Developing a Simulation Tool Box in MATLAB and Using It for Non-linear Adaptive Filtering Investigation*

Oleg Gorokhov and Innokenti Semoushin

Ulyanovsk State University, 42 Leo Tolstoy Str., 432970 Ulyanovsk, Russia
gorohov@icbank.ru SemoushinIV@ulsu.ru http://staff.ulsu.ru/semoushin/

Abstract. In this paper we develop a special purpose tool box for complex computational tasks solution in the area of stochastic adaptive system design. The proposed tool box is used to analyze the influence of different factors on the quality of numerical algorithms.

1 Introduction

Modern system design requires high performance computational modelling tools. MathCAD, MATLAB and Maple are examples. The complexity of problems in the adaptive filtering area often does not allow us to use the standard procedures and tool boxes to analyze the problems even of small dimensions. The reason lies in the amount of time necessary for computing while analyzing the influence of a set of many factors on the algorithm performance. This raises the problem of developing a special tool for complex system investigation.

The purpose of this paper is twofold. The first goal is to develop an efficient tool box for specific problems in the field of adaptive linear or non-linear filtering. The second part of the paper demonstrates the application of the designed tool box and presents the simulations results. The core of the designed tool box is implemented as a dynamic link library, and its interface part is made MATLAB-compatible.

2 Adaptive Non-linear Filtering Problem

Consider the linear stochastic time-invariant discrete-time model

$$\begin{aligned} x_{t+1} &= \Phi x_t + \Gamma w_t \\ z_t &= H x_t + v_t \end{aligned} \tag{1}$$

which is widely used in processing the experimental data for a state of dynamical plants in stochastic environment. Here $x_t \in \mathbb{R}^n$ is the state vector, $z_t \in \mathbb{R}^m$ is the

* This work was supported in part by the Russian Ministry of Education (grant No. T02-03.2-3427).

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2658, pp. 436–445, 2003.

measurement vector and $\{w_0, w_1, \ldots\}$ and $\{v_1, v_2, \ldots\}$ are zero-mean independent sequences of independent identically distributed random vectors $w_t \in \mathbb{R}^q$ and $v_t \in \mathbb{R}^m$ with covariances Q and R respectively. Measurements are assumed to be incomplete, i.e. $m < n$.

Classical approach to the state estimation for model (1) is that corresponding to the Kalman filtering. However, sometimes it is impossible to use the Kalman filter if the system contains a predesigned part which is nonlinear by its nature. When noise distributions are not Gaussian, optimal filter is not Kalman. In these cases, the whole system should be characterized by nonlinear equations, whose parameters are difficult to optimize. For such situations, an adaptive (or learning) approach seems to be the only possible way. It becomes absolutely necessary if, for a variety of reasons, parameters of model (1) are not precisely known and thus should be identified from experimental data [1], [2].

We assume that the initial time is placed at $-\infty$ and adopt the non-restrictive assumptions that $R > 0$, $Q \geq 0$, and system (1) is stabilizable and completely observable. Additionally, the spectral radius of Φ is assumed to be strictly less than one to have all the processes in (1) wide-sense stationary. In the below experiments, uncertainty inherent to (1) resides in Γ, Q and R only.

In the most case, the estimator $y_t \in \mathbb{R}^n$ is given by a nonlinear functional $\mathcal{O}(\cdot)$ intended to generate a p-step-ahead (predicted) estimate of the state x_{t+p} using all available measurements $z_{t-\tau}, 0 \leq \tau < \infty$. Thus, we have

$$y_t = \mathcal{O}(z_{t-\tau}, 0 \leq \tau < \infty) \tag{2}$$

where each k-th component y_t^k of y_t is desribed by the discrete time Volterra series

$$y_t^k = \sum_{l=1}^{m} \sum_{d=1}^{\infty} \{ \sum_{\tau_1=0}^{\infty} \cdots \sum_{\tau_d=0}^{\infty} h_{kld}(\tau_1, \ldots, \tau_d) z_{t-\tau_1}^{(l)} \cdots z_{t-\tau_d}^{(l)} \}$$

and where $l = 1, 2, \ldots, m$ denotes the component index in the vector of measurements z_t, d is the order of nonlinearity of $h_{kld}(\tau_1, \ldots, \tau_d)$, the d-order kernel of Volterra series. We assume the estimator to be stable so that $\{y_t\}$ is a wide-sense stationary sequence.

Let the system error, original performance index and a problem to be solved are as follows:

$$e_t = x_{t+p} - y_t, \qquad J_o = \mathrm{E}\{\|e_t\|^2\} \quad .$$

Problem 1. Given performance index J_o, formulate the auxiliary performance index J_a depending on some available sequence ε_t so that

$$J_a = \mathrm{E}\{\|\varepsilon_t\|^2\} = J_o + const \tag{3}$$

where *const* does not depend on set of the nonlinear estimator parameters, thus performance indices J_a and J_o have the same optimal points on the set.

We use a new solution to the problem given in [3] with the following notations: $z_{t+1-s}^t \in \mathbb{R}^{sm}$ is a column vector composed of column vectors z_{t+1-s} through z_t, s stands for the observability index, and T is the $n \times n$ observability matrix.

3 Nonlinear Filtering and MATLAB Algorithm Implementation

In this section we present the general computational algorithm for adaptive estimation of parameters. We also demonstrate how the adaptive filtering algorithm is implemented by the designed MATLAB tool box procedures. The MATLAB object `objExperiment` (Fig. 1) includes all information and settings for numerical simulations such as stabilization filter time, signal noise ratio, initial conditions.

```
objExperiment =struct('Tmax', 'TimeStab', 'TimeSwitching', ...
'Alpha', 'Qbefore', 'Qafter', 'Rbefore', 'Rafter', 'PHIbefore',...
'PHIafter', 'P0', 'AdaptiveProcedure');
```

Fig. 1. Tool box experiment object structure

Assuming, for generality sake, that the uncertainty can reside in matrices Φ, Γ and covariances Q and R, we denote the parameterized versions of these matrices as $\Phi_\theta, \Gamma_\theta, Q_\theta$ and R_θ respectively. The system (1) state generation procedure consists of two parts: before change point in the system and after change. The goal of the after-change algorithm is to identify the new parameter values. Tool box function `phiModel` (Fig. 2) demonstrates the top-level programming code without taking into consideration the details hidden in the low-level functions `phiModelInitialization`, `phiGenerateNoise` and others.

The model state estimates are obtained as a result of nonlinear transformation $\mathcal{L}_\theta\{\cdot\}$ of the feedback suboptimal filter estimate

$$\begin{aligned} \tilde{x}(t_{i+1}^-) &= \Phi_{\theta_0}\tilde{x}(t_i^+) \\ \tilde{x}(t_i^+) &= \mathcal{L}_{\theta_0}\{\tilde{x}(t_i^-) + K_{\theta_0}\nu(t_i)\} \\ \nu(t_i) &= z(t_i) - H\tilde{x}(t_i^-) \ . \end{aligned} \tag{4}$$

The gain K_θ is replaced by the result of each iteration (the whole identification process). The initial value for K_{θ_0} is set as some nominal value which is chosen a'priori to satisfy the stability conditions. The corresponding implementation is given at (Fig. 2) where the estimation of the system state vector is performed via Kalman filter before reaching stabilization and when via stabilized filter.

The adaptive model is appended to this system and started with initial state taken from the suboptimal filter (4). It has the following form

$$\begin{aligned} \tilde{g}(t_{i+1}) &= A_\theta\hat{g}(t_i) \\ \hat{g}(t_i) &= \mathcal{L}_\theta\{\tilde{g}(t_i) + D_\theta\eta(t_i)\} \\ \eta(t_i) &= z(t_i) - H_*\tilde{g}(t_i) \end{aligned} \tag{5}$$

```
function [objObject] = phiModel(objExperiment, objObject, t)
      if (t == 0)
            phiModelInitialization(objExperiment, objObject);
      else
            if (t < objExperiment.TimeSwitching)
                  objObject = phiBeforeSwitching(objObject,t);
            else objObject = phiAfterSwitching(objObject,t);
            end;
      objObject = phiGenerateNoise(objExperiment, objObject,t);
      objObject = phiGenerateDynamics(objExperiment, objObject,t);
      end;
end;
```

Fig. 2. Tool box model object code generating state of the system at time t

where $A_\theta = T_\theta \Phi_\theta T_\theta^{-1}, H_* = HT_\theta^{-1}$ and T_θ is the observability matrix defined in [3]. Let us assume that the collective parameter θ represents the set of adjustable parameters in the model indexed accordingly (the Kalman gain D_θ, the matrix A_θ and the nonlinear transformation $\mathcal{L}_\theta\{\cdot\}$).

```
function [objFilter] = phiFilter(objExperiment, objObject, t)
      if (t == 0)
            phiFilterInitialization(objExperiment, objObject);
      else
            if (t < objExperiment.TimeStab)
                  objFilter = phiKalmanFilter(objObject, objFilter,t);
            end;
      objFilter = phiStabFilter(objFilter,t);
      end;
end;
```

Fig. 3. Tool box filter implementation

Denote the stackable vector of $[\eta(t_{i-s+1}), \ldots, \eta(t_i)]$ as $\mathcal{H}_{i-s+1}^i$ where s is the maximal partial observability index. Then the model error between the adaptive and suboptimal models can be written in the following form

$$\varepsilon(t_i) = \mathcal{N}(D_\theta)\mathcal{H}_{i-s+1}^i \tag{6}$$

where $\mathcal{N}(D)$ is the structure transformation of adaptive model gain D as defined in [3].

```
function [objAdFilter] = phiAdaptiveFilter(objExperiment, objObject, t)
      if (t == objExperiment.timeSwitching)
            objAdFilter = phiAdFilterInitialization(objObject);
      else
            objAdFilter = phiKalmanFilter(objObject, objAdFilter,t);
            objAdFilter = phiSensitivityModel(objAdFilter,t);
                  if phiAdaptiveStep(objAdFilter, t)
                        objAdFilter = phiSensitivity(objAdFilter,t);
                        objAdFilter = phiGradient(objAdFilter,t);
                        objAdFilter = phiApproximation(objAdFilter,t);
                        objAdFilter = phiTrialStep(objAdFilter,t);
                        objAdFilter = phiStability(objAdFilter,t);
                  end;
      end;
end;
```

Fig. 4. Tool box adaptive filter implementation

The sensitivity model that reflects the influence of the adjustable parameters on the model error (6) and in fact is the partial derivatives of vector $\varepsilon(t_i)$ wrt. vector θ, is defined by three types of recursions according to the placement of adjustable parameter. Let μ denote the sensitivity model state vector. We have

$$\begin{array}{l} \tilde{\mu}_j(t_i) = A_\theta \hat{\mu}_j(t_{i-1}) \\ \hat{\mu}_j(t_i) = \frac{\partial \mathcal{L}_\theta}{\partial x}((I - D_\theta H_*)\tilde{\mu}_j(t_i) + \frac{\partial D_\theta}{\partial \theta_j}\eta(t_i)) \end{array} \tag{7}$$

where θ_j is a parameter of vector D_θ. The second type of sensitivity equations is defined for parameters θ_j of transition matrix A:

$$\begin{array}{l} \tilde{\mu}_j(t_i) = \frac{\partial A_\theta}{\partial \theta_j}\hat{g}(t_{i-1}) + A_\theta \hat{\mu}_j(t_{i-1}) \\ \hat{\mu}_j(t_i) = \frac{\partial \mathcal{L}_\theta}{\partial x}(I - D_\theta H_*)\tilde{\mu}_j(t_i) \ . \end{array} \tag{8}$$

The influence of the nonlinear estimator parameters on the error (6) are calculated as follows

$$\begin{array}{l} \tilde{\mu}_j(t_i) = A_\theta \hat{\mu}_j(t_{i-1}) \\ \hat{\mu}_j(t_i) = \frac{\partial \mathcal{L}_\theta}{\partial \theta_j} + \frac{\partial \mathcal{L}_\theta}{\partial x}\tilde{\mu}_j(t_i) \ . \end{array} \tag{9}$$

All recursions start with initial values $\hat{\mu}_j(t_0) = 0$ for each θ_j. Let vector $\xi_j(t_i)$ be used to denote $-H_*\tilde{\mu}_j(t_i)$. The history for vectors $\xi_j(t_i)$ and $\mathcal{H}_{i-s+1}^i$ should be accumulated during the iterations (4)-(9) till s last values are re-calculated.

The sensitivity matrix $\mathcal{S}(t_i)$ is computed as follows

$$\mathcal{S}(t_i) = \frac{\partial \mathcal{N}(D)}{\partial \theta_j} \mathcal{H}^i_{i-s+1} + \mathcal{N}(D) \frac{\partial \mathcal{H}^i_{i-s+1}}{\partial \theta_j} \tag{10}$$

where $\frac{\partial \mathcal{H}^i_{i-s+1}}{\partial \theta_j}$ is the stackable vector of the s last values $\xi_j(t_k)$. Then the gradient model is defined as the product of transposed sensitivity matrix $\mathcal{S}(t_i)$ and $\varepsilon(t_i)$

$$\begin{aligned} q(t_i) &= \mathcal{S}^T(t_i)\varepsilon(t_i) \\ \hat{q}(t_i) &= \beta \hat{q}(t_{i-1}) + (1-\beta) q(t_i) \end{aligned} \tag{11}$$

where β is the exponential smoothing factor, $0 \leq \beta < 1$.

```
objObject = [];
objFilter = [];
objAdFilter = [];
objObject = phiGetProblemDesc(objObject);
while phiIterationsRange(objExperiment) do
      while phiSnrRange(objExperiment) do
            while phiStabilityRange(objExperiment) do
                  for t = 0:objExperiment.Tmax;
                        objObject = phiModel(objObject, t);
                        objFilter = phiFilter(objFilter, t);
                        objAdFilter = phiAdaptiveFilter(objAdFilter, t);
                  end;
            phiSaveResults(objAdFilter);
            end;
      end;
objResults = phiAnalyseResults(objFilter, objAdFilter);
phiMakeGraphs(objResults);
end;
```

Fig. 5. Tool box numerical simulation function

The suboptimal adaptation procedure (SAP) shown here as one of possible variants, is defined for each adjustable parameter θ_j through the recursion

$$\begin{aligned} p_j(t_{i+1}) &= p_j(t_i) + \| \tfrac{\partial \varepsilon(t_i)}{\partial \theta_j} \|^2 \\ \pi(t_i) &= \hat{\theta}(t_i) - \mathrm{diag}(p_j(t_{i+1}))^{-1} \hat{q}(t_i) \ . \end{aligned} \tag{12}$$

(Here and below $\pi(t_i)$ denotes a trial value for $\hat{\theta}(t_{i+1})$). The stability condition of the linear part of estimator, $\rho\left[(I - DH_*)A\right] < 1$, should be checked for trial estimate $\pi(t_i)$. Also, the constraints of the nonlinear estimator should be satisfied for the next estimate $\hat{\theta}(t_{i+1}) = \pi(t_i)$.

The adaptive filter tool box function is depicted by Fig. 4. The renewal of the adaptive filter parameters with the new calculated values occurs only at each adaptive step, which is obtained by function `phiAdaptiveStep`.

4 Simulations Results

The integral percent error (IPE) will show the algorithm performance. The tool box numerical simulations function (Fig. 5) depends on the possible range of aspects to be evaluated. The extended experiment is planned to analyze the influence of a set of different aspects on the IPE-characteristic of the algorithm, to reveal some effects during the identification process and finally to illustrate the applicability of the nonlinear filtering algorithm and its MATLAB tool box implementation. The set of aspects includes signal-to-noise ratio (SNR defined by $\|Q\|/\|R\|$), stability property of the object, the presence of nonlinear estimator, the type of adaptation procedure (suboptimal, i.e. SAP, optimal, i.e. OAP, or simple stochastic approximation, i.e. SSAP), a number of iterations, and initial values for the estimates. Main user function (Fig. 5) is used to execute the algorithm for the ranges of analyzed aspects (`phiSnrRange`, `phiStabilityRange`), to compare results (`phiAnalyseResults`) and to plot graphs (`phiMakeGraphs`).

```
[RANGES]
      stabInterval=((0.01, 1.000); 0.0; 10.0);
      snrInterval=((0.001, 100.0); 0.0; 10.0);
      snrAdaptiveProcedure=SAP;
      snrvsiterationsIpeLevel=10.0;
      adjMaxAdapationTime=((10000, 10000000); 0.0; 10.0);
[SYSTEM]
      PHI = [-0.8, 0.1];
      Q = 0.04;
      R = 0.06;
      Gd = [0.0; 0.4];
      H = [1.0, 0.0];
```

Fig. 6. Experiment settings

We investigate the properties of the proposed algorithm and define the IPE as follows

$$\rho_{ipe} = \frac{\|\theta^*(t_i) - \theta^{**}\|}{\|\theta^{**}\|} \tag{13}$$

where $\theta^*(t_i)$ is the estimate of parameter θ obtained at time t_i and θ^{**} is the optimal (i.e. true) value of θ.

We consider the following example of the second order model

$$\begin{aligned} x(t_{i+1}) &= \begin{bmatrix} 0 & 1 \\ f_1 & f_2 \end{bmatrix} x(t_i) + \begin{bmatrix} 0 \\ \alpha \end{bmatrix} w_d(t_i) \\ z(t_i) &= Hx(t_i) + v_d(t_i) \end{aligned} \tag{14}$$

with unknown values of the state and measurement noise covariances Q and R and $\alpha = 0.4$, $\beta = 0.0$. The measurement matrix H is $\begin{bmatrix} 0, 1 \end{bmatrix}$ and parameters f_1 and f_2 are known and fixed. Nonlinear estimator has the nonlinear state transformation function which is linear for small state values and constant outside the linear area. The tangent of the linear part θ is unknown and should be identified. In this experiment we simultaneously identify the parameters of adaptive filter and the optimal tangent θ of the nonlinear estimator.

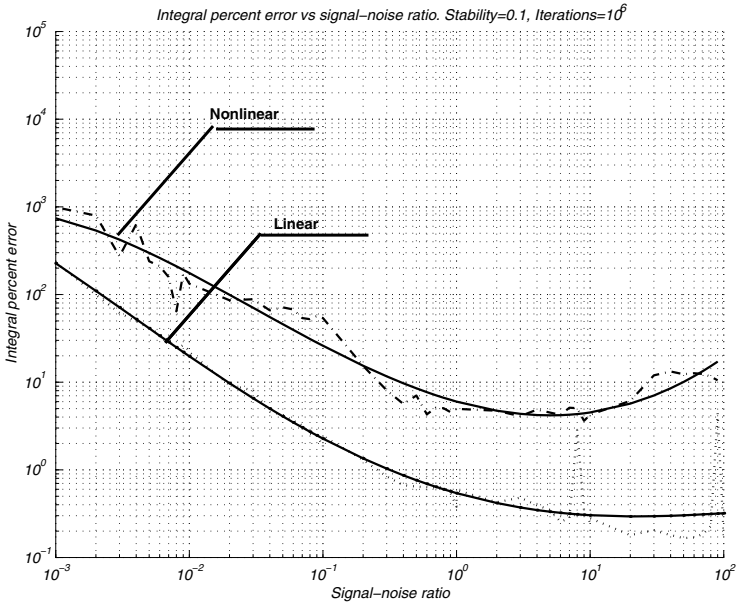

Fig. 7. IPE characteristic for linear and nonlinear problems, stability factor - 0.1

The experiment is a set of algorithm runs with chosen algorithm settings and defined values of modelling aspects. We distinguish the primary and secondary modelling aspects of the experiment that corresponds to an interval of values and single value accordingly. Experimental graphs depict the IPE-characteristic behavior vs the primary aspect interval with the secondary aspects values chosen arbitrarily. The experiment settings for considered example are set in tool box configuration file as shown in Fig. 6.

The primary modelling aspect is SNR, and we successively analyze the SNR-interval $[1.0 \cdot 10^{-3}, \ldots, 1.0 \cdot 10^{2}]$.

Graphs of Fig. 7 and Fig. 8 represent the IPE behavior vs the time scale for nonlinear estimation for different stability factors. In all other cases the quality of the estimates becomes better as the number of iterations grows.

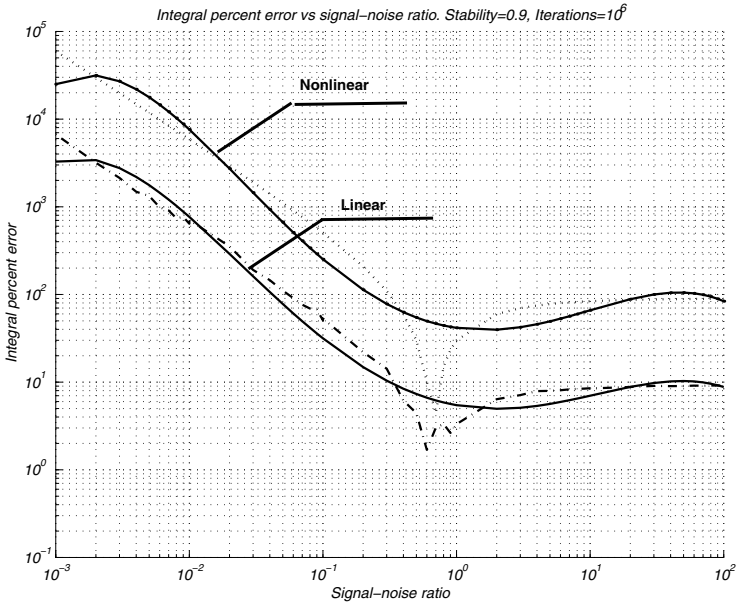

Fig. 8. IPE characteristic for linear and nonlinear problems, stability factor - 0.9

The stability factor depends on location of the eigenvalues of matrix Φ and it occurs that there exists an insensitivity area effect of the considered nonlinearity. This effect entirely defines the shape of the IPE-graph while the SNR interval $[1, \ldots, 10^2]$ does not influence on the quality of estimating process for stability factor 0.9. However, as the number of iterations is increased, the IPE-quality becomes better till the non-improvable level corresponding to the insensitivity area, is attained. This effect can be revealed on Fig. 9.

5 Conclusions

In this paper, we develop the special purpose MATLAB compatible tool box and demonstrate its applicability to computational investigation of complex systems described in terms of high-dimensional vector-matrix stochastic difference equations. This project was motivated by developing novel numerical algorithms for adaptive identification of a non-linear optimal discrete-time steady-state estimator intended to predict the state of the given stochastic system.

This is only one of the possible applications of the designed tool. Another problem investigated with the tool box is iterative control design [4]. The tool

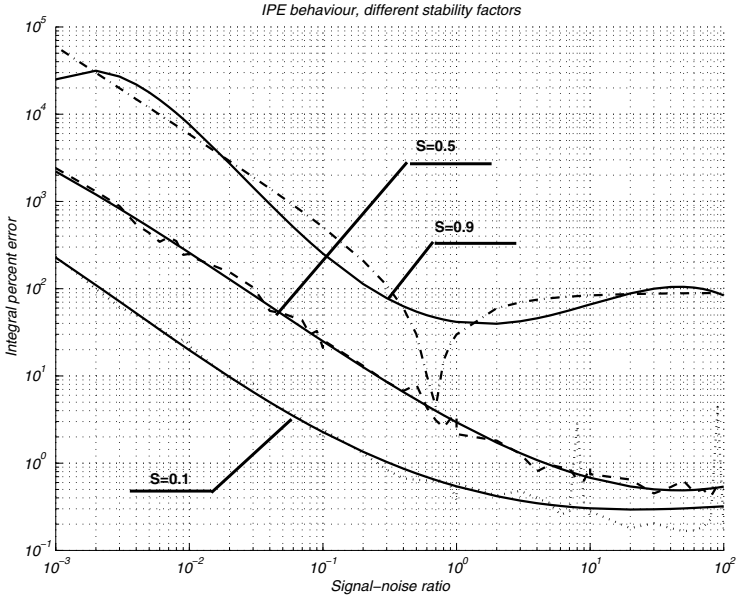

Fig. 9. IPE characteristic for different stability factors

box dramatically extends the capabilities of MATLAB as it makes computational experiments far easier to conduct.

References

1. Landau, I.D. (ed.): Identification des Systemes. Les Bases de l'Identification des Systemes. Hermes, Paris (2001)
2. Caines, P.: Linear Stochastic Systems. John Willey & Sons, New York Chichester Brisbane Toronto Singapore (1988)
3. Semoushin, I.V., Tsyganova, J.V.: Indirect error control for adaptive filtering. In: Neittaanmaki, P., Tiihonen, T., Tarvainen, P. (eds.): Proc. of the 3rd European Conference on Numerical Mathematics and Advanced Applications. World Scientific, Singapore New Jersey London Hong Kong (2000) 333–340
4. Semoushin, I., Gorokhov, O.: Computational processes in iterative control design. In: Sloot, P.M.A., Kenneth Tan, C.J., Dongarra, J.J., Hoekstra, A.G. (eds.): Computational Science – ICCS 2002. Lecture Notes in Computer Science, Vol. 2329. Springer-Verlag, Berlin Heidelberg New York Barcelona Hong Kong London Milan Paris Tokyo (2002) 186–195