# UTILIZATION OF MATLAB CLASSES TO STREAMLINE EXPERIMENTAL SOFTWARE

*Pavel Ettler* [1], *Ivan Puchr* [2]

COMPUREG Plzeň, s.r.o.
Nádražní 18, 306 34 Plzeň
Czech Republic

**Abstract**

**The paper deals with features of the experimental software which is being developed within the Matlab environment, prior to its final implementation in a common OOP (Object Oriented Programming) language. Both system of building blocks and calculus of artificial probabilistic reasoning, which are characteristic of the advanced diagnostic system being built within an international research project, call for utilization of classes and overloaded operators. Existing outputs are illustrated by experiments on both simulated and real data.**

## 1 Introduction

Authors belong to the consortium of the project the nature of which calls for the use of the Object Oriented Programming (OOP). Although the target application will be coded in some derivative of the C language, Matlab environment was selected for the phase of applied research and experiments due to its straightforwardness, graphical capabilities and its popularity among academic partners. Moreover, definition of classes in Matlab versions starting from 2008 became relatively suitable for experiments which are intended to be re-coded into another OOP programming language.

Following sections contain description of the system to be developed, summary of the related theory and an outline of the use of instrument of classes in two ways:

– For definition of blocks while benefitting from the property of inheritance from the superclass to its subclasses to reflect common and dissimilar properties of blocks of various types;

– For definition of the probabilistic calculus or more specifically calculus of artificial reasoning which, besides the inheritance principle, uses overloaded operators.

## 2 System to be built

The project in question aims to develop a hierarchical diagnostic system which will continuously evaluate health of an industrial control system taking pervasive uncertainty into account. Structure of the system is composed of two 'pyramids' as depicted in Fig. 1. The larger sub-pyramid is based on evaluation of healths of particular measured signals and cares about proper operation of the system. Its blocks on the lowest level which are distinguished by green color evaluate working conditions of single sensors and actuators. Green blocks on the upper levels correspond to the means how input data are utilized within the inspected system. The second sub-pyramid is related to the means by which the system is realized. Lowermost blue blocks monitor proper operation of hardware (computers, PLCs, etc.), load of the processors and network traffic. Information rises through common green blocks to the top and outputs of both sub-pyramids are combined to provide evaluation of health of the entire system.

[1] *ettler@compureg.cz*
[2] *puchr@compureg.cz*

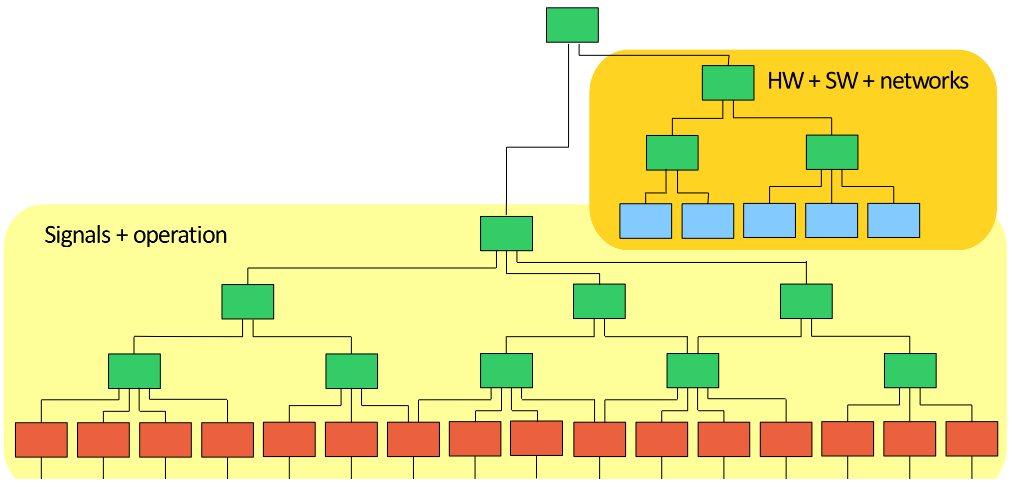

Figure 1: Example of a hierarchical diagnostic system with the pyramidal structure. Types of blocks are distinguished by different colors.

Main quantity which is used within the system denotes health $H \in [0, 1]$, where $H = 1$ represents perfect working condition of a unit while $H = 0$ corresponds to its total failure. Specifically for a single pyramid system, $H_{i,j}$ means health of the $j$-th block in the $i$-th level. For $n$ denoting the highest level, $H_{n,1} = H_S$ represent health of the entire system. Indices are omitted for the sake of simplicity wherever it is reasonable.

## 3 Probabilistic reasoning: Subjective logic

'So called probabilistic logic aims to combine the capacity of probability theory to handle uncertainty with the capacity of deductive logic to exploit structure. The result is a richer and more expressive formalism. . . . Subjective logic is a type of probabilistic logic that explicitly takes uncertainty and belief ownership into account. In general, subjective logic is suitable for modeling and analyzing situations involving uncertainty and incomplete knowledge. Arguments in subjective logic are subjective opinions about propositions. A binomial opinion applies to a single proposition, and can be represented as a beta distribution of probability.'

These sentences quoted from the web characterize theory which is behind the project. More details on probabilistic logic and on the subjective logic in particular can be seen e.g. in [3] or [4]. For the purpose of this paper it sufficient to mention, that the binomial opinion within the subjective logic is expressed by a quadruple

$$\omega_X = (b, d, u, a) \ , \tag{1}$$

where $X$ is a binary frame with $x$ corresponding to a defined assertion and $\bar{x}$ to its negation. Further, $b$ stands for belief mass in support of $x$ being true, $d$ is disbelief mass in support of $x$ being false, $u$ denotes uncertainty and $a$ is the base rate, corresponding to prior probability. It holds

$$b + d + u = 1 \qquad b, d, u, a \in [0, 1] \, . \tag{2}$$

The posterior mean value of binomial opinion or simply the expected value can be expressed as

$$E(x) = b + au \ . \tag{3}$$

Theory of subjective logic defines number of operators with opinions as counterparts to standard binary logic operators [4].

For $u > 0$ there exist a direct bijective mapping between opinion $\omega$ and corresponding beta probability density function (pdf). For $u = 0$ the pdf degenerates to a Dirac pdf concentrated at a point in the interval $[0, 1]$ given by values of $b$ and $d$.

## 3.1 Implementation

The project utilizes subjective logic to express opinions $\omega_{H_{i,j}}$ about health for each block of the pyramid. Thus, the system distinguishes between *plain* health $H$ and *opinion* $\omega_H$ about it.

Representation of particular opinions $\omega_{H_{i,j}}$ by the beta pdf [1] will be beneficial for the purpose of visualization as illustrated in Fig. 2. Operators defined for opinions enable to reflect relations within the inspected system. Details on preliminaries of particular utilization of the subjective logic can be found in [2].

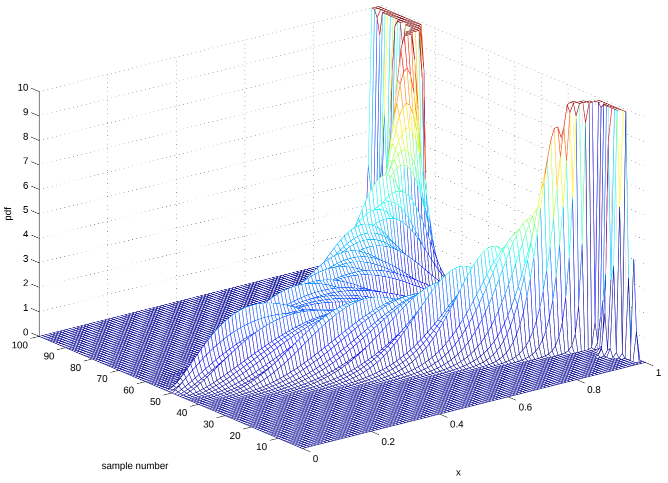

Figure 2: Progress of the beta pdf reflecting a simulated failure and its recovery.

Matlab implementation of objects needed for software realization of the project can benefit from class mechanism in several ways [5]:

- Defining a specialized data structure as a class has advantages over using a general-purpose data structure as users cannot accidentally misspell a field name without getting an error.
- A class is easy to identify. A class has a name so that objects can be identified with the `whos` and `class` functions and the Workspace browser. The class name makes it easy to refer to records with a meaningful name.
- A class can validate individual field values when assigned, including class or value.
- A class can restrict access to fields, for example, allowing a particular field to be read, but not changed.

Particular objects defined as Matlab classes are depicted in the following sections.

### 3.1.1 Class `slv` - subjective logic variable

The basic subjective logic variable which is characterized by four components $b, d, u, a$ (see (1)) is defined in the class `slv`. Elements `b`, `d`, `u`, `a` are defined as properties with their counterparts in `locb`, `locd`, `locu`, `loca` for a local access (having `private` attribute). Setter and Getter property access methods enable to define code to execute whenever a property value is queried or set and thus to ensure validity of relations (2). The class constructor ensures proper initial values of properties while additional private methods care about proper assignments between local and commonly accessible properties.

### 3.1.2 Class slo - subjective logic opinion

Class slo (subjective logic opinion) is defined as a subclass of slv, thus inheriting its properties. It defines mean value E and parameters alfa, beta of the beta pdf as class properties (with the dependent attribute), corresponding methods of which enable evaluation of the expected value $E(H)$ (3) and realize mapping $\omega \rightarrow$ beta pdf.

### 3.1.3 Overloaded operators

Subjective logic operators which are to be used within the system can be elegantly realized by user-defined functions having corresponding Matlab names which results in overloading of Matlab operators. For example, frequently used subjective logic operator of multiplication as a counterpart of the propositional/binary logic operator conjunction (= AND) is defined in the user-defined functions mtimes and times, overloading operators '$*$' and '.$*$', respectively.

Raising of an opinion to a power is realized by functions mpower and power overloading operators '$\wedge$' and '.$\wedge$', respectively.

Possible Matlab tip-offs like "Warning: Function xxx has the same name as a MATLAB builtin" drawing user's attention to a possibility of misuse of Matlab-defined names can be suppressed if they are felt as bothering [5].

## 4 Building blocks

Modularity of the developed system leads also to definition of its building blocks in the form of objects, i.e. classes.

### 4.1 Superclass blockKernel

Kernel of each building block is represented by the blockKernel class. Key properties of this class are of two types:

- Properties with Abstract and Protected attributes the concrete definitions of which are located in corresponding subclasses.
- Properties with Dependent attributes accessed through their Getter methods.

Moreover, the plainly defined property ignore enables to disregard status of an appropriate block in further evaluation: if ignore is set to true, the block is considered as perfectly "healthy".

### 4.2 Class blockUpper - Upper block

A common block of the system, i.e. a block on higher than the basic level, is defined by class blockUpper as a subclass of blockKernel. In addition to properties with the Access = protected attribute which include concrete definitions of abstract properties of its superclass, the "input" properties correspond to block inputs. For example, inOmegas property represent opinions $\omega_{H_{i-1,k}}$ coming from blocks on the lower level, where $k = 1, 2, \ldots, m_{i,j}$ and $m_{i,j}$ denotes number of blocks influencing the block in question. Proper dimensions of instances of these properties are adjusted in the class constructor.

Main methods of the class evaluate block's health $H_{i,j}$ and opinion $\omega_{H_{i,j}}$ by the means of weighted multiplications:

$$H_{i,j} = \prod_{k=1}^{m_{i,j}} f(H_{i-1,k}, w_{H_{i-1,k}}) \ , \tag{4}$$

$$\omega_{H_{i,j}} = \prod_{k=1}^{m_{i,j}} F(\omega_{H_{i-1,k}}, w_{H_{i-1,k}}) \ , \tag{5}$$

where $w_{H_{i-1,k}}$ denote weight of the incoming $H_{i-1,k}$, $\omega_{H_{i-1,k}}$ and $f(.,.)$, $F(.,.)$ mean appropriate weighting functions, respectively.

## 4.3 Class blockSignal - Signal block

The blockSignal is a subclass of blockKernel again. In addition to protected properties the reason of which was mentioned above, the class includes number of private and dependent properties. These reflect the main function of the block: to evaluate working condition of a particular measured signal within the inspected control system. Methods of the class evaluate for example how the signal fits into the pre-defined ranges (Fig. 3), whether it contains undesired outliers (Fig. 4), whether it contains a periodic disturbance with a significant dominant frequency (Fig. 5), etc. To do that, methods utilize various types of filtering, the FFT, etc. Further enlargement of block functionality is still in progress.

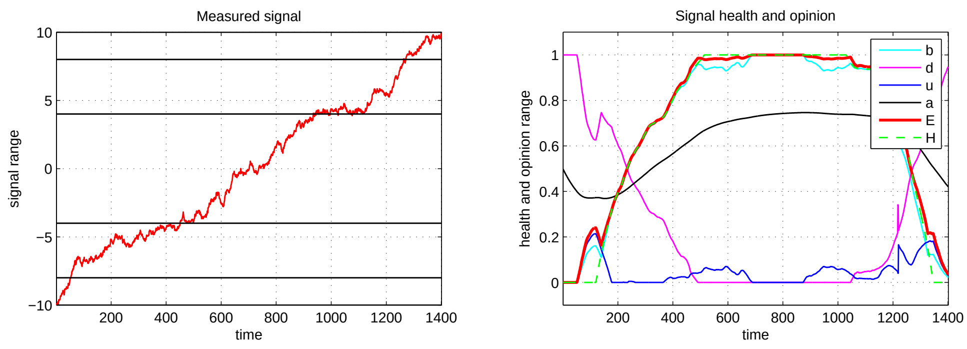

Figure 3: Range fitting. Left plot: simulated noisy signal traversing across its possible range. Right plot: evaluated health, components of the opinion and expected value.

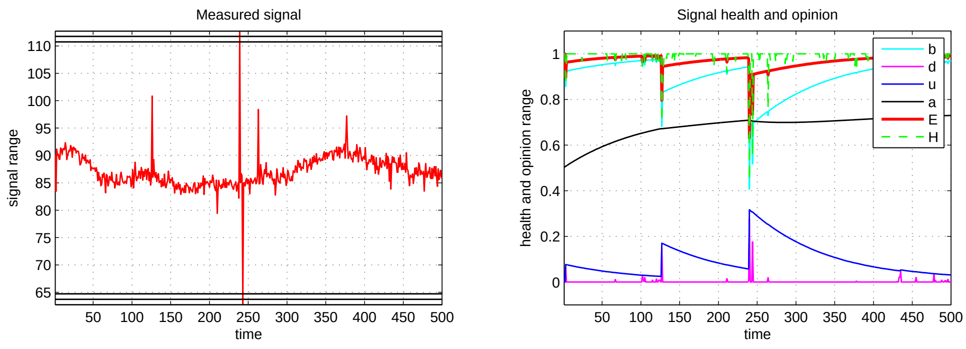

Figure 4: Detection of outliers. Left plot: measured signal of thickness deviation corrupted by outliers. Right plot: evaluated health, components of the opinion and expected value.

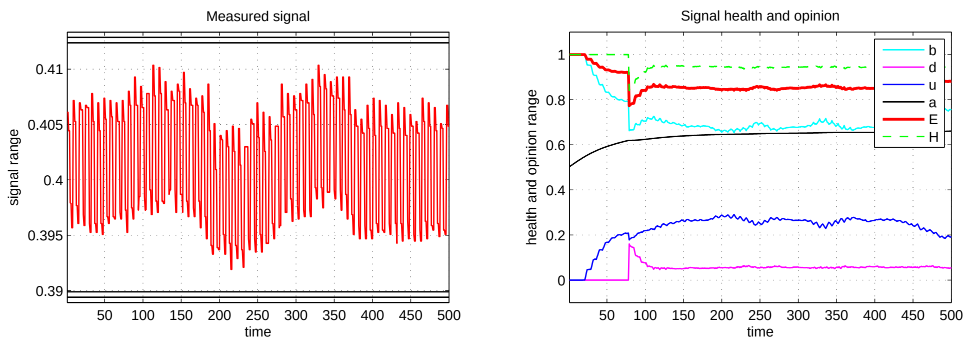

Figure 5: Detection of a dominant frequency. Left plot: Measured signal of speed of a rotational device corrupted by a periodic disturbance. Right plot: evaluated health, components of the opinion and expected value.

## 4.4 Classes Hardware block and Net block

The project intends to involve also blocks which would pass judgements on working conditions of the hardware both of the inspected control system and of the health evaluating system itself. Another blocks will guard network resources and evaluate their health. While there exist several elaborated packages to monitor the hardware and network, transformation of their outputs into the probabilistic opinion is still in progress. Intended classes blockHW and blockNet will have similar structure as existing classes while realizing developed algorithms.

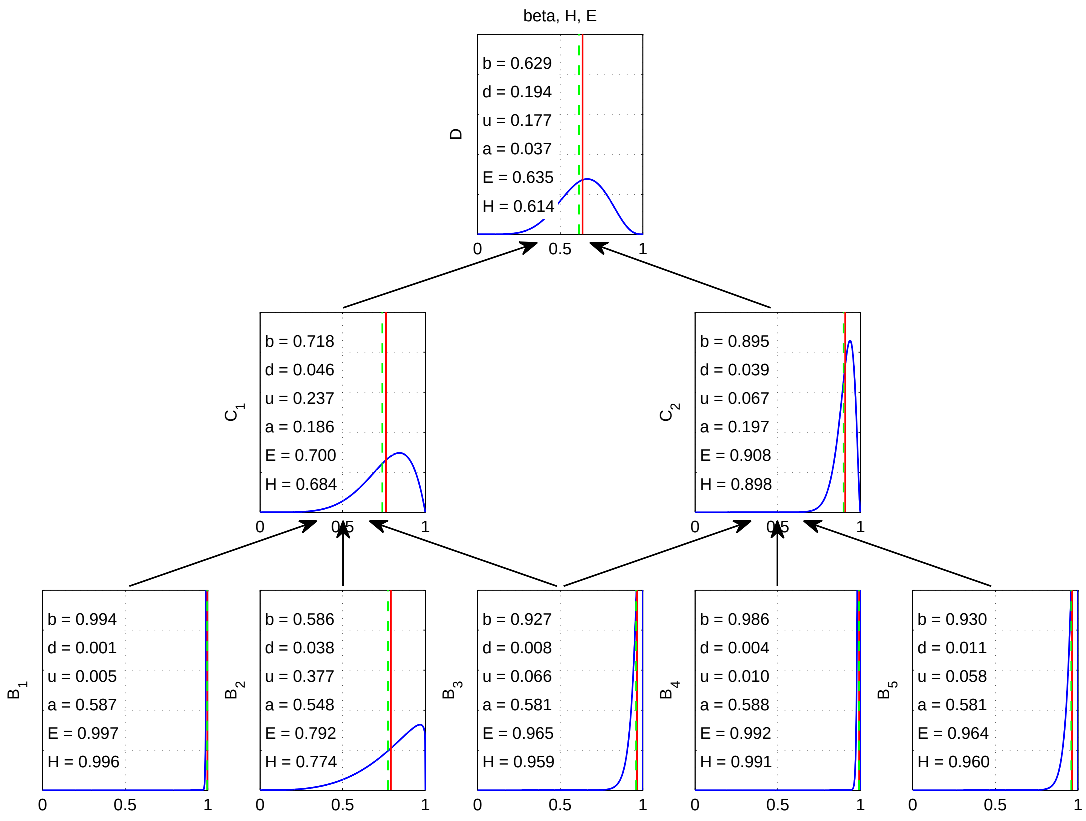

Figure 6: Experimental three-level system. For each block, dashed green line shows health $H$, red line correspond to the expected value $E(H)$, beta pdf is depicted in blue, meaning of variables is explained in section 3.

## 5 Example of the system

Fig. 6 depicts an example of a three level health-evaluating system. The lowermost blocks simulate the Signal blocks by generating particular healths $H_{1,j}$ and opinions $\omega_{H_{1,j}}$ as if they resulted from evaluation of measured signals. Please note that contrary to the envisaged real situation where all signals should be in perfect condition for most of the time the Fig. 6 illustrates imperfect signals for the sake of imagery.

Relations of blocks in the pyramid are depicted by arrows: each of the blocks in the second level is influenced by outputs of three lower blocks and the uppermost block provides opinion about the total system health. It can be seen how even slight malfunction of particular blocks on the lowermost level deteriorate health, i.e. working condition of the entire system.

## 6 Conclusions

The paper sketches the use of user-defined classes in Matlab environment for the purposes of the research project aiming to evaluate health of a complex industrial control system. Although the system is intended to be finally implemented under another platform, experiments in Matlab enable extended testing and sharing of its results among project partners. Proceeding theoretical research within the project will be reflected in the Matlab implementation successively.

## Acknowledgement

The research project is supported by the grant MŠMT 7D12004 (E!7662 ProDisMon).

## References

[1] K. Dedecius and P. Ettler. Overview of bounded support distributions and methods for Bayesian treatment of industrial data. In *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2013)*, pages 380–387, Reykjavík, Iceland, 2013.

[2] Ladislav Jirsa, Lenka Pavelková, and Kamil Dedecius. Preliminaries of probabilistic hierarchical fault detection. In *Proceedings of ECML/PKDD 2013 Workshop Scalable Decision Making: Uncertainty, Imperfection, Deliberation (SCALE)*, Prague, 2013.

[3] Audun Jøsang. Conditional reasoning with subjective logic. *Multiple-Valued Logic and Soft Computing*, 15(1):5–38, 2009.

[4] Audun Jøsang. *Subjective logic (draft).* University of Oslo, 2013.

[5] The MathWorks, Inc. Matlab documentation center. `http://www.mathworks.com/help/matlab/index.html`, 2013.