Policy Supervised Exact State Reconstruction in Real-Time Embedded Control Systems

Mariusz Pelc * **Richard J. Anthony** ** **Witold Byrski** ***

* *Opole University of Technology, 45-271 Opole, ul. Mikolajczyka 5, Poland, (e-mail: m.pelc@po.opole.pl).*
** *The University of Greenwich, Greenwich, SE10 9LS London, UK, (e-mail: r.j.anthony@gre.ac.uk)*
*** *AGH University of Science and Technology, 30-059 Krakow, Al. Mickiewicza 30, Poland (e-mail: wby@ia.agh.edu.pl)*

Abstract: In case of computer control systems which require application of optimal control algorithms, it is required to have immediate access to state variables of the controlled system. When these state variables are unavailable or cannot be measured, it is necessary to use state observers that the reconstruct required system state based on extrapolation of the system input and output signals. However, the latest trends in control systems applications express additional requirements regarding increased flexibility and ability to adjust to changing environment conditions. This results in a combination of classic control algorithms with various modern techniques of supervision and control which leads to design of hierarchical multilayer computer control systems. In some cases, depending on deployment environment, there may appear some constraints or additional requirements to be satisfied, which may reduce the range of possible solutions. This applies especially to embedded systems, where resource constraints are always an issue which has to be taken into account.
This paper presents a new architecture for control systems with hierarchical exact state observers. The architecture incorporates policy-based computing to facilitate flexible, reconfigurable supervision logic and the achievement of environment awareness.

Keywords: exact state observers, policy based computing, computer control systems

1. INTRODUCTION

Nowadays in the control systems domain one can meet many implementations that combine well known and commonly used classic control algorithms with various systems of higher level supervision. Most of these systems take advantage of expert knowledge, other ones use learning techniques. A combined control systems takes advantage of well known and popular control algorithms on the one side and flexibility and means of dynamic reconfiguration provided by additional supervision levels. This reconfiguration is considered to be a change in the decision making logic that affects the supervised lower level system, changing its behavior for a given set of circumstances..

Embedded platforms introduce many restrictions regarding possible methods of dynamic reconfiguration. These restrictions result from the typical constraints met in embedded systems, which include: limited processing power, small amounts of memory, low power availability and typically real-time ioperation because of the types of application present. These restrictions exclude some more resource hungry or time consuming reconfiguration algorithms and techniques.

This paper is concerned with exact state reconstruction by the use of integral state observers. In particular we are interested in how such algorithms can be dynamically reconfigured for optimal performance. These specific observers have some features that make them suitable for use in real time systems, including (Byrski and Fuksa (1984)), (Fuksa and Byrski (1984)):

- possibility of *on-line* computation of the exact state $x(t)$ based on finite time moving observation window $[t-T, t]$,
- possibility of exact state reconstruction for any assumed finite time window of width $T > 0$,

Actually, there are numerous techniques that can be used to introduce dynamic reasoning or dynamic system reconfiguration. These include inter alia fuzzy logic, ANNs (Artificial Neural Networks) and genetic algorithms. In case of rule-based (Mamdani's) fuzzy logic systems the reasoning or reconfiguration decisions are based on set of rules specified usually at expert level. However, in many cases using rules only may not be optimal (or not sufficiently flexible) from the point of view of reasoning efficiency in case of more complex dependencies between available inputs (context information). Though fuzzy-neuro systems take an advantage of learning process in terms of avoiding manual specification of long sets of reasoning rules, the learning process itself is in many cases too time consuming in order to adjust the system to the environment changes.

Artificial Neural Networks suffer from the same problems, additionally it is much more resource consuming than for example simple rule-based fuzzy systems. Quick reconfiguration of the decision making system seems to exclude as well all evolutionary algorithms, which have high lag as they converge to the target function over many iterations.

Another problem is the numerical complexity of the exact state reconstruction algorithm. This algorithm is based on numerical integration of matrix equations, thus it's quality (but also efficiency) is strongly dependant on the number of integration steps. Demand of good quality of the state reconstruction results in very high requirements regarding sufficient processing power and available operating memory which leaves a narrow margin for implementation of dynamic reconfiguration features.

In this situation policy-based computing seems to be a very reasonable reconfiguration technique, being an ideal trade-off between flexibility (high) and resource requirements (low).

The remainder of this paper is set out as follows. Section 2 contains theoretical background and provides a formal description of continuous integral state observer. Section 3 gives a brief description of policy-based computing, section 4 describes a new architecture of integral state observer with policy-based supervision. Section 5 evaluates the new integral state observer architecture Sections 6 and 7 contain respectively a conclusion and discussion of some open problems regarding implementation of policy-based computing in computer control systems.

2. BACKGROUND AND RELATED RESEARCH

2.1 State reconstruction in dynamic systems

In classic control theory and its applications a common practice in the estimation of inaccessible for measurement state vector of a linear system is the use of Luenberger type observers. D. G. Luenberger in his PhD dissertation (1963) and in [1] proposed the structure of the observer derived directly from the differential form of Kalman Filter (KF) (Kalman (1960), (Kalman (1961)), [3]. Next we see a few subsections.

$$\dot{\bar{x}}(t) = \underbrace{(A - GC)}_{F}\bar{x}(t) + Bu(t) + Gy(t) = \\ = F\bar{x}(t) + Bu(t) + Gy(t); \bar{x}(0) \neq 0 \qquad (1)$$

For observable systems there is possibility of choosing matrix G in many ways that the matrix F will be asymptotic stable. Then we have

$$\dot{\varepsilon}(t) = \dot{x}(t) - \dot{\bar{x}}(t) = F\varepsilon(t); \varepsilon(t) \neq 0; t \to \infty$$

Unfortunately the real state $x(0)$ and hence the initial error $\varepsilon(0)$ as well as the function error $\varepsilon(t)$ are unknown. The only certainty is that the solution of state estimate $\bar{x}(t)$ tends to the real state asymptotically and theoretically reaches it but in infinity $\bar{x}(t) \to x(t)$. The design technique for the gain matrix G in the Luenberger observer is the eigenvalue placement method. The optimal gain matrix G in KF is calculated based on stochastic properties of measurement disturbances in input u and system output y and least-squares error approach. Matrix G is the solution of well known Riccati differential equation or in simplified version - an algebraic Riccati equation. The formal solution of equation (1)

$$\bar{x}(t) = e^{Ft}\bar{x}(0) + \int_0^t e^{F(t-\tau)}Gy(\tau)d\tau + \int_0^t e^{F(t-\tau)}Bu(\tau)d\tau \quad (2)$$

Two integrals for every t represent two measurement windows with expanding width. However, calculation of $\bar{x}(t)$ is much easier by the use of any recursive numerical procedure for solution of equation (1) rather than by (2). The power of modern computers makes possible the application of quite different on-line observation algorithms. They reconstruct the value of the current state vector exactly. The calculations must be based on finite time moving windows and finite history of measurement samples of input u and output y.

In 1966 Gilchrist (Gilchrist (1966)) proposed an exact state observer based on so called n-observability condition. The exact state $x \in R^n$ was calculated from knowledge of the system's output at only n ***discrete*** measurement points (discrete history output window) and from the system's ***continuous*** input measurements (continuous input history window). The structure of the observer was derived directly from system output equation. In a similar way one can derive the full continuous version of an integral observer for exact reconstruction of initial state value for continuous measurements of both input and output signals. It is directly based on the definition of state observability.

Let's consider an LTI (Linear Time-Invariant) observable system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \qquad (3)$$

where $x(t) \in R^n$, $u(t) \in R^r$ and $y(t) \in R^m$ for $\forall t \geq 0$, $m < n$ and the initial state $x(0)$ is unknown. Assume that we measure the control u and the output y on the interval $[0, T]$ where T is the fixed observation time. Our purpose is to determine the state $x(0)$. The output of system for $t \in [0, T]$ has well known form

$$y(t) = Ce^{At}x(0) + C\int_0^t e^{A(t-s)}Bu(s)ds \qquad (4)$$

For $x(0)$ calculation one should multiply both side of the equation (4) by transposition $\left[Ce^{At}\right]'$ and next integrate equation over interval $[0, T]$. The real Gram matrix M is nonsingular for any T.

$$M = \int_0^T e^{A't}C'Ce^{At}$$

Then the formula for $x(0)$ reconstruction has a form

$$x(0) = \int_0^T M^{-1}e^{A't}C'y(t)dt - \int_S^T \left[\int_0^T e^{A't}C'Ce^{A(t-s)}dt\right]Bu(s)ds$$

or

$$x(0) = \int_0^T \bar{G}_1(t)y(t)dt + \int_0^T \bar{G}_2(s)u(s)ds \qquad (5)$$

In the same way one can built the exact state observer of final state $x(T)$ which is more convenient for on-line control than $x(0)$. The output of system (3) based on x(T) has the form

$$y(t) = Ce^{-A(T-t)}x(T) - C\int_t^T e^{A(t-s)}Bu(s)ds \qquad (6)$$

For $x(T)$ calculation one should multiply both side of the equation (6) by and integrate it within $[0,T]$. For observable system the real Gram matrix M is nonsingular for any T.

$$M = \int_0^T e^{-A'(T-t)}C'Ce^{-A(T-t)}$$

Then formula for $x(T)$ one can obtain by inversion of M

$$x(T) = \int_0^T e^{-A'(T-t)}C'y(t)dt - \\ - \int_0^T \left[\int_0^S e^{-A'(T-t)}C'Ce^{A(t-s)}dt\right] Bu(s)ds$$

or

$$x(T) = \int_0^T G_1(t)y(t)dt + \int_0^T G_2(s)u(s)ds \qquad (7)$$

In 1992 Medvedev and Toivonen (Medvedev and Toivonen (1992)) proposed another version of the above continuous initial state observer 5 with the use of discrete measurement of the output. Such types the observer were termed finite memory deadbeat observeránd were used in different application e.g. for fault detection (Medvedev (1996)).

A similar approach and application of exact observers to diagnosis was presented in (Nuninger et al. (1998)) by the use of full discrete version of the above observer (with discrete measurements of both output y and control u). Some other least squares version of finite memory deadbeat observer was presented in (James (1990)) and (Hocine et al. (2005)). Generally there is an infinite number of special matrices G_1 and G_2 which guarantee exactness of state observation by the observers of type (5) or (7).

The exactness in reconstruction of the state by the use of integral observer is valid only under assumption of perfect input-output measurements i.e. no input-output noise or disturbances occur. However, in practical case of noisy measurements the use of integral observers gives also reconstruction error. The optimal observer which should give minimal norm of reconstruction error should have also minimal value of its own norm from the point of view noisy measurement of u and y. To this end function matrices $G_1(t)$ and $G_2(t)$ should be carefully chosen from the class of all admissible observer matrices. Unfortunately the versions of exact observers (5) and (7) do not have their own minimal norm, this also applies to the above as mentioned applications. It is because the authors assumed that the input function (control produced by computer algorithm) is perfectly known and for state calculation there is no need for its measurement. The last assumption however in practical application is not always proper because the input signal produced by actuator may differ from the computer control signal.

Hence, all the above versions of exact state observers are only special subclasses, which may by derived from general exact and optimal observer theory. This theory was formulated and presented by Byrski and Fuksa in 1984. In (Byrski and Fuksa (1984)), (Fuksa and Byrski (1984)) the problems of general structure and optimality of the exact continuous state observers were solved. This theory also originates from definition of observability and uses functional analysis technique similar to presented in (Rolewicz (1972)). The authors proposed a deterministic approach to disturbances and to exact and optimal state observation for which the relations were formulated generally in function Hilbert spaces U, Y. Such type of the observer must have the structure of two linear continuous functions. It is because based on two continuous pieces of functions u and y given on finite time interval $[0,T]$, the observer should provide the real unknown number (vector) $x \in R^n$. On the other hand by the Riesz Representation Theorem every linear continuous function in Hilbert space can be expressed as inner product. Hence the structure of the observer has to be given by two inner products: one product of continuous output function $y \in Y$ and special observation (filtering) function $G1(\tau) \in Y$ and the second one with the input function $u \in U$ and special observation (filtering) function $G2(\tau) \in U$ on interval $[0,T]$. After the first observation interval $[0,T]$ the observer reconstructs the exact value $x(T)$ and hence $x(t)$ for $\forall t \in T$. Choosing different input-output Hilbert spaces one can obtain different formula for finite state exact observer. As the second problem - the optimal structure of the above observer was derived, with minimal norm. This optimal structure was based on optimal shape of functions $G_1(.)$, $G_2(.)$, which should be chosen in such a way that they fulfill observability requirements and minimize the norm of the observer in chosen Hilbert spaces Y and U. In this norm the weight factor α is connected with the norm of G_1 and represents the norm of output disturbances and the weight factor β is connected with the norm of G_2 and represents the norm of input disturbances. For different α, β different optimal observers are derived. The optimal observer with minimal norm guarantees minimal state reconstruction error for the worst case if disturbances of y and u are bounded and belong to unit balls.

The very special case (if absence of input measurement noises is assumed) is represented by the weight factor β equal to zero, $\beta = 0$. For this case the optimal integral observer gives continuous version of observer (Medvedev and Toivonen (1992)) as in (5) or (7).

In (Byrski and Fuksa (1984)) and (Fuksa and Byrski (1984)) it was assumed that the disturbances will have bounded norm (belong to the unit balls). If the spaces Y and U are chosen as $L^2[0,T]$ the inner products are repre-

sented by an integral operators. Hence in all authors publications the name 'integral observers' was used to underline its contrary type to differential structure of Kalman Filter or Luenberger observer. The extended results of the on-line exact observation and application were presented in (Byrski (1993b)), (Byrski (1993a)), (Byrski (1995)). In paper (Byrski (1993b)) the integral observers with Expanding and Moving Observation Window MWO (sliding window) and their differential versions were given. In (Byrski (1993a)) a generalization to disturbances from an ellipsoid was derived and in (Byrski (1995)) the applications to LQR stabilization of distillation column was presented. In (Byrski and Kubik (1998)) the integral observer was used for state observation in distributed parameter system (heat equation). In (Byrski (2003)) a survey for the exact and optimal state observers was given. In (Byrski and Pelc (2004)) authors investigated the cooperation of KF and integral observers. In (Byrski and Pelc (2005b)) the new discrete version of exact and optimal observer version was presented. In (Byrski and Pelc (2005a)), (Byrski and Pelc (2006)), (Pelc and Byrski (2007)), (Byrski and Pelc (2005c)) multi observer approach to optimal observation was proposed. Different observers with different windows widths used simultaneously can reconstruct the state with minimal error for different norm of disturbances. Computational algorithm by switching procedure and its parallel decomposition can choose the best observer for the current situation.

In on-line state reconstruction we can use the optimal integral observer as a moving observation window observer MWO with fixed width T of the window. Namely, from (7) we have for every $t \geq T$ and for any output y and input u, so also for arbitrarily shifted y and u.

$$x(t) = \int_0^T G_1(\tau) y(T-t+\tau) dt + \int_0^T G_2(\tau) u(T-t+\tau) d\tau$$

or after changing limits of integration

$$x(t) = \int_{t-T}^{t} G_1(T-t+\tau) y(\tau) dt + \int_{t-T}^{t} G_2(T-t+\tau) u(\tau) d\tau \quad (8)$$

From this it follows that the observer forms the windows of width T shifted along time axis. After the multiplication of functions $G_{1,2}()$ by y and u measurements on $[t-h, t]$ and product integration for each t it gives the exact state $x(t)$ for $t \geq h$. The value of $x(t)$ does not depend on the chosen interval h. Moreover, the matrices $G_1(\tau)$, $G_2(\tau)$ do not depend on the current time t and are calculated on interval $[0, h]$ only once (off-line). The problem of current and fast calculation of two integrals for every t does not belong to observation theory and is just numerical and hardware problem.

The important problem for the integral observer is the choice of the width T of the observation window. Theoretically, if there are no disturbances in measurements the exact state reconstruction does not depend on T and generally on the norm of the observer. Hence, from the computation effort point of view and for decreasing the delay of exact reconstruction the width T should be as small as possible. On the other hand the main statement is that the norm of the observer depends on T. The norm increases with decreasing T. When T tends to zero the norm tends to infinity (Byrski (1993a)). Hence, in practice for disturbed measurements of y and u the reconstruction error (estimated by the norm of observer) will be less for bigger T. Therefore we can determine the minimum value of T to guarantee admissible state reconstruction error or use simultaneously different observers with different values of T_i for adaptive switching.

The properties of closed-loop systems with MWO and static LQ controller were discussed in Byrski, Fuksa (Byrski and Fuksa (1984)). In particular, it was proved that system with static controller and MWO in a closed loop has the same eigenvalues as the system with controller only. This means that the rank of the closed system with the MWO observer is the same as without the observer; this is a better situation than in classical LQG systems.

2.2 The optimal form for integral observers in space $L_2[0, T]$

Let a linear system (3) be given. The optimal observer equation is

$$x(T) = \int_0^T G_1^0(T, \tau) y(\tau) dt + \int_0^T G_2^0(T, \tau) u(\tau) d\tau \quad (9)$$

where the dimensions of function matrices $G_1^0(T, \tau)$, $G_2^0(T, \tau)$ are $(n \times m)$ and $(n \times r)$, respectively. The optimal matrices $G_{1,2}$ are functions of two parameters - the fixed observation time T and the time $\tau \in [0, T]$. In the sequel we will omit the first argument in writing. One should choose matrix G_1 from the set of all admissible matrices which fulfill the constraint

$$\int_0^T G_1(\tau) C e^{-A(T-t)} d\tau = I \quad (10)$$

where $\boldsymbol{I}$ is an $n \times n$ identity matrix, (Byrski and Fuksa (1984)), (Fuksa and Byrski (1984)). Then admissible matrix G_2 should fulfill the constraint

$$G_2(\tau) = \int_0^\tau G_1(s) C e^{-A(\tau - s)} B ds \quad (11)$$

The squared norm of the observer is:

$$J = \int_0^T \left[\sum_{i=1}^n \alpha_i \sum_{j=1}^n (g_1^{ij}(\tau))^2 + \sum_{i=1}^n \beta_i \sum_{j=1}^n (g_2^{ij}(\tau))^2 \right] d\tau \quad (12)$$

The minimum of (12) is given by elements $g_1^{ij}(\tau)$, $g_2^{ij}(\tau)$ of matrices G_1 and G_2, which are as small as possible (in the sense of norm in L^2). Optimal matrices have the form, (Byrski and Fuksa (1984))

$$G_1^0(t) = P_1(t) C', G_2^0(t) = P_2(t) B \quad (13)$$

where i-th columns of $P_1(t)$ and $P_2(t)$ are given by

$$\begin{aligned} p_1^i(t) &= \Phi_{11}^i(t) \left[M_i'\right]^{-1} e_i \\ p_2^i(t) &= \Phi_{21}^i(t) \left[M_i'\right]^{-1} e_i \end{aligned} \quad (14)$$

and fundamental i-th matrices for i=1,..., n,

$$\Phi^i(t) = e^{W_i t} = \begin{bmatrix} \Phi^i_{11}(t) & \Phi^i_{12}(t) \\ \Phi^i_{21}(t) & \Phi^i_{22}(t) \end{bmatrix} \quad (15)$$

and Hamiltonian Matrix

$$W^i = \begin{bmatrix} A & \gamma_i BB' \\ C'C & -A' \end{bmatrix} \quad (16)$$

where $\gamma_i = \beta_i/\alpha_i$ and Gram matrices M_i

$$M_i' = \int_0^T e^{-A'(T-t)} C'C\Phi^i_{11}(\tau)d\tau$$

For weight factors $\alpha = 1$, $\beta = 0$ from the above formula one can obtain the observer derived by (7). For $\alpha = \beta = 1$ the solution of the optimization has simpler form

$$\begin{array}{l} G^0_1(T,\tau) = M^{-1}\Phi'_{11}(\tau)C' \\ G^0_2(T,\tau) = M^{-1}\Phi'_{21}(\tau)B \end{array} \quad (17)$$

The real Gram matrix M^{-1} is of the form:

$$M^{-1} = e^{AT}\left[\int_0^T \Phi_{11}(\tau)C'Ce^{A\tau}d\tau\right], \quad (18)$$

and two elements of the fundamental matrix $\Phi(t)$ are obtained from:

$$W = \begin{bmatrix} A & BB' \\ C'C & -A' \end{bmatrix}, \Phi(t) = e^{Wt} = \begin{bmatrix} \Phi_{11}(\tau) & \Phi_{12}(\tau) \\ \Phi_{21}(\tau) & \Phi_{22}(\tau) \end{bmatrix} \quad (19)$$

Matrices (17) for the given observation time T can be calculated off-line in interval $[0, T]$ and then applied on-line in optimal filtering moving window. The norm (12) of the observer is the function of chosen observation time T.

2.3 Policy-based reconfiguration in control systems

The idea of integration of policy-based computing with classical control algorithms is a new and promising research area especially in case of embedded control systems. The main reason is that policies usually are quite lightweight from the point of view of resource consumption in comparison to other methods of expert-knowledge based systems (ANN's, fuzzy logic systems, etc.). Our research seems to be a very promising and quite unique attempt of this kind of integration, though it is related to some work described in the reminder of this section.

In (Heinis (2008)) an experimental study of zero-configuration policies tuned automatically based on analytical models of the controlled systems is presented. The application to a distributed workflow execution engine called JOpera as well as a comparison with a standard PID controller is shown. The policy itself is used for tuning an autonomic PID controller in response to the engine's performance (provided by the performance monitor) according to specific control strategies. Based on the performance information (context information) the autonomic controller decides whether a reconfiguration is needed using a specified optimization strategy. The optimization strategy maps the context information into reconfiguration actions.

An idea of real-time reconfiguration in response to quality of service (as the context information) over the communication network is described in (Chen et al. (2009)). The context information is being processed based on a neuro-genetic method. In the result the NCS's (Networked Control System) bandwidth is shared properly among different parallel control tasks using a scheduling algorithm. The NCS has to have reconfiguration ability in order to accommodate to the environment changes quickly, smartly and flexibly to maintain real time performance. Contrary to our solution this approach has learning capability (supported by ANNs) which enables the reconfigurable scheduling to work in an intelligent way based on the history knowledge about QoS variations and to act in advance. The downside of this approach in comparison to our policy-based solution is significantly higher requirements regarding deployment system resources.

Adding any level of higher level supervision to an existing control system results in creation of hierarchical control system. An example of complex control system for autonomous vehicles (in this case a helicopter testbed was the implementation platform) is presented in (Wills et al. (2000)). In this approach the Open Control Platform (OCP) is presented where a new control algorithms are integrated with some components which interact in a distributed environment. This is to provide means of online reconfiguration and adaptation to react to unpredictable environmental changes. The OCP itself supports flexible component integration, providing an abstract interface based on familiar controls engineering concepts, such as block diagram components, input and output ports, and signals.

3. POLICY-BASED DECISION-MAKING SYSTEMS

Policy-based computing is a technique of separating out the static aspects of a system's control logic from the dynamic parts (policies) which are easily replaceable as a means of changing the systems' configuration and behavior. This approach thus supports the development of self-configurable or autonomous systems and provides an interface to describe interactions/dependencies between a system and its environment (these interactions are usually specified as a policy). The role of each policy is to decide how the system behaviour should change in response to contextual environmental conditions. These policies are designed to be replaced with other policies if a change in the logic is required.

Policy-based decision making systems are of high flexibility and have limited resource requirements the deployment system should satisfy (e.g. to store and evaluate the policy).

For large or complex systems, the logic is usually divided into functional components and centralized control is undesirable from scalability and robustness points of view. Similarly, the policy logic can be localized to the component level, placing policy logic at the point it is needed to make specific strategic decisions and avoiding the overheads and rigidities of centralized design. This was the approach taken in DySCAS (DySCAS (–)), in which policy logic was distributed throughout automotive middleware and control systems, and this served as an

evaluation testbed for the development of our policy techniques. Our approach of embedding the policy evaluation and decision-making logic into (software) components is facilitated by leaving open 'decision points' at design time, which are subsequently populated with the appropriate policy at run time, as illustrated in Fig.1 (Anthony et al. (2008b)). This increases efficiency and flexibility.

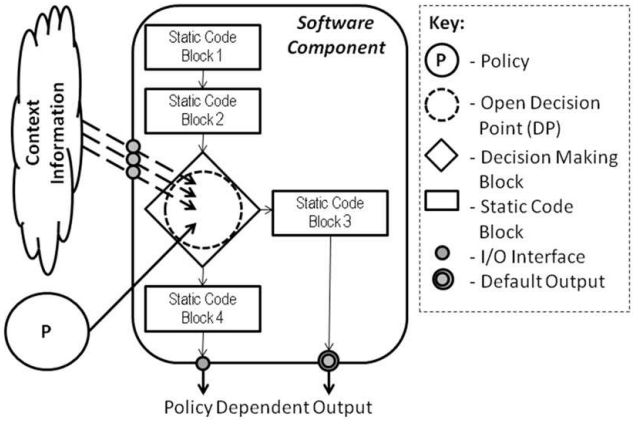


Fig. 1. A generic single software component with an open decision point left by the designer for embedding decision making modules

As discussed above, the decision-making system may be based on policies (which describe the way of how the context information is processed in order to make up a decision), but it may also use other techniques such as fuzzy logic or ANN, and thus the decision points must be populated with the appropriate configuration files containing a new set of rules for fuzzy logic controller or new weights for ANN. So, the type of technology used here is flexible (with respect to resource constraints). We have developed a component model which is open with respect to the decision technology used (refer to Fig.1), but our implementations are based on policies. In practice two components are required to fill such a decision point: a Decision Evaluation Module (DEM) containing the evaluation engine, and a Decision Configuration Module (DCM) containing the decision logic (policy) usually stored in a physical file. The type of evaluation module is matched to the configuration module; together they define the behaviour of a specific decision point. A DEM is compiled offline and is able to take the contextual inputs which describe the environment and system conditions, together with a DCM, and return the appropriate result. While the functionality of a DEM is generally fixed; there may be many versions of a DCM destined for the same component. The behaviour of a component changes depending on which of these variants are currently loaded. Also, different DCMs may require different context information to enact their desired function. The DEM should be ready before the software component is required to run. In order to prevent the dynamic decision making system to negatively impact on control system integrity or stability, a Dynamic Wrapper (DW) is used around each DEM. The Dynamic Wrapper makes it possible to provide a static safe output of the dynamic decision making system in case of the dynamic decision evaluation fails (Ward et al. (2008))(Anthony et al. (2008a)).

3.1 AGILE Policy Description Language

In the case of policy-based computing, the core part is a policy containing the changeable, context-aware strategic logic for the decision point, expressed as a series of rules and other constructs. Depending on application domain, the policy should be written using a Policy Description Language (PDL) offering the most flexible and efficient way of expressing dependencies between context information and decisions to be made based on this information. A PDL should be a compromise between flexibility and efficiency, because the more efficient (which means optimized for a selected application domain) a PDL becomes, the less flexible it is (as the flexibility can be achieved only by some level of generalization).

One of the very good examples of PDL where both, flexibility and efficiency are on a very high level is AGILE Policy Description Language (Anthony (2006))(Anthony (–)). AGILE is an XML-based language offering a wide set of run-time policy objects, of which the decision-making constructs include *Rules*, *ToleranceRangeChecks* and *UtilityFunctions*. AGILE's flexibility is further enhanced by supporting logical guards which can be used to dynamically enable / disable specific rules, as well as indirect addressing, so for example the weights of a utility function can be dynamically adjusted according to the outcome of a specific rule. The *ToleranceRangeCheck* construct enables inclusion of deadzones and three-way decision forks within a single policy statement. Templates allow a specific policy to be (re) initialized to some known state, by initially setting the values of the policy's internal variables. *Actions* allow logic statements to be grouped together and named, thus supporting structuring within policy scripts.

4. THE ARCHITECTURE OF POLICY-SUPERVISED EXACT STATE OBSERVER

The exact state observers have many features that make them strong candidates to benefit from integration with policy-based computing. One such feature is the dependency of the state reconstruction quality on the extent of disturbances / noise in the input and output signals. In the ideal case where the system input and output signals are noiseless, the observation window can be actually as narrow as possible. This will benefit in shortening of the computation time required to compute the system state. However, when disturbances occur in input or output signals, the state estimate quality decreases. Under these realistic conditions, the narrower the observation window, the bigger the state reconstruction error becomes. Naturally, widening of the observation window decreases the observation error, but it also lengthens the time required for the state computation. Dynamically choosing the appropriate observation window depending on the level of disturbances can significantly improve state reconstruction quality and simultaneously can result in the best possible real-time performance. This is an ideal scenario for using a policy which can make context aware decisions at run time to handle short-term or fast-changing environment changes, yet can also be replaced with a different policy version to deal with longer term changes - or deployment-specific characteristics of environment. Typically the use of state observer in the control system implies using state

controller. In case of noisy environments the optimal LQR controller is mostly used as it can be designed taking into account stochastic measures of the noise. The optimal control signal is calculated as the linear combination of estimated state variables provided by the policy-supervised state observer. The structure of a proposed control scheme with a policy-supervised state obs erver and optimal LQR state controller is shown in Fig.2.

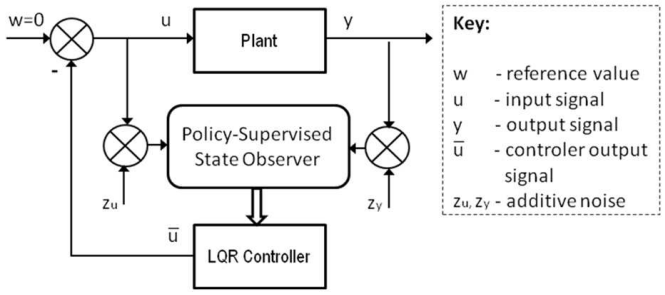


Fig. 2. Control system with Policy-Supervised State Observer

The exact state observers do not have any built-in feature supporting dynamic reconfiguration and decision making in response to the environment changes. Thus the only way to embed such a feature into the exact state observers is to build sort of hierarchical system by adding an additional layer of supervision to a set of observers. We achieve this by integrating a policy technology into the software components, and use policies to select the most appropriate observer for the given context. The architecture of the policy-supervised integral state observer is shown in Fig.3.

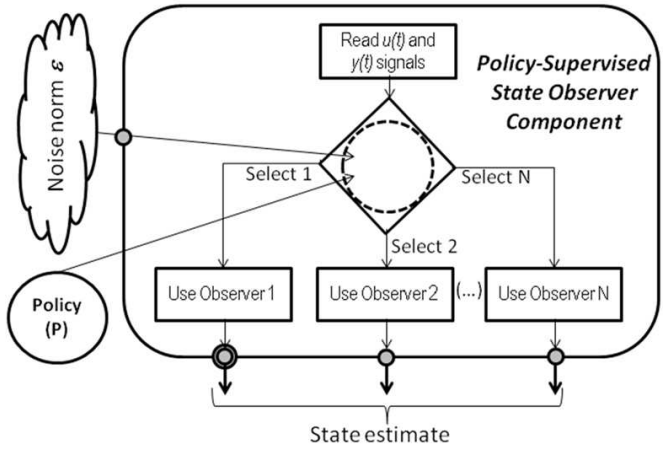


Fig. 3. Architecture of Policy-Supervised Integral State Observer

As shown in Fig.3, the policy logic is used to switch to the most appropriate observer (i.e. to select the one which guarantees the best quality of state estimate for given input signal $u(t)$ and output signal $y(t)$ from the set of available observers taking into account given context information. Depending on the policy, the observer context information can be for example the state estimation error, the norm of disturbances, real-time performance index, etc. In case of hierarchical systems, the natural consequence of implementation of some additional control layers within a control system is increased computational cost, as the supervision itself is time consuming. In order to limit the influence of the supervision on the overall system performance it is necessary to use the supervision as seldom as possible. This can be achieved by event driven supervision or cyclic supervision.

5. EVALUATION

In this section we present how policy-based computing can be integrated with the computer control systems as well as what benefits this can bring from the point of view of the control quality and the control system performance.

The use of exact state reconstruction is extremely important in various domains. One of the domain may be for example control of chemical processes. In case of distillation installations it is very important to know exact chemical composition of the distillation product at different stages of the process. Because of the process nature it is not possible to measure directly the composition at desired point as it might negatively influence the process itself or it would require using a very expensive equipment. For example, in order to measure the chemical composition of a liquid in the distillation column it would be necessary to use a spectral chromatography. Beside the price of spectral chromatograph, a much bigger disadvantage may be the real-time characteristic of the measurement process. In this case using the exact state reconstruction may be much more convenient as this can provide sufficient information for control purposes without any additional expenses or negative consequences.

5.1 Simulation scenario

For the evaluation purposes the control system structure shown in Fig.2. was implemented using TS-7260 hardware platform (for more details the reader is directed to (TechnologicSystems (–)). The control algorithm with state observer was developed in C programming language. Additionally the AGILE_Lite library (Pelc and Anthony (2007)) was used in order to provide reconfiguration and dynamic decision making features. Using the library a single decision point was embedded into the control algorithm in order to make up decisions about which observer from set of observers switch to depending on current noise norm. The controlled system was of second order with description in state space is given by the following state matrices:

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C = [20], D = [0]$$

and the initial conditions were assumed to be:

$$X_0 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

The LQR parameters are given by K vector as:

$$K = \begin{bmatrix} -2.2361 \\ -3.0777 \end{bmatrix}$$

In order to evaluate state reconstruction quality depending on observer a state estimation index (error) was used defined as:

$$\varepsilon = x(t) - \bar{x}(t)$$

where:
- $x(t)$ is the real state,
- $\bar{x}(t)$ is the state estimate.
The tests were conducted as follows:

- for the control purposes three continuous integral state observers with observation windows of width 0.5s, 1.0s and 4.0s were computed,
- first each observer was used separately in the control system with LQR controller and the state reconstruction quality was measured,
- then in the same control system all the observers were used in the switched structure and the state reconstruction quality was measured; the switching procedure was sequential beginning from the observer with the narrowest observation window and finishing on the observer with the widest observation window (Byrski and Pelc (2005c));
- and finally all the observer worked in the switched structure, but the switching process was being determined by three different policies,
- in each case the simulation time was measured additionally in order to determine how the choice of observer impacts on the control system performance.

Additional assumptions regarding the numerical tests include:

- the simulation time was set to 20s,
- the simulation step and the observers computation step was set to 0.01s,
- the noise characteristics for input and output noise was assumed to be exactly the same,
- additional step disturbance was generated after half of the simulation time.

During the simulation three different policies were used. Each policy was using one main functional logic block called *ToleranceRangeCheck* (Anthony (2006)),(Anthony (–)). This is a very efficient policy logic construct, specific to the AGILE policy language. It allows a test to see if the system lies within a dynamically specifiable tolerance from some goal state, or whether the system is above or below the goal state, by at least the tolerance amount. This three-way decision fork is expressed as a single AGILE statement. During each policy evaluation a selected *Check* variable (usually context variable) is being compared with some *Compare* value (or variable) in order to check whether it remains within a margin defined by *Tolerance* value. Depending on the checked value, a specific action can be taken (respectively *InZone*, *ActionHigher* and *ActionLower*). In the policy used during our simulation the current noise norm given by environment variable NoiseNorm is being compared with the fixed *ReferenceNorm* value and the *Tolerance* was set to 0.01. Depending on the noise norm an adequate action was called returning the decision regarding which state observer to switch to with respect to current environment conditions. The Tolerance was set to 0.01 for the *Policy1*, 0.05 for the *Policy2* and 0.03 for *Policy3*. The Tolerance value is set in the Template, which can be used to customize (adjust) the policy operation depending on preferences/needs (i.e. new Templates can be loaded at run-time, re-initialising control variables as necessary). The policy is shown in Fig.4.

```
<!-- Policy Definition XML file: Policy Language version 1.2 -->
<!-- Application: Temperature Control -->
<PolicySuite PolicyType="Observer Norm Scheduling">
  <EnvironmentVariables>
    <EVariable Name="NoiseNorm" Type="real"/>
  </EnvironmentVariables>
  <InternalVariables>
    <IVariable Name="ReferenceNorm" Type="real"/>
  </InternalVariables>
  <Templates>
    <Template Name="T1">
      <Assign Variable="ReferenceNorm" Value="0.05"/>
    </Template>
  </Templates>
  <ReturnValues>
    <ReturnValue Name="Window1" Value="0"/>
    <ReturnValue Name="Window2" Value="1"/>
    <ReturnValue Name="Window3" Value="2"/>
  </ReturnValues>
  <Actions>
    <Action Name="Start">
      <EvaluateTRC TRC="NormCheck"/>
    </Action>
    <Action Name="ChooseWindow1">
      <Return ReturnValue="Window1"/>
    </Action>
    <Action Name="ChooseWindow2">
      <Return ReturnValue="Window2"/>
    </Action>
    <Action Name="ChooseWindow3">
      <Return ReturnValue="Window3"/>
    </Action>
  </Actions>
  <ToleranceRangeChecks>
    <TRC Name="NormCheck" Check="NoiseNorm" Compare="ReferenceNorm"
Tolerance="0.01" ActionInZone="ChooseWindow2"
ActionHigher="ChooseWindow3" ActionLower="ChooseWindow1"/>
  <ToleranceRangeChecks/>
  <Policies>
    <Policy Name="Policy1" PolicyType="NormalPolicy">
      <Load Template="T1"/>
      <Execute Action="Start"/>
    </Policy>
  </Policies>
</PolicySuite>
```

Fig. 4. Policy used during simulation

5.2 Simulation results

For the simulation scenarios described in the previous subsection we have gained results showing how the control system works. In Fig.5 one can see the results of the control process. As the example the state variable x2 was chosen; in this case the state observer with the observation window of width 0.5s was used.

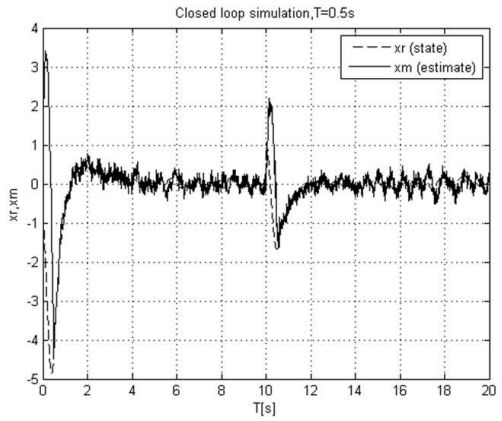
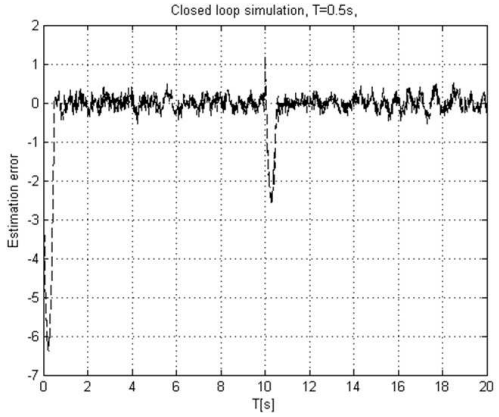


Fig. 5. The process of control for the system of rank 2

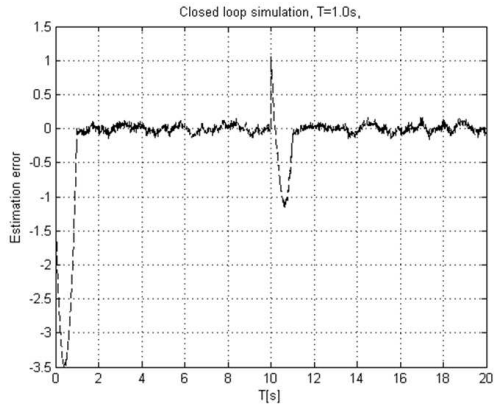
Fig.6 a-h). show the state estimation error in the control system with different configurations of state observers used and with different policies.

In table 1 one can see the simulation time depending on the state observer configuration.

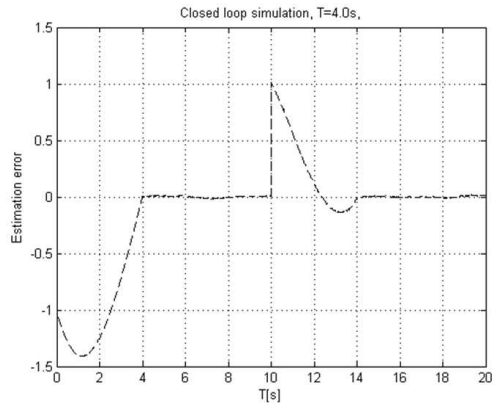
a)



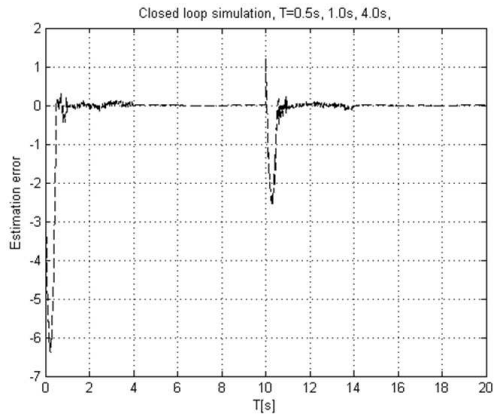
b)



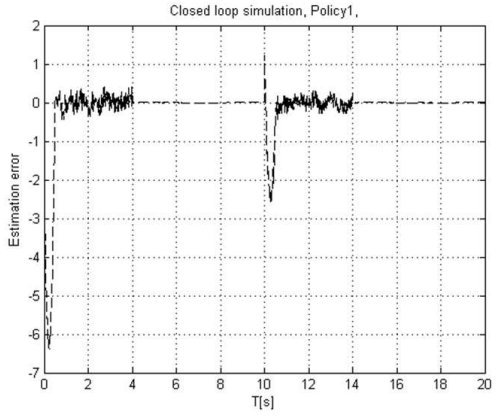
c)



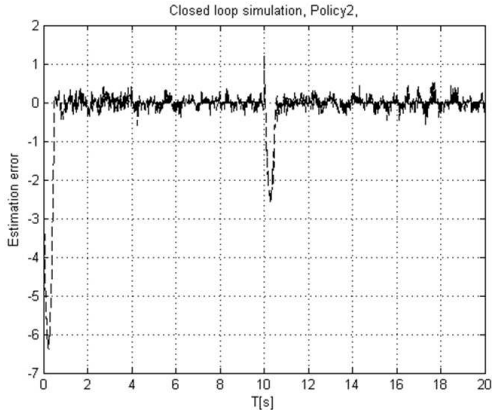
d)



e)



f)



g)

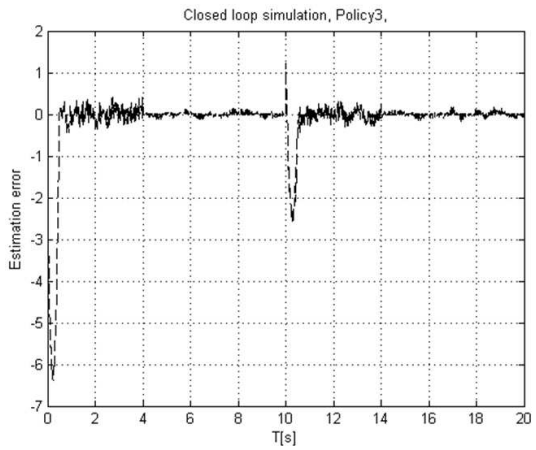


Fig. 6. Simulation results: the simulation error in case of the observation window of width: a) T=0.5s, b) T=1.0s, c) 4.0s d) sequential 0.5s, 1.0s, 4.0s, and the policy used: e) Policy1, f) Policy2, g) Policy3

Based on the simulation results on one side and on the control system performance on the other side one can see, that in some cases (depending on policy) the policy supervision may even increase the system performance. This can take place in the situation, when policy is choosing the observers with narrower observation windows more often, than observers with wider windows. In general it is clear that the policy can influence both the system performance and the state estimate quality.

Table 1. The simulation time depending on the state observer configuration

Observer configuration	Simulation time [s]
T=0.5s	0.233259
T=1.0s	0.338827
T=4.0s	0.799311
sequential switching T=0.5s, 1.0s, 4.0s	1.407799
With Policy1	1.468031
With Policy2	1.227429
With Policy3	1.228282

6. CONCLUSION

The research presented in this paper was an attempt of integration of the classical algorithm of integral state reconstruction with policy-based supervision layer. For this purpose we have presented an architecture of policy-supervised state observer, where the policy-based supervision level was responsible for choosing the most appropriate state observer from the set taking into account current context information. We thoroughly tested a couple of very important performance aspects of this hierarchical system with LQR state observer, i.e. the influence of the additional layer of supervision on:

- the performance of the control system,
- the quality of state estimate,
- the stability of the closed-loop control system.

As a result of our simulations we have shown that the presented architecture of policy-supervised state observer operates properly. Furthermore, the additional level of policy-based supervision may significantly change the system behavior depending on currently applied policy. The other aspect of the simulation results was finding out how much the decision-making system influences the entire control system performance. One can see that the control system with dynamic decision making may even work faster, than the static system with fixed switching mechanism. In order to get some more information about performance of the decision making system the reader is directed to (Pelc et al. (2009)).

There is no doubt that the system performance and the state estimate quality are strongly dependant. This is why the policies have to be optimized and purpose-made. But this is also beyond any doubt that introduction of the additional supervision level to the classical state reconstruction algorithm brings a lot of benefits to the control system with this kind of observers. But the most desired result of this kind of modification is increase of flexibility and possibility of adjusting the state reconstruction performance with respect to any selected control quality index or real-time constraints.

7. OPEN PROBLEMS

In case of computer control systems one of the most important issues is to ensure that the introduction of any additional level of supervision does not negatively influence the system stability or does not decrease its robustness.

This paper was not targeting these issues specifically, though our simulations were purposely conducted in the closed loop system in order to provide at least some simulation-based evaluation of the system operating. However, the issue of stability or robustness of a policy-supervised hierarchical system is a very important issue. Though there are a number of formal methods that evaluate stability or robustness of any classical control system, introduction of any additional higher level of supervision introduces additional degree of freedom and potential source of problems. Actually, there is no existing formal method that can be used for the complete validation of policies which have dynamic context dependent operation; which leaves a significant space to be filled in. As long as the policy is simple and is using expert-based knowledge, it can be easily evaluated in an empirical way. For more complex policies dealing with many context inputs the decision making system may become too complex for this kind of evaluation. Furthermore, the more complex the policy itself becomes, the easier is to make a (logical) mistake during its creation. In the result the policy may not do what it is supposed to.

Without the formal methods of evaluation the use of policies in the computer/embedded control systems may be a potential source of the system instability or may in some way decrease its robustness. Because of this reason the research of some formal methods of policies validation has started and will be subject of our further publications.

REFERENCES

Anthony, R.J. (–). Policy Autonomics Home Page. URL `http://www.policyautonomics.net`.

Anthony, R.J. (2006). A policy-definition language and prototype implementation library for policy-based autonomic systems. *Proceedings of 3rd International Conference on Autonomic Computing (ICAC2006)*, 265–276.

Anthony, R.J., Pelc, M., Ward, P., and Hawthorne, J. (2008a). Flexible and robust run-time configuration for self-managing systems. *SASO '08: Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, 491–492.

Anthony, R.J., Pelc, M., Ward, P., and Hawthorne, J. (2008b). A run-time configurable software architecture for self-managing systems. *Proceedings of ICAC 2008*, 207–208.

Byrski, W. (1993a). Integral description of the optimal state observers. *Proceedings of 2nd European Control Conference*, 4, 1832.

Byrski, W. (1993b). Optimal state observers with moving and expanding observation window. *Proceedings of IASTED 12th International Conference on Modelling & Simulation*, 68.

Byrski, W. (1995). Theory and application of the optimal integral state observers. *Proceedings of 3rd European Control Conference*, 52.

Byrski, W. (2003). The survey for the exact and optimal state observers in hilbert spaces. *Proceedings of 7th European Control Conference*, 6.

Byrski, W. and Fuksa, S. (1984). Optimal finite parameter observer. an application to synthesis of stabilizing feedback for a linear system. *Control and Cybernetics*, 13(1).

Byrski, W. and Kubik, P. (1998). Integral observer of finite state in heat equation approximated by ode. *IFAC/IMACS Large Scale Systems Symposium.*

Byrski, W. and Pelc, M. (2004). The finite time state observer and its cooperation with kalman filter algorithm. *Proceedings of 29th IASTED Conference on Modelling, Identification & Control*.

Byrski, W. and Pelc, M. (2005a). The adaptive integral multi-observer with switched moving windows. *Proceedings of 24th IASTED International Conference on Modelling, Identification & Control*.

Byrski, W. and Pelc, M. (2005b). Modelling and simulation of state observers in the computer control systems. *Proceedings of 39th International Conference on Modelling and Simulation of Systems*.

Byrski, W. and Pelc, M. (2005c). Multi-observer approach for the best state reconstruction in finite time. *Proceedings of 17th IMACS World Congress on Sciengific Computation, Applied Mathematics and Simulation*.

Byrski, W. and Pelc, M. (2006). Continuous and discrete integral state observers in on-line control systems. *Proceedings of 39th International Conference on Modelling and Simulation of Systems*.

Chen, H., Zhou, C., and Zhu, C. (2009). A hybrid neural-genetic approach for reconfigurable scheduling of networked control system. *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, 33–38.

DySCAS (–). Dyscas project home page. URL `http://www.dyscas.org`.

Fuksa, S. and Byrski, W. (1984). General approach to linear optimal estimator of finite number of parameters. *IEEE Transaction on AC-29*, 29(5), 470–472.

Gilchrist, J.D. (1966). n-observability for linear systems. *IEEE TAC*, 11(3), 388.

Heinis, T. (2008). Automatic configuration of an autonomic controller: An experimental study with zero-configuration policies. *Proceedings of ICAC 2008*.

Hocine, A., Maquin, D., and Ragot, J. (2005). Finite memory observer for switching systems: Application to diagnosis. *Proceedings of 16th IFAC Congress*.

James, M.R. (1990). Finite time observers and observability. *Procceedings of 29th IEEE Conf.on Decision & .Contr.*, 2, 770–771.

Kalman, R.E. (1960). Contribution to the theory of optimal control. *Boletin de la Sociedad Matematica Mexicana*, 5, 102–119.

Kalman, R.E. (1961). New results in linear filtering and prediction theory. *Trans of ASME Journal of Basic Engineering*, 83D, 95.

Medvedev, A. (1996). Fault detection and isolation by functional continous deadbeat observers. *Int. J. Control*, 64(3), 425.

Medvedev, A. and Toivonen, H.T. (1992). A continous finite memory deadbeat observer. *Proceedings of American Control Conference*, 180.

Nuninger, W., Kratz, F., and Ragot, J. (1998). Finite memory generalised state observer for failure detection in dynamic systems. *Proceedings of 37th IEEE Conf. on Decision & Control*.

Pelc, M. and Anthony, R.J. (2007). Towards policy-based self-configuration of embedded systems. *SIWN System and Information Sciences Notes*, 2(1), 20–26.

Pelc, M., Anthony, R.J., Ward, P., and Hawthorne, J. (2009). Practical implementation of a middleware and software component architecture supporting reconfigurability of real-time embedded systems. *Proceedings of 7th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC-09)*, 393–401.

Pelc, M. and Byrski, W. (2007). Full parallel decomposition of computational algorithm with discrete state estimator. *Proceedings of 19th IASTED International Conference on Parallel and Distributed Computing and Systems*.

Rolewicz, S. (1972). On optimal observability of linear systems in infinite-dimensional states. *Studia Math.*, 44, 411–416.

TechnologicSystems (–). Technologic systems home page. URL `http://www.embeddedarm.com`.

Ward, P., Pelc, M., Hawthorne, J., and Anthony, R.J. (2008). Embedding dynamic behaviour into a self-configuring software system. *Proceedings of 5th International Conference on Autonomic and Trusted Computing (Springer LNCS)*, 373–387.

Wills, L., Sander, S., Kannan, S., Kahn, A., Prasad, J., and Schrage, D. (2000). An open control platform for reconfigurable, distributed, hierarchical control systems. *Proceedings of Digital Avionics Systems*, 1, 4D2/1–4D2/8.