Semi-implicit Formulation of Proportional-integral Controller Block with Non-windup Limiter According to IEEE Standard 421.5-2016

Davide Fabozzi
Stefan Weigel
Bernd Weise

DIgSILENT GmbH
Heinrich-Hertz-Straße 9
72810 Gomaringen
Germany

D.Fabozzi@digsilent.de

Fortunato Villella

Elia Grid International
Rue Joseph Stevens 7 Joseph Stevensstraat
1000 Brussels
Belgium

Fortunato.Villella@eliagrid-int.com

***Abstract*—Appropriate models of excitation systems, voltage regulators, governors and power system stabilizers are fundamental for large-scale system stability studies which play a crucial role in modern system operations. The specification of the proportional-integral block with non-windup limiter, which is given in IEEE Standard 421.5-2016 for use in excitation system models, presents modelling challenges for standard power system time-domain simulation packages which are routinely used by transmission system operators for dynamic simulations. Three formulations of the same proportional-integral block with non-windup limiter are presented in this paper. The first two are input-output formulations: the first formulation is that described in the IEEE Standard 421.5-2016, the second one is an alternative formulation that does not exactly match the IEEE Standard, but is computationally more efficient. The main contribution of this paper is the introduction of a third, semi-implicit formulation which takes advantage of the possibility to change state variables from algebraic to differential and vice-versa, which is necessary to obtain a computationally-efficient simulation which closely matches the IEEE Standard 421.5-2016.**

I. Introduction

Large-scale system stability studies require accurate and computationally-efficient models of excitation systems, voltage regulators, governors and power system stabilizers. These kinds of studies play a crucial role in modern power system operations [1]. Since 1992, the IEEE maintains the IEEE Standard 421.5 [2] which was updated in 2005 [3] and, more recently, in 2016 [4]. This standard provides models which are valid for frequency deviations of 5 % from rated frequency and oscillation frequencies up to 3 Hz.

The specification of the Proportional-Integral (PI) controller block with non-windup limiter, which is given in IEEE Standard 421.5-2016 (see Fig. E.7 of [4]) for use in excitation system models, presents some modelling challenges for standard power system time-domain simulation packages which are routinely used by transmission system operators for dynamic simulations. Such challenges are inherent to the input-output causal modelling approach which is the modelling method of choice for commercial power system simulation packages.

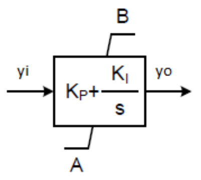

Fig. 1. PI controller block with non-windup limiter diagram in accordance with Figure E.7a of the latest IEEE Standard 421.5-2016 [4].

The authors are jointly working on the creation and optimization of a large-scale model for stability studies of the Belgian transmission system operated by Elia System Operator with the power system simulation software *PowerFactory* developed by DIgSILENT GmbH. Some of the challenges encountered by the authors in order to develop an accurate and computationally efficient model are detailed in this paper.

Three formulations of the same PI controller block with non-windup limiter depicted in Fig. 1 are presented in this paper. The first two are input-output formulations: the first formulation is that described in the IEEE Standard 421.5-2016, the second one is an alternative formulation that does not exactly match the IEEE Standard 421.5-2016, but is computationally more efficient than the first formulation.

The main contribution of this paper is the introduction of a third, semi-implicit formulation [5] of the equations of the PI controller block with non-windup limiter which takes advantage of the possibility to change state variables from algebraic to differential and vice-versa [6], which is necessary to obtain a computationally-efficient simulation that also closely matches the IEEE Standard 421.5-2016.

This formulation has been implemented in the standard version 2017 of the DIgSILENT *PowerFactory* software and is available to all DIgSILENT *PowerFactory* users through the introduction of a dedicated additional primitive function in the DIgSILENT Simulation Language (DSL) for the PI controller block with non-windup limiter according to IEEE Standard 421.5-2016.

The controller equations for the different controller models in DSL are provided and explained in the Appendix. The DSL syntax closely resembles the input-output equations, with the addition of some primitive functions to express switches and limiters [7].

A simple numerical example to show the short-comings of input-output causal modelling in the case of this PI controller block with non-windup limiter is shown in this paper along with results obtained using a power system example. Simulation trajectories obtained with different modelling techniques are compared in terms of accuracy and computational performance.

II. PI CONTROLLER BLOCK WITH NON-WINDUP LIMITER ACCORDING TO IEEE STANDARD 421.5-2016

IEEE Standard 421.5 introduced a PI controller block with non-windup limiter (see Fig. E.7 of [4]) for use in the excitation system models ST4C, ST6C, ST8C, AC7C and AC11C.

The IEEE Standard gives the model description in the form of input-output block diagram (see Fig. 2) with some additional written constraints (1):

$$\begin{aligned} if\ yaux \geq B \rightarrow \quad & yo = B, \quad \frac{dxs}{dt} = 0 \\ if\ yaux \leq A \rightarrow \quad & yo = A, \quad \frac{dxs}{dt} = 0 \\ otherwise \quad \rightarrow \quad & yo = yaux, \quad \frac{dxs}{dt} = K_I yi \end{aligned} \tag{1}$$

Equation (1) expresses that the state variable xs is kept constant, i.e. has a zero derivative, when the total control action $yaux$ is outside limits. Otherwise, i.e. when the total control action is within limits, the derivative of the state variable xs is equal to $K_I yi$. The latter state is referred to as the *integrating* state, while the limited states are referred to as the *lower* and *upper limited* state respectively.

Figure 3 shows the corresponding state-machine implementation with three states, i.e. one per each line in Eq. (1). Each of the three states is formulated as a differential equation in terms of the state variable xs.

While in principle it may be possible to find closed-form analytical solutions of simple ordinary differential equations, in practice the system of equations resulting from power systems have much greater complexity. Equations as given in Eq. 1 or in Fig. 3 must be discretized in time and algebraized according to an integration formula in order to be solved in the context of a power system simulation, together with all the other representative equations of the system under study.

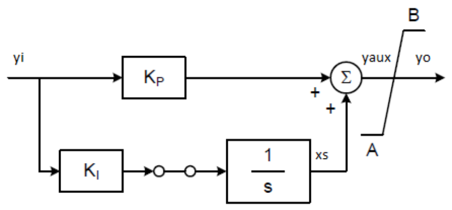

Fig. 2. PI controller block with non-windup limiter implementation in accordance to Figure E.7b of the latest IEEE Standard [4].

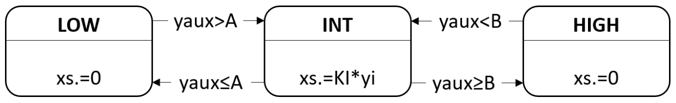

Fig. 3. State machine of PI controller block with non-windup limiter.

III. DISCRETIZATION AND ALGEBRAIZATION ISSUES

This section investigates a problem that may arise with the model presented in the previous section, as described by Eq. 1 or Fig. 3, once it has been discretized and algebraized.

A small example easily illustrates that this implementation cannot be robustly simulated in some cases. We take a case in which the state machine is already in the upper limited state (HIGH in Fig. 3).

At the moment of upper limitation, some time before the actual time, it could be stated that:

$$xs_0 + KP * yi_0 \quad = \quad B \tag{2}$$

where xs_0 and yi_0 were the values of state variable xs and input yi at the moment of limitation.

The state machine condition to switch back to the integrating state (INT in Fig. 3) is:

$$yaux \quad < \quad B \tag{3}$$

Since the state variable xs is constant in the upper limited state at its value xs_0 (since the moment the limitation became active) the condition (3) can be rewritten in terms of the input yi (assuming $KP > 0$ for simplicity):

$$yi \quad < \quad \frac{B - xs_0}{KP} = yi_0 \tag{4}$$

The state machine would then switch to the integrating state if condition (4) is fulfilled, i.e. if the input yi decreases below its value at limit activation yi_0. Let us assume that at a given integration step, the input yi, previously constant at yi_0, decreases from yi_0 to yi_1 with $yi_1 < yi_0$ and $\Delta yi = yi_1 - yi_0 < 0$.

Assuming an integration step size of length h and applying the backward Euler rule of integration, one can obtain the corresponding variation of $\Delta xs = xs_1 - xs_0$ as:

$$\Delta xs \quad = \quad KI * h * yi_1 \tag{5}$$

Assuming $yi_1 > 0$ and $KI > 0$, then it holds that $\Delta xs > 0$.

In order to obtain a feasible transition from the upper limited state to the integrating state, it must hold that $yaux < B$ at the end of the integration step h. This is equivalent to the condition:

$$\Delta xs + KP * \Delta yi \quad < \quad 0 \tag{6}$$

The substitution of (5) into (6) gives:

$$\begin{aligned} KI * h * yi_1 + KP * (yi_1 - yi_0) &< 0 \\ \frac{KI * h * yi_1}{KP * yi_0} + \frac{yi_1}{yi_0} - 1 &< 0 \\ \frac{yi_1}{yi_0} &< \frac{1}{1 + \frac{KI * h}{KP}} \end{aligned} \tag{7}$$

Equation (7) ultimately indicates that a feasible transition from the upper limited state to the integrating state can be obtained only when the input is reduced below a certain value that depends on the integration step size h and controller gains KP and KI.

Equation (7) has been derived for the Backward Euler integration formula. It is however possible to show that for other integration formulae feasible transitions exist depending on the values of h, KP and KI.

An interesting observation is that condition (7) is always satisfied at the limit $h \to 0$, since $yi_1 < yi_0$: this indicates that the state-machine implementation is correct in the continuous-time domain and that the transition problem arises because of the numerical discretization and algebraization.

The transition from the upper limited state to the integrating state is possible whenever the decrease in the proportional part of the control action $\Delta P = KP * \Delta yi$ is (as an absolute value) larger than the increase in the integral part of the control action $\Delta I = \Delta xs = KI * h * yi_1$. In the opposite case, neither the upper limited state nor the integrating state correctly represent the system.

In other words, the state machine malfunctions because its three states do not correctly model all possible modes of functioning of the system in the discretized time domain. An example of this would be a case in which the proportional part of the control action $\Delta P = KP * \Delta yi$ is (as an absolute value) larger than the increase in the integral part of the control action $\Delta I = \Delta xs = KI * h * yi_1$. This case only be simulated while accepting that the model keeps switching between infeasible integrating state in one integration step and infeasible upper limited state in the next.

IV. Alternative model of PI controller block with non-windup limiter

Section III illustrated that an infeasible transition may occur in the PI controller block as described by the IEEE Standard 421.5-2016 [4]. This section briefly describes an alternative model of PI controller block with non-windup limiter which suffers less from the described issues while not exactly matching the response of the IEEE Standard controller.

Figure 4 shows the block diagram and implementation of the alternative model. With respect to the model described in Fig. 2, this model implements the anti-windup limitation

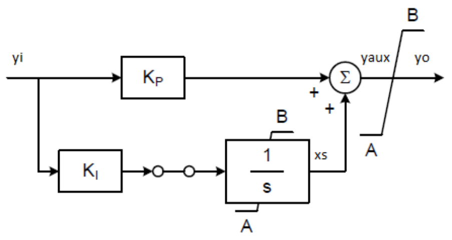

Fig. 4. Alternative PI controller block with non-windup limiter.

in a standard way on the state variable xs, i.e. disregarding the additional written constraints (1). This means that the derivative of the state variable xs is equal to $K_I yi$ while xs is within limits and zero otherwise.

Allowing the state variable to keep integrating when the total control action is outside limits and until the integral action is within limits leads to a delayed response at limit deactivation.

V. Semi-implicit formulation of PI controller block

This section describes the main contribution of the paper, which is a formulation of the PI controller block which strictly matches the behaviour described in the IEEE Standard while avoiding the infeasible transitions described in Section III.

This semi-implicit formulation [5] of the PI controller block with non-windup limiter takes advantage of the possibility to change state variables from algebraic to differential and vice-versa [6].

The infeasible transition described in Section III may be avoided with the introduction of an additional algebraic mode in the state machine that describes the functioning of the system while the proportional part of the control action $\Delta P = KP * \Delta yi$ is (as an absolute value) smaller than the increase in the integral part of the control action $\Delta I = KI * h * yi_1$. The additional ceiling state (CEIL in Fig. 5) represents the system behaviour when the control action is kept at the maximum by an integral action which is smaller than that given by the state derivative and opposite to the proportional action. This is best expressed by the algebraic constraint $yaux = B$ which results in the algebraic equation $0 = yaux - B$ or $0 = xs + KP * yi - B$.

The ceiling state may then transition to the integrating state, as soon as the proportional part of the control action $\Delta P = KP * \Delta yi$ is (as an absolute value) larger than the increase in the integral part of the control action $\Delta I = KI * h * yi_1$.

Conversely, the ceiling state may also transition back to the upper limited state when the proportional action is positive.

Figure 5 shows that the equations in the different modes may be differential (INT and HIGH states) or algebraic (CEIL state) for the same state variable xs. This is not possible in standard input-output formulations which require ordinary differential equations to be formulated as $\frac{dxs}{dt} = f(xs)$.

Conversely, the extended formulation $\Gamma \frac{dxs}{dt} = f(xs)$ [6], which is a particular case of the general semi-implicit for-

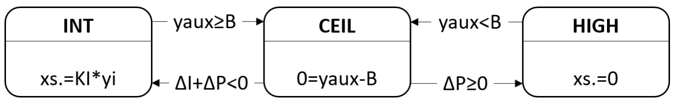

Fig. 5. State machine of semi-implicit formulation of PI controller block with non-windup limiter (only three states out of five are represented).

mulation $0 = F(\frac{dxs}{dt}, xs)$ [5], allows the changing of state variables from algebraic to differential and vice-versa.

In order to obtain a complete state machine, an algebraic flooring state should also be added to ensure feasible transitions between the integrating state and the lower limited state. The flooring state equations and transition conditions may be determined in a similar way to those derived for the ceiling state.

VI. Simple example

In this section, the control response of the different implementations with respect to a sinusoidal input are investigated, assuming the gains $KP = 0.5$ and $KI = 1$ and the limits $A = -1$ and $B = 1$.

For the example, a sinusoidal input is assumed in the form:

$$\begin{aligned} if\ time \leq 0 \quad &\rightarrow \quad yi = 0 \\ otherwise \quad &\rightarrow \quad yi = 2 * sin(time) \end{aligned} \tag{8}$$

These simulations have been performed in the standard version 2017 of the DIgSILENT *PowerFactory* software. A fixed integration step size h equal to 10 ms is used throughout the simulation. Larger step sizes and automatic step size adaptation strategies have also been tested, but are not reported here for brevity.

Figures 7 and 8 show the simulated trajectories. Models are labelled as follows:

- $LimSelect$: PI controller block with non-windup limiter according to IEEE Standard 421.5 (Section II);
- $LimLimState$: Simplified model of PI controller block with non-windup limiter (Section IV);
- $PiControl$: Semi-implicit formulation of PI controller block (Section V).

Figure 7 shows in the upper diagrams that the simulation trajectories of the $LimSelect$ and the $PiControl$ models match in a satisfactory way at a macroscopic level. Comparing any of these models trajectories with those obtained with $LimLimState$ in the lower left diagram shows that the latter model response does not closely match that of the IEEE Standard. The overall control response, however, is slightly delayed as shown in the lower right diagram, where the $LimSelect$ trajectory has been omitted because it is indistinguishable from the one obtained with $PiControl$.

The computational effort for the $PiControl$ and the $LimLimState$ models is very similar. The $LimSelect$ model, however, is more computationally demanding because of the infeasible transitions that occur as described in Section II.

This can be observed in Fig. 8 where a zoom is provided over the simulation time in which the infeasible transitions occur. The trajectories computed with the $LimSelect$ model do not follow the limit curve smoothly as is the case for the $PiControl$ model; instead they either overshoot in integrating mode or undershoot in upper limited mode, as shown in the two upper plots. Since the $LimSelect$ model has no ceiling state, this is obtained with the alternance of integrating and upper limited states for very small periods of time (usually one or two integration steps). The theoretical explanation for this numerical issue has been given in Section III.

Such discrete transitions perturb the simulation algorithm and would pose particular difficulties to automatic integration step size adaptation algorithms which would be forced by the many occurring events to use smaller steps than otherwise allowed. This may in turn greatly impact their performance.

Figure 8 also shows in detail in the lower plots the difference between the control action of the models; the $LimLimState$ response is delayed with respect to those obtained with both the $LimSelect$ and the $PiControl$ models.

VII. Power system example

This section illustrates the differences between the different modelling formulations on a power system example. The simulated test case is based on the *Test Case 2* from the ENTSO-E documentation on controller tests in test grid configurations [8]. The original test case has been modified to remove islanded operation and the simulated contingency is the connection of a 380 MW load on the machine terminal at $t = 1$ s. A fixed integration step size h of 1 ms is used.

While the IEEE Standard 421.5-2016 refers to excitation systems, the PI control used in this example is within a gas turbine-governor model for illustrative purposes. The simplified control scheme of this model [9] is illustrated in Fig. 6. The PI part of the proportional-integral-differential controller under study has been modelled according to the different modelling techniques discussed in this paper ($LimSelect$, $LimLimState$ and $PiControl$, as described in Section VI).

The input yi, the output yo and the mechanical power pm are indicated in Fig. 6. The state variable within the PI controller block is labelled xs in Figs. 9 and 10 which show the simulation trajectories of the $LimLimState$ and the $PiControl$ models. The $LimSelect$ trajectories have been omitted because they are indistinguishable from the latter.

Figure 9 shows that the control action on pm is slightly higher in the $LimLimState$ model than in the $PiControl$ model. In this example the trajectories are quite close but they may be more distant in other cases. The reason for their difference can be observed in Fig. 10 which shows that slightly before $t = 1.15$ s, when yo gets to its upper limit of 1, the xs variable computed with the $PiControl$ model remains constant, while the same variable computed with $LimState$ keeps integrating and eventually reaches its upper limit much later at $t = 1.4$ s. This delays the release of the output yo from its upper limit at around $t = 1.6$ s in the $LimLimState$ model, which ultimately is the cause of the discrepancies encountered.

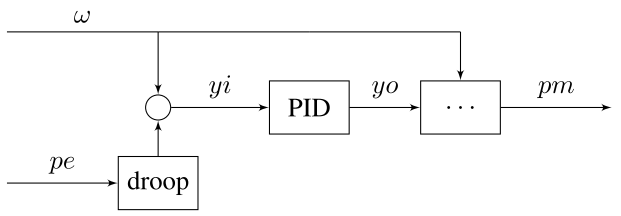

Fig. 6. Simplified block diagram of the example gas turbine-governor model showing the proportional-integral-differential controller under study.

VIII. Conclusion

This paper investigates the simulation behaviour of three implementations of the PI controller block with non-windup limiter introduced by IEEE Standard 421.5-2016 [4]. This IEEE Standard describes appropriate excitation systems and power system stabilizer models which are fundamental for large-scale system stability studies.

It has been shown that the standard implementation of the PI controller block with non-windup limiter suffers from the drawback of infeasible transitions that produce both slightly inaccurate and computationally demanding simulations. An alternative model with increased computational efficiency has been also investigated.

A novel implementation that makes use of a semi-implicit formulation [5] to change a state variable from algebraic to differential and vice-versa [6] leads to accurate behaviour and computationally efficient simulations. This is the main contribution of the paper.

The numerical issues described have been shown on a simple simulation example and on a power system example.

References

[1] F. Villella, S. Leclerc, I. Erlich, and S. Rapoport. PEGASE pan-European test-beds for testing of algorithms on very large scale power systems. In *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, pages 1–9, Oct 2012.

[2] IEEE recommended practice for excitation system models for power system stability studies. *IEEE Std 421.5-1992.*

[3] IEEE recommended practice for excitation system models for power system stability studies. *IEEE Std 421.5-2005 (Revision of IEEE Std 421.5-1992).*

[4] IEEE recommended practice for excitation system models for power system stability studies. *IEEE Std 421.5-2016 (Revision of IEEE Std 421.5-2005).*

[5] F. Milano. Semi-implicit formulation of differential-algebraic equations for transient stability analysis. *IEEE Transactions on Power Systems*, 31(6):4534–4543, Nov 2016.

[6] D. Fabozzi, A. S. Chieh, B. Haut, and T. Van Cutsem. Accelerated and localized newton schemes for faster dynamic simulation of large power systems. *IEEE Transactions on Power Systems*, 28(4):4936–4947, Nov 2013.

[7] DIgSILENT PowerFactory 2017 User Manual.

[8] ENTSO-E sub-group “System Protection and Dynamics”. Documentation on controller tests in test grid configurations. Nov 2013.

[9] Draft IEC 61970: Energy Management System Application Program Interface (EMS-API) - Part 302: Common Information Model (CIM) for Dynamics Specification. Sept 2013.

Acknowledgement

The authors wish to thank Dr Genevieve Lietz (DIgSILENT GmbH) who assisted in the proof-reading of the manuscript.

IX. Appendix

The controller equations for the different controller models in the DIgSILENT Simulation Language (DSL) are provided and explained in this appendix.

A. *LimSelect* - PI controller block with non-windup limiter according to IEEE Standard 421.5 (Section II)

Equation (1) can be rewritten in the DSL language as follows:

$$\begin{aligned} xs. &= select(yaux < B\ .and.\ yaux > A, KI*yi, 0) \\ yaux &= xs + KP*yi \\ yo &= lim(yaux, A, B) \end{aligned} \tag{9}$$

Equation (9) makes use of the (.) operator to indicate the derivative with respect to time and of two DSL primitives:

- $select$: evaluates the first argument (in this case whether $yaux$ is within its A,B bounds), then returns the second argument if the first argument is true and the third argument otherwise;
- lim: limits the first argument between the second argument (lower limit) and the third argument (upper limit).

B. *LimLimState* - Simplified model of PI controller block with non-windup limiter (Section IV)

The model represented by the block diagram in Fig. 4 can be rewritten in the DSL language as follows:

$$\begin{aligned} xs. &= KI*yi \\ yaux &= xs + KP*yi \\ yo &= lim(limstate(xs, A, B) + KP*yi, A, B) \end{aligned} \tag{10}$$

Equation (10) makes use of the DSL primitive $limstate$ whose first argument is the state variable; and the second and third arguments are the lower and upper limits, respectively. This primitive implements a simple anti-windup limiter on the state variable xs.

C. *PiControl* - Semi-implicit formulation of PI controller block (Section V)

The model represented by the state machine in Fig. 5 can be written as follows:

$$\begin{aligned} xs. &= KI*yi \\ yaux &= xs + KP*yi \\ yo &= picontrol\_const(xs, A, B, KP*yi) \end{aligned} \tag{11}$$

Equation (11) makes use of the DSL primitive $picontrol\_const$ which implements the state machine as described in Section V. The first argument is the state variable; the second and third arguments are the lower and upper limits, respectively; and the fourth argument is the proportional part of the control action.

D. Simple example (Section VI)

The sinusoidal input (8) is described in DSL as follows:

$$yi = select(time() < 0, 0, 2*sin(time())) \tag{12}$$

where the function $time()$ returns the current simulation time.

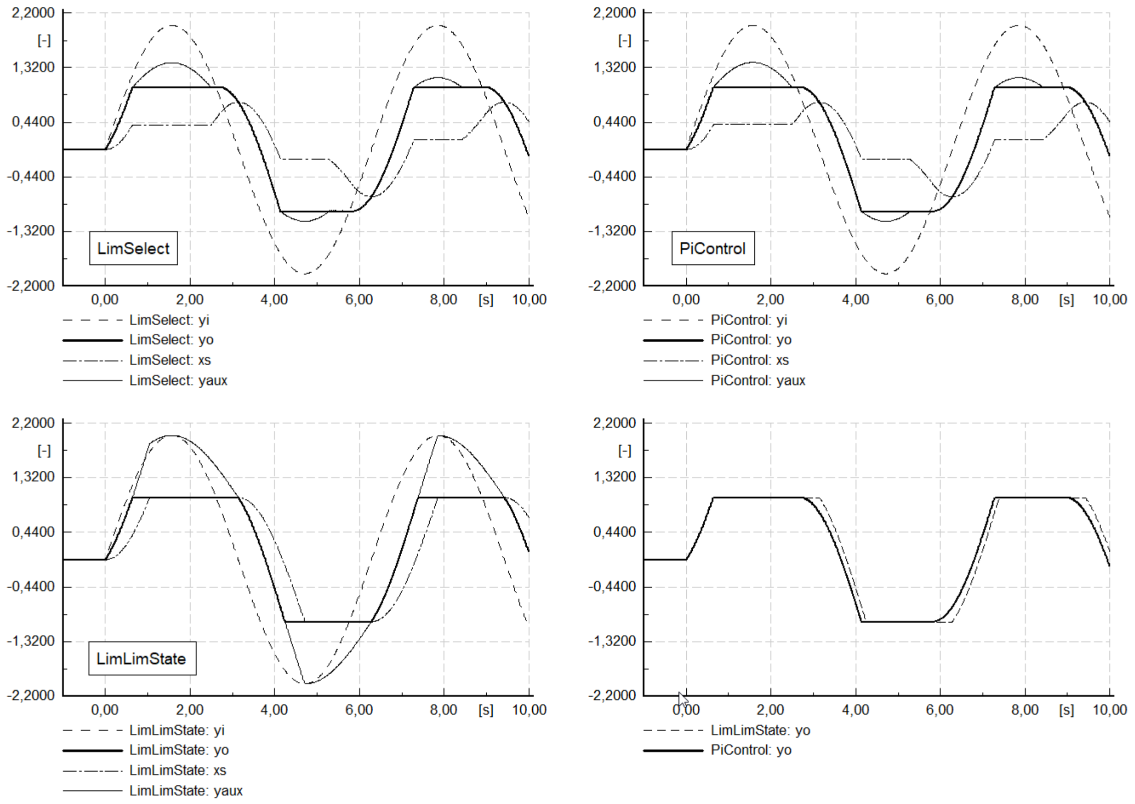

Fig. 7. Simulation results for the sinusoidal input example.

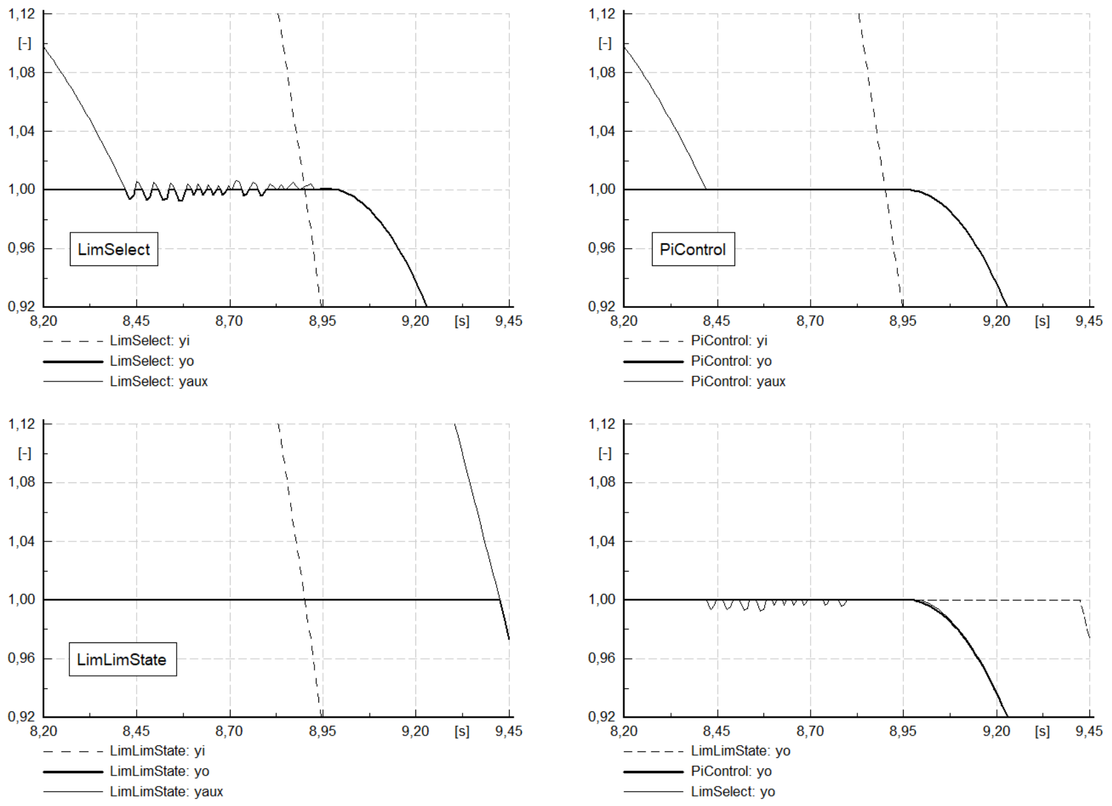

Fig. 8. Simulation results for the sinusoidal input example (zoom).

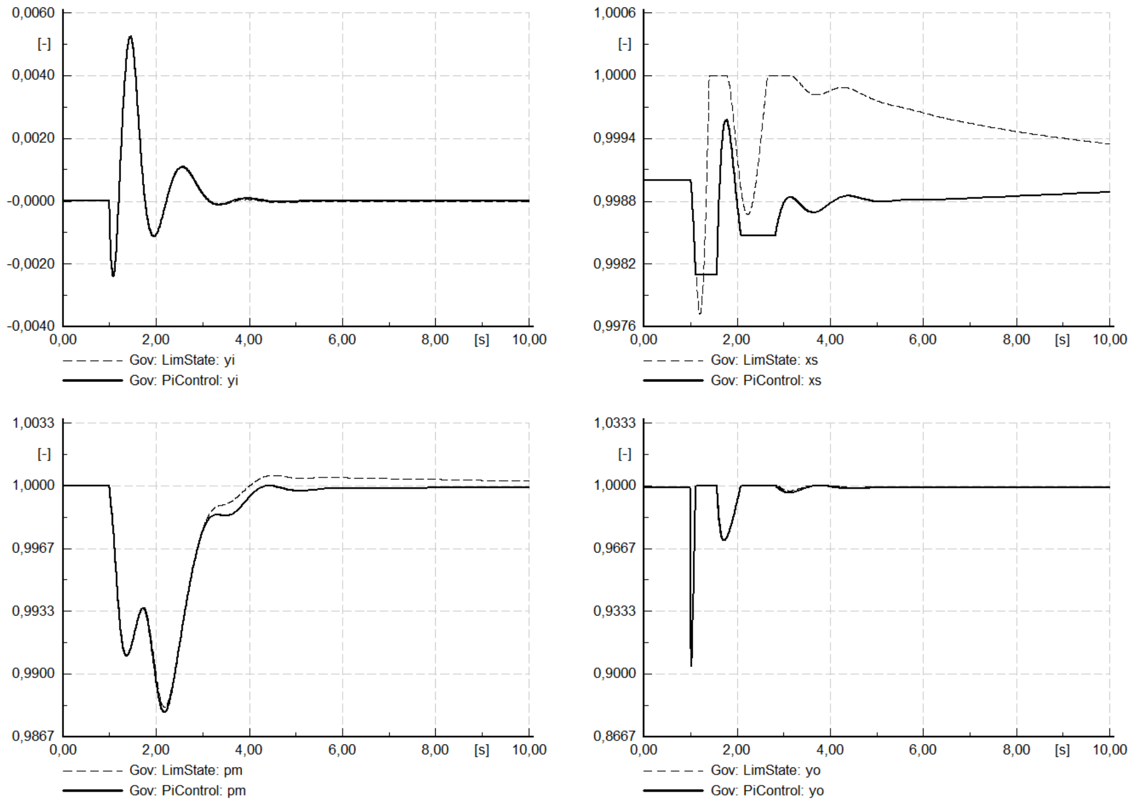

Fig. 9. Simulation results for the power system example.

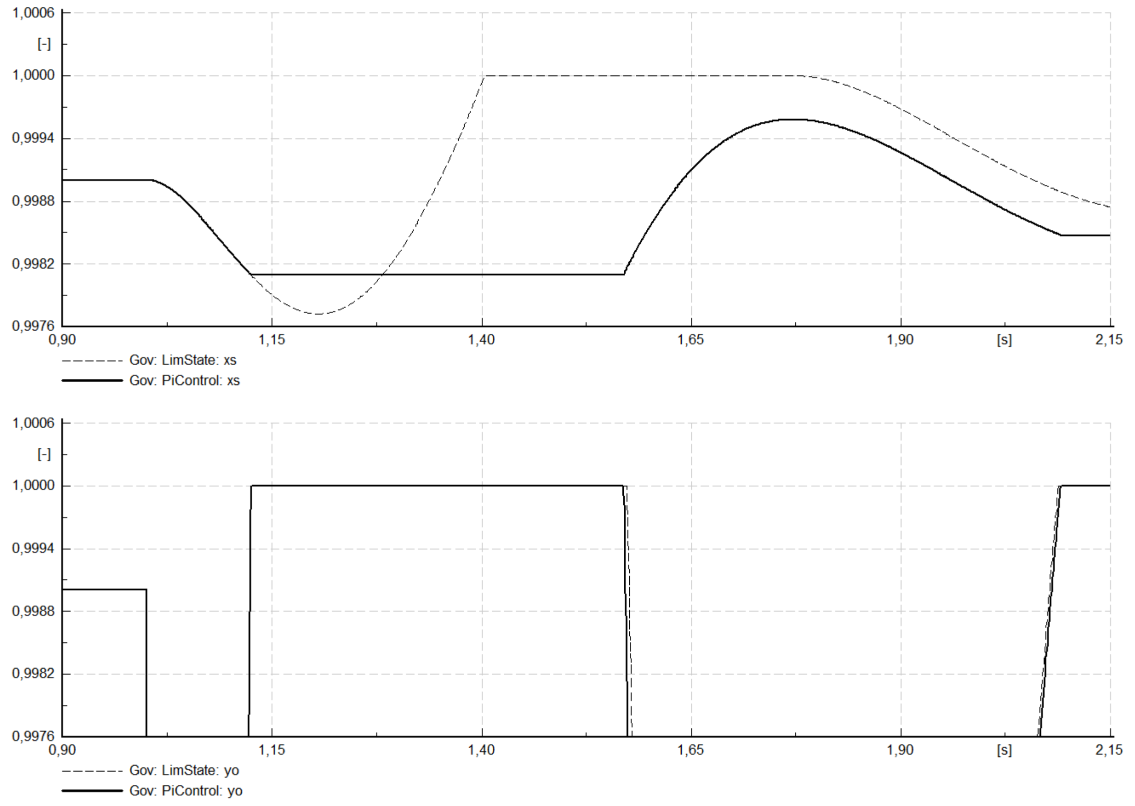

Fig. 10. Simulation results for the power system example (zoom).