Advanced PID Control Algorithms Built into the REX Control System

Pavel Balda * **Miloš Schlegel** **

* *Department of Cybernetics, University of West Bohemia, Pilsen, Czech Republic (e-mail: pbalda@kky.zcu.cz).*
** *Department of Cybernetics, University of West Bohemia, Pilsen, Czech Republic (e-mail: schlegel@kky.zcu.cz).*

Abstract: REX is an industrial control system which has been developed by the authors of this paper and by several their colleagues during the last decade. Control algorithms of REX are contained in a large function block library (block set). Controller blocks, including various PID controllers, cover a significant part of the library. This paper briefly explains main ideas of REX and it focuses on description of two advanced PID controller function blocks with built-in auto-tuning facilities. Both of these controllers use active identification experiment for the process identification, first of them uses a pulse experiment, second of them uses a relay experiment. After finishing of the identification experiment, the designed controller parameters are immediately computed in both cases. These controllers and some additional function blocks are presented in several examples demonstrating various control structures.

Keywords: PID controllers, PID control, Autotuners, Automatic control, Control systems, Industrial control, Real time.

1. INTRODUCTION

The general statement "there is a wide gap between academic research and practice" is valid also in automatic control. Application results of academic research in automatic control are usually only in the simulation form, without a direct practical verification.

Verification of a newly developed "academic" algorithm is a very long-term and usually iterative task. Utilization of a standard, commercially available programmable logic controllers (PLCs) for this purpose is a very laborious, ineffective and error prone because the developer has to transfer the algorithm from the simulation form to the target form, which are almost always different.

Matlab-Simulink (see MathWorks (2011b) for the current version) was probably the most frequently used simulation tool in the last decade. The Mathworks, the producer of Matlab, Simulink and other toolboxes, wanted to bridge the gap by a rapid prototyping toolbox Real-Time Workshop (RTW), which was recently renamed to Simulink Coder (see MathWorks (2011a)). RTW shortened the verification process of a new control algorithms because it allowed to generate the C-language code from Simulink block diagrams, to compile them to various target platforms (hardware devices) and to debug these algorithms online using Simulink environment and tools (e.g. the Scope block, parametric dialogs).

But still using Simulink and Real-Time Workshop have the following main disadvantages (from the authors' point of view):

- Lack of high-quality control algorithms suitable for industry
- High price of necessary tools (Matlab, Simulink, RTW, etc.) and suitable target devices.

These facts (but not only these) were important triggers of our own control system development, which would also help to bridge the mentioned gap.

2. REX CONTROL SYSTEM

REX (Balda et al. (2005)) is a software control system compatible with Matlab-Simulink. The compatibility is based on two facts:

- REX contains the large function block library RexLib (Schlegel et al. (2001)), which has been developed using Simulink. Each function block is a standalone Simulink C MEX file. The whole library is integrated with Simulink as `REX Industrial Blockset` in the Simulink library browser.
- REX uses the `.mdl` (Simulink model) text file format for control algorithm configuration. It means that the same configuration files can be used for the development of control algorithms in Simulink as well as for real-time control of the particular process or machine.

This section brings a brief description of REX architecture and several supported target platforms.

2.1 Architecture of REX

The simplified overall architecture of REX is depicted in fig. 1. REX is not a monolithic program, it has a modular structure. Real-time control is executed on a target device, while configuration (control algorithms) development

tools, visualization (Human-Machine Interface, HMI) and diagnostics tools are running on a single or more host computers. Target and host environments are connected with a communication layer.

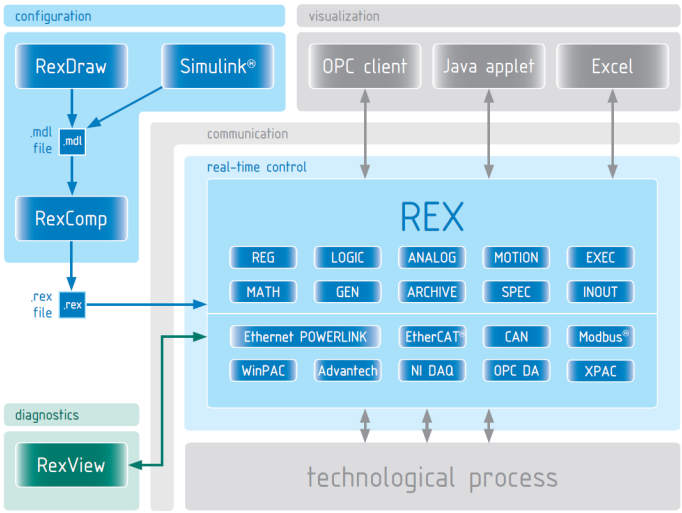


Fig. 1. Architecture of REX (simplified)

Real-time control algorithms are executed by the REX core (the RexCore program), see the biggest blue area in fig. 1. It contains function blocks in RexLib which are divided into smaller sub-libraries (placed above the horizontal line in the figure):

- `REG` – function blocks for regulation including PID controllers, the other advanced controllers, process models and many more.
- `LOGIC` – blocks for logic control (combinational and sequential).
- `ANALOG` – blocks for processing of analog signals.
- `MOTION` – motion control blocks.
- `EXEC` – blocks for configuration of the execution environment in RexCore.
- `MATH` – math functions and simple blocks.
- `GEN` – various signal generators.
- `ARCHIVE` – trending and alarming blocks which can store results to memory and disk archives.
- `SPEC` – special advanced function blocks, e.g. universally programmable function block
- `INOUT` – function blocks connecting input and output process signals to control algorithms

The detailed description of function blocks is contained in REX Controls (2011).

Inputs and outputs of a controlled technological process (or a machine) are available via input/output drivers (placed below the horizontal line in the figure). Drivers implement various communication protocols, e.g. Ethernet POWERLINK, EtherCAT, CAN, Modbus, OPC Data Access Client, or directly read inputs from and write outputs to plug-in cards or modules, e.g. WinPAC and XPAC by ICP DAS, Advantech I/O cards, National Instruments DAQ.

Control algoritms of REX can be configured in the RexDraw program or in Matlab-Simulink. RexDraw is independent of Matlab-Simulink and it does not require any license from The Mathworks products. But a user, who has the Matlab-Simulink license, can use Simulink for development and simulation of the control strategy before putting the control application into operation. REX target devices do not use the `.mdl` files directly, they are first compiled by RexComp to a more compact binary format `.rex`. RexComp is called internally from RexDraw as well as Simulink. Moreover, download of the compiled application to the target device, online monitoring and debugging is built into RexDraw.

REX itself does not implement visualization tools. Instead, it supports the OPC Data Access standard via the REX OPC server, which enables utilization of any OPC client available in the market. Moreover, REX communication protocol based on TCP/IP has been ported to Java classes, which can be used for communication with a Java application or an applet. Also, the Automation (formerly OLE Automation) is supported and it is suitable for communication with any software using Visual Basic scripting, e.g. Microsoft Excel.

Diagnostic information is especially crucial during the commissioning of the control system into operation. The RexView program (see left bottom part of fig. 1) provides real-time information about the timing of control tasks and input output drivers, inputs, outputs and parameters of each task, subsystem and function block in the control algorithm, allows the user to change block parameters (including controller parameters), displays real-time trend signals of selected variables, displays alarms and archive history, and many more.

2.2 REX Target platforms

Originally REX was developed on a standard PC platform in Windows operating system (OS). Since then, REX has been ported to several operating systems and target platforms. Fig. 2 shows some of them.

a) b) c)

Fig. 2. Examples of REX hardware platforms

The lowest price control units are represented by massively produced single board computers. Fig. 2a) depicts the ALIX 2D13 computer by PC Engines which is equipped with an AMD Geode processor, a Compact Flash slot, 3 Ethernet ports, 2 USB ports and 1 miniPCI slot. The REX implementation runs in GNU Linux or GNU Linux with real-time extension Xenomai. Typically, remote input/output modules communicating with control system via an industrial Ethernet are used. The shortest sampling period is 1 millisecond.

A very cost-effective solution is provided by the WinPAC-8000 programmable automation controller (PAC) by ICP DAS, see fig. 2b). Input and output signals are connected to plug-in modules. This PAC runs REX ported to Windows CE 5.0. The shortest sampling period is 2 milliseconds.

The highest computing power of REX supporting hardware is offered by industrial PCs (IPC). Fig. 2c) shows the V2402 fanless model by MOXA with Intel Atom processor, Compact flash slot, 2 Ethernet ports and 4 serial lines. The REX implementation runs on GNU Linux, optionally with Xenomai and the shortest sampling period is slightly under 1 millisecond. If a more powerful model is requested, an IPC with PCI expansion slots with plug-in boards can be used. Typically, an IPC (with PCI cards) by Advantech and IntervalZero ETS (formerly Phar Lap ETS) real-time operating system is used. The minimum sampling period of this solution is 0.1 millisecond.

3. PID CONTROL IN REX

Various PID controllers are essential function blocks of REX from the very beginning. All of them belongs to the RegLib sub-library of RexLib (see the REG box in fig. 1). This section deals with two degrees of freedom (2DOF) PID controllers which are a standard part of RexLib. The next section describes PID controllers with advanced autotuners.

3.1 2DOF PID controllers in REX

Standard REX PID controllers are implemented by a `PIDU`, `PIDUI` and `PIDGS` function blocks. Their symbols are depicted in fig. 3.

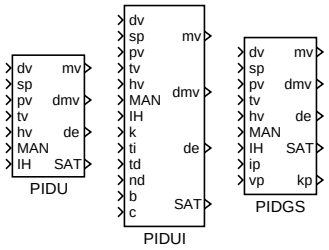


Fig. 3. Symbols of standard 2DOF PID controllers

The `PIDU` function block is a basic block for creating a complete PID controller or P, I, PI, PD, PI+S (PI controller with Smith predictor) controllers. The `PIDUI` function block differs from the `PIDU` block by connecting the controller parameters to the block inputs which can be changed dynamically (programmatically) from the control algorithm. The `PIDGS` is a gain scheduling variant of the `PIDU` block. The `PIDGS` can bumplessly switch at most six sets of basic PID controller parameters using the additional block inputs (`ip` or `vp`).

All three blocks can operate in automatic mode (`MAN=off`) or manual mode (`MAN=on`).

In the automatic mode (`MAN=off`), the block `PIDU` implements the PID control law with two degrees of freedom in the form

$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s}\left[W(s) - Y(s)\right] + \frac{T_d s}{\frac{T_d}{N}s + 1}\left[cW(s) - Y(s)\right] \right\} + Z(s)$$

where $U(s)$ is Laplace transform of the manipulated variable `mv`, $W(s)$ is Laplace transform of the setpoint variable `sp`, $Y(s)$ is Laplace transform of the process variable `pv`, $Z(s)$ is Laplace transform of the feedforward control variable `dv` and K (controller gain), T_i (integral time constant), T_d (derivative time constant), N (derivative filtering parameter), b (setpoint weighting factor of the proportional part) and c (setpoint weighting factor of the derivative part) are the parameters of the controller. The sign of the right hand side depends on the parameter `RACT`. The range of the manipulated variable `mv` (position controller output) is limited by parameters `hilim`, `lolim`. The parameter `dz` determines the dead zone in the integral part of the controller. The integral part of the control law can be switched off and fixed on the current value by the integrator hold input `IH` (`IH=on`). For the proper function of the controller it is necessary to connect the output `mv` of the controller to the controller input `tv` and properly set the tracking time constant `tt` (the rule of thumb is `tt` $\leq \sqrt{T_i T_d}$). In this way we obtain the bumpless operation of the controller in the case of the mode switching (manual, automatic) and also the correct operation of the controller when saturation of the output `mv` occurs (antiwindup). The additional outputs `dmv`, `de` and `SAT` generate the velocity output (difference of `mv`), deviation error and saturation flag, respectively.

In the manual mode (`MAN=on`), the input `hv` is copied to the output `mv` unless saturated. The overall control function of the `PIDU` block is quite clear from the diagram in fig. 4.

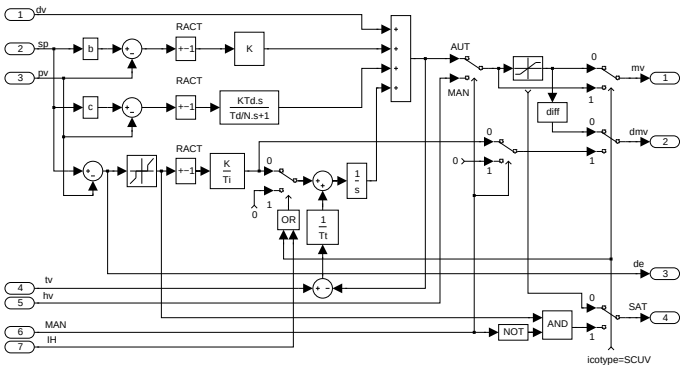
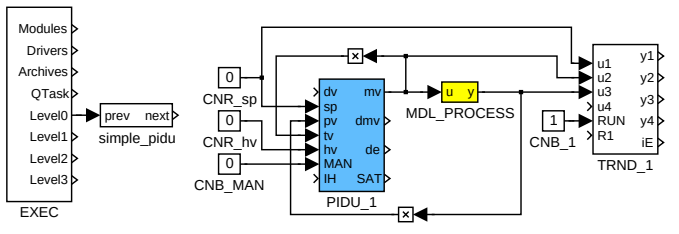


Fig. 4. Internal structure of the PIDU function block

A simple PID control loop with the `PIDU` function block is depicted in fig. 5.



a) Main file b) `simple_pidu` task

Fig. 5. Simple PID control loop in REX

Unlike Simulink, REX uses at least two files for its configuration. The first main file in sub-fig. 5a) specifies the real-time core parameters (the block `EXEC`). Further, additional

modules, drivers, archives and tasks (the highest priority QTask and tasks connected to different priority levels) can be connected to the `EXEC` block. In this case, only one task `simple_pidu` is used.

The control algoritm of the `simple_pidu` task is shown in sub-fig. 5b). Controlled process is simulated by the `MDL_PROCESS` second order block with dead time. It is controlled by the `PIDU_1` controller. Real constants `CNR_sp` and `CNR_hv` correspond to the controller setpoint `sp` and the value `hv` which is set to the controller output `mv` in the manual mode. The controller in fig. 5b) works in the automatic mode because the output of the binary constant `CNB_MAN` is `0` (`off`). The controller variables `sp`, `pv` and `mv` are being strored to the trending block `TRND_1`. Small crossed squares correspond to the `Loop break` blocks indicating the feedback edges which are temporarily removed from the control scheme to determine the proper execution order of all blocks.

More detailed information about the 2DOF PID control algorithms in REX can be found in REX Controls (2011).

3.2 PID controller with pulse width modulated output

Pulse width modulation (PWM) is a well known technique for proportional conversion of a continuous signal `u` from the interval $[0;1]$ to the ratio T_{on}/T_{pm} where the digital output `UP` is `on` for the T_{on} time of the modulation period T_{pm}. The output signal `UP` is `off` (resp. `on`) for the whole period T_{pm} for `u`=0 (resp. `u`=1). In the case of two binary outputs `UP` and `DN` the input interval $[-1;1]$ can be used where the negative values are mapped analogously to the `DN` output.

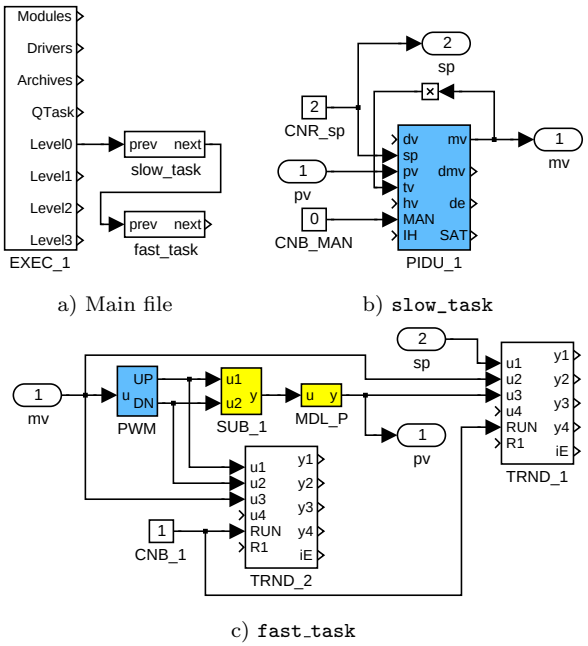


Fig. 6. PID control example with PWM output

For discrete implementation of the PWM algorithm, the ratio T_s/T_{pm} where T_s is the sampling period determines resolution and precision of PWM because the algorithm can set outputs from `on` to `off` only once during the sampling period. In other words, the shorter the sampling period, the higher the PWM accuracy. This situation is illustrated with the configuration in fig. 6. The `PWM` block and the process model are contained in the `fast_task`, the `PIDU` controller is in `slow_task`.

REX implementation of the `PWM` block is a more sophisticated, the user can define minimum width of the output pulse, minimum delay between output pulses, minimum delay between `UP` and `DN` pulses (reversing the direction) etc., see REX Controls (2011) for more details.

3.3 PID controller with three state output

Not all actuators are equipped with an analog input signal which can be directly connected to a PID controller output. Besides PWM, also actuators with two digital inputs are used very often. Control actions like change a value "up"–"do not change it"–"down" or rotate a motor to "left"–"stay in place"–"right" can be implemented by means of these two digital inputs.

Motorized valves use this strategy, which is demonstrated in fig. 7 (the project main file is omitted).

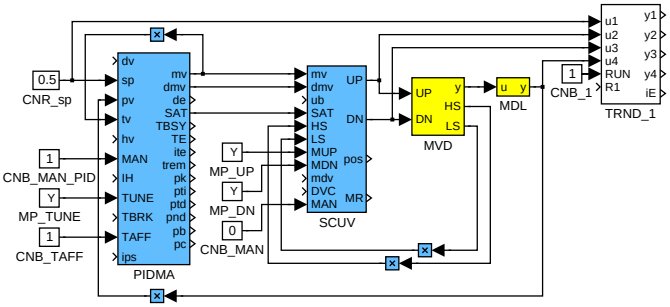


Fig. 7. PID control example with three state output

The `PIDMA` block which is a PID controller with built-in autotuner (see the next section) is used as a primary controller instead of the `PIDU` block. The block `SCUV` is used in the step controller loop when the position signal is not available. The primary controller is connected with the block `SCUV` using the block inputs `mv`, `dmv` and `SAT`.

If the primary controller uses `PI` or `PID` control law then all three inputs `mv`, `dmv` and `SAT` of the block `SCUV` are sequentially processed by the special integration algorithm and by a three state element. Pulse outputs of the three state element are further shaped in such a way that the minimum pulse duration time and minimum pulse break time are guaranteed at the block outputs `UP` and `DN`.

The position `pos` of the valve is estimated by an integrator with the specified time constant `trun`. If signals from high and low limit switches of the valve are available, they should be connected to the inputs `HS` and `LS`.

If the primary controller uses `P` or `PD` control law then the deviation error of the primary controller can be eliminated by the bias `ub` manually. In this case, the control algorithm is slightly modified, see REX Controls (2011) for details.

There is also a group of input signals for manual control available. The manual mode is activated by the `MAN=on` input signal. Then it is possible to move the motor back and forth by the `MUP` and `MDN` input signals. It is also

possible to specify a position increment/decrement request by the `mdv` input. In this case the request must be confirmed by a rising edge in the `DVC` input signal.

The controlled valve is simulated by the `MVD` (Motorized Valve Drive) block and the process by the `MDL` block. Further, a manual pulse (`MP`) block is used three times (`MP_TUNE`, `MP_UP` and `MP_DN`). The `MP` block generates the binary pulse of the specified duration at its output after a binary parameter is set to `on`. The selected signals are stored to the trend block `TRND_1`.

4. PID CONTROLLERS WITH AUTOTUNERS

The most advanced controllers in RexLib are equipped with autotuners. Two of them (`PIDMA` and `PIDAT`) are PID controllers which are depicted in fig. 8. These controllers, with the same control function as the `PIDU` block, are briefly described in the next subsections.

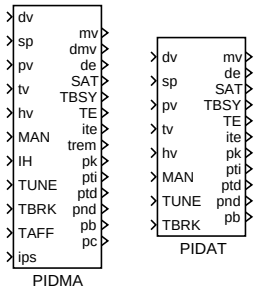


Fig. 8. Symbols of the PIDMA and PIDAT function blocks

4.1 PID controller with pulse tuning experiment

The `PIDMA` (PID controller with Moment Autotuner) block uses a pulse experiment for the controlled process identification. The approach is based on papers Schlegel and Čech (2005) and Schlegel and Večerek (2003).

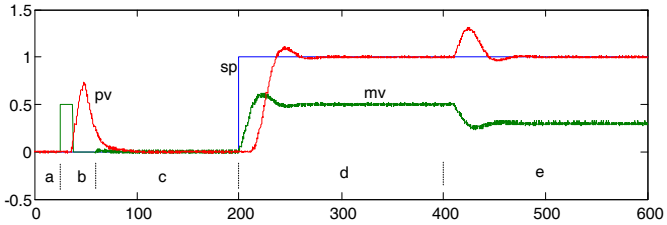


Fig. 9. Tuning experiment of the `PIDMA` controller

The autotuner function is illustrated in fig. 9. The experiment consists of the following phases:

a Waiting for the steady state.
b Pulse experiment itself, which is determined by its amplitude (the `amp` parameter of the PIDMA block) and by a threshold (the `dy` parameter) of the process variable (`pv`). When the difference between the current value of `pv` and its value in the preceding steady state exceeds the threshold the pulse is automatically finished.
c Controller works in automatic mode with a newly computed parameters.
d Step response of the closed loop.
e Response to the disturbance.

The detailed description of the `PIDMA` block can be found in the manual REX Controls (2011).

4.2 PID controller with relay tuning experiment

The `PIDAT` (PID controller with relay AutoTuner) block uses a relay experiment for the controlled process identification. The relay autotuner is based on the method described in Schlegel (2011).

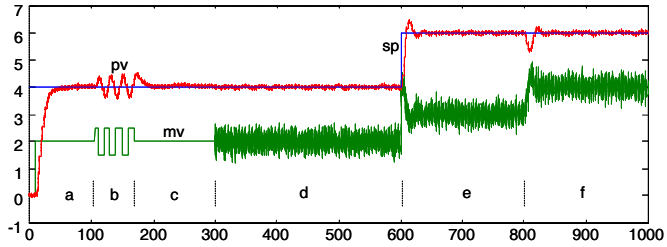


Fig. 10. Tuning experiment of the `PIDAT` controller

Fig. 9 shows a relay experiment example, which consists of the phases:

a Waiting for the steady state.
b Relay experiment itself, which is determined by its amplitude (the `amp` parameter of the PIDAT block) and by a maximum number of half periods of the experiment (the `n1` parameter)..
c Controller works in manual mode.
d Controller works in automatic mode with a newly computed parameters.
e Step response of the closed loop.
f Response to the disturbance.

Again, more details can be found in the manual REX Controls (2011).

5. EXAMPLES OF ADVANCED CONTROL STRUCTURES

5.1 Center seeking control

Center seeking control is a control strategy which can be used when two actuators (usually valves) with different ranges (fine and coarse) of the control action are available. Fig. 11 shows one of the possible REX implementations.

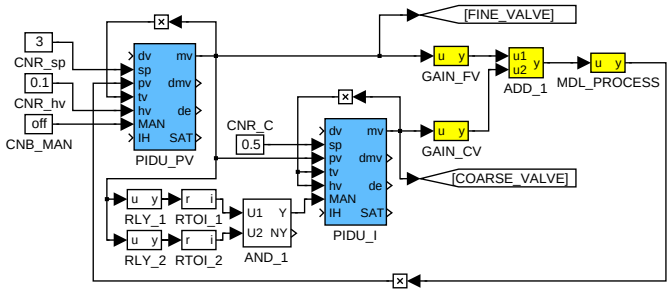


Fig. 11. Center seeking control block diagram

Control function is always performed by the fine (small) valve. The coarse (big) valve is controlled by the integration controller so that the fine valve is in the centre of

its range. It can be ensured by an appropriate choice of the relay blocks `RLY_1` and `RLY_2`. The integration controller `PIDU_I` should be sufficiently slow (i.e. its integral time constant must be sufficiently large) so that the main control loop with the `PIDU_PV` controller remains stable. The position of the coarse valve will be corrected by switching the `PIDU_I` controller to the automatic mode (`MAN=off`) whenever the fine valve leaves the specified (recommended) interval $(0.25; 0.75)$ and correction is finished when the (recommended) interval $(0.45; 0.55)$ is reached. The `RTOI_1` and `RTOI_2` blocks perform only type conversion of real to integer numbers which are inputs of the logical `AND` block.

5.2 Nuclear reactor power controller

A power controller of the nuclear reactor LR0 (Schlegel and Balda (2008)) in Nuclear Research Institute Řež, Czech Republic, is one of the most interesting applications of REX. The nuclear reactor power control is a very demanding task because the controller must work in the power scale of seven orders of magnitude. The overall control scheme is depicted in fig. 12.

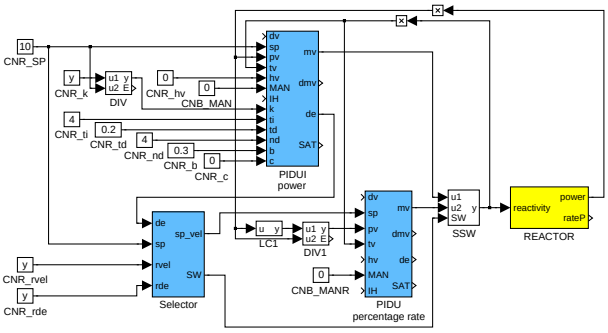


Fig. 12. Selector control of nuclear reactor

The control algorithm is based on a suitable switching of two controllers which is provided by the `Selector` subsystem. The active controller is selected by its output `SW` connected to the `SW` input of the `SSW` (Simple SWitch) block. When `SW=off`, the `PIDUI` controller is active (its output `mv` is copied through the `SSW` block to the reactor model). The `PIDU` controller is active when `SW=on`.

The `Selector` subsystem makes the `PIDUI` controller active when the reactor power (`pv`) is near to the desired power (`sp`), i.e. the absolute value of the deviation error (`de=sp-pv`) of the `PIDUI` is small relatively to the `sp`, which is specified by the `CNR_rde` constant.

Otherwise, the `PIDU` intergration controller is active. It controls the relative velocity of the reactor power increase (decrease), which is determined by the `CNR_rvel` constant. The `sp_vel=rvel` (`sp_vel=−rvel`) for power increasing (decreasing).

6. CONCLUSION

This paper briefly inform about the possibilities of the PID control in the REX control system, which are demonstrated on several examples. The examples in figures 5, 6, 7 and 11 are included in the `EXAMPLES\REX_TUTOR` subdirectory of the REX for Windows installation directory.

But REX offers more than only PID control, other advanced controllers (several of them with autotuners) are available, e.g. the `PSMPC` block (Pulse-Step Model Predictive Controller, Schlegel and Sobota (2008)), `SMHCCA` block (Sliding mode Heating Cooling Controller with Autotuner, Schlegel and Mertl (2006)), and `SC2FA` block (State Controller for 2nd order system with Frequency Autotuner). Very interesting are also sequential control blocks `ATMT` (automat) and `EATMT` (extended automat) supporting the Sequential Function Chart (SFC) formalism (IEC 61163-3 standard) including a user-friendly editor (Kocánek and Balda (2011)), and many more.

Free demonstration version of REX is available at
`www.rexcontrols.com`

ACKNOWLEDGEMENTS

This work has been partially supported by the Czech Ministry of Industry and Trade, project No. FR-TI1/394. This support is very gratefully acknowledged.

REFERENCES

Balda, P., Schlegel, M., and Štětina, M. (2005). *Advanced control algorithms + Simulink compatibility + Real-time OS = REX*, 121–126. Elsevier, Oxford.

Kocánek, M. and Balda, P. (2011). General sequential function charts editor. In *12th International Carpathian Control Conference (ICCC), 2011*, 191–194. Velké Karlovice, Czech Republic.

MathWorks (2011a). *Simulink Coder User's Guide*. The MathWorks, Inc., Natick, MA, USA.

MathWorks (2011b). *Simulink User's Guide*. The MathWorks, Inc., Natick, MA, USA.

REX Controls (2011). *REX system function blocks – reference manual*, 2.03 edition.

Schlegel, M. (2011). Exact revision of the ziegler-nichols frequency response method. In *IASTED International Conference Control and Application, 2002*, 121–126. Cancun, Mexico.

Schlegel, M. and Balda, P. (2008). Power controller of nuclear reactor. In *Process Control 2008*, 1–7. University of Pardubice, Pardubice, Czech Republic.

Schlegel, M., Balda, P., and Štětina, M. (2001). C mex blockset for industrial control with examples. In *Proceedings of the 9th Matlab Conference (in Czech)*, 361–369. Vydavatelství VŠCHT, Prague.

Schlegel, M. and Mertl, J. (2006). Sliding mode controller for heating-cooling temperature processes. In *MATEO - The European Network of Mechatronics Centres and Industrial Controllers 2006 (in Czech)*, 171–177. University of West Bohemia, Železná Ruda, Czech Republic.

Schlegel, M. and Sobota, J. (2008). Simple pulse-step model predictive controller. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 8401–8406. Elsevier, Korea.

Schlegel, M. and Čech, M. (2005). *Computing value sets from one point of frequency response with applications*, 325–330. Elsevier, Oxford.

Schlegel, M. and Večerek, O. (2003). Robust design of smith predictive controller for arbitrary order lag systems with dead time. In *Process Control 03, 2003*, 1–10. Bratislava, Slovakia.