# DESIGNER 2000 PROJECT

J Bůcha, M Kárný, P Nedoma, J Bohm, J Rojíček

Institute of Information Theory and Automation, Academy of Sciences, Czech Republic

Abstract: Designer 2000 project is presented. Designer 2000 system will support a systematic probabilistic design of LQG adaptive multidimensional controllers based on ARX model. The project goal is to improve the accessibility of already available tools. This is achieved mainly by incorporating available knowledge into the system and by a careful object-oriented computer-aided design of the system and especially of the user communication.

Keywords: Adaptive control, AI techniques and knowledge-based systems, Computer aided control systems design

## INTRODUCTION

In this paper the Designer 2000 is briefly presented. It is a project under development. The overall **project goal** is to improve the accessibility and usage of algorithmic and software tools for the design of adaptive systems[1]. The project is based on the availability of necessary theory, algorithms, software and expertise. It is the continuation of a long-term development in the authors' Department, Peterka (10), Kárný et al (5).

The **theoretical basis** of the project is Bayesian approach to the estimation and control (5, 10) that gives a common point of view to these traditionally separated problems and is capable to respect their mutual influence. The probabilistic insight accompanies most of steps of the design and offers a more realistic interpretation of results.

The mentioned algorithmic and software **tools and expertise** are available in the Department, but their usage is restricted mainly to the Department. There has been a **goal** to spread tools' usage outside, for a long time, see Kárný and Halousková (3, 4). A significant work has been done in this direction. A complete design cycle of adaptive controller has been covered. There have been prepared **previous** versions of **Designer** providing corresponding software tools. The last one is available as ABET (Adaptive Bayesian Estimation Toolbox), Nedoma et al (7, 8), Nedoma (9). The last versions were written in MATLAB (6). It provides an experimental environment, but from the point of view of the **ordinary user**, i.e. ordinary control engineer, it is still very hard to start with it; because of a great knowledge barrier. The user can hardly overcome this barrier. To **solve this problem** we intend to use "all available means to succeed".

This **paper** is divided into several **sections**. In the project we use object-oriented (OO) approach to analysis and design (A&D) of the Designer and the same goes for the illustrations, where we use the corresponding graphical language, UML (Unified Modeling Language), see Fowler (2). These are, together with the analysis results, shortly described in the Analysis section. The next section, Architecture, describes the main architectural features of the Designer, with special attention paid to object-oriented CASE (Computer Aided Software Engineering) and knowledge base (KB). The next section, Communication, describes the main features of communication between the user and Designer. The next section, Processing, describes the main features of Designer processing. The last section is the Conclusion. Some project details are briefly illustrated in Figures. The more detailed project description can be found in Bůcha et al (1).

## ANALYSIS

The analysis is mainly oriented both to identification of users, to users' requirements and to the previous version of the Designer. We recognize four types of **users**, i.e. ordinary user, expert user, administrator and developer. In order to determine who is an ordinary user and who is an expert user, we compiled two lists of terms: The first one contains the terms that should be known to the ordinary user, the second one the terms that need not be

[1] As "project" and "design" are synonyms, our goal is "design of design". To distinguish these designs we use "design" for the design of adaptive control, we use "project" for the overall analysis and design of design of adaptive control and we use "proposal" for the design (without analysis) of design of adaptive control, mainly.

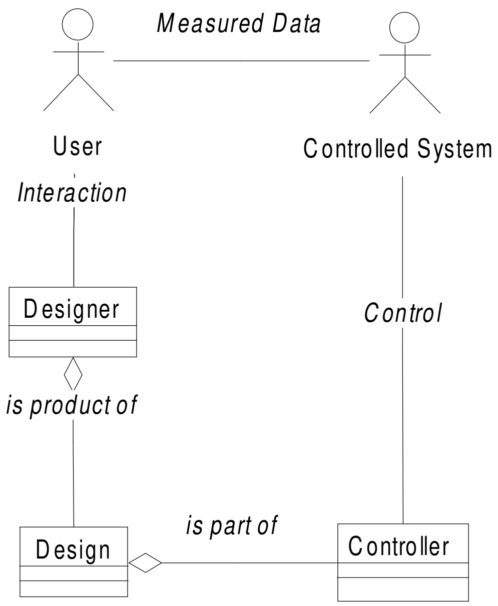


Figure 1: Main Class Diagram

known to him (1). In cooperation with potential users we created a list of main **requirements** on the Designer (1). The aim of the third part is to understand why the mentioned goal (to spread usage) has not been reached. We begin with a context.

The **overall context** of the previous Designer is shown on the Figure 1. We use UML graphical language in following Figures. The short explanation of used **language symbols** follows: Named rectangle, with two additional lines inside, means a class, e.g. Designer. The class describes its objects. Named man icon means an active (external) class, e.g. User. Controlled System is considered active as the man icon indicates. (Named) line connecting two classes means an association relation, e.g. Interaction. It is used for a data flow, event and other relations. (Named) arrow with a diamond head means that a class is a part of another class, e.g. Design is part of Designer. (Named) arrow with a triangle head means a class-subclass relation, e.g. Data Handling is a subclass of Step (not in Figures).

The main classes in this context are Designer, User and Controlled System. In Figure 1, there are also Design and Controller classes that are parts of Designer class. They are considered in Figure 1 because they have a special relation to the context. The named lines indicate relations between classes. The meanings of classes and relations are self-evident. Controlled System can be both real and simulated. There are also classes corresponding to association relations (not to generalization relations). An example is Interaction. For simplicity, they are shown neither in Figure 1 nor in other similar Figures. (They are shown in Figure 3.)

The context of previous Designer is practically the same as the context of the (new) Designer. The differences are on lower architectural levels, i.e. in the contents, exact meanings and implementations of classes and their relations. The ABET contains about 150 m-functions, i.e. 170 KB of MATLAB source code, other C++ code, other code in demonstration programs etc. It is a set of programming tools, i.e. subroutines, classes, objects, helps, case examples and documentation that allow to an expert user to produce design, i.e., to create and run a program that can produce control action(s).

We can say that the previous approach has **its pros and cons**. Pros are that in this bottom-up way the necessary foundations for the Designer have been built. Cons are that to use the previous Designer mean to program in its language and system. It is not easy as the problems are not simple, system and language is neither stabilized, nor well documented, nor very understandable, nor very reliable, nor fully implemented. The **requirements** resulting from the analysis are clear: Make it easy, stabilized, …

## ARCHITECTURE

The Designer architecture has the following **main features**: 1) OO CASE as a nucleus of the project, 2) knowledge base as a nucleus of the Designer control, 3) efficient users' communication, 4) use of prior knowledge, 5) extension of the existing algorithmic and software tools if it is necessary to ensure completeness of the system.

1) The **nucleus** is needed to stabilize and standardize somewhere the project terminology, concepts and approaches. The **CASE** is needed to provide for project methods, documentation, dynamics, reliability, implementation support and user involvement. The CASE changes the character of the project from the currently prevailing bottom-up to top-down, too. The **OO CASE** is needed to provide for the solution of project complexity and for developers cooperation via UML. All main features are very interdependent. The object-orientation and CASE-orientation influences importantly all the other architectural features. This is why we put this feature on the top of main feature list.

The orientation on OO CASE is extended into a more general use of **CAD** tools to support the project, i.e. use e.g. GUI (Graphical User Interface), MATLAB, expert system shell, testing tools, to further support high quality, correctness, requirements attainment, dynamics of development, easy future development and administration of the project and product.

**OO CASE tool** should support the following functions: the standard range of object-oriented analysis and design, UML, production of documentation and help, single user environment, reengineering, testing, version control. The parts of previous projects may be overtaken, i.e. copied, transferred and more or less changed, but once inserted into the project **depository**

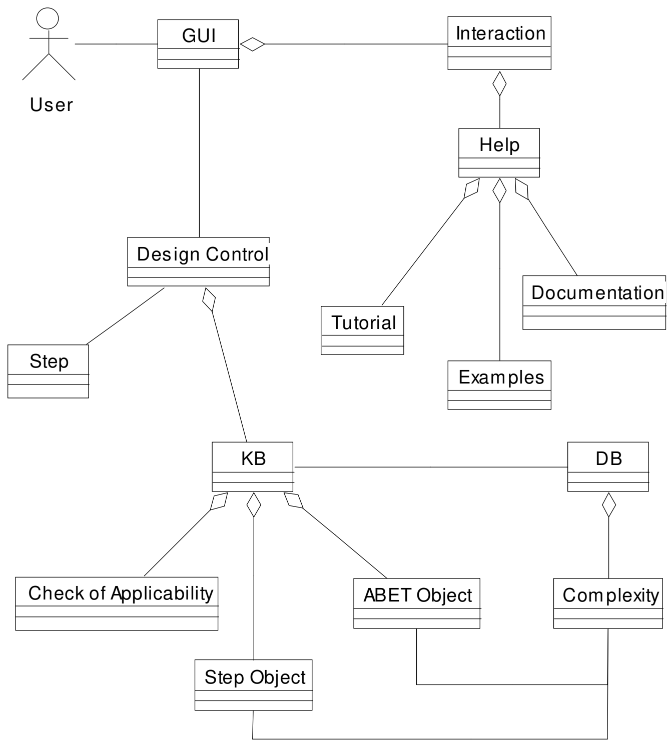


Figure 2: Detailed Class Diagram

they become obligatory for the whole project. They become the primary and only one source of information for the project.

2) **KB** is needed to allow using knowledge about the existing tools and about design process to integrate these tools into one comprehensible and reliable system. Knowledge is spread in the documentation, examples and helps and among expert designers in the Department. There is general and specific **knowledge** used by the Designer. This more general and/or more stable one is held in KB, that more specific and/or more variable one is held in DB, as DB can be more easily actualized. There is technical and problem-oriented knowledge. The **technical** knowledge specifies how to use ABET functions and objects, what are relations among them, etc. It is hidden to the ordinary user and it is implemented by standard programming tools. But, this knowledge is accessible to the expert user and in this case it is implemented in KB. The second part of knowledge, the **problem-oriented** knowledge specifies how to manage the whole design in order to reach reliably a successful solution. It has mainly a heuristic character. The part of this knowledge is already identified. The extent of the rest of this knowledge is dependent on experiments that will be done with the early versions of the system. This processing knowledge is accessible to the user. In this part of KB, there can be "contradictory" knowledge, dependent on expert who provided it.

For example, specific characteristics of m-functions like their input and output parameter names, their usage of and utilization in other functions, their hierarchical relations, their characteristics of computational complexity, characteristics of information processed by these functions like their location, extent and form, their hierarchical relations are held in DB. The general characteristics like specification of cooperation of these functions, knowledge, how to derive the overall computational complexity, are held in KB.

KB extent is not large, at present. It has about 20 rules. This is also caused by our concentration on the creation of the first prototype, where only a restricted set of rules is considered. The rules are fairly complex. Several rules of KB are, in a user-readable form, illustrated in the next sections of the paper.

KB is **object-oriented**. The reasons are: a) There is the fairly complex hierarchy of objects in the design, with composition and inheritance relations, with needs for encapsulation. For example, our steps and/or covering functions (see section Processing) are self-contained and the concept of hiding unnecessary information (see section Communication) aims at encapsulation and polymorphism, too. For example, there are send, receive, plot operations of polymorph type. b) The previous approach had some object-oriented features. There are, e.g., the following classes of objects: Project, Step, M-function, Problem, data flows, e.g. Design Specification, Control Action, User and Expert User.

Figure 2 illustrates some classes of the Designer and their relations, on a more detailed level, i.e. Design Control, KB, DB, GUI, Help, Tutorial, Examples, Documentation, Check of Applicability, Step Object, ABET Object and Complexity. They illustrate some of above-mentioned features.

## COMMUNICATION

Efficient users' communication is needed to decrease the complexity of usage and to provide for better users' understanding of the system. It is solved by proposing a comprehensible communication language, by providing the user with "exactly" as much information as he needs, i.e. hiding as much information as possible from him, providing as much information as possible, if he needs it, providing exact and consistent information and knowing "exactly" what are his needs in various situations.

To implement these concepts, the interaction with the user, GUI, is organized a) in a cooperation with several potential users, b) in a top-down manner, c) with a hypertext-oriented help, produced directly from the project CASE depository, d) with a good documentation, also produced directly from the project depository, e) with access to the system via Internet.

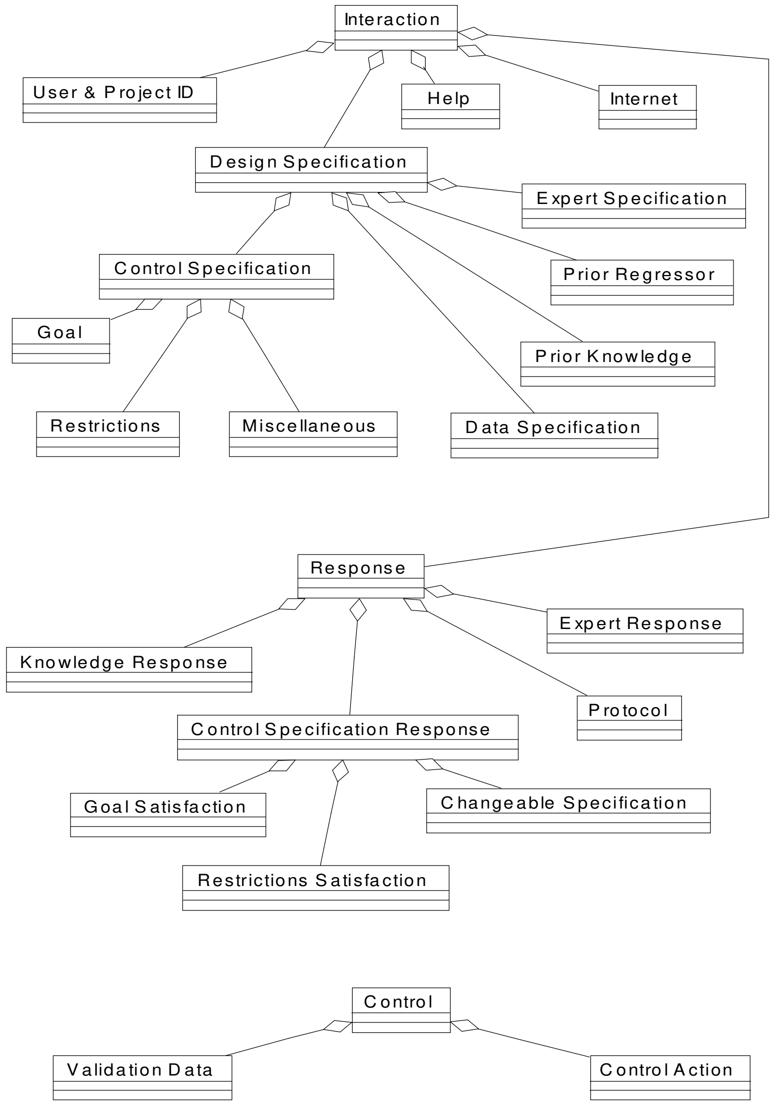


Figure 3: Interface Structure

Interactive conversations begin like in the following style: “Do you like to solve control problem?” “Well, it is nice and wise of you to chose the Designer 2000.” “To begin with it, we are to 1) check the applicability of the system to your problem, 2) if you feel like, tell something about you and 3) specify your problem. Let us start by checking whether Designer 2000 suits to your problem. …”

For the check of applicability, we specified precisely the **range of control problem** that can be solved (and must be checked with the user): There are the following **rules**:

1) Designer is to be used mainly when the classical solution has reached its limits. It means, use Designer when the control problem is currently being solved by set of mutually independent control loops (or when it can be solved in such a way), e.g. there is a similar system or there is a corresponding project, but this solution has reached its limits because it is a too complex problem

2) Designer is to be used also when some loops are more complex, e.g. the solved control problem has long delays or variable delays, there is feedforward, or need for multivariate solution

3) Designer is to be used only when the solved control problem can be linearized, e.g. the user knows that it is not necessary to add nonlinear elements to solve the control problem, and when the (economical) importance of the problem is not high, e.g. it means that it is not worth of hiring a Ph.D. level professional to build and/or operate a specific nonlinear model

4) Designer is not to be used, when the solved control problem is a simple one, e.g. when PID control is sufficient.

There are further preconditions of the use, but they are on a different level, they cannot be checked in advance:

1) Good result could be obtained if the data are measurements of sufficiently excited system and/or there exists a sufficient prior knowledge. But, this can be recognized only after the most of design is performed, sometimes

2) The design goal should be at least approximately describable by a quadratic loss.

In the range of the concept of hiding information the user learns as little as possible. GUI is as familiar to him as possible. For example the whole GUI is in the style of MS Word (Office) and/or MATLAB that are typical environments for the user, i.e. File, Edit, View, Options, Window, Help in the Main menu, their pull-down menus, tool-bars etc. Within this, the concept of providing as much

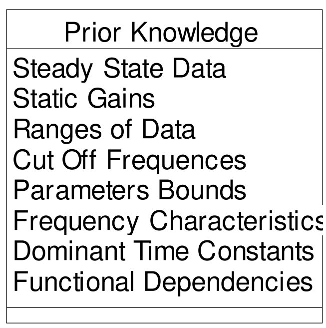


Figure 4: Attributes of Prior Knowledge

as necessary information is implemented. The user can obtain all definitions of used concepts in a hypertext form, including all usual synonyms, explanation and justification of chosen solution etc.

To illustrate these generally specified directions we list some of the user **inputs and outputs**. On the most abstract level the user can specify structures (classes) shown in Figure 3. In the Prior Knowledge part he can specify items and structures shown as the attributes of Prior Knowledge class in Figure 4. The user can obtain as a Response e.g. Level of Goal Satisfaction, Level of Restrictions Satisfaction. All “mathematical” and/or “formal” information, like model structures and theirs parameters, penalization matrices, theirs probability distributions, covariance matrices and guesses, internal ABET objects and theirs pointers are hidden to the ordinary user. They are available to the expert user, only.

## PROCESSING

In Figure 5 there are shown the main steps of the Designer processing with data flows between these steps and/or User and Controlled System. Some of these data flows are described in the Communication section.

steps have special features regarding the possibility to include user’s knowledge. Step Prior Knowledge Processing computes Initial Parameters Distribution from Prior Knowledge. Step Data Handling provides the information obtained by handling (measured) data, Handled Data. Currently, two functions are supported in this step, the elimination of outlying measurements and the normalization of measurements. Step Structure Estimation searches the best Regressor of the Controlled System, i.e. the qualitative dependency of outputs on delayed inputs and outputs that has the maximum likelihood. Step Forgetting Estimation selects the optimum forgetting technique, Forgetting. It allows to Parameter Estimation step to compute Parameters Distribution from Initial Parameters Distribution and Handled Data. Parameters Distribution together with Regressor settles the description of Controlled System model. Step Control Design builds and sets up the Controller in order to reach the desired closed loop behavior. Step Control Verification checks the set up of the Controller either on a real system or on some user-provided model. The steps are run in a sequential order, mainly. The successful conclusion of one step is a prerequisite for the following step.

For the batch version of the system there are provided so called **covering functions** that correspond to each

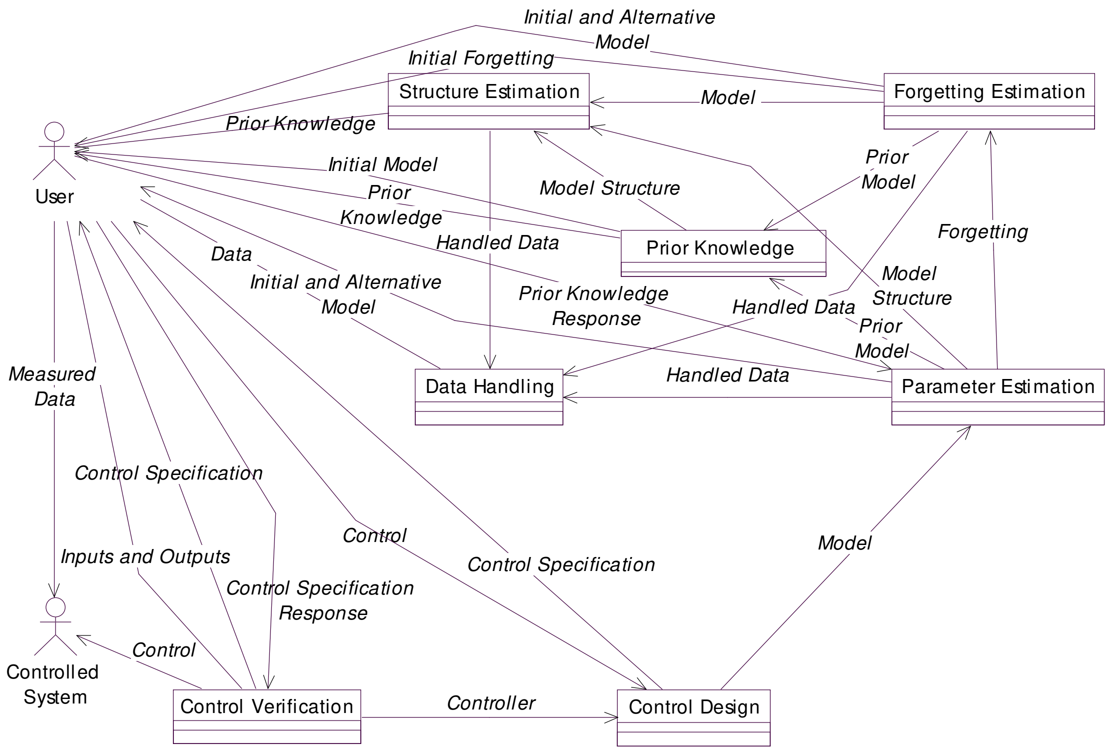


Figure 5: Class Diagram of Steps

The design has seven main **steps**: Data Handling, Prior Knowledge Processing, Structure Estimation, Forgetting Estimation, Parameter Estimation, Control Design and Control Verification. These more or less traditional

step. The steps can use ABET functions on a more detail level, also. The technical knowledge is used to specify how to integrate these functions, what are relations among them etc. This knowledge is hidden to the user. The second part of knowledge specifies how to manage the whole design in order to reach reliably a successful solution. It has a heuristic character. To illustrate the use of that knowledge we present two simplified rules that guide the overall operation and interaction with the user. This part is accessible to the user, e.g. in the form of explanations.

**Main Loop Rule**: If Inputs Satisfactions are greater then Limits of Inputs Expectations, then this "space" can be "traded" for Outputs Satisfactions by increasing Limits of Inputs Expectations, and vice-versa. If there is no space either in Inputs Satisfactions or in Outputs Satisfactions, then try to gather more data.

**Rule Applied after Step Structure Estimation**: If some output is independent of inputs then the reason can be that there is no Prior Knowledge specified. There will be many cases with independent output if there is no Prior Knowledge specified.

The use of prior knowledge allows and provokes the user to use his knowledge of controlled system to support successful design. The concept of using this knowledge is fundamental for the Designer. In the range of this concept it is necessary to stimulate the user to provide as much as possible of his Prior Knowledge and restrict an incompleteness of Prior Knowledge in such a way, and to stimulate the user to precise his goals, again, as much as possible.

## CONCLUSION

The discussions with users assured that the system like the projected one is very much needed. Our work confirmed the project's assumption that there are tools that allow building the Designer. One of the important project's assumptions was the existence of experience and knowledge about the utilization of Bayesian methods in analysis and design of controllers. This assumption was confirmed only partially. There is same knowledge that can be used, but it is not sure if any significant knowledge can be further gathered. More research is needed in this direction.

OO CASE is not only A&D tool for current project, but in its UML representation it is an important tool for a future cooperation on the further development.

In the paper we are using many, roughly 50, not precisely specified terms. They are used to illustrate the project on some more detailed level. We have identified about 310 such terms in the project. More details can be found in (1).

An important effort has been done to create the foundations for the project. We estimate it on level of hundred(s?) research-years. The rough estimate of overall necessary effort for this project is on level of 4 man-years. We are now (at the time of writing this paper) at the beginning of detailed Analysis and design stage. The project roughly follows a classical division into project stages, i.e. Feasibility study, Analysis and design, Implementation and Operation stages. From this point of view the future is predictable.

## REFERENCES

1. Bůcha J., Kárný M., Nedoma P., Bohm J. and Rojíček J., 1998, "Designer 2000, Feasibility Study", Technical Report, UTIA, AVCR, Prague, Czech Republic
2. Fowler M. and Scott K., 1997, "UML Distilled", Addison-Wesley, N.Y., USA
3. Kárný M. and Halousková A., 1994, "Pre-tuning of self-tuners", in: Advances in Model-Based Predictive Control, D. W. Clarke (Ed.), pp. 333-343, Oxford University Press, Oxford, UK
4. Kárný M. and Halousková A., 1991, "Preliminary tuning of self-tuners", in: Advanced Methods in Adaptive Control for Industrial Application, K. Warwick, Kárný M. and Halousková A. (Eds.), 158, Springer Verlag, Germany
5. Kárný M., Halousková A., Bohm J., Kulhavý R. and Nedoma P., 1985, "Design of linear quadratic adaptive control: Theory and algorithms for practice", Kybernetika, 21, Supplement to No. 3, 4, 5, 6
6. MATLAB, 1996, "Using MATLAB, Version 5", The Math Works, Natick, Mass
7. Nedoma P., Kárný M. and Bohm J., 1996, "DESIGNER: Preliminary Tuning Adaptive Controllers", Proc. of the 2nd Process Control, ŘÍP 1996, University of Pardubice, Horní Bečva, Czech Republic
8. Nedoma P., Kárný M. and Bohm J., 1996, "Project DESIGNER", Proc. of the spring school Adaptive and Predictive Control, Oxford University, Oxford, UK
9. Nedoma P., 1998, "ABET97, Adaptive Bayesian Estimation Toolbox, A Tutorial", Technical Report, UTIA, AVCR, Prague, Czech Republic
10. Peterka V., 1981, "Bayesian system identification", in: Trends and Progress in System Identification, P. Eykhoff (Ed.), pp. 239-304. Pergamon Press, Oxford, UK

**Acknowledgement**: This project is partially supported by the grant "Adaptive Systems: Theory, algorithms and software practice" GA AV ČR No. 102/97/0118.