2002 IEEE International Symposium on Computer
Aided Control System Design Proceedings
September 18-20, 2002 • Glasgow, Scotland, U.K.

A multi-objective optimisation-based software environment for control systems design

Hans-Dieter Joos,
Johann Bals, Gertjan Looye, Klaus Schnepper, Andras Varga
Institute of Robotics and Mechatronics
DLR - German Aerospace Center
D-82230 Weßling

Abstract

Multi-objective optimisation is a proven, well-known parameter tuning technique in control design. It is especially suited to solve complex, multi-disciplinary design problems. This paper describes a software environment, called MOPS (Multi-Objective Parameter Synthesis), which supports the control engineer in setting up his design problem as a properly formulated multi-objective optimisation task. To this end, MOPS offers a basic control system criteria library, a generic multi-model structure for multi-disciplinary problems and a generic multi-case structure for robust control law design, as well as visualisation tools for monitoring the design progress. Several additional features for dealing with a large amount of parameters and criteria, distributed computation for time consuming computations and the use of external simulation and analysis servers are also provided. MOPS also supports parameter estimation in identification problems and optimisation based design assessment for robustness. The user is provided with a clear application program interface and a graphical user interface both implemented in MATLAB. To solve the underlying optimisation problem different powerful optimisation procedures are available.

1. Introduction

Control law design problems are often multi-disciplinary in their nature where many different, often conflicting design requirements have to be fulfilled simultaneously. In the case of many design objectives the control systems designer needs to compare different design alternatives. Further, he needs to know to which extent certain design objectives are satisfied and in case of conflict, he needs quantitative information about degradation of individual objectives while other objectives are improved. Design objectives can usually be expressed as mathematical criteria representing quantitative measures of achieved performance. The solution of such a control design problem with many criteria can be carried out by solving a multi-objective optimisation problem. As a computer aided design technology, multi-objective optimisation-based design is able to address all design goals and constraints simultaneously, while compromising them individually according to given demands. Due to the complexity of the design task, a multi-objective optimisation-based design usually involves experimenting with different set-ups for criteria formulation and weighting, different controller structures and parameterisations, as well as alternative (e.g. global or local) optimisation methods.

This paper describes a software environment, called MOPS (Multi-Objective Parameter Synthesis) [1] for optimisation-based computer aided control system design. In contrast to earlier implementations, see for example [2] [3], MOPS is completely redesigned based on powerful internal data structures. It now explicitly supports features like multi-model/multi-case design problems, robustness assessment, parallel computation of criteria, Monte-Carlo simulation etc.

MOPS supports a general controller design process, as it is illustrated in Figure 1, especially in the following three tasks: robust control law tuning, control law robustness assessment and parameter estimation of non-linear dynamical systems. These tasks can be solved most valuably by optimisation. The underlying multi-objective optimisation problem is solved as a min-max parameter optimisation, or in the case of parameter estimation, as a non-linear least-squares problem.

0-7803-7388-X/02/$17.00 © 2002 IEEE

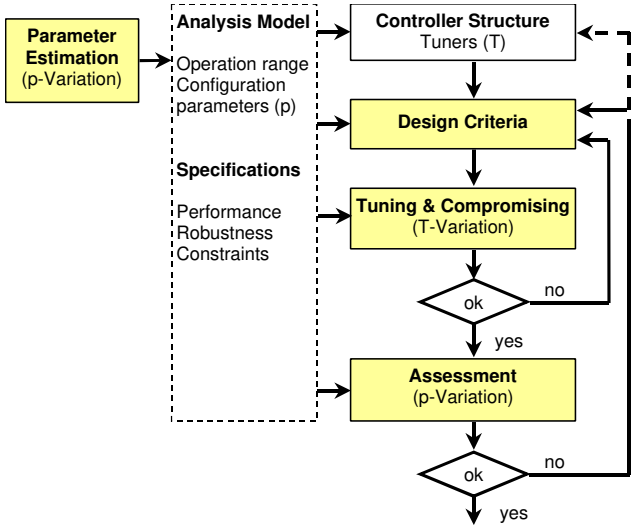

Figure 1: General controller design process with components supported by MOPS (grey).

As any complex software system MOPS can be structured into several software levels, see Figure 2.

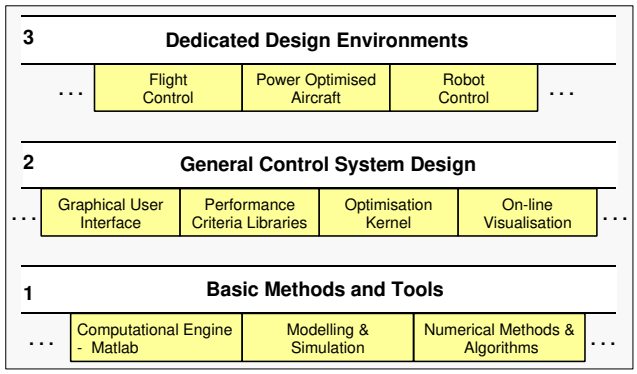

Figure 2: Three principal software levels of MOPS.

MOPS is primarily based on the widespread technical computing environment MATLAB [4] taking advantage of the powerful MATLAB-language features, like flexible data structures and handle graphics. The employed optimisation solvers are proprietary codes implementing several powerful algorithms. The level 2 of general control system design is described in this paper. The third level indicates the current fields of applications of MOPS. Application dedicated environments are planed to be implemented.

2. Basic Problem Formulation

MOPS provides the following functionality to support the user to properly formulate multi-criteria control design problems.

Definition of design criteria

As first step, optimisation requires a thorough formulation of design objectives through smooth optimisation criteria. A set of basic functions for the most commonly used time and frequency domain criteria is provided within a MOPS criteria library, which may serve as a basis to define more complicated design criteria.

Normalisation of criteria

To compare criteria in a multi-objective optimisation problem, a proper normalisation of criteria is necessary via appropriate transformations (e.g., scaling and shifting). MOPS provides a convenient framework to normalise automatically criteria by generating appropriate scaling and shifting on basis of specified good/bad limiting values (similar to fuzzy logic membership functions) [2]. The criteria transformations, (see Figure 3), ensure the separation between the 'acceptable' and 'not acceptable' values with respect to a normalised value of one. All best possible values are mapped to zero, i.e. to the smallest criterion value.

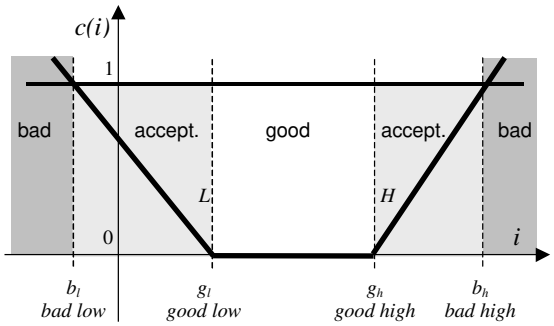

Figure 3: Normalisation of a not necessarily positive indicator value *i* to a positive criterion function *c* with a level of acceptance at one, by means of bad/good limiting values.

Multi-model based criteria evaluation

Realistic control law design is a multidisciplinary task [5] involving the simultaneous minimisation of many design criteria in the presence of various constraints. Typically, in a constrained multi-objective optimisation-based design, the different criteria and constraints are evaluated using computational models developed for different engineering disciplines or resulting from different modelling formalisms. MOPS explicitly supports the usage of different multidisciplinary models (or *multi-models*) to evaluate the design criteria. To each analysis model (e.g. non-linear simulation

model, frequency domain models, etc.) a set of criteria is associated.

Multi-case approach to robust control law design

Robust controller design can be achieved in several ways by appropriately mapping the robustness requirements into design criteria [6]. A kind of 'global' robustness can be achieved by using the multi-case approach. For example, for analysis models depending on uncertain parameters, the robustness against parameter variations can be achieved by trying to apply a unique controller to a whole set of model instantiations, corresponding to different values of physical parameters. Such a set of model instantiations is called a *multi-case model* and ideally characterises the whole range of dynamics variations over the parameter range. In contrast to multi-models, the associated design criteria are the same for each member of the multi-case model. MOPS explicitly supports the multi-case approach for robust controller design, by automatic generation of multi-case models from a given parameterised analysis model (e.g., by using linearisations of a non-linear model in different stationary points and/or for different parameter values).

To structure the multi-model/multi-case approach, MOPS distinguishes common, normal and final models, see Figure 4.

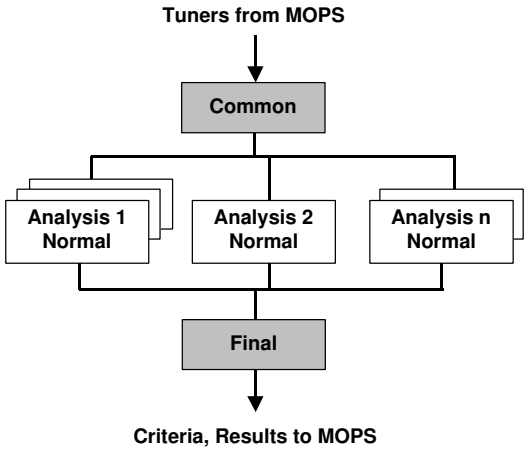

Figure 4: Structuring the multi-model/multi-case approach by common, normal and final models. Normal models can be multi-case models.

The common model is computed first and the results and criteria from there can be used as input to all other models. The purpose of a common model is for example to perform the synthesis of a control law in dependency of the tuners (e.g. LQG), while the resulting controller gains are input to all other multi-case models representing a closed loop system, called normal model. The final model is computed last. It can have input from any other model output. This can be used for example to compute overall criteria, based on the results of all other models.

As a multi-model/multi-case application example Figure 5 shows the problem formulation for a flare control law design in an aircraft automatic landing system [6], as it is presented to the designer by the graphical user interface of MOPS.

3. Solving the basic optimisation problem

The multi-objective/multi-model/multi-case design problems are mapped to the weighted min-max optimisation problem

$$\begin{aligned} &\min_{T} \max_{ijk \in S_m} \{c_{ijk}(T, p_{ij}) / d_{ijk}\} \\ &c_{ijk}(T, p_{ij}) \le d_{ijk}, \qquad ijk \in S_i \\ &c_{ijk}(T, p_{ij}) = d_{ijk}, \qquad ijk \in S_e \\ &T_{min,l} \le T_l \le T_{max,l} \end{aligned} \tag{1}$$

where: S_m is the set of criteria to be minimised, S_i is the set of inequality constraints and S_e is the set of equality constraints; T is a vector containing the tuning parameters T_l to be optimised, lying between the upper and lower bounds $T_{min,l}$ and $T_{max,l}$, respectively; $c_{ijk} \in S_m$ is the k-th normalised criterion of the j-th case of the i-th model and d_{ijk} is the corresponding demand value which serves as a criterion weight [2]; p_{ij} denotes a parameter of the i-th model defining the j-th case. $c_{ijk} \in S_i, S_e$ denotes normalised criteria which are used as inequality or equality constraints. The affiliation of a criterion to one of the groups S_m, S_i or S_e respectively can be changed at any time according to design progress. This provides utmost flexibility in expressing design requirements. For example, a criterion c_{ijk} to be minimised, which satisfies the according demand d_{ijk} after an optimi-

sation run, can be set to an inequality constraint $c_{ijk} \leq d_{ijk}$ in further optimisations. This ensures that the demand for this criterion remains satisfied, while other criteria can be further improved.

The min-max multi-criteria optimisation problem (1) is solved by reformulating it as a standard Non-Linear Programming problem (NLP) with equality, inequality and simple bounds constraints. This NLP-problem is then solved in MOPS by using one of several available powerful solvers implementing local and global search strategies. Besides efficient gradient-based solvers (well-suited primarily for smooth problems), less efficient, but usually more robust gradient-free direct-search based solvers are available to address problems with non-smooth or noisy criteria. Such kind of criteria often occur when engineering design specification are directly translated into optimisation criteria or when truncation errors manifest in the criteria (e.g., from numerical simulation, approximations, etc.) To overcome the problem of local minima to some extent, solvers based on statistical methods or genetic algorithms can be alternatively used.

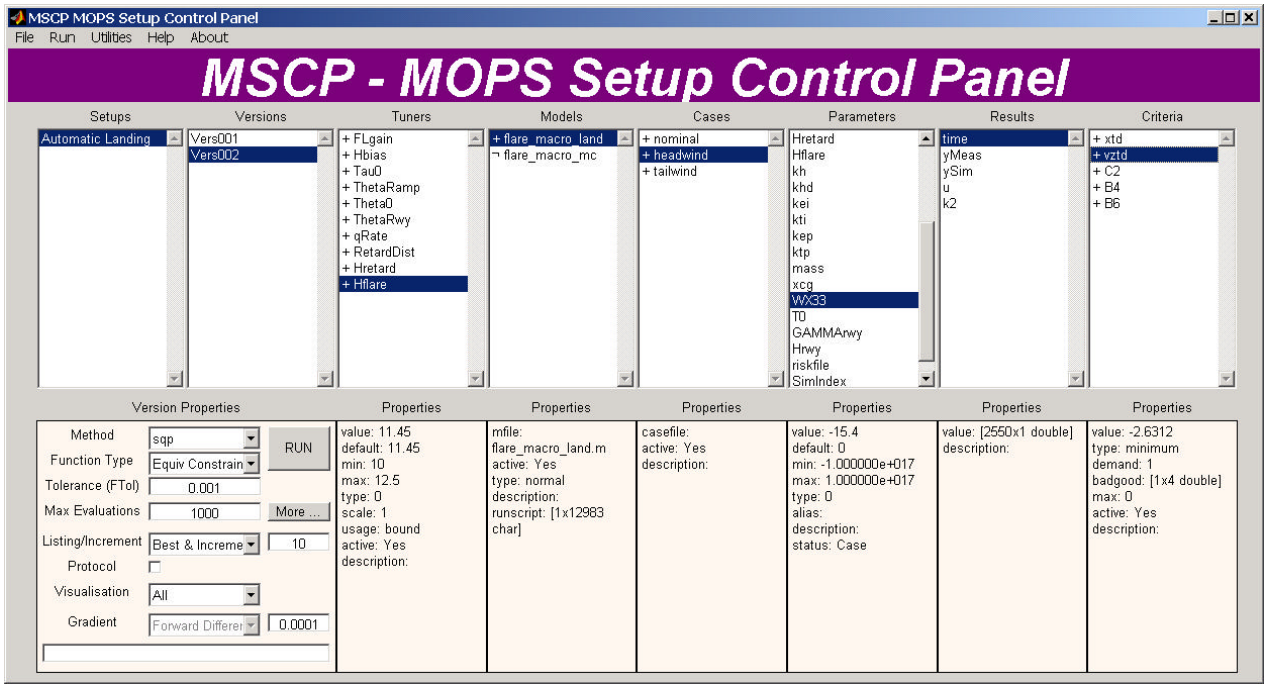

Figure 5 MOPS Set-up Control Panel for flare control laws in an aircraft automatic landing system. The list boxes of the upper row present all objects (Tuners, Models, Cases, Parameters, Results, and Criteria) defined in the set-up and its current version. In the row below the actual values and properties of the selected objects are displayed. Optimisation method properties are set by the menu panel in the lower left corner. Tuners are for example feedback gain **Flgain** and flare initiation height **Hflare**. Criteria are computed from two models: **flare_macro_land**, performing a single landing simulation to compute criteria such as sink rate at touchdown (**vztd**) and landing distance from runway threshold (**xtd**), and **flare_macro_mc** (currently inactive), to compute mean values, standard deviations and risk-based criteria for touch down parameters based on on-line Monte-Carlo analysis.

4. Special Features of MOPS

MOPS provides additional useful functions to enhance design productivity and control the tuning process.

Visualisation and on-line monitoring of design results and design progress

During a multi-objective parameter tuning performed with MOPS, all intermediate iteration steps can be visualised to allow the user a maximum insight into the tuning process. The criteria values are simultane-

ously displayed in a normalised parallel co-ordinate plot [7]. This plot gives an overview of all (normalised) criteria values at successive iterations, allowing the user to quickly figure out which criteria are easy to be fulfilled, or in contrast, are hard to be satisfied. By just watching this plot, it is often easy to detect conflicting criteria which need to be compromised during the design process. In addition, the user is able to conveniently specify his own graphical output of interest related to the criteria computation, such as simulation responses, pole maps, frequency responses etc. All curves belonging to the same design iteration automatically get the same colour. Clicking on any visualised curve or point causes all other graphical outputs at the same iteration to be also highlighted.

At the end of the tuning process (or after an interrupt), the user has the possibility to examine the last as well as all intermediary results by simply selecting a curve/point corresponding to a particular design candidate. Facilities are built in to browse the whole design history step by step (either forward or backward). The user typically accepts the best iteration found by the optimiser, but occasionally, based on graphical outputs, he can decide to select results computed at an intermediary iteration.

Figure 6 shows a typical visualisation set-up used during optimisation of controller parameters with MOPS (in this case, for tuning an autopilot for the automatic landing of a civil aircraft). There are five design criteria which are optimised for a multi-case model, consisting of three wind situations (head wind, tail wind, and average wind). As can be seen from the parallel co-ordinates (lower right), all five normalised criteria (see also Figure 5) are shown in parallel for the three wind cases. For each case of multi-case model, a window is visible showing simulation results (upper right). Textual information with actual values of tuning parameters and associated criteria values corresponding to the highlighted design iteration (fat) is displayed in the MATLAB command window (lower left).

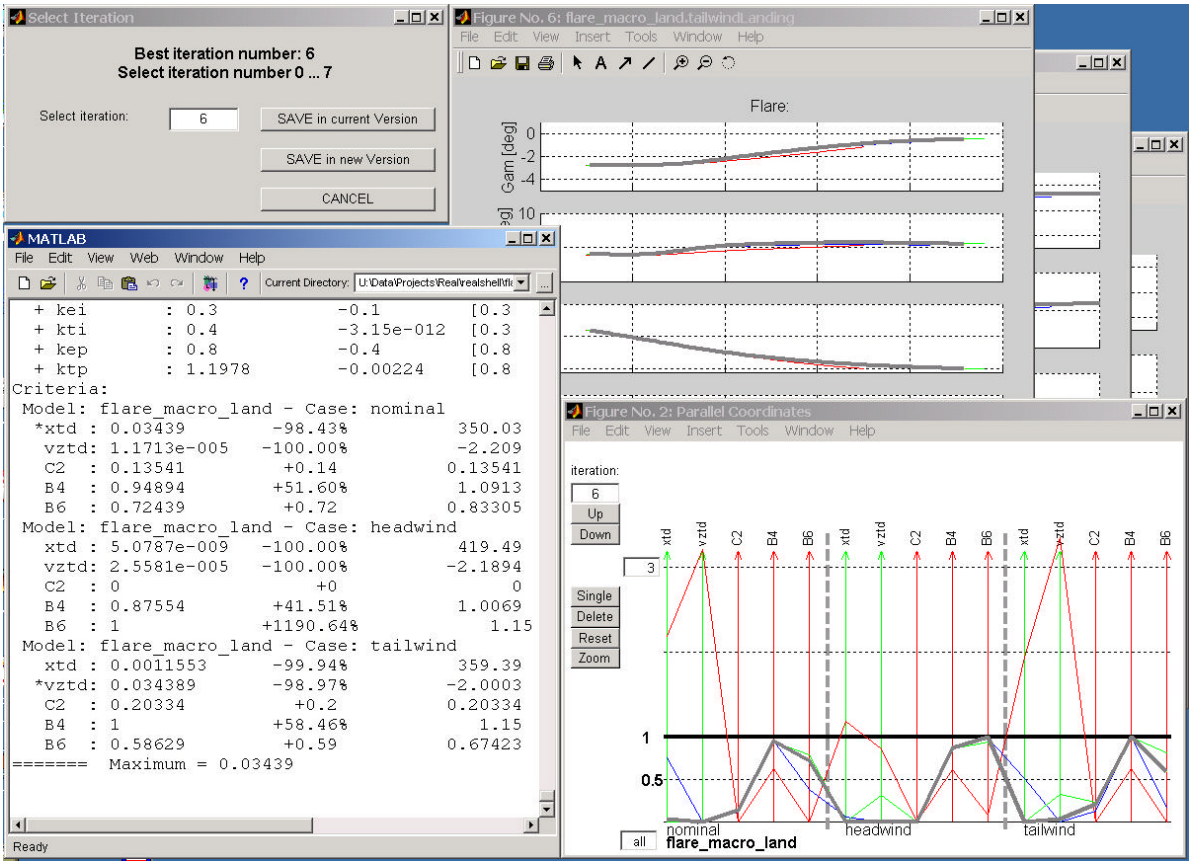

Figure 6 Typical on-line visualisation in MOPS: Time response indicators are displayed in combination with performance measures in parallel co-ordinates to select the result preferred by the designer.

Robust tuning via Monte-Carlo simulation

An alternative to the multi-case approach for robust control law design is to use statistical characteristics of control law design criteria like mean, standard deviation or limit risks with respect to the uncertain parameters of the model. To evaluate such quantities, Monte-Carlo simulations can be performed with randomly disturbed parameters. MOPS provides tools to support this kind of statistical analysis within the optimisation process, by allowing repeated evaluations of a set of selected criteria using randomly generated parameter sets with specified statistics. The computed statistical characteristics of the selected criteria (e.g., mean value) are used in the optimisation instead the original criteria. Figure 7 shows optimisation progress for the statistical indicators and criteria of the automatic landing design problem [6].

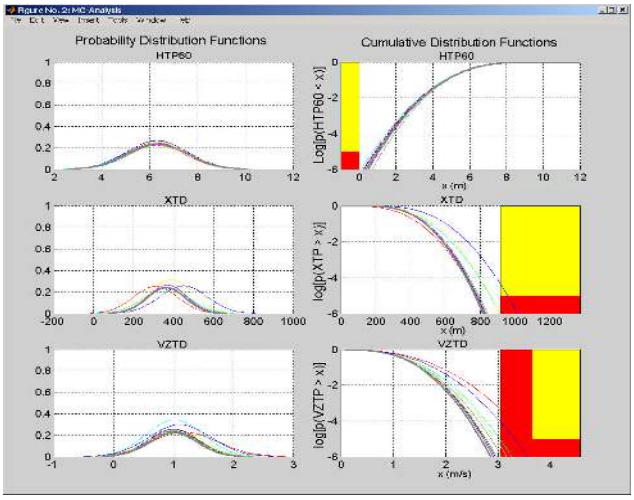

Figure 7: Optimisation with Monte-Carlo based criteria for the automatic landing controller. For example, the risk of landing with more that 3.1 m/s sink rate (**VZTD**) should be less than 10^{-6}. This implies that the curve of the cumulative distribution plot (lower right) should be outside the shaded area. This is achieved by the optimiser for VZTD, as well as the other touch down parameters. The distribution functions (left) are assumed to be Gaussian.

Control law robustness assessment

The assessment of robustness of designed control laws over a whole region of operation points and/or in the presence of parametric variations is an important task which completes the design process, see Figure 1. This can be done within MOPS by formulating the assessment problem for each criterion as an "anti"-optimisation problem to determine the parameter combination leading to the worst performance [2]. In this way, possible design deficiencies can be detected and the computed worst-case parameter combinations can also serve to redefine the multi-case models employed in the control law tuning. Since assessment problems are global optimisation problems, optimisation tools based on global search strategies or combinations of parameter raster and local optimisation can be employed in MOPS to solve this kind of problems.

Parameter estimation

MOPS supports the parameter estimation of nonlinear dynamic models by using the nonlinear least-squares approach. Since any parameter estimation problem can be formulated also as a general multi-criteria optimisation problem, other existing solvers can be equally employed as well. In any case, the criteria scaling features provided by MOPS are useful for proper formulation of estimation errors.

Handling many parameters and criteria

Each tuning parameter and each design criterion is typically handled in MOPS individually, having their own names, limiting values, scaling factors etc. In the case of many parameters, these can be grouped in a matrix and referred by a single name. Similarly, many criteria can be grouped together into a vector, and are handled through a unique name. These recent enhancements built in MOPS allow to easily handle large scale optimisation problems with many parameters and many criteria.

Data and version management

The solution of realistic design problems typically require many experimental steps/iterations to arrive to the best controller structure/setting. This leads to a great amount of results and associated information which are stored during computations. MOPS provides a convenient framework to support the user's design decisions by recording complete information on various design steps, and storing all relevant data to allow the comparison of various design outcomes. It is possible within MOPS to recover all data used to produce graphical and textual outputs within different experiments, allowing backtracking and branching of the design process.

Use of external simulation or analysis servers

The computation of the criteria may require the use of existing simulation/analysis programs running on specific hardware architectures. An easy to use application program interface (API) has been implemented to facilitate the interfacing of such simulation tools with MOPS.

Parallel computation of criteria

Criteria evaluations may be very time consuming, especially when long simulations or complicated analyses are involved. Distributed computation, allowing parallel evaluation of criteria, can alleviate this problem. The underlying multi-model/multi-case formulation of the design problems within MOPS is well suited to a natural parallelisation of criteria evaluations. Based on the *remote shell* concept for process communication, MOPS API-functions (see chapter 5) are available to distribute the computations needed for criteria evaluations on external simulation or analysis servers. In this way, the criteria evaluations for multi-case models can be done in parallel on different machines in a heterogeneous network. MOPS automatically ensures the synchronisation of the submitted processes.

5. The MOPS software architecture

The software architecture of MOPS has several functional software layers (see Figure 8.). To solve the basic optimisation problem (1), two basic layers, the *solver layer* and *MEX-interface layer*, strongly interact via dedicated process communication protocols. The *API-layer (Application Program Interface)* provides a comprehensive user interface to the two basic layers for easily defining and solving complex multi-objective/multi-model/multi-case based robust design problems.

The *solver layer* consists of a collection of NLP-solvers. Since each solver is implemented as an independent task, new solvers can be easily added as the need arises. At both MEX- and API-interface levels, only a minimal programming effort is required to support a newly included solver. This modular organisation basically confers an open software architecture to the MOPS optimisation environment.

The *MEX-interface layer* can be seen as the primary layer to solve min-max multi-objective optimisation problems by calling various available NLP-solvers. This interface is provided by a unique *mex*-function, which converts a weighted min-max multi-objective optimisation problem of the form (1) into a standard NLP and calls one of the available solvers. A reverse-communication like interface is the key feature which allows to easily integrate, in a single flexible optimisation environment, a heterogeneous collection of optimisation tools (e.g., written in different programming languages, having different options and parameters lists, using different function and gradient evaluation strategies, etc.).

The *API-layer* provides a modular set of basic functions to be used in MATLAB-scripts and *m*-files and for command input to define design problem set-ups and solve the underlying optimisation task. In addition, these API-functions serve as call-backs for the existing graphical user interface. The API-functions comprehend:

Run script API	*define interface between MOPS/ analysis model: parameters, criteria, results*
Set-up API	*define and edit a set-up*
Visualisation API	*define visualisation and on-line monitoring*
Criteria API	*computation of basic criteria*
Parallel API	*define distributed computation*
Monte-Carlo API	*basic Monte-Carlo functions*

The computation of criteria has to be formulated in so called ‘model-run scripts’. These m-file scripts define the interface between MOPS and an arbitrary, problem specific analysis model together with the corresponding criteria computation in a structured manner.

6. Conclusions and Outlook

The described software environment offers flexibility and visibility both in controller design process and controller design problem set-up, which is necessary to successfully handle complex design tasks. Visibility of design is based on the formulation of design requirements as normalised criteria functions, which can be systematically compromised by multi-objective optimisation.

Multi-objective optimisation based design is still an iterative design technique, since in practice no single ideal ‘optimal’ solution exits. The software described here supports the designer in interactive experimenting with various criteria, controller structures and

parameterisations to achieve best-possible performance. It also supports the designer in the formulation of his design problem as a clearly structured multi-objective optimisation problem by offering the multi-model/multi-case approach. Several facilities like on-line visualisation or distributed computation of criteria increase design efficiency.

MOPS has been applied in several application fields, such as robotics, aerospace, automotive control, etc. To further improve design efficiency, dedicated criteria libraries and user interfaces (level 3 in Figure 2) are in preparation.

The aim of the multi-objective tuning approach is to achieve solutions that satisfy all requirements concurrently. The attained solutions are compromises between competing requirements. Design-tuning ends when the optimiser finds a satisfactory solution with agreed upon trade-offs. Local parameter optimisation techniques only find local pareto-optima. To find a global solution set, optimisation of non-convex functions has to be applied. In this case, guided random search, like response surface techniques or evolutionary genetic algorithms, have to be investigated further for their usability in control synthesis tuning.

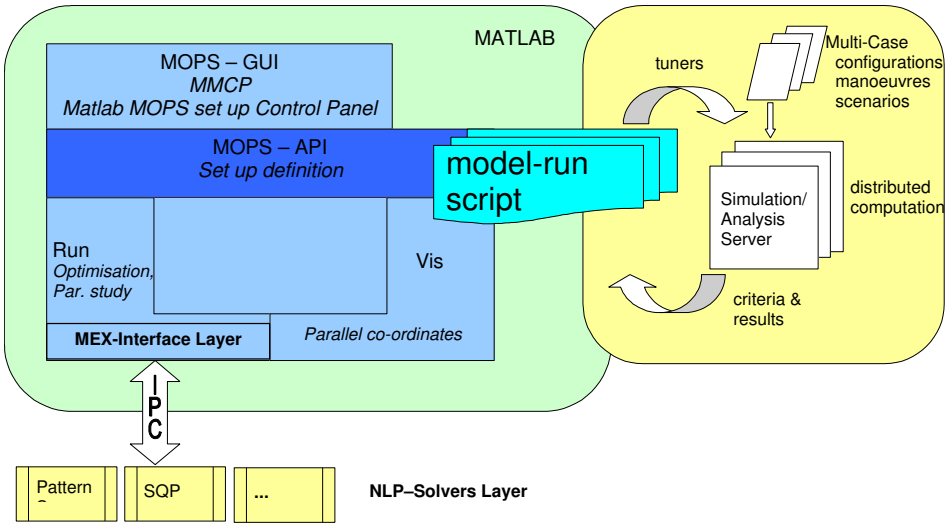

Figure 8. MOPS-Software Architecture (layers, process communication (IPC), interface via model-run-scripts to criteria servers and external processes).

7. References

[1] H.-D. Joos. MOPS - Multi-Objective Parameter Synthesis, User's Guide V1.21, DLR Technical Report IB-515-02-01, 2002.

[2] H.-D. Joos. A Methodology for Multi-Objective Design Assessment and Flight Control Synthesis Tuning. Aerospace Science and Technology, no. 3, pp. 161-176, 1999.

[3] H.-D. Joos, A. Varga and R. Finsterwalder. Multi-Objective Design Assessment. IEEE Symposium on Computer-Aided Control System Design, Kohala, Hawai'i, USA, 1999.

[4] MATLAB®, The Language of Technical Computation. Using MATLAB, The MathWorks, Inc., 2000.

[5] M. B. Tischler, J. D. Colbourne, M. R. Morel, D. J. Biezad, W. S. Levine and V. Moldoveanu. CONDUIT – A New Multidisciplinary Integration Environment for Flight Control Development. AIAA-97-3773, pp. 1759-1781, 1997.

[6] G. Looye, H.-D. Joos and D. Willemsen. Application of an Optimisation-based Design Process for Robust Autoland Control Laws. In Proc. of The AIAA Guidance, Navigation and Control Conference 2001, Montreal CA, 2001.

[7] A. Inselberg., The plane with parallel coordinates. In The Visual Computer, 1985