. With MPC controllers that feature handling of constraints, the computation time required for on-line optimization is an obvious obstacle. Another issue is that for a meaningful definition of the cost function, the prediction horizon should cover a certain time period, therefore shortening of T_s requires an increase of N (even infinite-horizon MPC formulations which do not need a high N for a meaningful cost function formulation still require it for a timely response to anticipated constraints violations).

4.3 KF tuning

The KF is tuned by specifying the covariance matrices $\mathbf{Q}_K$ and $\mathbf{R}_K$. In theory, these two matrices may be tuned by estimating noise covariances $\mathrm{E}\{\mathbf{w}_a\mathbf{w}_a^T\}$ and $\mathrm{E}\{\mathbf{v}\mathbf{v}^T\}$, respectively. However this is infeasible with a not-yet-existing system. Another issue is that such an optimal filtering approach leads to filtering which is mathematically optimal with respect to the defined quadratic norm and the type of noise present in the system, however the impact of this optimality on actual control requirements are unclear. Furthermore, it is well known that the design of an optimal filter (KF) and its dual LQ optimal controller leads to favourable properties of each and good overall performance as long as the certainty equivalence assumption holds; however the composite system may not be robust to modelling error. Loop Transfer Recovery (LTR) is one known approach to overcome the robustness issue by either matching KF tuning to the LQ controller or vice versa; however, it is not clear how to apply it with augmented models. Therefore, independent tuning of the KF is possible only at the initial stage, while the final phase is carried out together with tuning the controller, where cross-validation on a set of different actual models is essential.

In practice, manual tuning is often used, using identity matrices or more general diagonal matrices. Simple identity matrices lack tuning flexibility, while considering diagonal elements individually results in an excessive number of tuning parameters. A compromise may be achieved by considering the structure of the state and output vectors and blocking related parameters together. The output vector is known to comprise elements of $\mathbf{I}_{PF}$, I_p and $\mathbf{g}$ which have different properties and value ranges. The state vector also has a particular structure when using the modal form. The last 11 elements are inherent integrating dynamics, while the others represent transient stable dynamics; the appended modes are integrating disturbance modes at the respective outputs. So, one possible structure of the covariance matrices is

$$\mathbf{Q}_K = \begin{bmatrix} k_3\mathbf{C}\mathbf{C}^T & & & \\ & k_4\mathbf{I}_{11} & & \\ & & k_5\mathbf{Z}_{12} & \\ & & & k_6\mathbf{Z}_{13:41} \end{bmatrix}, \qquad \mathbf{R}_K = \begin{bmatrix} \mathbf{Z}_{1:11} & & \\ & k_1\mathbf{Z}_{12} & \\ & & k_2\mathbf{Z}_{13:41} \end{bmatrix} \tag{9}$$

with 6 tuning parameters k_1 ... k_6, where $\mathbf{Z} = \text{diag}(\mathbf{B}^T\mathbf{B})$. Using the values $k_1 = 1$, $k_2 = 1$, $k_3 = 10^{-1}$, $k_4 = 10^6$, $k_5 = 10^2$, $k_6 = 10^2$, the output estimates shown in Figs. 7 - 11 were obtained (dotted lines in displacement values). The simulations of a "Minor disruption" event were carried out with a variant of the control scheme with CSC in open loop, with the MPC controller removed as shown in Fig. 6, with only the VS control loop closed.

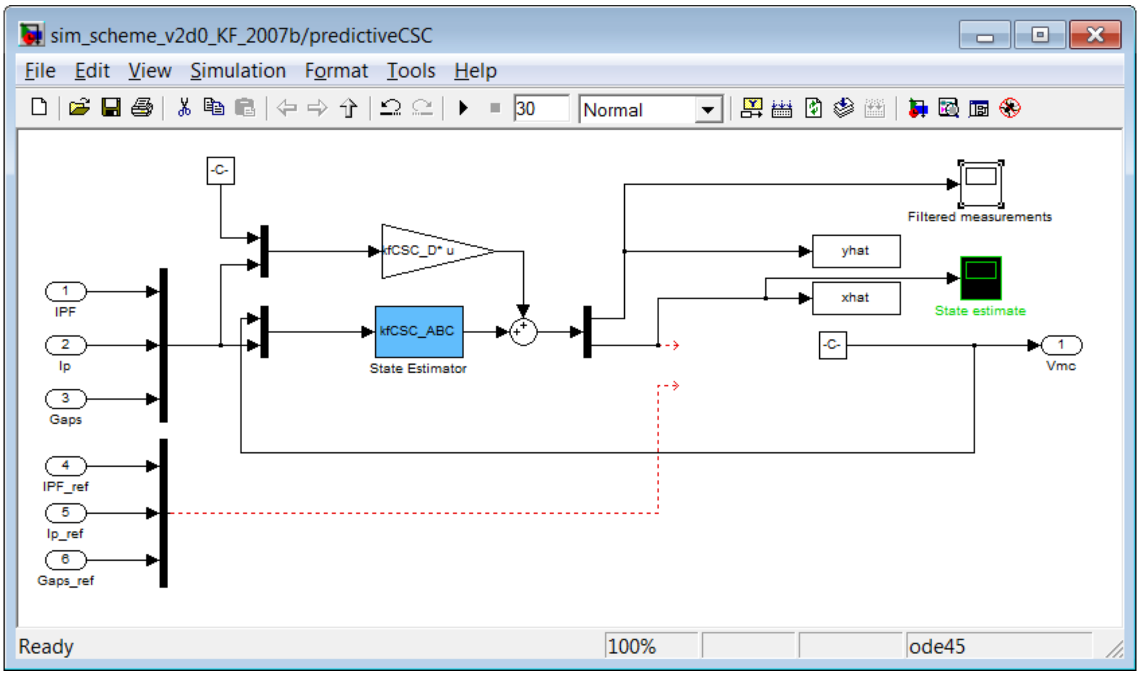


Fig. 6. Block "predictiveCSC" with KF state estimation only (no MPC controller).

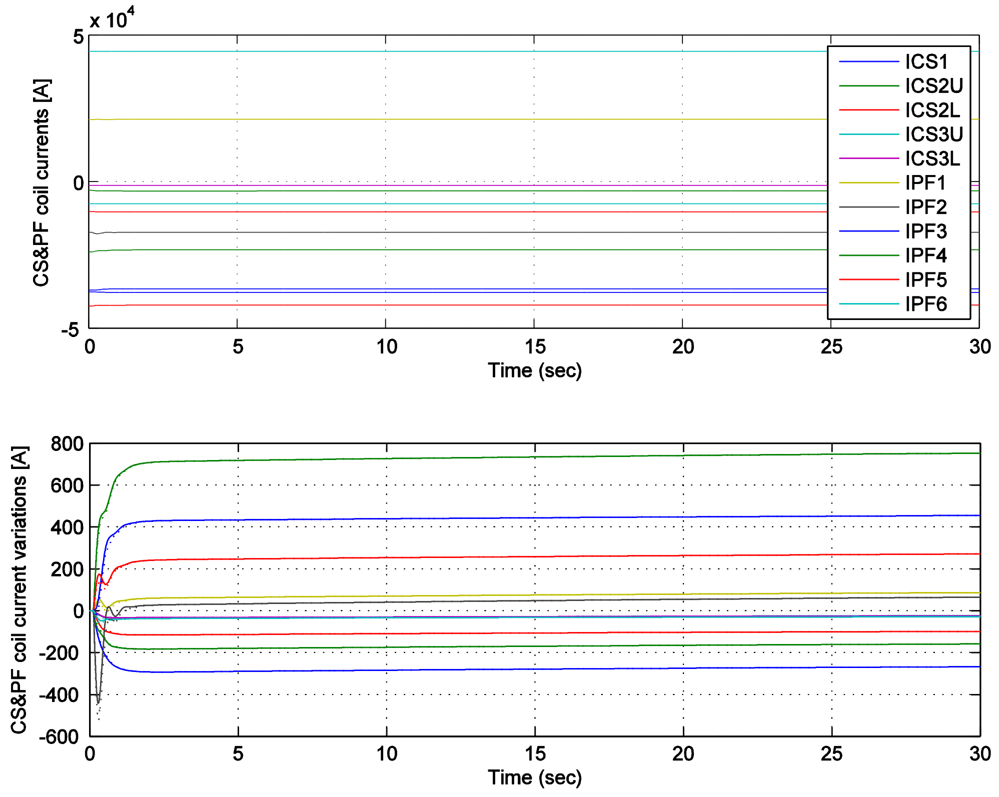


Fig. 7. Minor disruption simulation: SC&PF coil currents. Estimates are drawn with dotted lines on the lower graph only (displacement values).

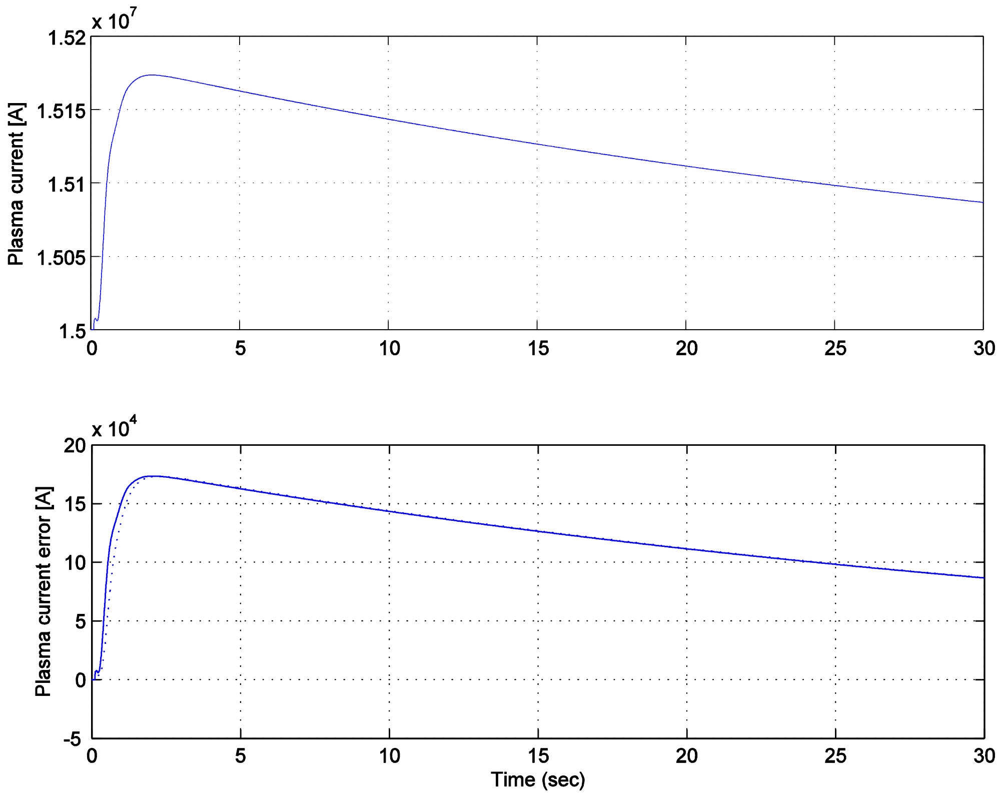
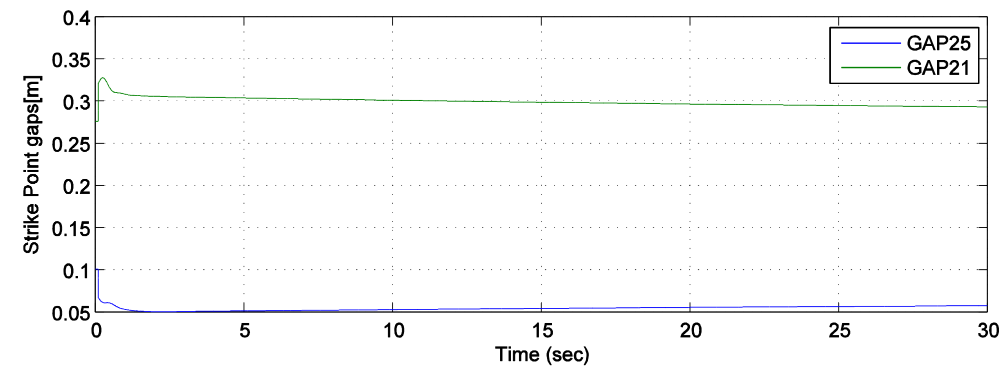


Fig. 8. Minor disruption simulation: Plasma current. Estimates are drawn with dotted line on the lower graph only (displacement values).



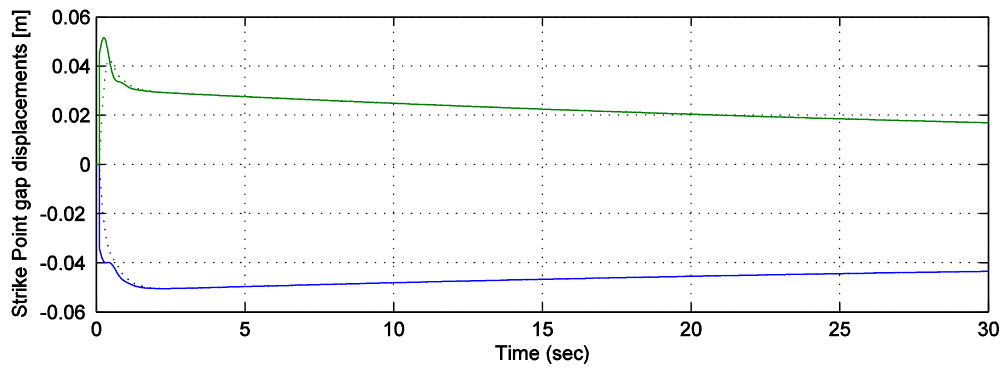
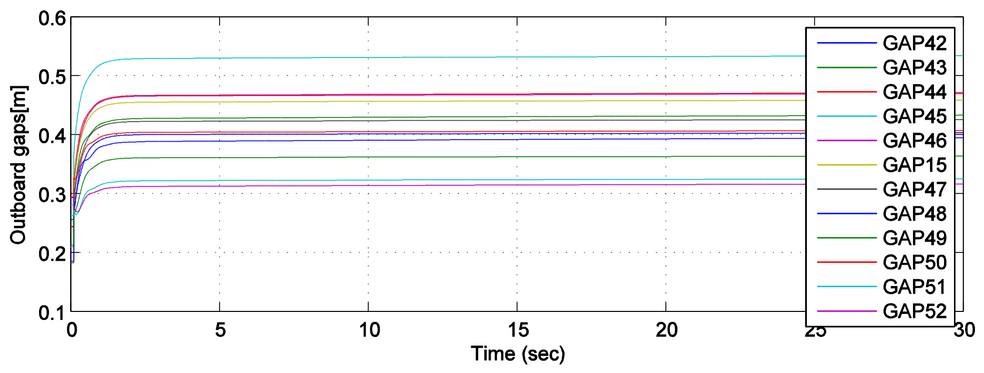


Fig. 9. Minor disruption simulation: strike points. Estimates are drawn with dotted lines on the lower graph only (displacement values).



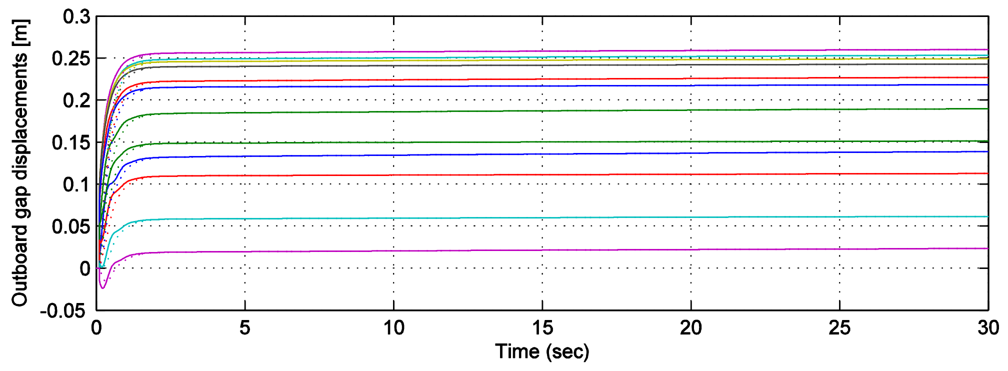


Fig. 10. Minor disruption simulation: Outboard gaps. Estimates are drawn with dotted lines on the lower graph only (displacement values).

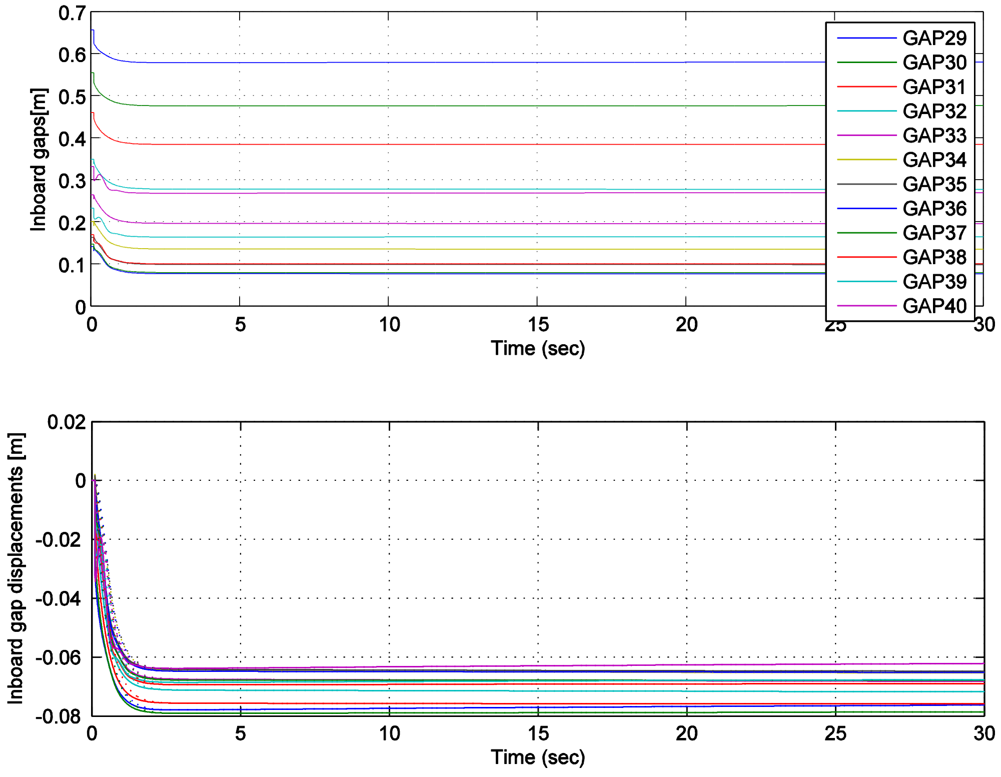


Fig. 11. Minor disruption simulation: Inboard gaps. Estimates are drawn with dotted lines on the lower graph only (displacement values).

A very similar approach is used when tuning MPC controller cost matrices $\mathbf{Q}_{C,y}$ and $\mathbf{R}_{C,u}$.

4.4 MPC CSC controller variants

Several variants of the MPC controller have been investigated, with the aim of finding a scheme with desirable control performance and suitable for numerically efficient controller implementation.

4.4.1 Initial MPC CSC prototype

An initial MPC CSC prototype scheme was available at the start of the project already [33]. In this scheme the controller regulates the signal deviations from the nominal operating point to zero, without support for set-point tracking. It is made for a fixed selection of 6 pre-selected gaps in the vector of geometrical descriptors **g** (specifically, 4 gaps and 2 strike-points). A different set of CREATE plasma models, with respect to the one introduced in Table 1, were used in the design.

This prototype without further modifications, except for complexity reduction techniques, was being used in the parallel effort of examining candidate fast online QP solvers.

4.4.2 MPC CSC with full output vector

The basic version of the new MPC controller concept (based on the new set of plasma models, designed for a scheme with "absolute" signals, and with set-point tracking support) as described in Section 4, uses the full output vector **y** including the vector **g** with 29 geometrical descriptors. While it is possible to design a controller with many more controlled outputs than manipulated variables, it is known that offset-free performance in the steady state is not possible and that tuning of such controller is difficult, as the offsets determined by minimizing the cost functions may not coincide with practically sensible performance – it is difficult to tune trade-offs between different offsets and constraint violations. In addition, there are numerical issues in the practical design of such a controller with some MPC design libraries (MPT Toolbox) due to the large matrices involved.

4.4.3 MPC CSC with the output vector reduced via manual selection

The simplest form of output vector reduction is achieved by introducing an output selection matrix $\mathbf{M}_{\mathrm{sel}}$ (containing mostly zeros, and n_g elements equal to 1, one in each row) which maps a certain number of gaps n_g to from $\mathbf{g}$ to the vector of selected gaps $\mathbf{g}_{\mathrm{sel}}$

$$\mathbf{g}_{\mathrm{sel}} = \mathbf{M}_{\mathrm{sel}}\mathbf{g} \qquad (10)$$

Subsequently, the controller is designed for the output vector $\mathbf{y}_{\mathrm{CSCsel}} = [\mathbf{I}_{\mathrm{PF}}\ I_p\ \mathbf{g}_{\mathrm{sel}}]^{\mathrm{T}}$ instead of the original output vector $\mathbf{y}_{\mathrm{CSC}} = [\mathbf{I}_{\mathrm{PF}}\ I_p\ \mathbf{g}]^{\mathrm{T}}$. In the base model $\{\mathbf{A}_{CSC},\mathbf{B}_{CSC},\mathbf{C}_{CSC},\mathbf{0}\}$, the matrix $\mathbf{C}_{\mathrm{CSC}}$ is replaced with a reduced matrix $\mathbf{C}_{\mathrm{CSCsel}}$, which only retains rows corresponding to the gaps included in $\mathbf{g}_{\mathrm{sel}}$. The modified controller block is shown in Fig. 12, where $\mathbf{M}_{\mathrm{sel}}$ is used in the gain blocks "gtosvd".

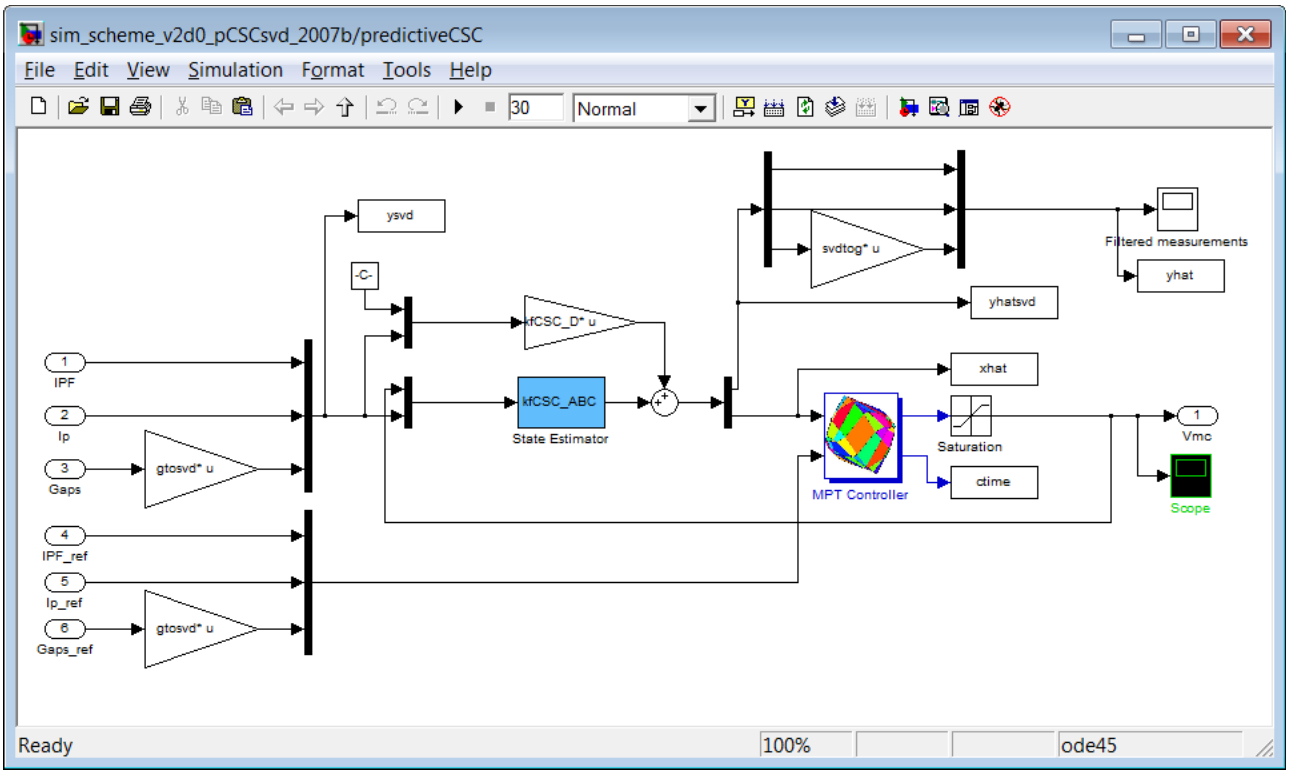


Fig. 12. Block "predictiveCSC" with output vector reduction.

If the choice of the gaps is the same as in the initial scheme from Section 4.4.1, very similar performance may be obtained.

4.4.4 MPC CSC with the output vector reduced via manual selection and averaging

A generalization of the simple concept from the previous section is made by replacing individual gaps with weighted sums (averages) of neighbouring gaps. With this approach, a much larger number of gaps are being considered by the controller, despite the controller having a smaller number of virtual outputs. For instance, the vector $\mathbf{g}_{\mathrm{sel}}$ and the matrix $\mathbf{M}_{\mathrm{sel}}$ can be generated by averaging signals using the following list

```
gsel{1} = 1:12;   % inboard gaps
gsel{2} = 13:15;  % top gaps
gsel{3} = 16:19;  % top outboard gaps
gsel{4} = 20:27;  % bottom outboard gaps
gsel{5} = 28;     % strike point GAP25
gsel{6} = 29;     % strike point GAP21
```

Here, in the base model $\{\mathbf{A}_{CSC}, \mathbf{B}_{CSC}, \mathbf{C}_{CSC}, \mathbf{0}\}$, the matrix $\mathbf{C}_{CSC}$ is replaced with a reduced matrix $\mathbf{C}_{CSC}$, in which the rows corresponding to the averaged signals are obtained by averaging the corresponding rows of $\mathbf{C}_{CSC}$. The controller simulation block in Fig. 12 is used.

Figures 13-17 show the resulting simulation response to Minor disruption; tuning is provisional, because the controller concept is not final yet.

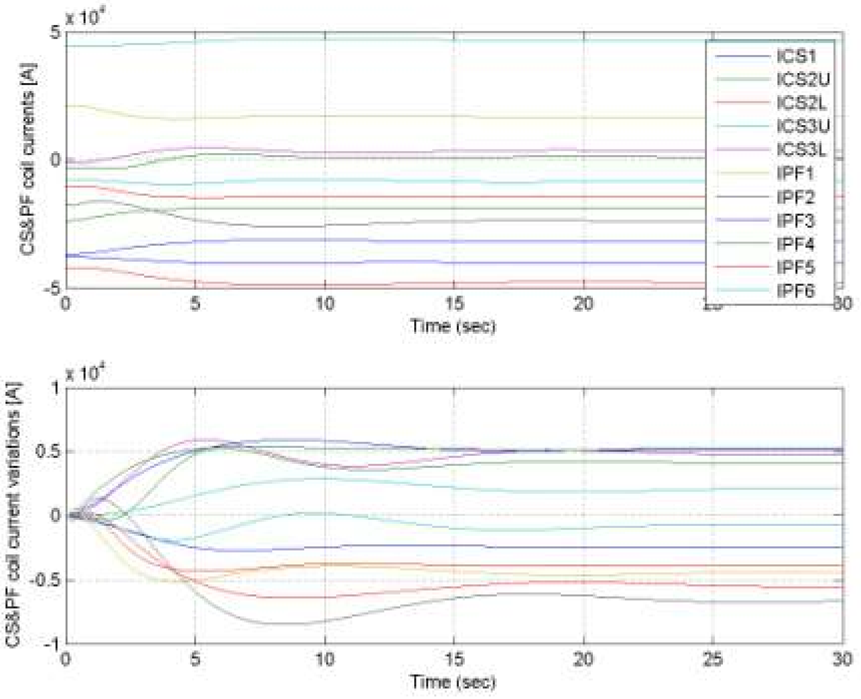


Fig. 13. Minor disruption simulation: SC&PF coil currents.

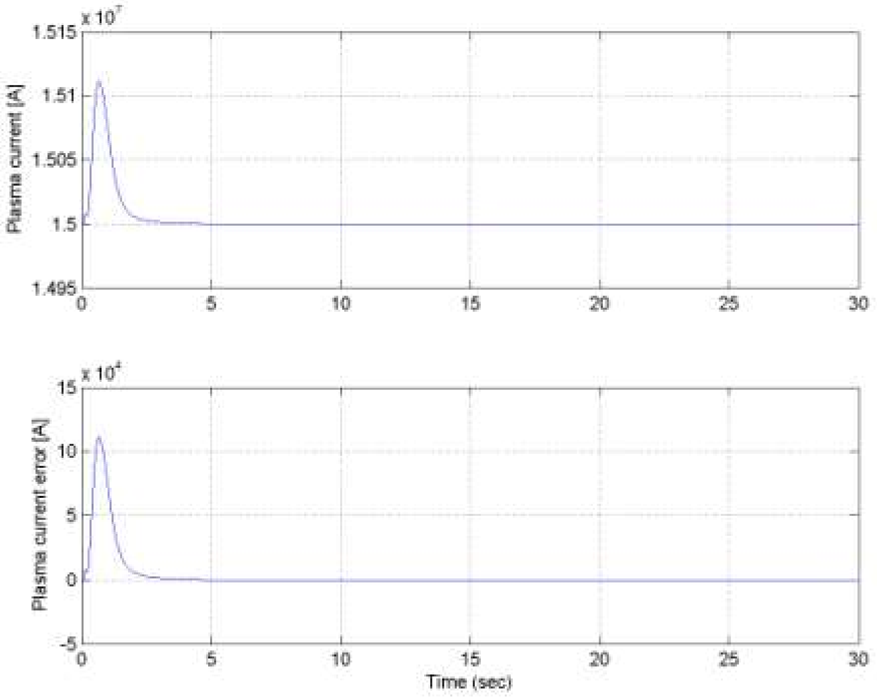


Fig. 14. Minor disruption simulation: Plasma current.

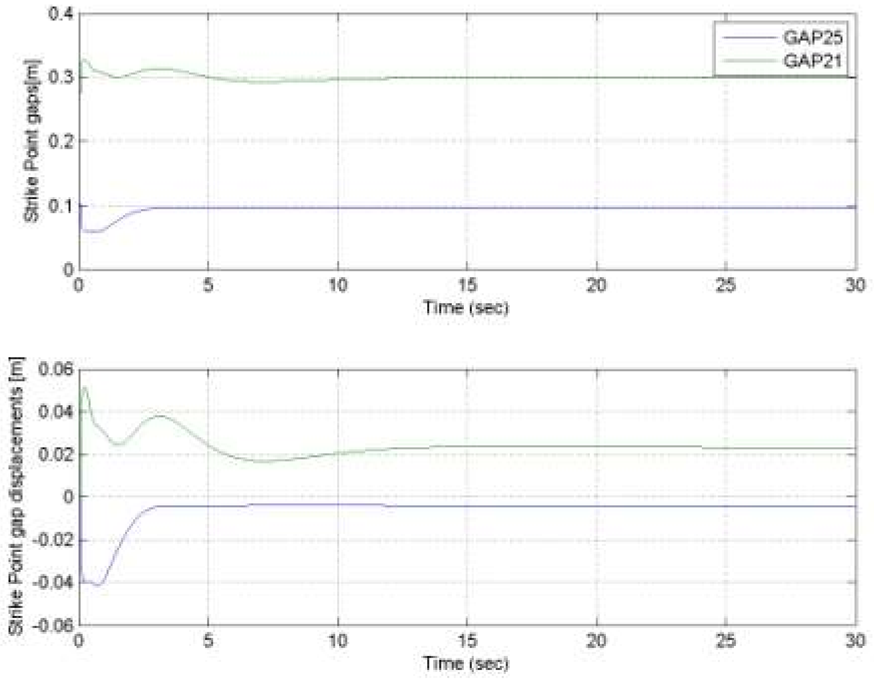


Fig. 15. Minor disruption simulation: strike points.

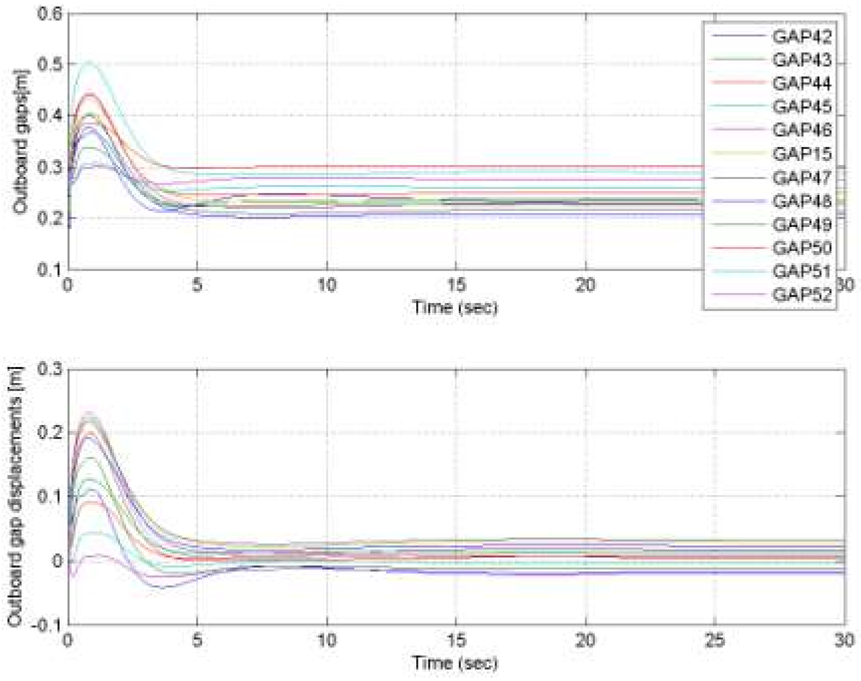


Fig. 16. Minor disruption simulation: Outboard gaps.

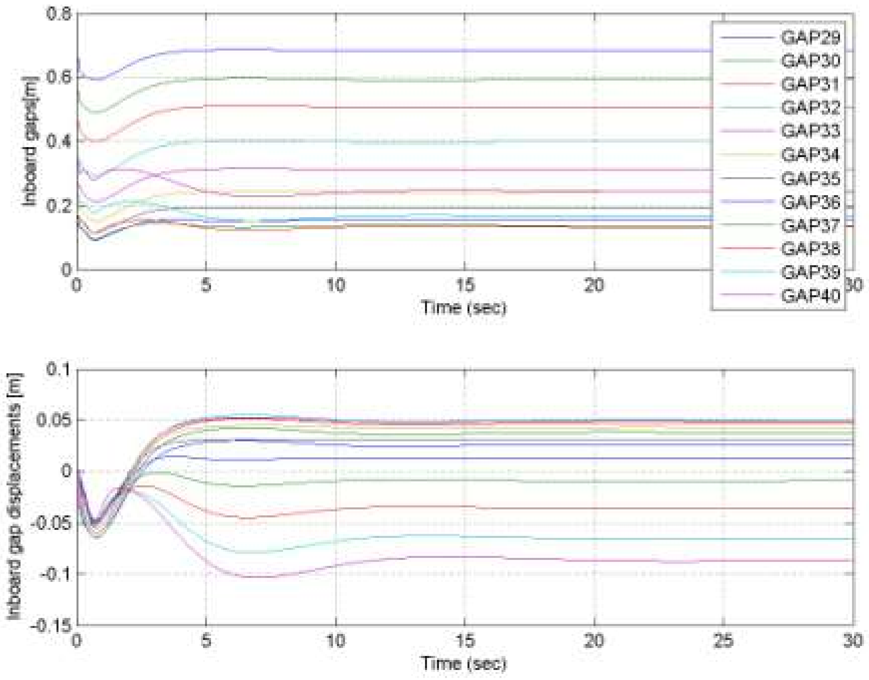


Fig. 17. Minor disruption simulation: Inboard gaps.

4.4.5 MPC CSC with SVD-based output space reduction

In this scheme, the idea is to apply singular values decomposition (SVD) to $\mathbf{C}_\mathbf{g}$, the part of the output matrix $\mathbf{C}_{CSC}$ producing the geometrical descriptors $\mathbf{g}$

$$\mathbf{C}_{\mathbf{g}} = \mathbf{U}_0\mathbf{S}_0\mathbf{V}_0^T \tag{11}$$

Then, a truncated SVD approximation using the first n_g singular values

$$\mathbf{C}_{\mathbf{g}1} = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^T \tag{12}$$

and an artificial output vector $\mathbf{g}_{\mathrm{svd}}$ with reduced dimension n_g is introduced

$$\mathbf{g} = \mathbf{U}_1\mathbf{S}_1\mathbf{g}_{\mathrm{svd}}, \quad \mathbf{g}_{\mathrm{svd}} = \mathbf{V}_1^T\mathbf{x} \tag{13}$$

The controller is designed for the output vector $\mathbf{y}_{\mathrm{CSCsvd}} = [\mathbf{I}_{\mathrm{PF}}\ I_p\ \mathbf{g}_{\mathrm{svd}}]^{\mathrm{T}}$ instead of the original output vector $\mathbf{y}_{\mathrm{CSC}} = [\mathbf{I}_{\mathrm{PF}}\ I_p\ \mathbf{g}]^{\mathrm{T}}$. The simulation scheme in Fig. 12 is used.

The matrix $\mathbf{U}_1\mathbf{S}_1$ can be used for computing estimates of $\mathbf{g}$ estimates from KF estimates of $\mathbf{g}_{\mathrm{svd}}$, which cannot be done in selection/averaging schemes. The reverse mapping, in place of $\mathbf{M}_{\mathrm{sel}}$, is

$$\mathbf{g}_{\mathrm{svd}} = \mathbf{S}_1^{-1}\left(\mathbf{U}_1^T\mathbf{U}_1\right)^{-1}\mathbf{U}_1^T\mathbf{g} \tag{14}$$

Fig. 18 displays surface plots of elements of the SVD approximation matrix $\mathbf{C}_{\mathbf{g}1}$ (left), and the difference ($\mathbf{C}_{\mathbf{g}}$ - $\mathbf{C}_{\mathbf{g}1}$) (right) using $n_g = 6$. It is evident that a rather rough approximation is obtained. A simulation of the scheme containing the KF only, without the CSC controller, shows considerable offsets in **g** estimates, as shown in Fig. 19. These offsets can be reduced only by using a considerably higher n_g (25 or higher), as shown in Fig. 20, but in this case erratic performance of the KF was noticed at fast initial transients, and the desired degree of reduction is not achieved. This indicates that SVD cannot be applied for dimension reduction of the output vector when considering system dynamics. However, it may still be possible to use SVD to reduce the output dimension in a static manner, considering only the steady-state relation between $\mathbf{I}_{\mathrm{PF}}$ and **g** as in [11].

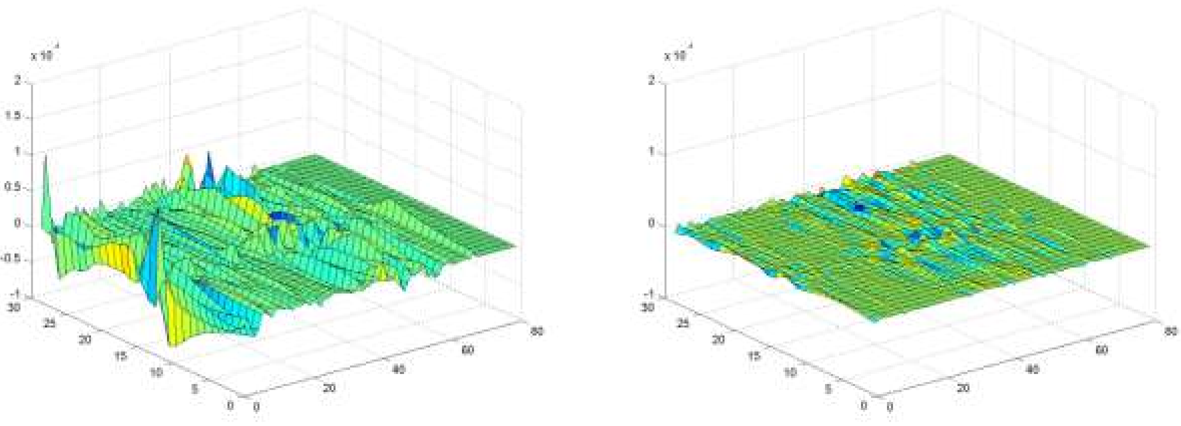


Fig. 18. Surface plots of elements of $\mathbf{C}_{\mathbf{g}1}$(left), and the difference ($\mathbf{C}_{\mathbf{g}}$ - $\mathbf{C}_{\mathbf{g}1}$) (right), using $n_g = 6$

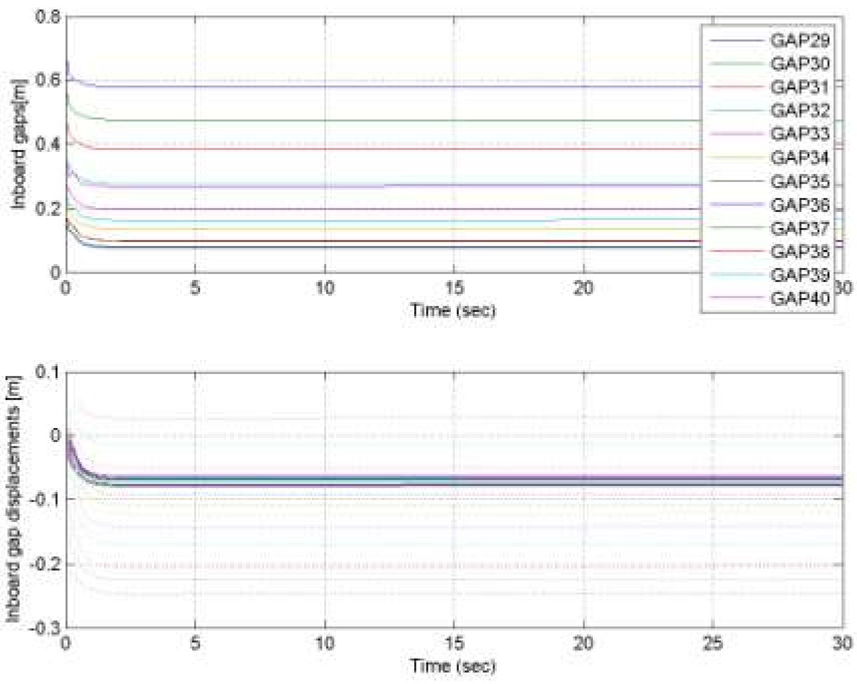


Fig. 19. Minor disruption simulation with no CSC, KF only: Inboard gaps. Estimates are drawn with dotted lines on the lower graph only (displacement values). SVD-based output space reduction, $n_g = 6$

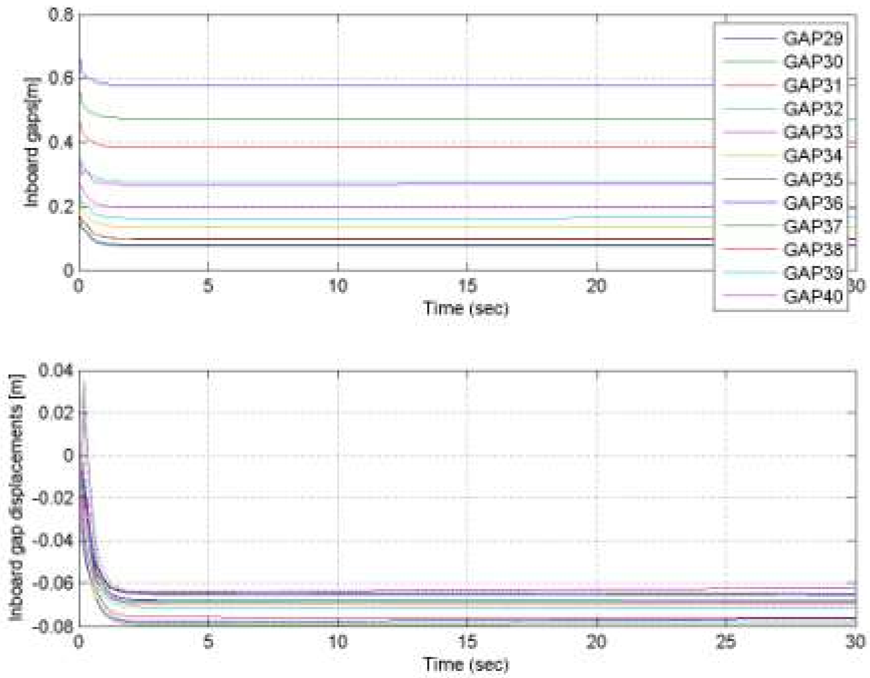


Fig. 20. Minor disruption simulation with no CSC, KF only: Inboard gaps. Estimates are drawn with dotted lines on the lower graph only (displacement values). SVD-based output space reduction, $n_g = 25$

Part II: MPC implementation using fast online optimization

5 Fast online MPC methods

5.1 MPC and QP

Model Predictive Control (MPC) is a control methodology in which future control actions are determined by optimization of a performance criterion defined over a future horizon in which control signals are predicted using a dynamic process model. It is related to Linear Quadratic optimal control (LQ), in particular to Constrained LQ optimal control. It is able to handle constraints on process signals over a finite horizon.

Consider a linear dynamic system

$$x_{k+1} = Ax_k + Bu_k, \quad y_k = Cx_k, \tag{15}$$

where x_k is system state at time k, u_k is input, and y_k is output, and $A, \ B, \ C$ are system dynamics matrices. By adding cost function

$$J = \sum_{j=0}^{N-1} \left(x_{k+j|k}^T Q_x x_{k+j|k} + u_{k+j|k}^T R_u u_{k+j|k} \right) + x_{k+N|k}^T Q_{xN} x_{k+N|k} \tag{16}$$

and constraints

$$u_{min} \le u_k \le u_{max}, \quad x_{min} \le x_k \le x_{max}, \quad y_{min} \le y_k \le y_{max}, \tag{17}$$

the MPC problem is formed. Notice that the cost function *J* is not a convenient practical measure of good control performance, but rather a mathematically convenient simple norm.

The problem that needs to be solved to obtain the control input value in each time-step of MPC is a quadratic programming (QP) problem in structure. The optimization variable z may contain only elements of *u* ("condensed form"), or possibly also *x* and *y* ("structured form"). The MPC problem may be restated as a QP problem

$$z^* = argmin\left(z^T Hz + g^T z\right) \tag{18}$$

subject to

$Az = b$ (equality constraints that follow from dynamical system dynamics) (19)

$z_{min} \le Zz \le z_{max}$ (inequality constraints that follow from dynamical system constraints) (20)

Without inequality constraints, the problem may be solved analytically via the least-squares method. Various solution methods thus mostly differ in the approach they take with respect to inequality constraints.

5.2 Multiparametric or explicit method

Fast online MPC can be carried out using a pre-calculated solution of the QP [15].

The vector *b* of the equality constraints equation depends linearly on the initial state *x*. The inequality constraints of the QP for which equality holds are called active. The set of active constraints depends on *b*. It is shown that for a given set of inequality constraints to be active, x belongs to a polyhedral subset of the state space. Within such a polyhedron, one in effect only has equality constraints, which means the problem is easy to solve and the transformation from x to z^* is affine.

In principle, one can build the list of all the polyhedra and corresponding affine transformations in advance. Online, one only has to find the polyhedron x is in, for instance using a binary search tree, and transform x into the exact solution z^* using the corresponding affine transformation.

The method is practically useful only for small problems because of the combinatorial explosion of the number of possible combinations of active constraints. For problems with the size of the one considered for plasma current and shape control, the memory required to store the solution and the time required to find the polyhedron x is in, make the method impractical [16, Ch. 12, 13, Preface]. Its use for CSC can thus be excluded.

5.3 Fast online solvers

Fast online MPC can also be performed using a fast QP solver. Traditionally, using general-purpose online QP solvers (for instance, CPLEX [31] or NAG [32]) for real-time MPC was considered feasible only for processes with slow dynamics, with sampling time in the order of minutes or hours.

Recently, a number of solvers have been designed specifically for solving QP problems for MPC. They take into account that MPC requires repeated solutions of very similar QP problems of relatively small dimensions. Online computation time required can be shortened considerably by using separate offline pre-processing, optimized solver core, and code-generation approach, which may produce optimized solver core code for different MPC configurations and target platforms. They can utilize the fact that H, A, and Z are known in advance. Matrix products, decompositions, and/or algorithm weights are calculated offline; the code is optimized for calculating z^* as quickly as possible after x is known and b can be obtained.

Fast online QP solvers for MPC have been developed based on all traditional types of QP solvers, as described below.

5.3.1 Active set methods

When the set of active inequality constraints for a given x is known, z^* is easy to calculate. To find the active set, efficient iterative methods that avoid checking all the possible combinations of active constraints exist. In each step, a constraint is either added to or removed from the working set. The number of iterations needed tends to be quite small in practice; however, no active set method is certified to reach the correct solution in polynomial time [16, part 4.3.3].

qpOASES is an open-source fast online active set solver written in C++ [22, 23, 24]. This solver was tested with the oscillating masses benchmark example [38].

5.3.2 Interior point methods

Primal barrier methods convert the constrained optimization problem into an unconstrained problem (with respect to inequalities) by means of a barrier function [16, part 4.3.2]. The barrier function that is added to the cost function is infinite outside of the feasible set defined by inequality constraints, finite inside the feasible set, and twice continuously differentiable. For solving the problem obtained this way, an unconstrained minimization method is used. The barrier function is then iteratively decreased (multiplied by a factor smaller than 1) and the new problem is solved until the solution is close to the solution of the original problem.

The general idea of primal-dual interior point methods is to solve Karush-Kuhn-Tucker (KKT) conditions with a modified version of Newton's method [16, part 4.3.2]. The complementary slackness condition is relaxed [17, part 11.7] and the relaxation parameter is lowered through successive iterations.

In [25], a fast primal barrier method is introduced and applied to oscillating masses benchmark example, and in [26, 27], it is further developed into CVXGEN code generator. It has been tested with a pre-ordering and an energy storage example [39]. In [28], GPU is used for solving the problem. FORCES [29] is a primal-dual interior point solver, and HPMPC includes an interior point method implemented in C [30]. The last two solvers have been tested with the oscillating masses benchmark example.

5.3.3 First order methods

First order methods are iterative methods in which an iteration consists of a gradient step and projection to feasible domain [16, part 4.3.1]. First order methods are formally general QP solvers; however, the complexity of the projection operation depends a lot on the types of the constraints, and the convergence may be very slow in case of correlated optimization variables. Therefore, manual customization to specific QP problems is typically required for numerically efficient implementation. The methods being considered are optimized for MPC problems.

With some types of constraints, for instance with simple upper and lower bounds on the actuator signal *u*, the projection is a simple min/max operation. On the other hand, the projection is difficult to carry out efficiently with output inequality constraints of MPC. This leads to projection onto a polyhedron, for which no simple algorithm exists. Such projection may in some cases be pre-computed parametrically [43], or it can even be evaluated with other QP solvers such as CPLEX [43] (where the focus of the study was on another part of the algorithm). More typically, a dual approach is adopted for state/output constraints.

In particular, implementing soft constraints of MPC through the introduction of auxiliary slack variables is suboptimal because it leads to differently shaped constraint polyhedron and slower convergence [18, part 3.2.5]. Other means of taking soft constraints into account have thus been developed [16].

The convergence rate of first order methods is linear or sub-linear [18, part 2.1], which is slow compared to interior point methods. However, it has been shown that they can be efficient for obtaining approximations of relatively low accuracy, sufficient for control (considering the limited accuracy of the actuators and sensors).

Their advantages are easy implementation due to short computer code, short iteration time, and the availability of meaningful complexity certificates [18, 19].

The basis for these methods is the classic gradient method [16, part 4.2.1] and its fast (accelerated) variant. For constrained optimization, it was extended into gradient projection method [16, part 4.2.1]. Gradient and gradient projection methods can be accelerated [16, part 4.2.1] in a similar fashion.

As an example of a first order method, let us analyze the example of generalized fast dual gradient method as described in [20] with the iteration loop in Eqs. (47–50). It is designed to minimize a function of the form

$$f(y) = \frac{1}{2} y^T H. \tag{21}$$

subject to constraints

$$\begin{aligned} Ay &= b\bar{x} \\ By &= v \\ \underline{d} \le v &\le \bar{d}. \end{aligned} \tag{22}$$

We define $h(y)$ to be the indicator function corresponding to equality constraints of the QP $h(y) = I_{Ay=b\bar{x}}$, where the value of the indicator function is 0 in the set defined by the constraints and ∞ outside. Similarly, $g(y) = I_Y, Y = \{y \in {}^n; d \le y \le \bar{d}\}$ The original problem can thus be reformulated as

$$\begin{aligned} &\text{minimize} \quad f(y) + h(y) + g(v) \\ &\text{subject to} \quad By = v. \end{aligned} \tag{23}$$

Dual variables are introduced only for the remaining constraints, and the problem is solved iteratively. In the line (47), KKT system is solved. In (48), generalized dual gradient and projection step is carried out using generalized proximity operator. In contrast with ordinary gradient and projection steps, a Lipschitz matrix is used in place of the scalar Lipschitz constant as a pre-conditioner to improve convergence. The lines (49-50) carry out acceleration. The equations are spelled out as

$$\begin{aligned} y^k &= H^{-1}\left(A^T H_{A^{-1}}\left(AH^{-1}B^T v^k + b\bar{x}\right) - B^T v^k\right) \\ \mu^k &= \operatorname{prox}_g^{L_\mu}\left(v^k + L_\mu B y^k\right) \\ t^{k+1} &= \frac{1 + \sqrt{1 + 4\left(t^k\right)^2}}{2} \\ v^{k+1} &= \mu^k + \left(\frac{t^k - 1}{t^{k+1}}\right)\left(\mu^k - \mu^{k-1}\right). \end{aligned} \tag{24}$$

Another example is ADMM – Alternating Direction Method of Multipliers [17]. It consists of the iterations:

$$\begin{aligned} x^{k+1} &:= \operatorname{argmin}_x L_\rho\left(x, z^k, y^k\right) \\ z^{k+1} &:= \operatorname{argmin}_z L_\rho\left(x^{k+1}, z, y^k\right) \\ y^{k+1} &:= y^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right) \end{aligned} \tag{25}$$

when solving the problem

$$\begin{aligned} &\text{minimize} \quad f(x) + g(z) \\ &\text{subject to} \quad Ax + Bz = c \end{aligned} \tag{26}$$

and

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2. \tag{27}$$

where ρ is a positive parameter.

5.4 Criteria for choosing the method

The method to be used in FMPCFMPC must fulfil the following criteria:

- The solution must be sufficiently accurate;

- The solution must be sufficiently fast;

- The first two criteria must be met as reliably as possible using as simple hardware as possible.

The prototype plasma current and shape controller, used for the evaluation of the QP solvers, uses sample time 100 ms [33]. Our goal is to achieve a much shorter peak calculation time, around 10 ms. With a general-purpose commercial solver CPLEX, the achieved computation times were 30 ms average, 50 ms maximum.

5.5 Fast online MPC solvers

The available open-source QP solvers are being carefully examined for the purpose of plasma magnetic control implementation. While QPgen [34, 20] has already been tested extensively and appended with some additionally required code, the work on FiOrdOs [36] and HPMPC [37] is still in progress.

5.5.1 QPgen

QPgen can generate C code for the alternating direction method of multipliers (ADMM), the fast dual proximal gradient method (FGMdual), and the fast primal proximal gradient method (FGMprimal) to solve the optimization problem instances [34]. It is supplied with AFTI-16 aircraft model case-study, a poorly-conditioned open-loop unstable system with 2 manipulated variables, 2 controlled variables, and 4 states.

The generated C code for the aircraft MPC problem supplied with QPgen and “FGMdual” algorithm has been verified to be equal to the algorithm in the Eqs. (47-50) of [20], upgraded with reference tracking, restarting, and soft constraints.

In the code, KKT system is solved using LDL^T decomposition with pivoting instead of solving the KKT system from line (47) directly, because the original matrix and the decomposition matrices are sparse so that this method requires fewer multiplications and is thus faster. Matrices are calculated in advance and hard-coded into the solver. In 8 lines of code, the right hand side vector is constructed, two triangular systems with permutation of the right hand side vector elements are solved, one multiplication of a vector with a diagonal matrix is carried out, and the primal solution is extracted.

To perform a generalized proximity operation as required in the line (48), the only modification needed to the procedure for ordinary proximity operation is a change in the matrix *B*. It is multiplied with $L_\mu^{\frac{1}{2}}$, where L_μ is Lipschitz matrix. It results in differently scaled Lagrange multiplier but as the “new” *B* is also used in vector construction for KKT system, it cancels out. It is known that the sum of proximity operator of a conjugate to a function and proximity operator of the original function of the same argument equals the argument of the proximity functions: $\operatorname{prox}_g(x)+\operatorname{prox}_{g^*}(x)=x$ (Moreau decomposition, Rockafellar, 1970, Theorem 31.5, in [17]). This method is used in the code, as the proximity operation of the original indicator function of a box constraint is projection onto the box. Actually, soft clipping as in [18] is used instead if soft constraints are desired. In 5 lines of code, the argument of the proximity operator is calculated, the projection (or soft clipping) is done and the difference is returned. L_μ and the “new” *B* are calculated offline and are included as a part of the code.

The acceleration part (49–50) is trivial.

If the difference between the current accelerated dual variable and the previous one, and the difference between the current one and the previous non-accelerated one, have a positive scalar product, the algorithm is restarted with all the dual variables reset to the old accelerated one's value. Such restarting improves the convergence rate [16, part 4.2.1; 60].

5.5.2 FiOrdOs

The FiOrdOs toolbox implements code generators for gradient method and fast gradient method [36]. For both methods, also a variant with adaptive step size is available. Primal, dual, and primal-dual approaches are available. We have implemented AFTI-16 aircraft model and analyzed its convergence. We followed both sizes of iteration steps and differences between the correct solution (given by `quadprog` function from the Matlab optimization Toolbox) and FiOrdOs solution at a given iteration.

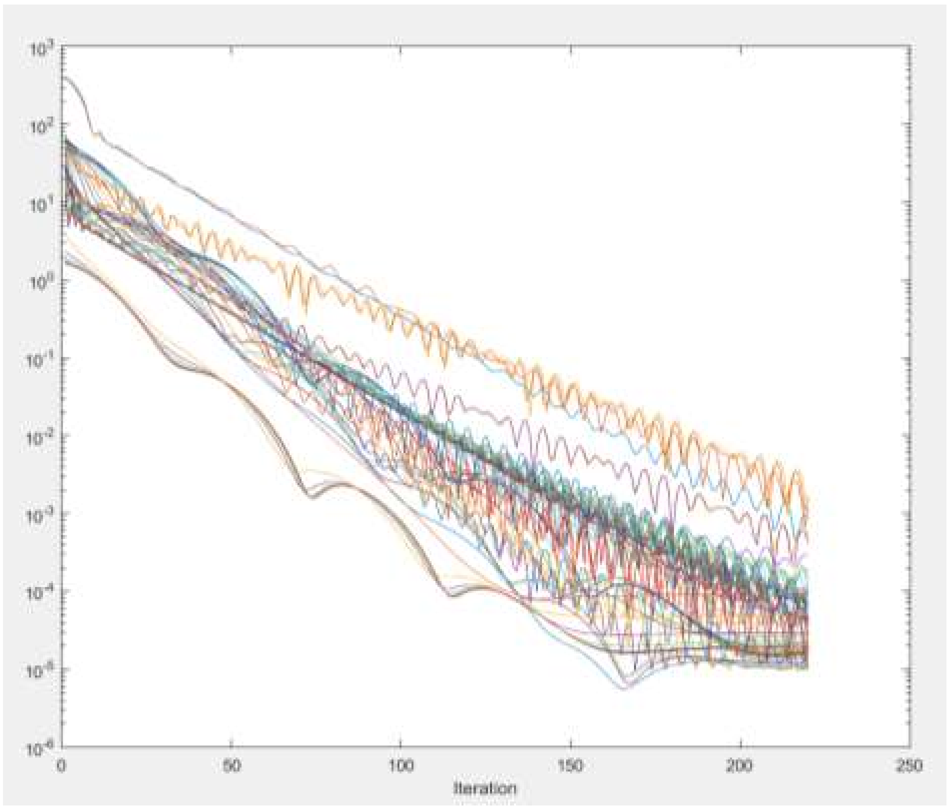


Fig. 21. Convergence illustrated by log-scale norm of the difference between FiOrdOs and quadprog results as a function of the number of iterations. AFTI-16 aircraft model example case from QPgen is used. Every line represents one timestep of the simulation.

5.5.3 HPMPC

HPMPC is a C-code library aimed at providing routines for high-performance implementation of solvers for linear MPC and MHE [37]. It contains ADMM and an interior point method. Its time-critical routines are carefully optimized by hand for a number of architectures, including various low-power processors [40]. It contains a mass-spring-damper system example and was benchmarked using a mass-spring system with box constraints [41].

6 Implementation of a prototype MPC controller for ITER plasma magnetic control using a QP solver based on the fast gradient method

For the evaluation of fast online MPC solvers, an earlier version of the ITER plasma magnetic control scheme [33] is being used. Firstly, a detailed description of this scheme is given, then the achieved results are presented.

6.1 Simulation setup

The simulations and controller design methods are based on high-order local linear dynamical models of the tokamak plasma from CREATE-L or CREATE-NL [9, 10], at several different equilibrium points, defined by the nominal plasma current I_p, poloidal beta β_p, and internal inductance l_i, for the anticipated ITER plasma. The models are listed in Table 2.

Table 2. Local linear dynamic models.

Model code	I_p (MA)	β_p	l_i	Number of states
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
LMNE	15.0	0.10	1.21	120

The Matlab/Simulink simulation scheme comprises:

- the plasma/circuits linearized model,
- a simplified model of plasma diagnostics for the plasma vertical velocity v_p and position z_p (a first-order dynamic lag filter with the time constant equal to $7\cdot10^{-3}$ s is considered),
- simplified models of the power supplies for the superconductive coils VS1 and for the in-vessel ohmic coils VS3 (a first-order dynamic lag with the time constant equal to $7.5\cdot10^{-3}$ s; a delay equal to $2.5\cdot10^{-3}$ s; saturations ±6 kV and ±1.5 kV for VS1 and VS3, respectively),
- simplified models of the main power supplies (saturations ±1.5 kV, except for VCS1 ±3 kV, and first-order dynamic lag with the time constant equal to 0.015 s and a delay equal to 0.015 s),
- the inner cascade control loop of the VS system, which aims at vertically stabilizing the plasma column,
- the outer cascade control loop of the SC, which controls plasma current and shape,
- blocks enabling the simulation of vertical displacement events (VDE), using a corresponding plasma model initial state, and H-L transitions, by injecting recorded profiles of β_p and l_i (BPLI) [11]

The simulation solver ode23tb is used, with relative tolerance 10^{-5}.

6.2 Vertical stabilization

For vertical stabilization, a variant of continuous-time LQG controller which also controls plasma position z_p (ctLQGz) [8] is used.

The ctLQGz VS controller act on the control variables $\mathbf{u}_{\mathrm{VS}} = [u_{\mathrm{VS},1}\, u_{\mathrm{VS},2}]^{\mathrm{T}}$, where:

- $u_{\mathrm{VS},1}$ is the voltage applied to the IV coils VS3,
- $u_{\mathrm{VS},2}$ is the voltage applied to the SC circuit VS1,

while it attempts to drive to zero the controlled inputs $\mathbf{y}_{\mathrm{VS}} = [y_{\mathrm{VS},1}\, y_{\mathrm{VS},2}\, y_{\mathrm{VS},3}]^{\mathrm{T}}$, where

- $y_{\mathrm{VS},1}$ is the VS3 power supply current,
- $y_{\mathrm{VS},2}$ is the plasma vertical velocity v_p,
- $y_{\mathrm{VS},3}$ is the plasma vertical position z_p.

It is built by firstly designing a basic ctLQG controller [12, 8] governing only $[y_{\mathrm{VS},1}\, y_{\mathrm{VS},2}\,]^{\mathrm{T}}$, with a third-order model $\{\mathbf{A}_r, \mathbf{B}_r, \mathbf{C}_r, \mathbf{0}\}$ obtained from a nominal model with model reduction. The LQ controller is tuned by adjusting the diagonal elements of the cost matrices $\mathbf{Q}_{\mathrm{C},\mathbf{y}}$ and $\mathbf{R}_{\mathrm{C},\mathbf{u}}$, and the covariance matrices of the Kalman filter (KF) are tuned by using $\mathbf{Q}_{\mathrm{K},\mathbf{y}} = \mathbf{B}_r\mathbf{B}_r^{\mathrm{T}}$ and tuning the diagonal elements of $\mathbf{R}_{\mathrm{K},\mathbf{y}}$ [8]. Then, the additional control loop from z_p is added by augmenting the nominal model with an integrator

$$\mathbf{A}_a = \begin{bmatrix} \mathbf{A}_r & \mathbf{0}_{3\times1} \\ \mathbf{C}_{r,2} & 0 \end{bmatrix},\ \mathbf{B}_a = \begin{bmatrix} \mathbf{B}_r \\ \mathbf{0}_{2\times1} \end{bmatrix},\ \mathbf{C}_a = \begin{bmatrix} \mathbf{C}_r & \mathbf{0}_{2\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{28}$$

where $\mathbf{C}_r^T = \begin{bmatrix} \mathbf{C}_{r,1}^T & \mathbf{C}_{r,2}^T \end{bmatrix}$.

6.3 Plasma current and shape controller

The CSC output is the vector of the 11 main power supply voltages $\mathbf{V}_{\mathrm{PF}}$.

The CSC inputs may include:

- the vector of controlled gaps $\mathbf{g}$, comprising four gaps and two strike-points,
- the plasma current I_p,
- the currents in the 11 superconductive coils $\mathbf{I}_{\mathrm{PF}}$.
- the plasma vertical position z_p and the VS3 power supply current I_{VS3} may also be considered, although they are controlled by the VS.

6.3.1 Model predictive control (MPC)

The MPC CSC presented in this study is based on the nominal plasma model LM52 in the state-space form. The simplified models of the power supplies and sensors (diagnostics) are appended. Then, the ctLQGz VS feedback loop is added, and the subsystem from the process inputs $\mathbf{u}_{\mathrm{CSC}} = \mathbf{V}_{\mathrm{PF}}$ to the outputs $\mathbf{y}_{\mathrm{CSC}} = [\mathbf{I}_{\mathrm{PF}}\ I_{\mathrm{VS3}}\ z_p\ I_p\ \mathbf{g}]^{\mathrm{T}}$ is extracted. With a model reduction procedure, the order of the subsystem is reduced from 199 to 44. The base model for the MCP CSC

$\{\mathbf{A}_{CSC}, \mathbf{B}_{CSC}, \mathbf{C}_{CSC}, \mathbf{0}\}$ is finally obtained with model conversion to discrete time with the sampling time T_s = 0.1 s, assuming zero-order hold.

For the purposes of this study, the CSC should facilitate offset-free control of I_p and **g** to zero with integral action, without set-point tracking. In our implementation, integral action is based on the *disturbance estimation* (DE) concept [13], and the *velocity form* (without tracking) is used to prevent offset due to the control cost when the control signal is non-zero at the steady-state.

For the estimation of asymptotically non-zero disturbances, the base model is augmented with DE integrators at the outputs which require offset-free control. Consider the discrete-time state-space model

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{w}(k), \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{v}(k) \end{aligned} \tag{29}$$

where **w** and **v** are white noise signals to the state and output, respectively. DE integrator states **d** with the associated white-noise signal $\mathbf{w}_d$ are appended to the state **x**, so that the augmented state is $\mathbf{x}_a = [\mathbf{x}^T \ \mathbf{d}^T]^T$, and $\mathbf{w}_a = [\mathbf{w}^T \ \mathbf{w}_d^T]^T$. The augmented system is

$$\begin{aligned} \begin{bmatrix} \mathbf{x}(k+1) \\ \mathbf{d}(k+1) \end{bmatrix} &= \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{d}(k) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} u(k) + \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{w}(k) \\ \mathbf{w}_d(k) \end{bmatrix} \\ \mathbf{y}(k) &= \begin{bmatrix} \mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{d}(k) \end{bmatrix} + \mathbf{v}(k) \end{aligned} \tag{30}$$

and is rewritten as

$$\begin{aligned} \mathbf{x}_a(k+1) &= \mathbf{A}_a\mathbf{x}_a(k) + \mathbf{B}_a\mathbf{u}_a(k) + \mathbf{w}_a(k), \\ \mathbf{y}(k) &= \mathbf{C}_a\mathbf{x}_a(k) + \mathbf{v}(k) \end{aligned} \tag{31}$$

The steady-state Kalman filter (KF)

$$\begin{aligned} \mathbf{x}_a(k|k-1) &= \mathbf{A}_a\mathbf{x}_a(k-1|k-1) + \mathbf{B}_a\mathbf{u}(k-1) \\ \mathbf{x}_a(k|k) &= \mathbf{x}_a(k|k-1) + \mathbf{M}_\mathbf{K}[\mathbf{y}(k) - \mathbf{C}_a\mathbf{x}_a(k|k-1)] \end{aligned} \tag{32}$$

is used state estimation with the disturbance-augmented model, where $\mathbf{M}_K$ is computed via the steady-state solution of the Riccati equation from the covariance matrices $\mathbf{Q}_K = \mathrm{E}\{\mathbf{w}_a\mathbf{w}_a^T\}$ and $\mathbf{R}_K = \mathrm{E}\{\mathbf{v}\mathbf{v}^T\}$. The KF is used in the sense of an observer, where the diagonal elements of $\mathbf{Q}_K$ and $\mathbf{R}_K$ are used as tuning parameters to achieve desired dynamics.

For the velocity form, the disturbance-augmented system $\{\mathbf{A}_a, \mathbf{B}_a, \mathbf{C}_a, \mathbf{0}\}$ is augmented again. In the velocity-augmentation, the change of the input signal δ**u** becomes the new input; the state expands to $\mathbf{x}_{av} = [\mathbf{x}_a^T \ \mathbf{u}(k-1)^T]^T$; the new output is $\mathbf{y}_{av} = [\mathbf{y}_a^T \ \mathbf{u}(k-1)^T]^T$

$$\begin{aligned} \begin{bmatrix} \mathbf{x}_a(k+1) \\ \mathbf{u}(k) \end{bmatrix} &= \begin{bmatrix} \mathbf{A}_a & \mathbf{B}_a \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_a(k) \\ \mathbf{u}(k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{B}_a \\ \mathbf{0} \end{bmatrix} \delta\mathbf{u}(k) \\ \begin{bmatrix} \mathbf{y}_a(k) \\ \mathbf{u}(k-1) \end{bmatrix} &= \begin{bmatrix} \mathbf{C}_a & \mathbf{D}_a \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_a(k) \\ \mathbf{u}(k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{D}_a \\ \mathbf{0} \end{bmatrix} \delta\mathbf{u}(k) \end{aligned} \tag{33}$$

with $\mathbf{D}_a = \mathbf{0}$ rewritten as

$$\begin{aligned} \mathbf{x}_{av}(k+1) &= \mathbf{A}_{av}\mathbf{x}_{av}(k) + \mathbf{B}_{av}\delta\mathbf{u}(k) \\ \mathbf{y}_{av}(k) &= \mathbf{C}_{av}\mathbf{x}_{av}(k) \end{aligned} \tag{34}$$

Then, the MPC controller is built using the Multi-Parametric Toobox (MPT) [4] in the output-cost formulation, with the cost function

$$J(\delta\tilde{\mathbf{u}}) = \sum_{k=0}^{N-1} \mathbf{y}_{av,k}^T \begin{bmatrix} \mathbf{Q}_{C;y} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{y}_{av,k} + \delta\mathbf{u}_k^T \mathbf{R}_{C,\delta u} \delta\mathbf{u}_k \tag{35}$$

where the diagonal elements of the cost matrices for the outputs $\mathbf{Q}_{C,y}$ and the control moves $\mathbf{R}_{C,\delta u}$ are used as tuning parameters, where $N = 30$ is the prediction horizon length. The control law is obtained by minimising J with respect to the vector of the future control moves $\delta\tilde{\mathbf{u}}$ subject to constraints (currently, only control amplitude constraints $\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}$ are used). To reduce the computational demand, the number of free control moves is reduced from 30 to 3 using move blocking to intervals [2 2 26]. Due to the finite horizon length, the control law is computed as a least-squares problem in the unconstrained case, or as a quadratic programming problem in the constrained case [13].

6.3.2 Singular perturbation decomposition (SPD)

For comparison, the plasma current and shape control algorithm based on SPD of Ariola and Pironti [7] is used. The SPD CSC implements a multivariable proportional-integral control law from **g** and I_p, with an additional proportional contribution from $\mathbf{I}_{\mathrm{PF}}$. It also includes windup protection in case of actuator saturation.

6.4 MPC controller model adaptation for use with fast online MPC/QP solver

6.4.1 Problem background

The reference controller model has been built using the MPT Toolbox [4] and general-purpose QP solvers (Matlab `quadprog`, IBM ILOG Cplex).

To investigate the potential computation performance the using a fast first-order QP solver, the QPgen code generation library has been selected. In following, the required MPC problem formulation for use with a fast QP solver and some required modifications to QPgen are described.

In the original implementation, the MPC problem setup was made using the `MPT_yalmipCFTOC` function of the MPT Toolbox and internally parsed using YALMIP. For use with QPgen it was found more appropriate to carry out the MPC problem setup manually, because grouping of different constraint types is required, and because the implementation of soft constraints is different. The QP problem formulation requires the system to be first augmented by integrators for disturbance estimation (30) and second to the velocity form (33). Further, the process the output cost matrix $\mathbf{Q}_y$ is provided but the output cost formulation (35) is not supported within QPgen, which requires the following state and input cost matrices to achieve same effect:

$$\mathbf{Q}_{av} = \begin{bmatrix} \mathbf{Q}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}, \quad \mathbf{R} = \mathbf{R}_{\delta u} \tag{36}$$

where

$$\mathbf{Q}_x = \mathbf{C}^T \mathbf{Q}_{C;y} \mathbf{C} \tag{37}$$

The augmented state and input constraints are then defined by

$$\mathbf{x}_{av\_LB} = \begin{bmatrix} \mathbf{x}_{a\_LB} \\ \mathbf{u}_{LB} \end{bmatrix} \quad \mathbf{x}_{av\_UB} = \begin{bmatrix} \mathbf{x}_{a\_UB} \\ \mathbf{u}_{UB} \end{bmatrix} \quad \delta\mathbf{u}_{UB} = \left[\mathbf{u}_{UB} - \mathbf{u}_{LB}\right] \quad \delta\mathbf{u}_{LB} = \left[\mathbf{u}_{LB} - \mathbf{u}_{UB}\right] \tag{38}$$

where UB and LB stands for upper and lower state/input bound, respectively. Additionally, soft output constraints are set.

For enabling the full-featured implementation of the MPC problems using QPgen solver, allowing MPC and QP optimization, some modifications of the QPgen algorithm were required:

- Move blocking – stacking of control signal along the prediction horizon in same-valued blocks (in this case, 3 blocks with lengths [1 2 27])
- Sparse output constraints – partial elimination of output constraints along the prediction horizon (constraints applied to each 3rd sample in this case)
- Elimination of redundant constraints equations where the variables are unbounded.
- Parallelization – fully exploitable in bare C without Matlab interface - using Intel compiler, as the originally coded parallelization procedure is ineffective with the plasma control problem
- Quadratic cost on soft output constraints – only linear cost is implemented in the original QPgen solver.

6.4.2 MPC problem simplification

The main MPC parameters affecting the QP size and complexity are the process states $\mathbf{x}$, inputs $\mathbf{u}$, constraints on them and the horizon length N. These are affecting the size of the QP optimization vector $\mathbf{z}$, Hessian matrix $\mathbf{H}$ and inequality constraint matrices $\mathbf{A}_{\text{ineq}}$, $\mathbf{B}_{\text{ineq}}$.

$$size(\mathbf{z}) = \left(size(\mathbf{x}) + size(\mathbf{u}) + size(\mathbf{s})\right) \times N \tag{39}$$

$$size(\mathbf{A}_{ineq}) = \left(size(\mathbf{y}) + size(\mathbf{u}) + size(\mathbf{s})\right) \times N \tag{40}$$

6.4.2.1 Efficient handling of soft constraints

For handling soft constraints in MPC, slack variables added to constrained process variables are typically introduced. When the value of the process variable does not violate the constraint values, the slack variable is zero; when the process variable exceeds the constraint, the value of the slack variable is equal to the amount of constraint violation. Thus softened constraints allow constraint violations, but with a high penalty on the violation in the cost function. The introduced slack variables significantly expand the optimization problem (1 or 2 slacks and 2 or 4 additional constraint inequalities for each sample of the prediction) and its computational cost. On the other hand, the QPgen approach to soft constraints handling [18], without the need for additional slack variables maintains the QP size of the original hard-constraint optimization problem, as well as the solver computational demand.

6.4.2.2 Move blocking

It is well known that by considering constant disturbance and reference signals along the horizon, the major part of the resulting controller action is performed in the early part of the horizon. The move blocking technique [3] exploits this fact and is used to significantly reduce the number of active inputs along the control horizon. It is achieved by stacking consecutive inputs in blocks with the same control value and by this effectively reducing the size of the optimization vector (and size of the resulting QP).

6.4.2.3 Sparse placement of output constraints

To further reduce the optimization problem size, an efficient approach [12] is to reduce the number of output constraints placed along the prediction horizon, by enforcing constraints every *k*-th sample only (e.g. every 2nd or every 3rd etc.). Effectively this means reduction of the number of constraint equations (represented by $\mathbf{A}_{ineq}$ matrix) by factor of *k*. This results in less strict enforcement of constraints, however constraints enforcement remains reasonable when the sampling time is relatively short. Notice that relatively short sampling time, considering the process dynamics, is used in order to maintain fast response to disturbances.

6.4.2.4 Elimination of redundant constraints

By removing the equations for the unbounded variables (states, inputs, outputs), the constraints number (represented by $\mathbf{A}_{ineq}$ matrix) can be decreased without any influence on MPC control.

6.4.2.5 Plasma MPC problem simplification

To present the achieved QP problem reduction via the above-mentioned MPC simplifications, the plasma-controller parameters are given in the Table 3. The original problem considers a system with 62 states, 11 inputs and 20 outputs, (of which 11 are bounded), free control moves allowed at each sample along the horizon (i.e. 30), and with all horizon samples subject to constraints. The resulting simplified MPC structure maintains the 11 inputs and 62 states, but reduces affected outputs to 11, the free control moves to 3, and with output constraints placed on every third sample.

Table 3. MPC problem simplification influence to QP complexity

Parameter size	***u***	***x***	***y***	***s***	N_z[2]	N_{cstr}[3]	A_{ineq} (y+u+s)*N×	*z* (x+u+s)*N	***H***	***memory***
Original[4]	11	62	20	20	30	30	1530×2790	2790	2790×2790	
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

6.4.3 Implementation of quadratic cost on for soft constraints

The implementation of quadratic cost on soft constraints into QPgen toolbox has been made based on [18], section 3.2.5**.** Therefore quadratic cost on slacks can be added without expanding the optimization variable. Firstly we solve the problem without constraints and then apply the Karush-Kuhn-Tucker conditions to the unconstrained solution with quadratic cost on the soft constraints variable.

[2] N_z – optimization-vector size affecting horizon length

[3] N_{cstr} – constraint equation number affecting horizon length

[4] For information only, not feasible for use due to slow convergence

Option *MPC.W.soft* was added to the library as a constant value for the implementation of the quadratic cost on soft constraints. The value is later on multiplied by an identity matrix. The size of the matrix is equal to the number of the constraints, therefore is transformed via

$$\mathbf{W}_s = \mathbf{C}^T \mathbf{W} \mathbf{C} \tag{41}$$

The implementation requires cost on states and inputs in the CSC MPC cost function, for which the **Q** state cost matrix is introduced into the MPC problem formulation, only for the quadratic cost on soft constraints. All related modifications are documented in a separate file "*QPgen_IJS_changelog.pdf*".

6.5 Simulation results

6.5.1 Validation of fast QP solver

Table 4 lists RISE values (root integral squared error, from the corresponding equilibrium point) of the main process signals in 25 s simulations using the same ctLQGz VS controller,

- with either the MPC or the SPD CSC,
- for VDE events with the initial amplitude –0.1 m and for BPLI disturbances,
- with the five local models, to roughly assess the sensitivity to the varying operating conditions.

Figs. 22-23 show the main process signals in VDE events, and Figs. 24-27 with BPLI disturbances, respectively, simulated in the same time interval; only the simulations with the models LM53 and LM52 are included; the performance with MPC is compared with SPD. "Delta" signal values (the deviations from the equilibrium point of the linear model) are displayed.

The results of the MPC and the SPD scheme are difficult to compare because both methods involve a large number of degrees of freedom in tuning that allow various trade-offs, and the presented simulation results only show the performance with particular sets of tuning parameters. With MPC we have achieved a generally faster suppression of disturbances at I_p and better at $\mathbf{g}$. The z_p response is mostly improved in BPLI simulations but is slower in VDE simulations. The improved tracking performance comes at the price of a slight increase of the voltages $\mathbf{V}_{\mathrm{PF}}$ and currents $\mathbf{I}_{\mathrm{PF}}$, but there are no rapid reactions indicating over-sensitivity.

Fig. 28 shows a simulation similar to the one in Fig. 26, where, in addition to **u** constraints, soft constraints $\max(\mathbf{I}_{\mathrm{PF}}) \leq 4$ kA were set in the MPC controller. It is not feasible to fully adhere to this limit, yet the controller manages to reduce the current peaks visibly.

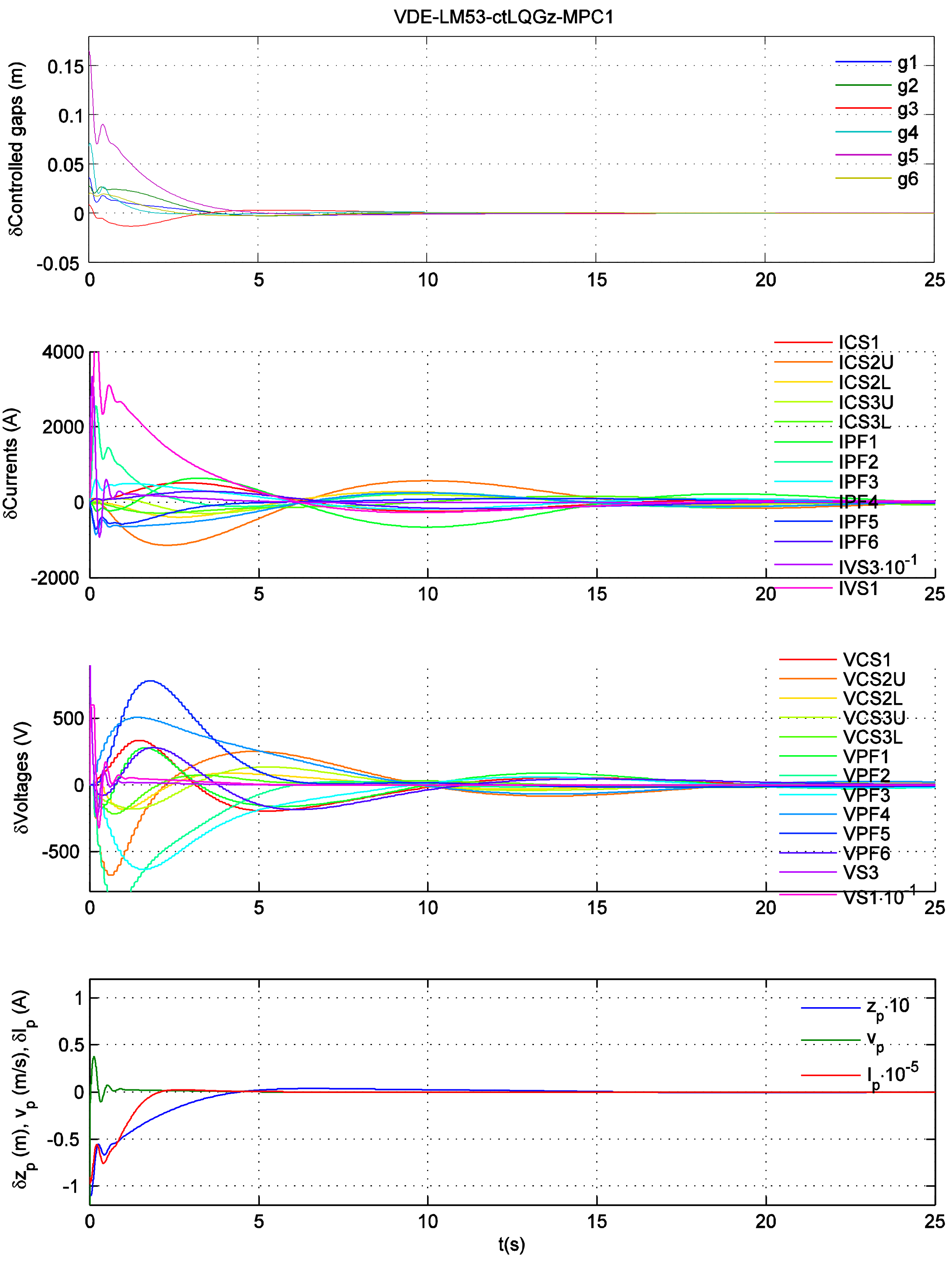


Fig. 22. VDE simulation: ctLQGz-MPC, model LM53.

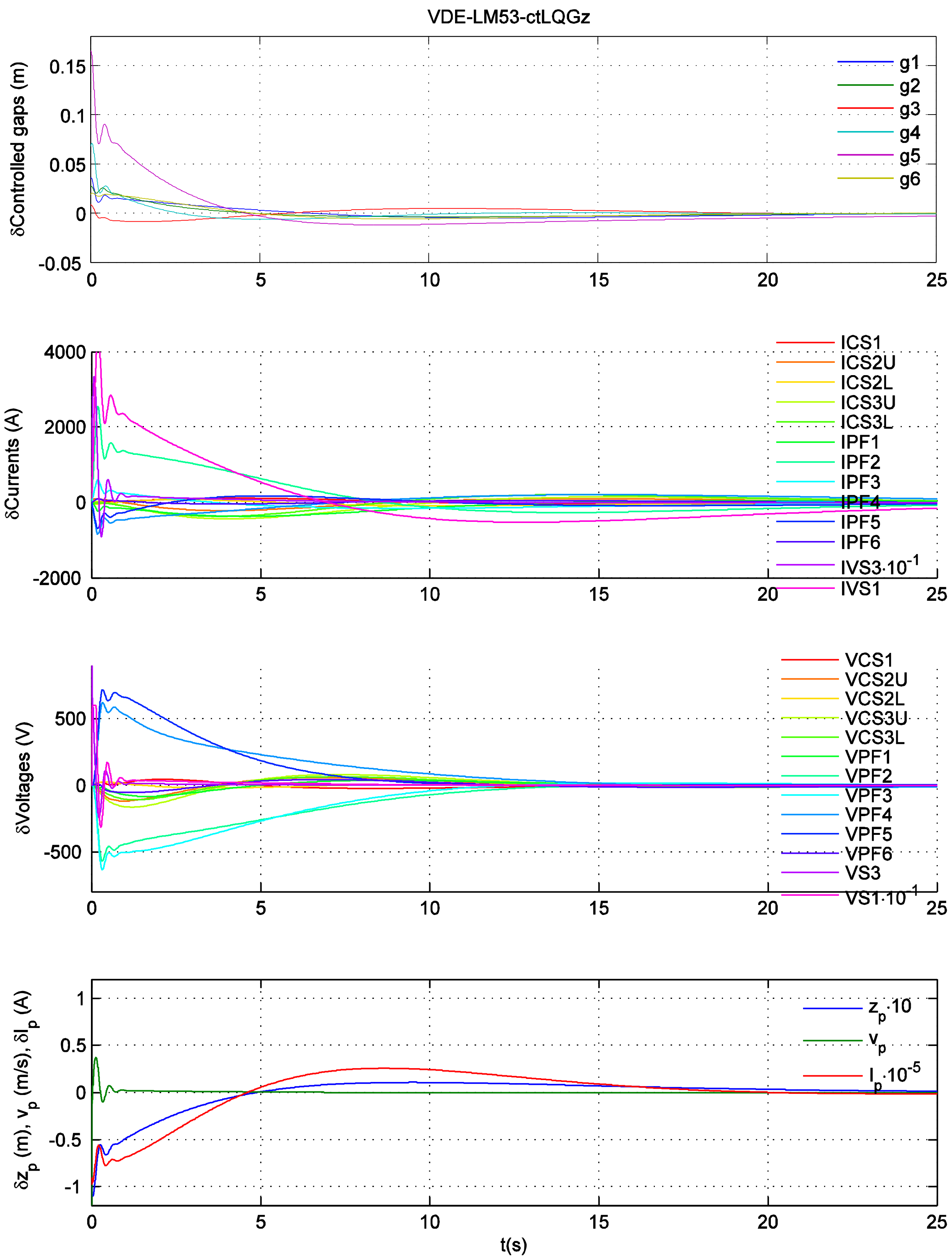


Fig. 23. VDE simulation: ctLQGz-SPD, model LM53.

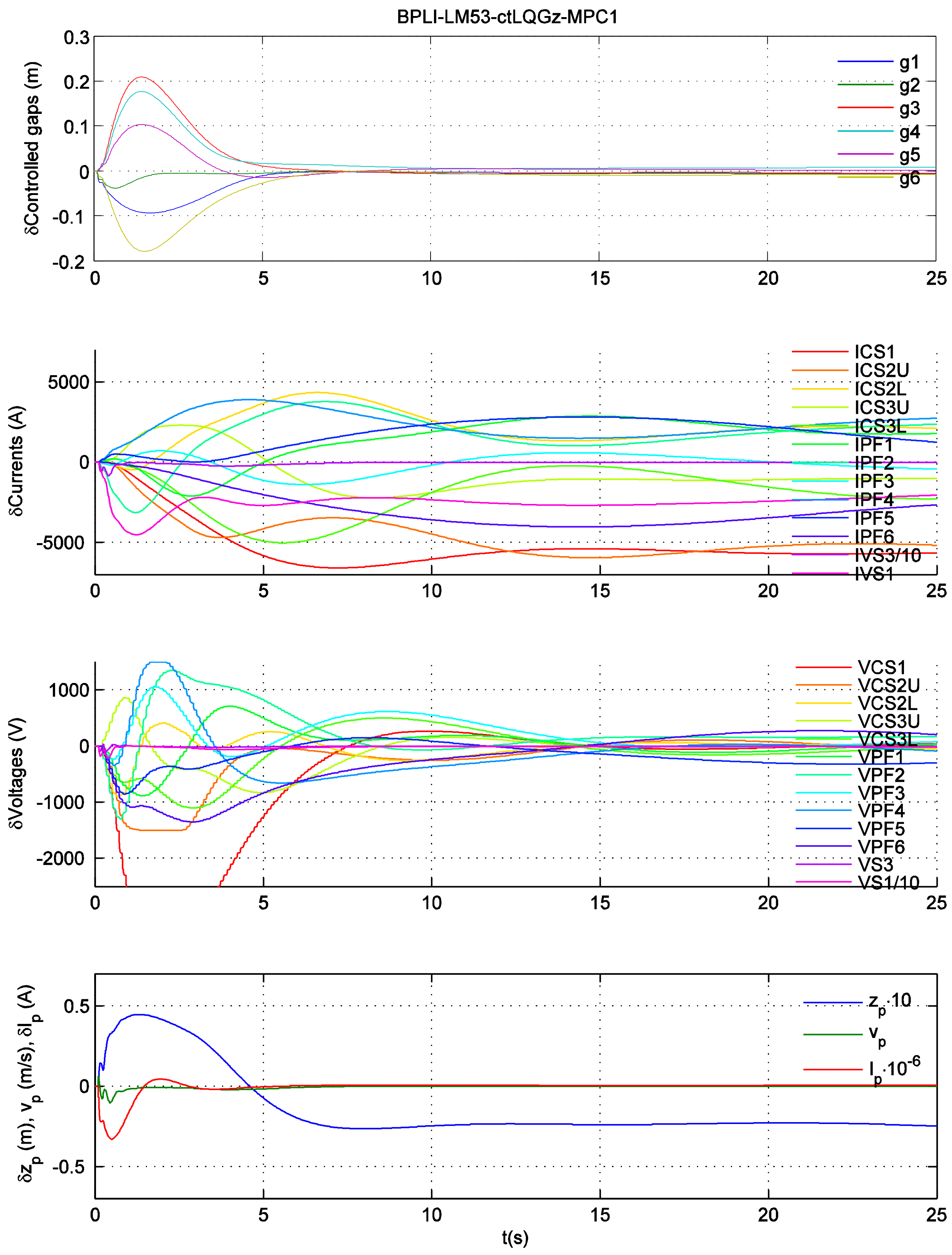


Fig. 24. BPLI simulation: ctLQGz-MPC, model LM53.

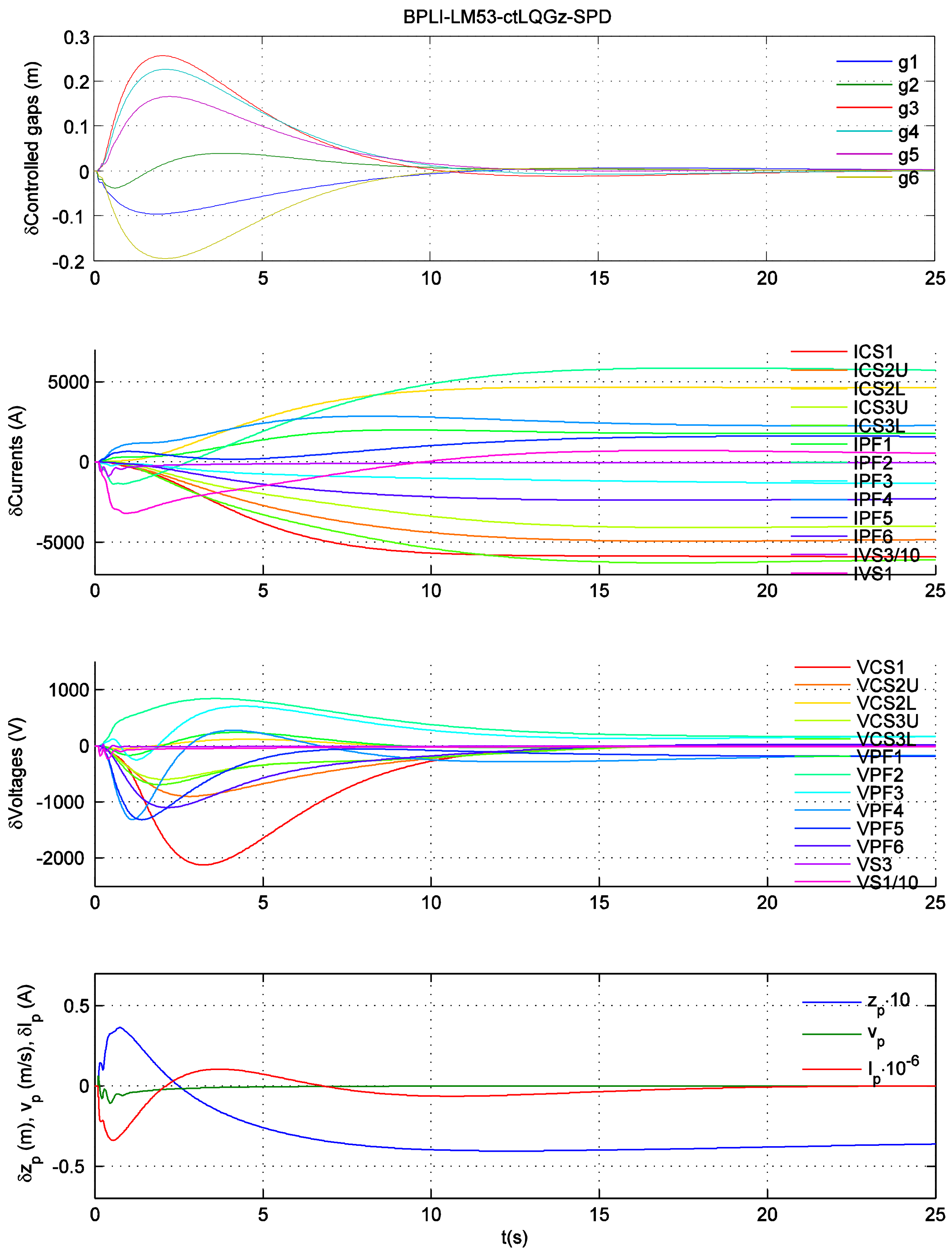


Fig. 25. BPLI simulation: ctLQGz-SPD, model LM53.

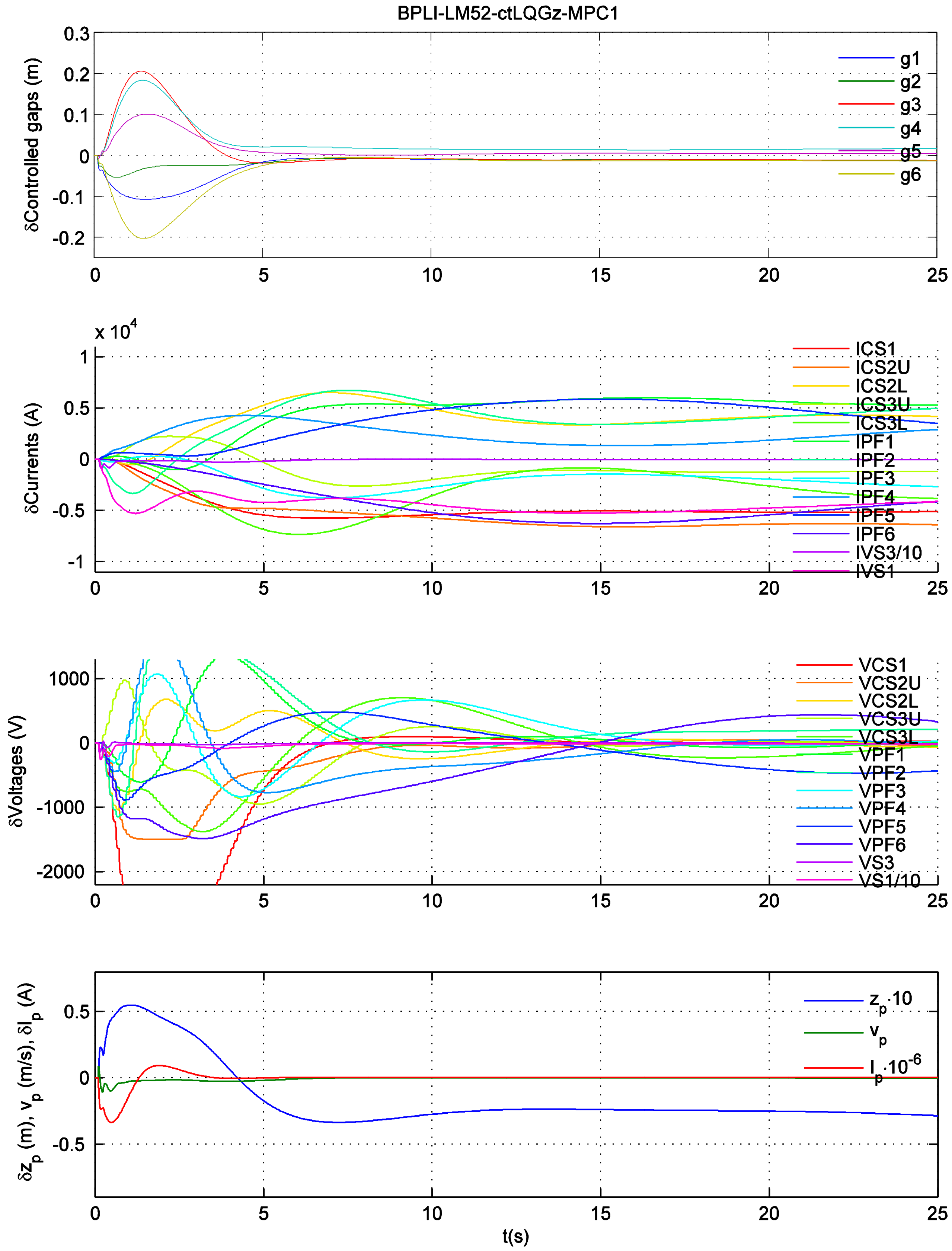


Fig. 26. BPLI simulation: ctLQGz-MPC, model LM52.

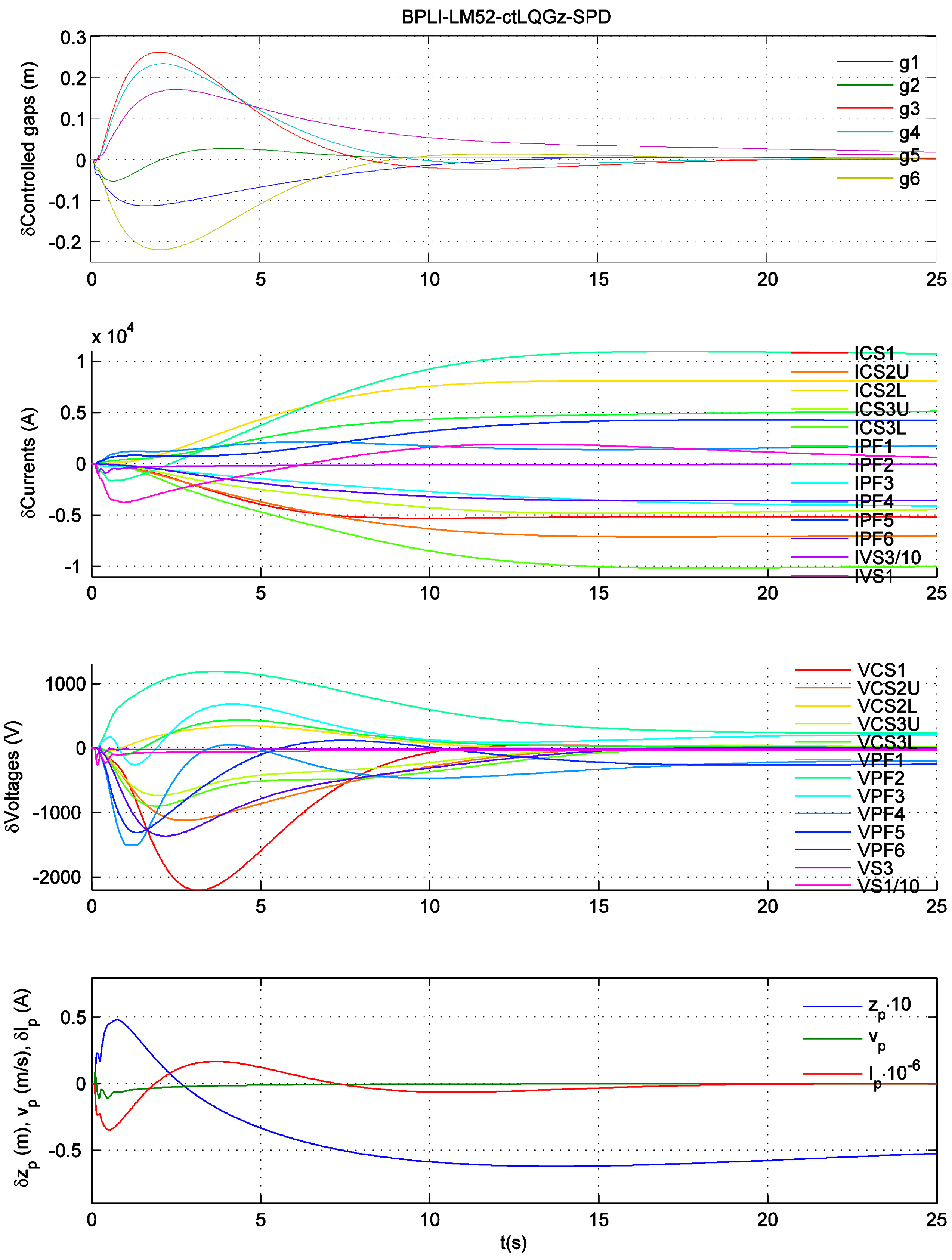


Fig. 27. BPLI simulation: ctLQGz-SPD, model LM52.

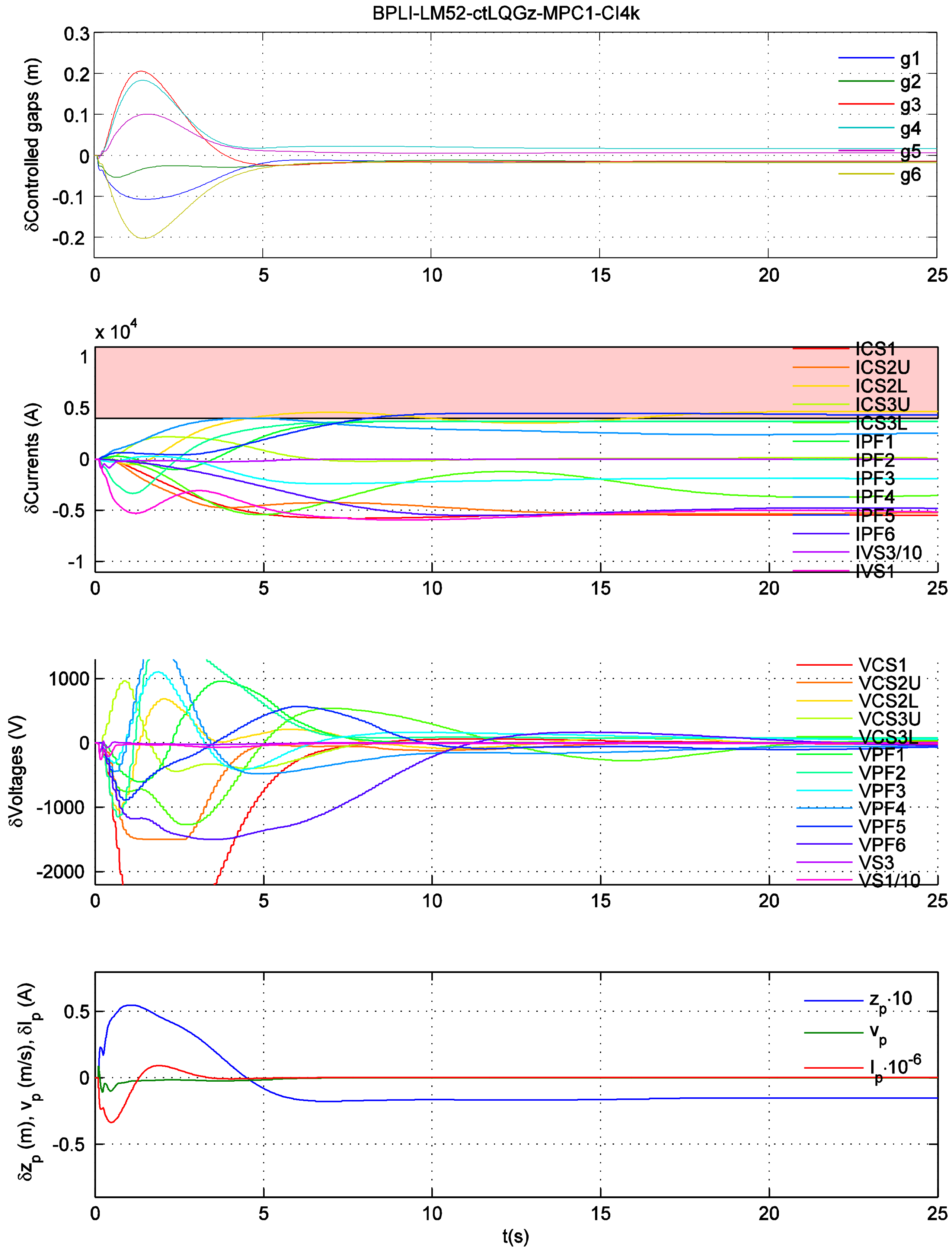


Fig. 28. BPLI simulation: ctLQGz-MPC, model LM52, with soft constraints max($\mathbf{I}_{PF}$) ≤ 4 kA.

Table 4. RISE values (from the corresponding equilibrium).

LMNE	VDE-ctLQGz-MPC	VDE-ctLQGz-SPD	BPLI-ctLQGz-MPC	BPLI-ctLQGz-SPD
max(CtrlGaps)	0.067183	0.07322	0.24188	0.6735
avg(CtrlGaps)	0.019551	0.02092	0.16442	0.447
max(AllGaps)	0.088145	0.09702	0.48326	1.0339
avg(AllGaps)	0.017834	0.01875	0.20083	0.48994
IVS3	36320	36357	7358.9	8557.4
IVS1	1262	1344.2	4731.5	7366.8
VS3	97.011	97.126	17.127	65.23
VS1	3264.5	3283.2	1420.5	2411.2
v_{p}	0.6012	0.60184	0.11636	0.087254
z_{p}	0.025029	0.02783	0.087254	0.25402
I_{p}	25926	31177	442220	575550
LM52	VDE-ctLQGz-MPC	VDE-ctLQGz-SPD	BPLI-ctLQGz-MPC	BPLI-ctLQGz-SPD
max(CtrlGaps)	0.11312	0.13224	0.28927	0.45392
avg(CtrlGaps)	0.03931	0.04727	0.19914	0.32625
max(AllGaps)	0.21748	0.24904	0.6673	1.2303
avg(AllGaps)	0.052378	0.06047	0.2547	0.42863
IVS3	10014	9840.8	6860.6	8539.7
IVS1	3861	4097.4	22622	8149.3
VS3	158.11	156.92	63.439	69.548
VS1	2484.3	2435.7	1825.7	2412.4
v_{p}	0.13284	0.13179	0.075632	0.086616
z_{p}	0.082884	0.09539	0.14286	0.26018
I_{p}	79948	134900	275000	431300
LM53	VDE-ctLQGz-MPC	VDE-ctLQGz-SPD	BPLI-ctLQGz-MPC	BPLI-ctLQGz-SPD
max(CtrlGaps)	0.10138	0.11235	0.27591	0.46948
avg(CtrlGaps)	0.037829	0.04289	0.18146	0.30869
max(AllGaps)	0.25307	0.28416	0.56506	0.8483
avg(AllGaps)	0.055365	0.06094	0.2296	0.36091
IVS3	11949	11860	5730.9	6909.6
IVS1	4075.3	4311.1	12920	5927
VS3	192.85	192.27	54.533	56.088
VS1	2615.1	2586.5	1548.7	1897.1
v_{p}	0.16552	0.16504	0.062971	0.073573
z_{p}	0.078885	0.08749	0.12626	0.17578
I_{p}	68748	122600	265130	380330
LM59	VDE-ctLQGz-MPC	VDE-ctLQGz-SPD	BPLI-ctLQGz-MPC	BPLI-ctLQGz-SPD
max(CtrlGaps)	0.12084	0.16562	0.30581	0.54037
avg(CtrlGaps)	0.048016	0.05993	0.25257	0.29891
max(AllGaps)	0.20879	0.28263	1.2666	2.3938
avg(AllGaps)	0.059293	0.07147	0.41902	0.58263
IVS3	9016.3	8624.3	8502.6	10344
IVS1	3401.4	3459	55016	7778.7
VS3	134.88	132.64	86.887	94.695
VS1	2623.4	2533.1	2282.3	3016.1
v_{p}	0.10447	0.10228	0.091223	0.10034
z_{p}	0.091871	0.11635	0.15196	0.35798
I_{p}	128420	192320	291360	378100
LM60	VDE-ctLQGz-MPC	VDE-ctLQGz-SPD	BPLI-ctLQGz-MPC	BPLI-ctLQGz-SPD
max(CtrlGaps)	0.14021	0.17821	0.36421	0.6735
avg(CtrlGaps)	0.049426	0.06082	0.23439	0.447
max(AllGaps)	0.25214	0.29641	0.68351	1.0339
avg(AllGaps)	0.062564	0.07296	0.28841	0.48994
IVS3	8424.5	7989.6	6378.5	8557.4
IVS1	3923.5	4168.2	19028	7366.8
VS3	119.07	116.17	55.333	65.23
VS1	2356.3	2248.4	1691.6	2411.2
v_{p}	0.10323	0.10036	0.071016	0.087254
z_{p}	0.085086	0.10566	0.13991	0.25402
I_{p}	79535	120170	308480	575550

6.5.2 MPC problem computational efficiency

To investigate the possible improvements of the computational efficiency, the following tasks were

- to compare the generic CPLEX solver performance with original and simplified MPC problem,
- to repeat the experiment using the developed QPgen based fast MPC setups,
- to verify the control quality.

Figures 29 and 30 presents the plasma control time plot comparison using the generic CPLEX solver of original and simplified MPC problem, respectively. Figures 31 and 32, respectively, give the iteration count and computation time at each simulation sample.

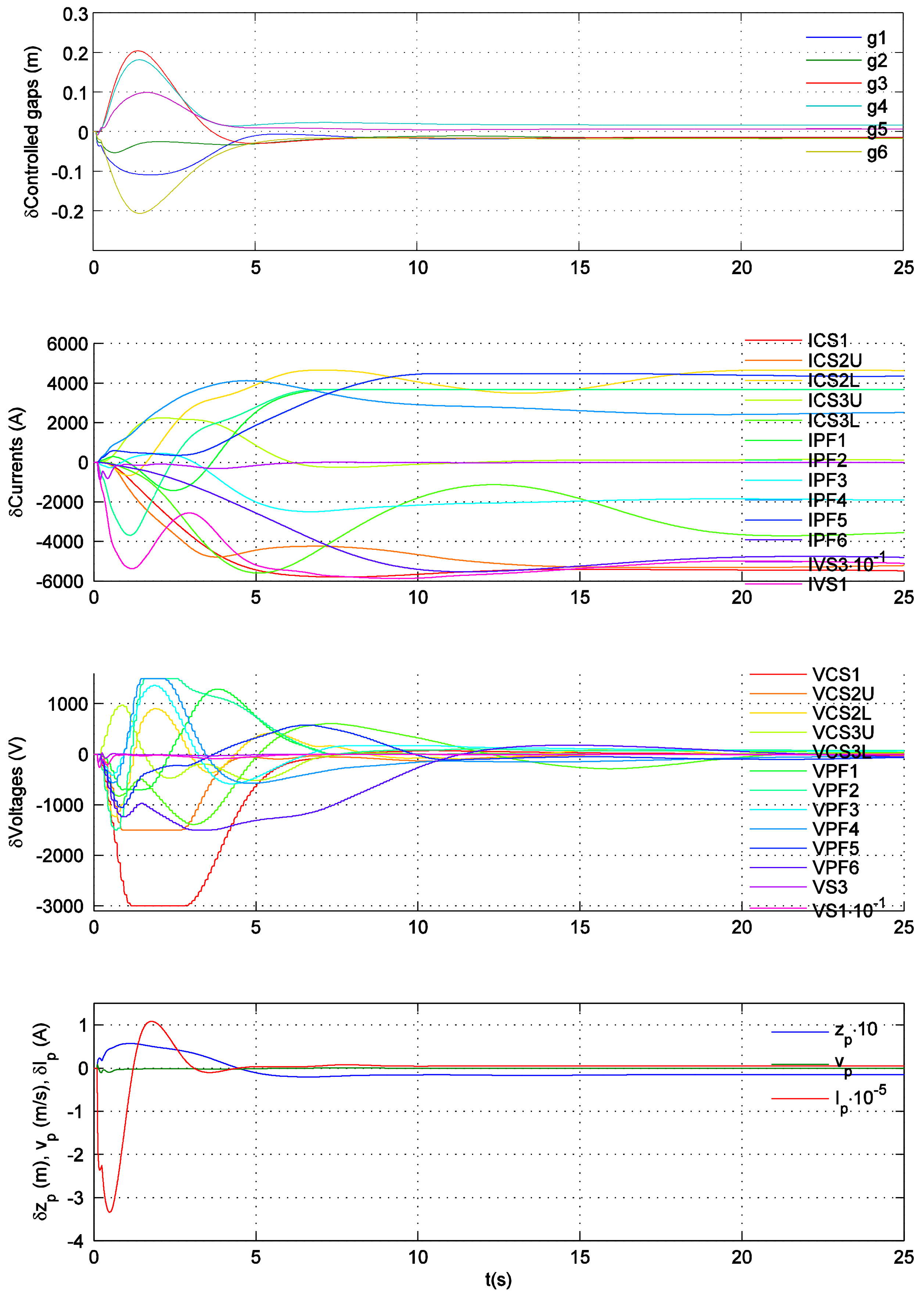


Fig. 29. BPLI simulation: ctLQGz-MPC, model LM52, using original model and CPLEX solver.

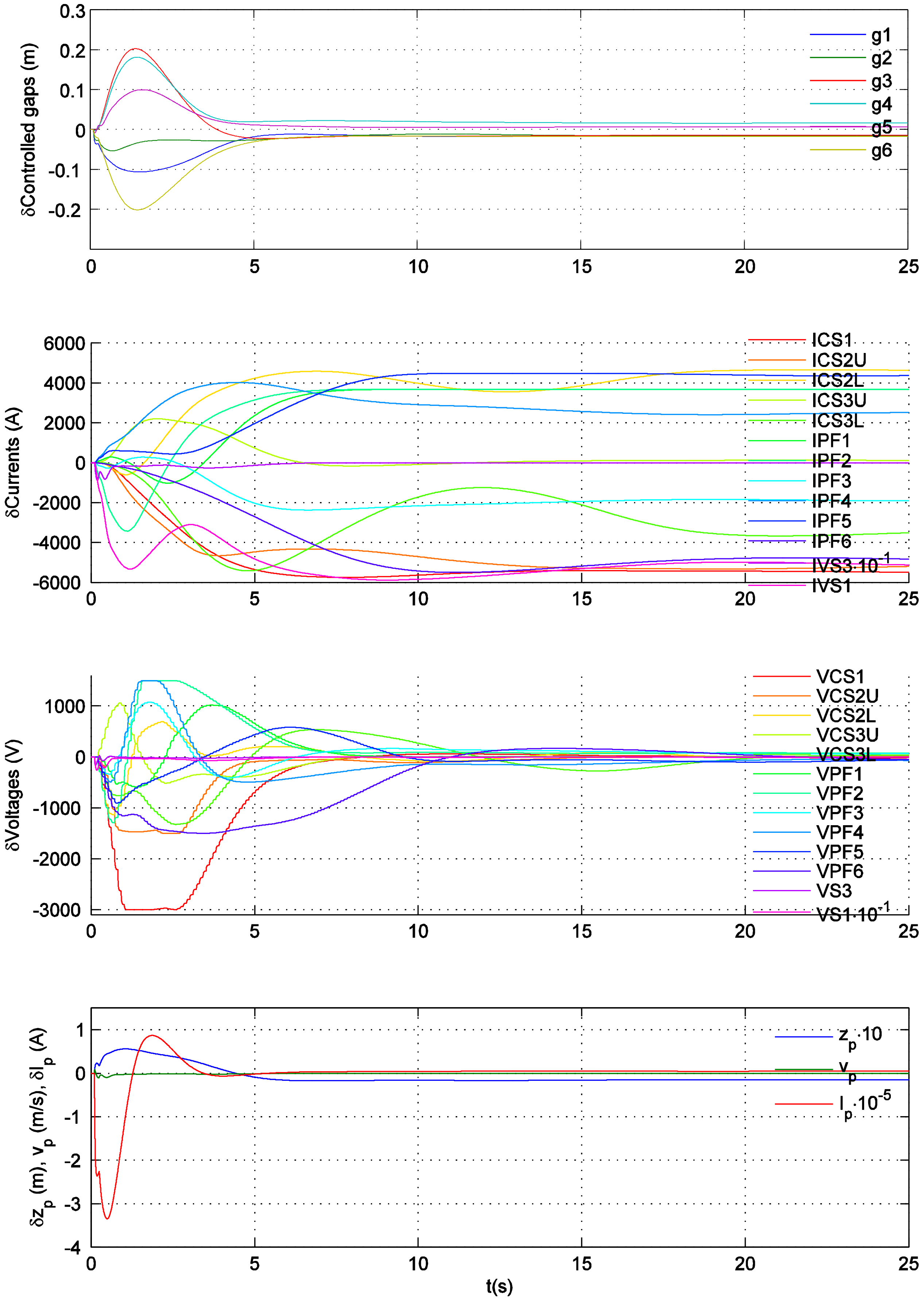


Fig. 30. BPLI simulation: ctLQGz-MPC, model LM52, using simplified model and CPLEX solver.

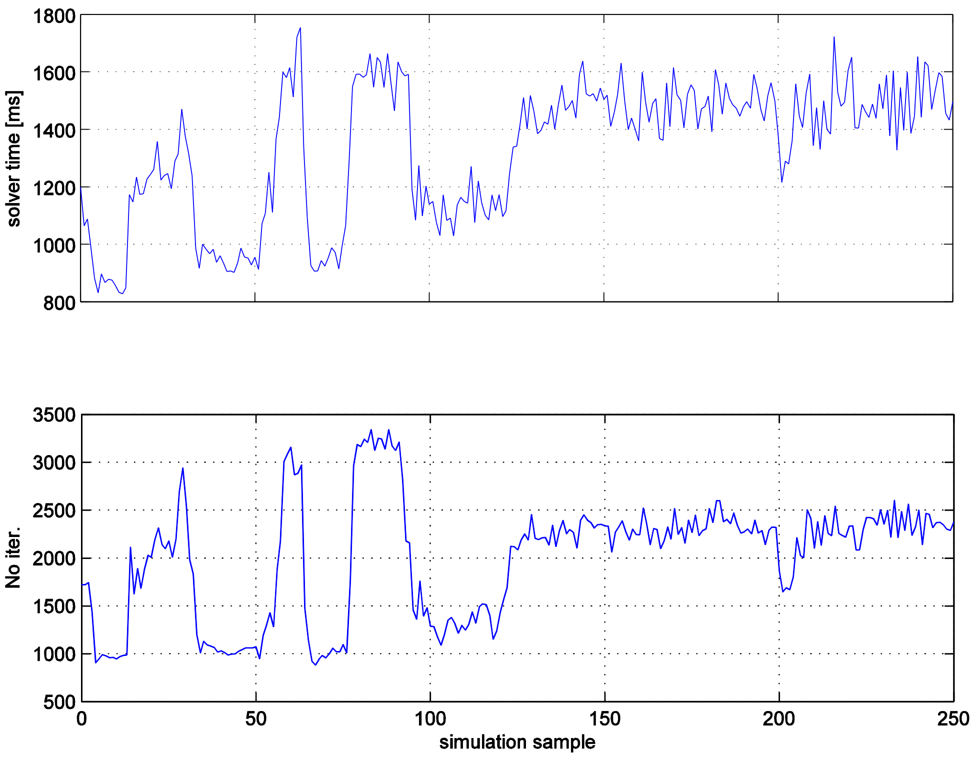


Fig. 31. No. of iterations and computation time: ctLQGz-MPC, model LM52, using original model and CPLEX solver.

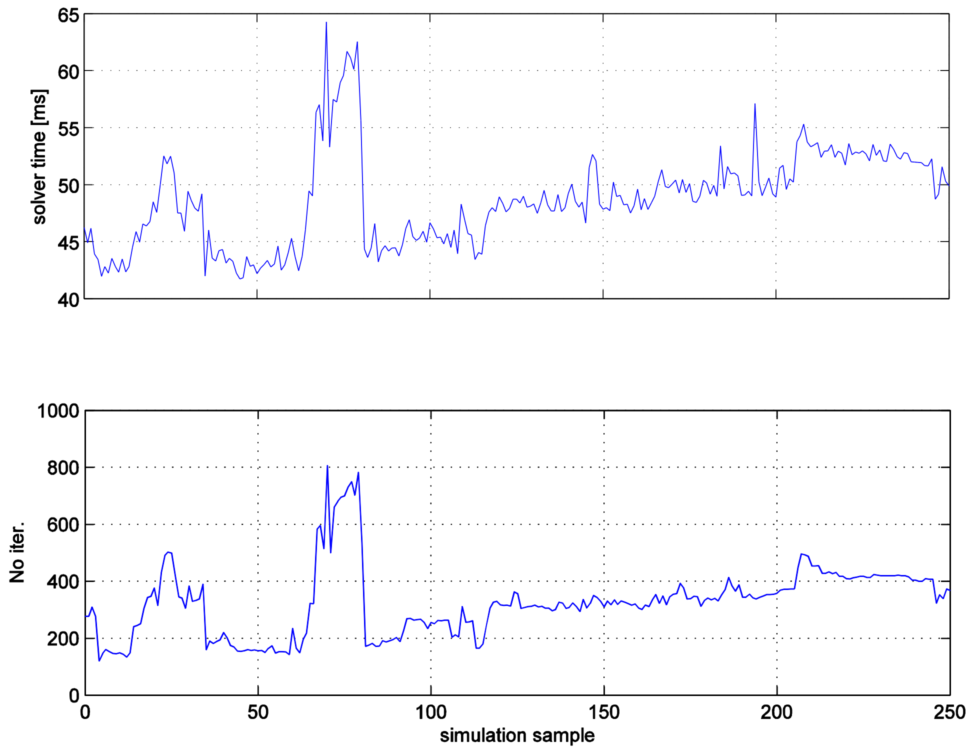


Fig. 32. No. of iterations and computation time: ctLQGz-MPC, model LM52, using simplified model and CPLEX solver.

In Table 5, the results of all the performed experiments with the original and simplified model, using CPLEX and QPgen are listed. For each experiment, the average number solver iterations and the computation time per time sample are given. The simulations have been performed using a laptop computer with a 2.4 GHz Inter Core2Duo processor, with 8 GB RAM, using a single core.

In the simplified problem, complexity-reduction techniques of move blocking [1 2 27] and sparse output constraints (with constraints remaining at each third sample). In both original and simplified problems, linear cost on soft constraints is used. The impact of numerical precision on the computational performance was also investigated with QPgen, by testing two values of tolerance in the plasma-control specific relevant range, 10^{-3} and 10^{-4}.

Further internal adjustments available within QPgen were tested:

- two available algorithms, ADMM and FGMdual
- QP size reduction using null-space: the SVD derived null-space obtained via Matlab function `null`, and the dynamical-system based null-space derived by

$$\mathbf{N} = \begin{bmatrix} \mathbf{B} & & & \\ \mathbf{AB} & \mathbf{B} & & \\ \vdots & & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \dots & \mathbf{AB} & \mathbf{B} \\ \mathbf{I} & & & \\ & \mathbf{I} & & \\ & & \ddots & \\ & & & \mathbf{I} \end{bmatrix}, \qquad (42)$$

The QPgen solver performance is displayed in Figs. 33-34, and summarized in Tables 3 and 5. From the results the following can be observed:

- For the original MPC problem, the ADMM and FGM dual methods provide similar results, with slightly better computational efficiency of FGM when handling constraints.
- Using dynamical-system derived null-space matrix (42) improves computation stability and efficiency to a degree. Since the iteration number per sample is very similar, one can deduct that the computation efficiency per single iteration benefits from such null-space matrix.
- The simplification of the MPC problem via techniques described in chapter 6.4.2 significantly affects the solver performance, and reduces the computation time 20-times (Fig. 31), while maintaining close to original control quality (Fig. 29)
- For the simplified MPC problem, the FGM dual algorithm yields notably (cca 2 times) faster convergence. Additional solver acceleration (1,4x) can be gained by using the SVD-derived null space.

Table 5. Plasma MPC control QPgen solver performance summary, single-core implementation

Parameter \ solver		Original problem		Simplified problem	
	tolerance	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
QPgen FGM dual, DS null-space		/	/	385	13.1
QPgen FGM dual, SVD null-space	10^{-4}	/	/	286	9.7

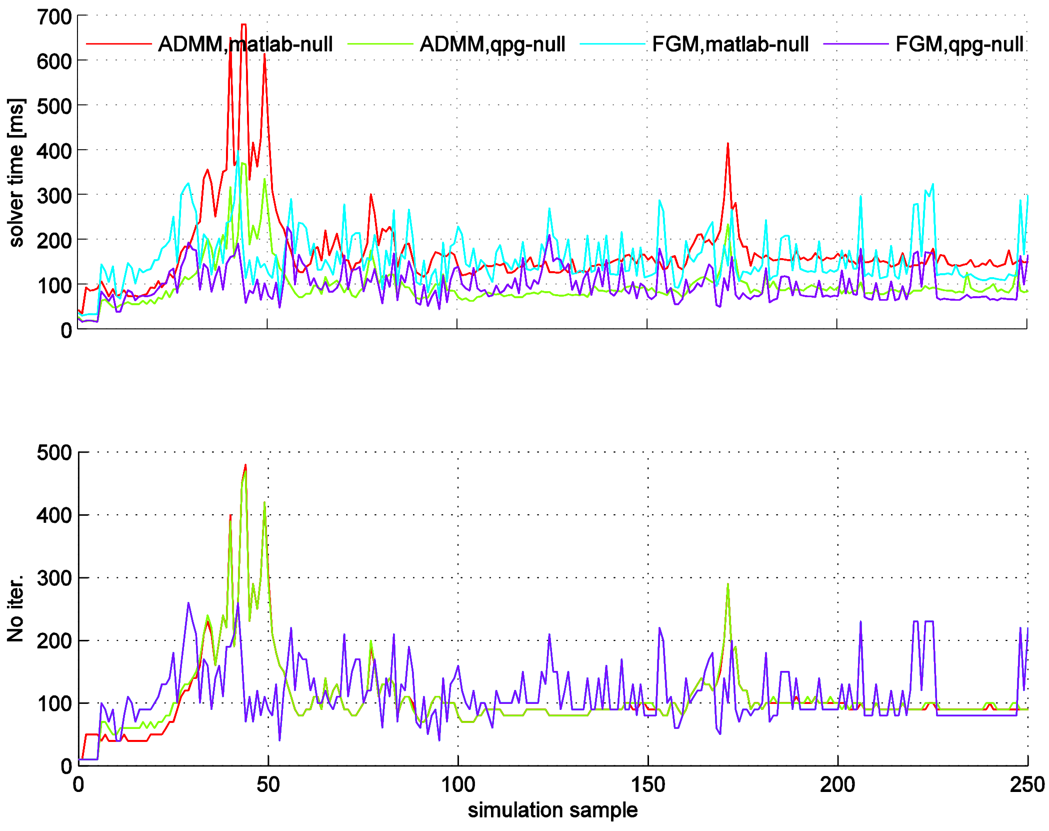


Fig. 33. QPgen solver execution time and number of iterations based on original MPC problem

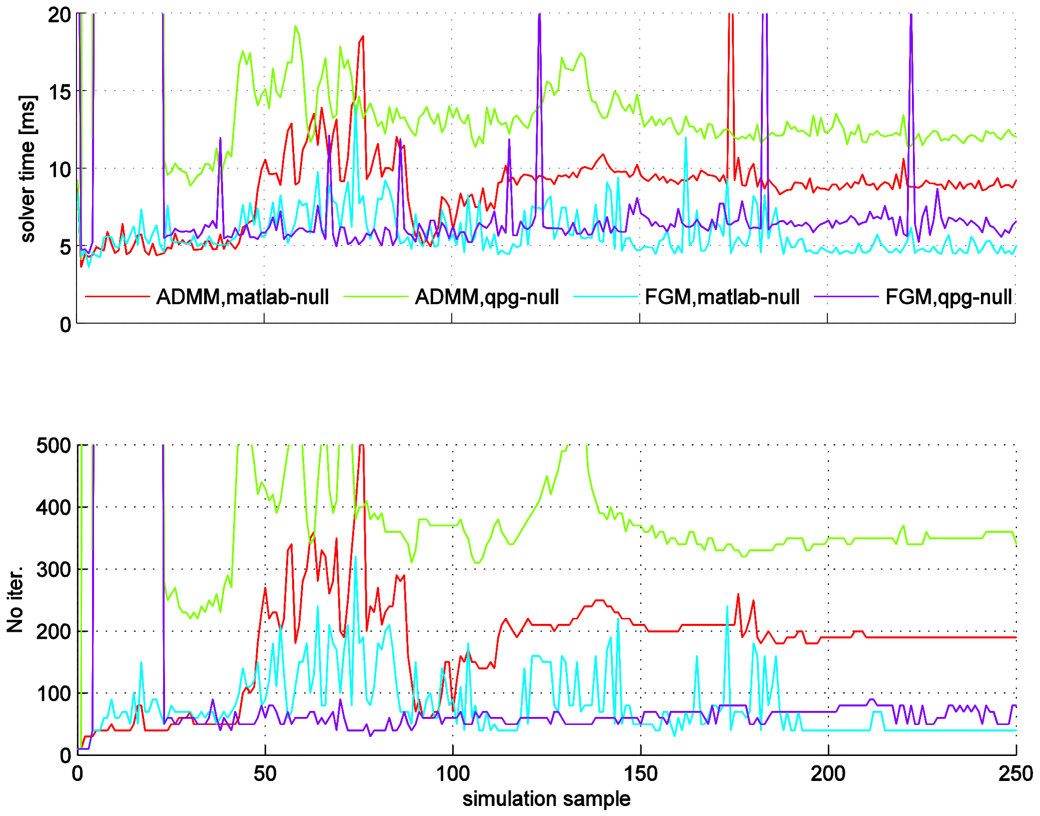


Fig. 34. QPgen solver execution time and number of iterations based on simplified MPC problem

6.5.3 Parallelization

Further acceleration was expected by employing a parallel implementation of the algorithms on a multi-core processor. QPgen includes support for parallel execution, but it was not functional and efficient for our example originally, so certain modifications were required. The Intel C++ compiler is used with parallel flag enabled to parallelize simple loops in code. It has the ability to analyze the dataflow in loops to determine which loops can be safely and efficiently executed in parallel, which has proven to be very efficient [42].

As the full performance advantage gained by parallelization cannot be achieved using Matlab (MEX) interface to the function, the parallelization performance study has been performed on off-line data. The generated code is compiled using the Intel C++ compiler to a standalone application and run separately. The results are presented in Figures 35-38 and summarized in the Table 6, where the average number of iterations and the minimum and peak execution times are listed. In this case computations have been performed using Intel Core i5, 2.7GHz processor, with 8 GB ram, by using a single core or all four cores. Because of spikes due to the operating system and other running programs in time measurements, multiple measurements are plotted, and such spikes are removed from the data reported in the table.

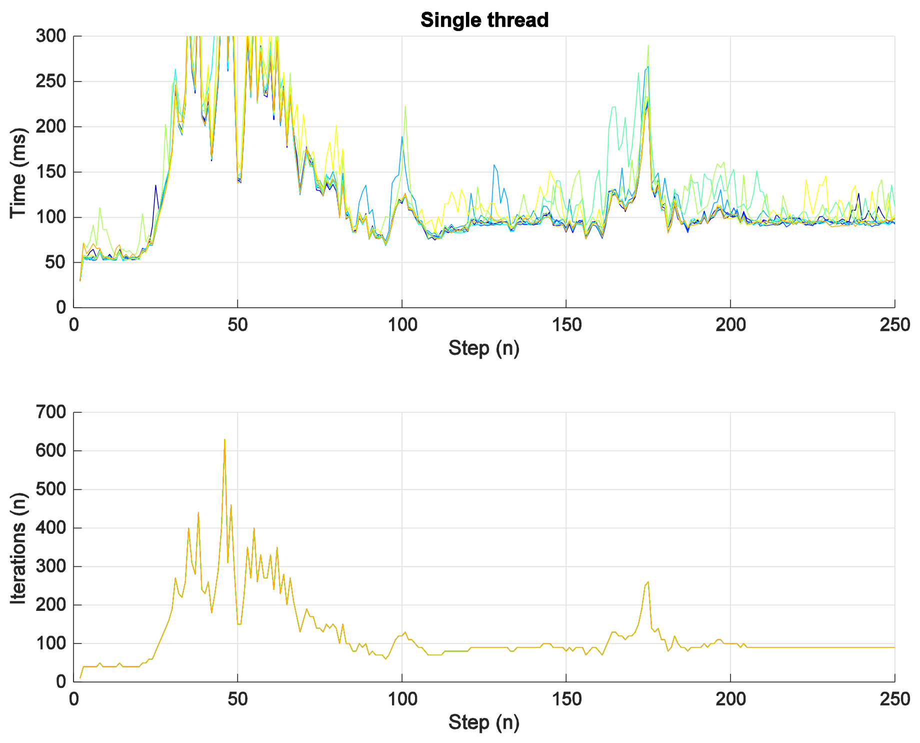


Fig. 35. QPgen solver executions time and number of iterations, off-line model-generated data based on original MPC problem, single thread

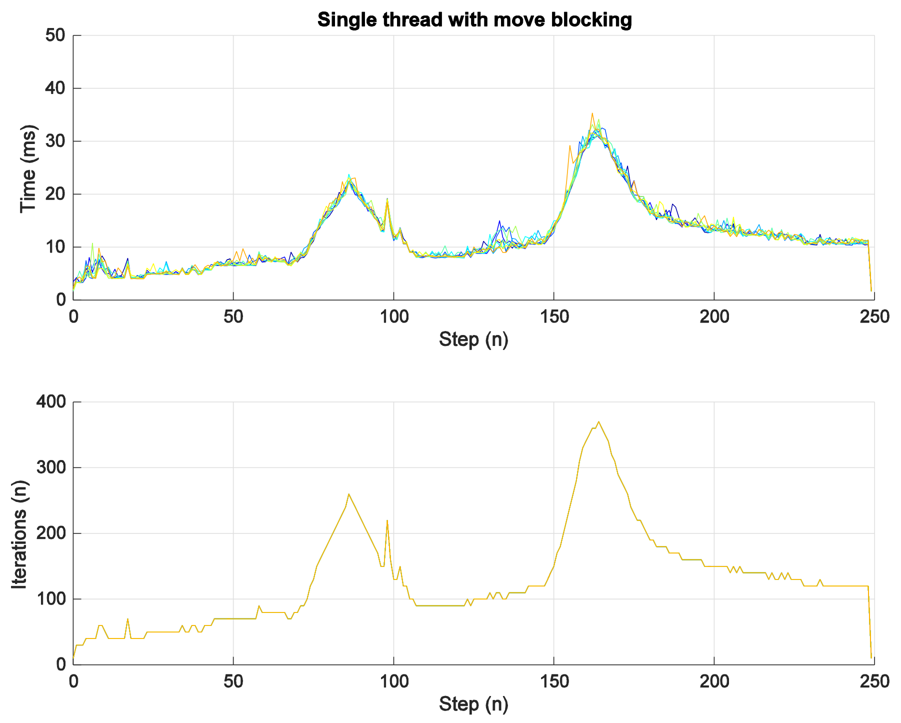


Fig. 36. QPgen solver executions time and number of iterations, off-line model-generated data based on simplified MPC problem, single thread

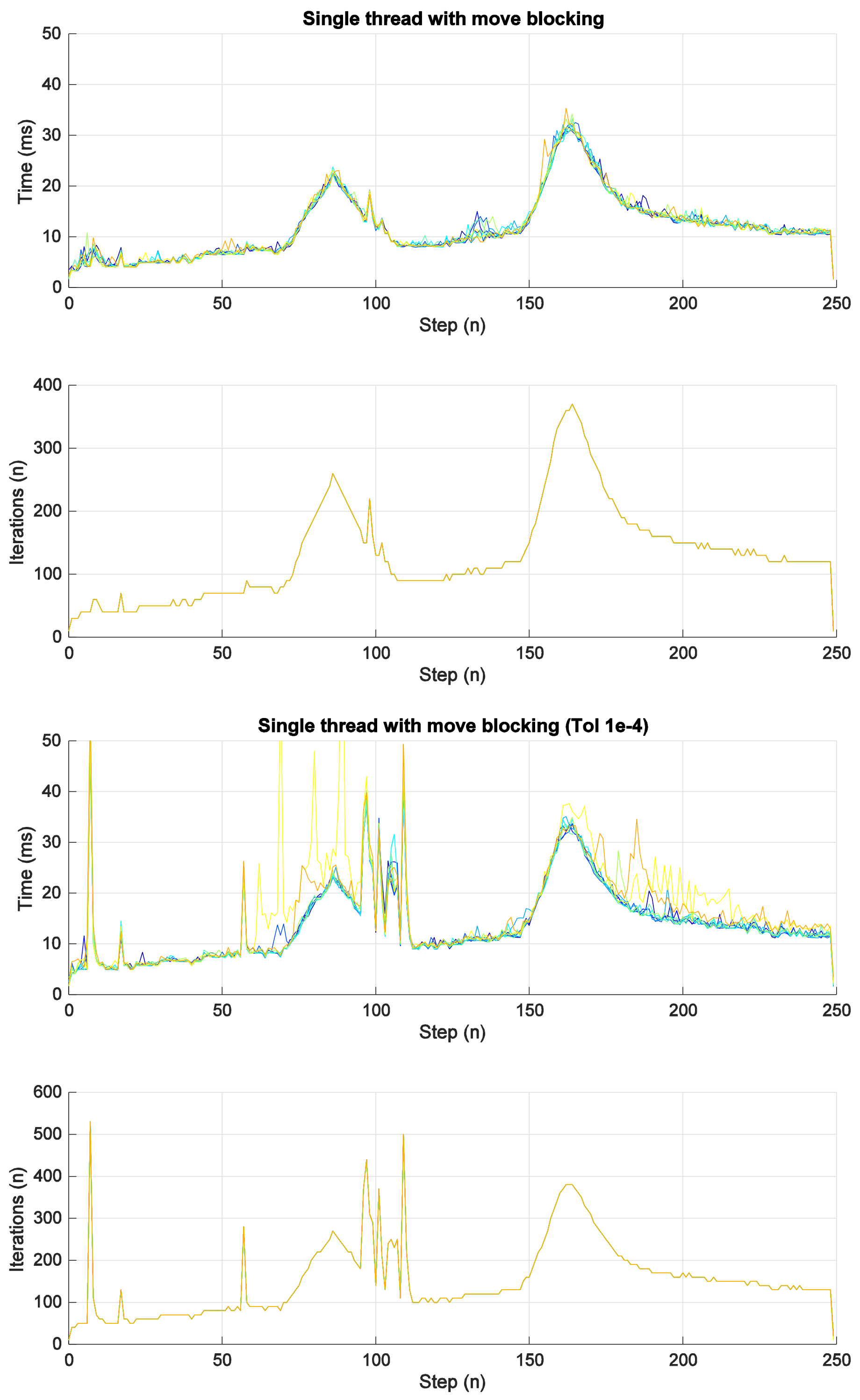


Fig. 37. QPgen solver executions time and number of iterations, run on off-line model-generated data based on original MPC problem, single thread

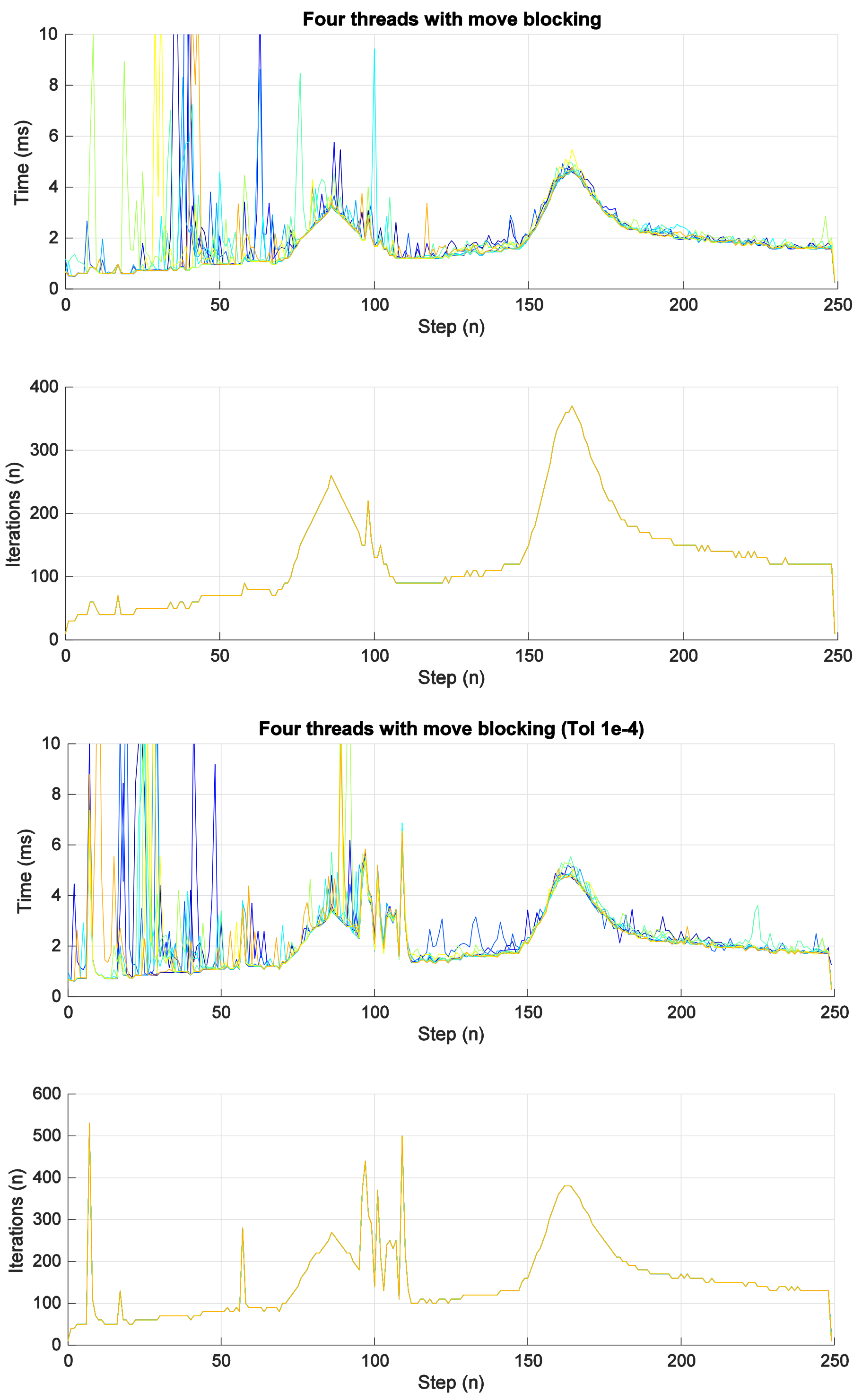


Fig. 38. QPgen solver executions time and number of iterations, off-line model-generated data based on simplified MPC problem, four threads

Table 6: Plasma MPC control QPgen solver performance summary, parallel implementation

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

7 Conclusions

In this report we describe the results of the first year of the three-year project FMPCFMPC. The report has two parts which roughly correspond to two parallel but interconnected work-packages, WP1: Plasma current and shape (CSC) control for ITER, and WP3: Fast online Quadratic Programming.

In Part I we expand the preliminary steps of MPC CSC with actual MPC controller design. Compared to a previous MPC CSC prototype, we have added set-point tracking and the ability to control more gaps, comparable to the "extreme shape controller" (XCS) of CREATE.

In Part II we present a survey of fast online QP methods, with emphasis on first-order methods. A case study is presented where a previous prototype version of MPC CSC for ITER is implemented. The dimensions of the system (11 manipulated variables, 6 controlled variables, 60 states, soft output constraints) are *much* larger that the dimensions of benchmark systems found in the available literature. The implementation is based on QPgen, an open-source solver comprising first-order methods FGM and ADMM. With complexity reduction techniques and certain modifications of QPgen we were able to achieve peak sample computation times in the order of 10 ms, which is a five-fold speed-up compared to the state-of-the-art commercial solver CPLEX, and is considered sufficiently fast for actual controller implementation.

References

[1] T. Bellizio et al., Control of elongated plasma in presence of ELMs in the JET tokamak, IEEE T. Nucl. Sci. **58**, 4 (2011)

[2] A. Neto et al., Exploitation of Modularity in the JET Tokamak VS System, Control Eng. Pract. **20**, 9 (2012)

[3] S. J. Qin and T. A. Badgwell, A survey of industrial model predictive control technology, Control Eng. Pract., **11** (2003)

[4] Kvasnica, M.: "*Real-time model predictive control via multi-parametric programming*". VDM verlag, Saarbrücken, 2009.

[5] M. N. Zeilinger et al., Real-time suboptimal Model Predictive Control using a combination of Explicit MPC and Online Optimization, IEEE Trans. Auto. Contr., **56** (2011) 1524–1534

[6] E. N. Hartley et al., Predictive control using an FPGA with application to aircraft control, IEEE Trans. Control Systems Technology, **22**, 3 (2014)

[7] M. Ariola and A. Pironti, An Application of the Singular Perturbation Decomposition to Plasma Position and Shape Control, Eur. J. Control **9** (2003) 410–420

[8] S. Gerkšič, G. De Tommasi, Improving magnetic plasma control for ITER Fusion Eng. Des. **89,** 9-10 (2014) 2477–2488

[9] R. Albanese and F. Villone, The linearized CREATE-L plasma response model for the control of current, position and shape in tokamaks, Nucl. Fus. **38** (5) (1998)

[10] R. Albanese et al., Plasma response models for current, shape and position control at JET, Fusion Eng. Des. **66–68** (2003)

[11] G. Ambrosino et al., Design of the Plasma Position and Shape Control in the ITER Tokamak Using In-Vessel Coils, IEEE Trans. Plasma Science **37** (2009)

[12] S. Gerkšič and G. De Tommasi, Vertical stabilization of ITER plasma using explicit model predictive control, Fusion Eng. Des. **88,** 6-8 (2013) 1082–1086

[13] S. Gerkšič and B. Pregelj, Tuning of a tracking multi-parametric predictive controller using local linear analysis, IET Control Theory Appl. **6**, 5 (2012), 1–11

[14] G. Ambrosino et al., Plasma Vertical Stabilization in the ITER Tokamak via Constrained Static Output Feedback, IEEE T. Contr. Syst. T. **19**, 2 (2011)

[15] Bemporad A., Morari M., Dua V. and Pistikopoulos E. N., The explicit linear quadratic regulator for constrained systems, Automatica **38**, 1 (2002), 3–20

[16] Borrelli F., Bemporad A. and Morari M.: "*Predictive Control for linear and hybrid systems*". Berkeley , Lucca , Zurich, 2015.

[17] Boyd S. and Vandenberghe L.: "*Convex Optimization*". Cambridge University Press, Cambridge 2004.

[18] Kouzoupis D.: "*Complexity of First-Order Methods for Fast Embedded Model Predictive Control (Master Thesis)*". Eidgenössische Technische Hochschule, Zürich 2014.

[19] Richter S.: "*Computational Complexity Certification of Gradient Methods for Real-Time Model Predictive Control (Dissertation)*". Eidgenössische Technische Hochschule, Zürich 2012.

[20] Giselsson P., Improving Fast Dual Ascent for MPC - Part II: The Embedded Case, arXiv (2014)

[21] O'Donoghue B. and Candès E., Adaptive Restart for Accelerated Gradient Schemes, Found Comput Math **15,** (2015), 715–732

[22] https://projects.coin-or.org/qpOASES

[23] Ferreau H. J., Bock H. G. and Diehl M., An online active set strategy to overcome the limitations of explicit MPC, Int. J. Robust Nonlinear Control **18,** (2008), 816–830.

[24] Ferreau H. J., Kirches C., Potschka A., Bock H. G., Diehl M., qpOASES, a parametric active-set algorithm for quadratic programming, Math. Prog. Comp. **6,** 4 (2014), 327–363

[25] Wang Y. and Boyd S., Fast Model Predictive Control Using Online Optimization, IEEE Trans Control Syst Techn **18**, 2 (2010), 267–278

[26] Mattingley J. and Boyd S., CVXGEN: a code generator for embedded convex optimization, Optim Eng **13**, (2012), 1–27

[27] Mattingley J., Wang Y. and Boyd S., Receding Horizon Control, Automatic Generation of High-Speed Solvers, IEEE Control Syst. Mag. **31**, 3 (2011), 52–65

[28] Huang Y., Ling K. V. and See S., Solving Quadratic Programming Problems on Graphics Processing Unit, ASEAN Engineering Journal **1**, 2 (2011), 76–86

[29] Domahidi A., Zgraggen A. U., Zeilinger M. N., Morari M. and Jones C. N., Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control, 51st IEEE Conference on Decision and Control, December 10-13, 2012. Maui, Hawaii, USA

[30] Frison G., Kufoalor K. M., Imsland L. and Jørgensen J. B., Efficient Implementation of Solvers for Linear Model Predictive Control on Embedded Devices, 2014 IEEE Conference on Control Applications (CCA), Part of 2014 IEEE Multi-conference on Systems and Control, October 8-10, 2014. Antibes, France

[31] http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/

[32] http://www.nag.co.uk/numeric/numerical_libraries.asp

[33] Gerkšič, S. and De Tommasi, G., A model predictive controller for ITER plasma current and shape control, 2014, unpublished

[34] http://www.control.lth.se/QPgen/index.html

[35] Boyd S., Parikh N., Chu E., Peleato B. and Eckstein J., Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, Foundations and Trends in Machine Learning **3**, 1 (2011), 1–122

[36] http://fiordos.ethz.ch/dokuwiki/doku.php

[37] https://github.com/giaf/hpmpc/blob/master/README.txt

[38] Patrinos P. and Bemporad A., An Accelerated Dual Gradient-Projection Algorithm for Embedded Linear Model Predictive Control , IEEE Transactions on Automatic Control **59**, 1 (2014), 18–33

[39] Mattingley, J. E., Wang Y. and Boyd S., Code generation for receding horizon control. In: Proceedings IEEE multi-conference on systems and control (2010), 985–992

[40] Frison G.: "*HPMPC regerence guide",* 2015.

[41] http://smart-cities-centre.org/wp-content/uploads/06-Frison-cities.pdf

[42] Optimizing Applications/Using Parallelism: Automatic Parallelization", Intel® C++ Compiler User and Reference Guide, November 2, 2011

[43] Kufoalor D. K. M., Richter S., Imsland L., Johansen T. A., Morari M. and Eikrem G. O., Embedded Model Predictive Control on a PLC Using a Primal-Dual First-Order Method for a Subsea Separation Process, MED 2014 Palermo (2014)