Use of MATLAB Environment for Simulation and Control of CSTR

Jiri Vojtesek, Petr Dostal

***Abstract*—This contribution presents the usability of the mathematical software MATLAB ® (MATrix LABoratory) in the field of simulation of the steady-state, dynamic behaviour and adaptive control of the Continuous Stirred Tank Reactor (CSTR). These types of chemical reactors belong to the class of nonlinear lumped-parameters systems mathematical model of which is described by one or more Ordinary Differential Equations (ODEs). The simple iteration method was used for steady-state analysis of the system while the Runge-Kutta's method was employed for the numerical solution of the set of ODE. Both methods are simple, provides sufficient results and they are easily programmable which was important in our case. The presented adaptive approach used for controlling of the system provides sufficient results although the system has negative properties from the control point of view. The benefit of this paper can be found in the simulation program made in MATLAB with the use of Graphical User Interface (GUI) that provides user possibilities to examine simulations without changing of the program code.

***Keywords*—Matlab, modeling, simulation, adaptive control, recursive identification.

I. Introduction

THE most of processes in the real world, not only in the industry has nonlinear behaviour. On the other hand, chemical reactors belong to the most often equipments in the chemical and biochemical industry [1] and that is why this paper is focused on one particular member of this family – the Continuous Stirred Tank Reactor (CSTR) with exothermic reaction inside.

Specific design of the controller is usually proceed by few very important steps. Not every property of the controlled system is known before we start and that is why we perform simulation experiments on the system. There are two main types of the investigating of the system's behaviour – (1) experiment on the real model and (2) computer simulation. Computer simulation is very often used at present as it has many advantages over an experiment on a real system, which is not feasible and can be dangerous, or time and money demanding.

Simulation and modelling possibilities rise with the increasing impact of the digital technology and especially with the computer technology which grows exponentially every moment.

The mathematical model of this particular CSTR is described by the set of two nonlinear Ordinary Differential Equations (ODEs) which are constructed with the use of material and heat balances inside. Examples for deriving such mathematical models can be found in [2]. The steady-state analysis investigates behaviour of the system in the steady-state which from the mathematical point of view means numerical solving of the set of nonlinear algebraic methods. The simple iteration method [3] was used in this case because the system fulfills the convergence condition. The next step is the dynamic analysis which practically means numerical solving of the nonlinear set of ODEs. A lot of numerical solution methods have been developed, especially for the ODE, such as Euler's method or Taylor's method [4]. Runge-Kutta's methods are very popular because of their simplicity and easy programmability [5].

Although there could be used several modern control methods [6], [7], the adaptive approach is used here. The basic idea of adaptive control is that parameters or the structure of the controller are adapted to parameters of the controlled plant according to the selected criterion [8]. The adaptive approach in this work is based on choosing an *external linear model* (ELM) of the original nonlinear system whose parameters are recursively identified during the control. Parameters of the resulted continuous controller are recomputed in every step from the estimated parameters of the ELM.

The polynomial method introduced by Kucera [9] used for designing of the controller here together with the pole-placement method ensures basic control requirements such as stability, reference signal tracking and disturbance attenuation. The basic control system configurations with one degree-of-freedom (1DOF) and two degrees-of-freedom have been used. The proposed controller is hybrid because polynomial synthesis is made for continuous-time but recursive identification runs on the δ-model, which belongs to the class of discrete-time models.

The MATLAB (MATrix LABoratory) [10] is mathematical software often used for computation and simulation [11], [12]. Although this software has it own programming language, it also provides the tool for creating window-like

Manuscript received August 9, 2011.

J. Vojtesek and P. Dostal are with Department of Process Controll, Faculty of Applied Informatics, Tomas Bata University in Zlin, Czech Republic (phone: +420576035199; fax: +420576032716; e-mail: {vojtesek,dostalp}@ fai.utb.cz).

programs which can be used for testing and simulation without changing of the program and knowledge about this programming language. This tool called GUIDE was used here for creating of the simulation program as a practical output of this contribution.
The main goal is to provide MATLA program which provides basic simulation of one nonlinear system. Created program is available for free. People, who are interested, please contact author on his email address – *vojtesek@fai.utb.cz*.

II. Model of the Plant

The proposed control strategy was tested on the mathematical model of Continuous Stirred Tank Reactor (CSTR). This model has simple exothermic reaction inside the tank which is cooled via cooling coil – see Fig. 1.
Mathematically speaking, this plant is represented by the mathematical model which describes all quantities is of course very complex and we need to introduce some simplifications. First, we expect that reactant is perfectly mixed. Then, we also assume that volume, heat capacities and densities do not change rapidly during the control.

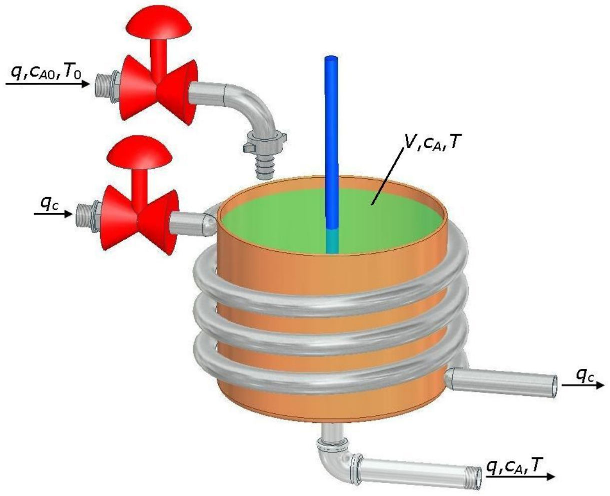


Fig. 1 The schematic representation of CSTR

These assumptions results in the mathematical model represented by the set of two Ordinary Differential Equations (ODE) [12] which are derived from the material and heat balances of the reactant and cooling, i. e.

$$\frac{dT}{dt} = a_1 \cdot (T_0 - T) + a_2 \cdot k_1 \cdot c_A + a_3 \cdot q_c \cdot \left(1 - e^{\frac{a_4}{q_c}}\right) \cdot (T_{c0} - T) \qquad (1)$$

$$\frac{dc_A}{dt} = a_1 \cdot (c_{A0} - c_A) - k_1 \cdot c_A$$

where $a_{1\text{-}4}$ are constants computed as

$$a_1 = \frac{q}{V};\ a_2 = \frac{-\Delta H}{\rho \cdot c_p};\ a_3 = \frac{\rho_c \cdot c_{pc}}{\rho \cdot c_p \cdot V};\ a_4 = \frac{-h_a}{\rho_c \cdot c_{pc}} \qquad (2)$$

The fixed values of the system are shown in Table 1.
The nonlinearity of the model is hidden mainly in the computation of the reaction rate, k_1, which is nonlinear function of the temperature, T, and it is computed from Arrhenius law:

$$k_1 = k_0 \cdot \mathrm{e}^{\frac{-E}{R \cdot T}} \qquad (3)$$

TABLE 1.
FIXED PARAMETERS OF THE REACTOR

Quantity	*Symbol and value*
Reactor's volume	$V = 100\ l$
Reaction rate constant	$k_0 = 7.2 \cdot 10^{10}\ min^{-1}$
Activation energy to R	$E/R = 1 \cdot 10^4\ K$
Reactant's feed temperature	$T_0 = 350\ K$
Inlet coolant temperature	$T_{c0} = 350\ K$
Reaction heat	$\Delta H = -2 \cdot 10^5\ cal.mol^{-1}$
Specific heat of the reactant	$c_p = 1\ cal.g^{-1}.K^{-1}$
Specific heat of the cooling	$c_{pc} = 1\ cal.g^{-1}.K^{-1}$
Density of the reactant	$\rho = 1 \cdot 10^3\ g.l^{-1}$
Density of the cooling	$\rho_c = 1 \cdot 10^3\ g.l^{-1}$
Feed concentration	$c_{A0} = 1\ mol.l^{-1}$
Heat transfer coefficient	$h_a = 7 \cdot 10^5\ cal.min^{-1}.K^{-1}$

III. Steady-state and Dynamic Analyses

The pre-control simulation often includes steady state and dynamic analyses which help us with the understanding, how system works in different states and behaves after various changes on the input.

A. Steady-state Analysis

The steady-state analysis shows behaviour of the system in the steady-state, i.e. in $t \rightarrow \infty$ and results in optimal working point in the sense of maximal effectiveness and concentration yield. Mathematical meaning of the steady-state is that derivatives with respect to time variable are equal to zero, $d(\cdot)/dt = 0$.
The previous studies [14] have shown interesting steady-state feature of this reactor. It is clear, that the reactant and cooling heat must be equal in the steady-state, i.e. $Q_r = Q_c$, which means that the equation (1) in steady-state is rewritten to:

$$a_1 \cdot (T_0 - T) + a_2 \cdot k_1 \cdot c_A + a_3 \cdot q_c \cdot \left(1 - e^{\frac{a_4}{q_c}}\right) \cdot (T_{c0} - T) = 0 \qquad (4)$$

and results in relations for these heats Q_r and Q_c:

$$Q_r = a_2 \cdot k_1 \cdot c_A^s$$
$$Q_c = \left[a_1 + a_3 \cdot q_c \cdot \left(1 - e^{\frac{a_4}{q_c}}\right)\right] \cdot T^s - \left[a_1 \cdot T_0 + a_3 \cdot q_c \cdot \left(1 - e^{\frac{a_4}{q_c}}\right)\right] \qquad (5)$$

If we compute Q_r and Q_c for various values of the temperature T = <300, 500> K for working point q = 100 $l.min^{-1}$ and q_c = 80 $l.min^{-1}$, we obtain three steady-states – see Fig. 2.
As you can clearly see, this system has two stable steady-states (S_1 and S_2) and one unstable steady state (N_1). The steady-state values of the state variables in these points are:

$$\begin{aligned} S_1: &\quad T^s = 354.23\ K \quad c_A^s = 0.9620\ mol.l^{-1} \\ N_1: &\quad T^s = 392.45\ K \quad c_A^s = 0.6180\ mol.l^{-1} \\ S_2: &\quad T^s = 456.25\ K \quad c_A^s = 0.0439\ mol.l^{-1} \end{aligned} \qquad (6)$$

It is clear, that the second operating point S_2 has better efficiency (95.6 % reacts) for the same input settings than on the point S_1 (3.8 % reacts).

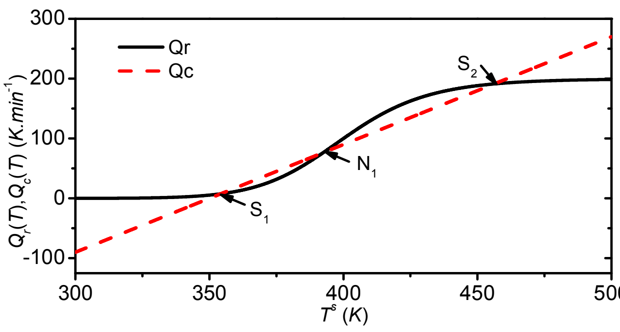


Fig. 2 Heat balance inside the reactor

So the steady-state model is finally described by the set of nonlinear functions

$$T^s = \frac{a_1 \cdot T_0 + a_2 \cdot k_1 \cdot c_A^s + a_3 \cdot q_c \cdot T_0 \cdot \left(1 - e^{\frac{a_4}{q_c}}\right)}{a_1 + a_3 \cdot q_c \cdot \left(1 - e^{\frac{a_4}{q_c}}\right)} \tag{7}$$

$$c_A^s = \frac{a_1 \cdot c_{A0}}{a_1 + k_1}$$

which are easily solved numerically by the simple iteration method.

B. Dynamic Analysis

This analysis means that we observe course of the state variables in time after the step change of some input variable. The step changes of volumetric flow rates q and q_c are input variables in our case and the steady-state values in Equation (6) are initial conditions for the set of ODE (1). The Runge-Kutta's fourth order method was used for numerical solving of the set of ODE.

IV. Adaptive Control

There could be used several so called "modern" techniques to control of this process such as robust control, predictive control, fuzzy control etc. In our case, the adaptive control was used mainly because of its strong theoretical background and usability for such kind of processes.

The Adaptive control is based on the quality of real organisms which can change behavior according to environmental conditions. This process is usually called "*adaptation*". There are several ways of use of the adaptation. It can be done for example by the modification of the controller's parameters by the change of the controller's structure or by generating an appropriate input signal, which is called "adaptation by the input signal".

The adaptive approach in this work is based on choosing an *external linear model* (ELM) of the original nonlinear system whose parameters are recursively identified during the control. Parameters of the resulted continuous controller are recomputed in every step from the estimated parameters of the ELM. The advantage of this method is that we do not care about the system nonlinearity. First we do the dynamic analysis that shows us the dynamic behavior of the output variable which is then used for the choice of the ELM which describes the output in the most accurate way. The possible change of the ELM parameters is taken into account by the recursive identification of ELM during the control.

A. External Linear Model (ELM)

The ELM could be generally described by the transfer function:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b(s)}{a(s)} \tag{8}$$

The presence of the variable s in the Equation (8) indicates continuous-time (CT) model. The online identification of such processes which is necessary in this case is not very easy.

One way, how we can overcome this problem is the use of so called δ–model. These special types of models formally belong to discrete models but it was proofed for example in [15] that their parameters are close to the continuous ones for very small sampling period.

The δ–model introduces a new complex variable γ ([16]):

$$\gamma = \frac{z-1}{\beta \cdot T_v \cdot z + (1-\beta) \cdot T_v} \tag{9}$$

Where β is a parameter from the interval $0 \le \beta \le 1$ and T_v means a sampling period. It is clear that we can obtain infinite number of δ-models for various β. A so called *forward δ-model* for β = 0 was used and γ operator is then

$$\gamma = \frac{z-1}{T_v} \tag{10}$$

The continuous model (8) is then rewritten to the form

$$a^\delta(\delta) y(t') = b^\delta(\delta) u(t') \tag{11}$$

where polynomials $a^\delta(\delta)$ and $b^\delta(\delta)$ are discrete polynomials and their coefficients are different, but for the small sampling period very close to those of the CT model $a(s)$ and $b(s)$.

These parameters identified recursively, which means that they are computed by the Recursive Least Squares (RLS) method from differential equation

$$y_\delta(k) = \boldsymbol{\theta}_\delta^T(k) \cdot \boldsymbol{\varphi}_\delta(k-1) + e(k) \tag{12}$$

Where $\boldsymbol{\phi}_\delta$ stands for known regression (data) vector, $\boldsymbol{\theta}_\delta$ represents vector of parameters and $e(k)$ is a general random immeasurable component.

B. Recursive Identification

As it is written above, the well known and easily programmable Recursive Least-Squares (RLS) method is used for the on-line identification. This method is usually modified with some kind of forgetting; exponential or directional [17] mainly due to specific features of the identified system like nonlinearity etc.

The basic RLS method is described by the set of equations:

$$\begin{gathered}\varepsilon(k)=y(k)-\boldsymbol{\varphi}^T(k)\cdot\hat{\boldsymbol{\theta}}(k-1)\\ \gamma(k)=\left[1+\boldsymbol{\varphi}^T(k)\cdot\mathbf{P}(k-1)\cdot\boldsymbol{\varphi}(k)\right]^{-1}\\ \boldsymbol{L}(k)=\gamma(k)\cdot\mathbf{P}(k-1)\cdot\boldsymbol{\varphi}(k)\\ \mathbf{P}(k)=\frac{1}{\lambda_1(k-1)}\left[\mathbf{P}(k-1)-\frac{\mathbf{P}(k-1)\cdot\boldsymbol{\varphi}(k)\cdot\boldsymbol{\varphi}^T(k)\cdot\mathbf{P}(k-1)}{\lambda_1(k-1)+\boldsymbol{\varphi}^T(k)\cdot\mathbf{P}(k-1)\cdot\boldsymbol{\varphi}(k)}\right]\\ \hat{\boldsymbol{\theta}}(k)=\hat{\boldsymbol{\theta}}(k-1)+\boldsymbol{L}(k)\varepsilon(k)\end{gathered}\tag{13}$$

Where the "forgetting" could be affected by the choice of the forgetting factor λ_1.

The identification methods used in the program are:

Without forgetting, e.g. no forgetting factor is inserted.

Constant exponential forgetting is for $\lambda_1 < 1$ and $\lambda_2 = 1$. The values of forgetting factor λ_1 are from the range <0.95; 0.99>. Parameter λ_1 influences gradual forgetting of the old values and the most weight is put on the last values. This relation can be described by criterion

$$J=\sum_{i=1}^{k}\lambda^{k-i}\varepsilon_i^2 \tag{14}$$

This algorithm can be used for systems with changing parameters.

Increasing exponential forgetting has forgetting parameters $\lambda_2 = 1$ and λ_1 is computed from

$$\lambda_1(k)=\lambda_0\lambda_1(k-1)+1-\lambda_0 \tag{15}$$

Typical values of the forgetting parameters are $\lambda_1(0)=\lambda_0\in\langle 0.95,0.99\rangle$. The value of this forgetting factor is asymptotically approaching to 1, which means that the old data is forgotten.

Changing exponential forgetting has again the value of forgetting parameter $\lambda_2 = 1$ and exponential forgetting λ_1 is recomputed in every step as

$$\lambda_1(k)=1-K\cdot\gamma(k)\cdot\varepsilon^2(k) \tag{16}$$

where K is a very small value (e.g. 0.001).

Directional forgetting. This algorithm forgets information only in the direction from which it comes. General description of this method can be formulated by the following equations:

$$r(k-1)=\boldsymbol{\varphi}^T(k)\cdot\mathbf{P}(k-1)\cdot\boldsymbol{\varphi}(k)$$

$$\boldsymbol{L}(k)=\frac{\mathbf{P}(k-1)\cdot\boldsymbol{\varphi}(k)}{1+r(k-1)}$$

$$\beta(k-1)=\begin{cases}\lambda_1(k-1)-\dfrac{1-\lambda_1(k-1)}{r(k-1)} & \text{for } r(k-1)>0\\ 1 & \text{for } r(k-1)=0\end{cases} \tag{17}$$

$$\mathbf{P}(k)=\mathbf{P}(k-1)-\frac{\mathbf{P}(k-1)\cdot\boldsymbol{\varphi}(k)\cdot\boldsymbol{\varphi}^T(k)\cdot\mathbf{P}(k-1)}{\beta^{-1}(k-1)+r(k-1)}$$

where λ_1 can be chosen similarly as in exponential forgetting.

-

C. Control System Synthesis

The control system configuration with one degree-of-freedom (1DOF) was used this work – see Fig. 3.

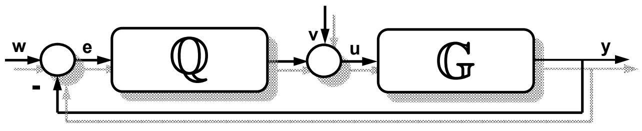


Fig. 3 1DOF control configuration

G in Fig. 3 denotes transfer function (8) of controlled plant, w is the reference signal (wanted value), v is disturbance, e is used for control error, u is control variable and y is a controlled output.

The feedback part of the controller are designed with the use of polynomial synthesis:

$$Q(s)=\frac{q(s)}{s\cdot p(s)} \tag{18}$$

where parameters of the polynomials $p(s)$ and $q(s)$ are computed by the Method of uncertain coefficients which compares coefficients of individual s-powers from Diophantine equation [9]:

$$a(s)\cdot s\cdot p(s)+b(s)\cdot q(s)=d(s) \tag{19}$$

The resulted, so called "hybrid", controller works in the continuous time but parameters of the polynomials $a(s)$ and $b(s)$ are identified recursively in the sampling period T_v.

The feedback controller $Q(s)$ ensures stability, load disturbance attenuation for both configurations and asymptotic tracking.

The polynomial $d(s)$ on the right side of (19) is an optional stable polynomial. Roots of this polynomial are called poles of the closed-loop and their position affects quality of the control.

This polynomial could be designed for example with the use of Pole-placement method. The degree of the polynomial $d(s)$ is in this case

$$\deg d(s)=\deg a(s)+\deg\tilde{p}(s)+1 \tag{20}$$

A choice of the roots needs some a priory information about the system's behaviour. It is good to connect poles with the parameters of the system via spectral factorization. The polynomial $d(s)$ then for our ELM (8) can be rewritten for aperiodical processes to the form

$$d(s)=n(s)\cdot(s+\alpha)^{\deg d-\deg n} \tag{21}$$

where $\alpha > 0$ is an optional coefficient reflecting closed-loop poles and stable polynomial $n(s)$ is obtained from the spectral factorization of the polynomial $a(s)$

$$n^*(s)\cdot n(s)=a^*(s)\cdot a(s) \tag{22}$$

V. Used Numerical Methods

The main numerical methods used in the program are the Simple iteration method in the steady-state analysis and the Standard Runge-Kutta's method used for the solving of ordinary differential equations in the dynamic analysis and also in the hybrid adaptive control where the action value (i.e. output from the controller) is computed also from the ordinary differential equation for the continuous-time.

A. *Simple Iteration Method*

Consider nonlinear system in the form of state equation $\dot{\boldsymbol{x}}(t)=\boldsymbol{f}\left[\boldsymbol{x}(t),\boldsymbol{u}(t)\right]$ with initial condition $\boldsymbol{x}(0)=\boldsymbol{x}^s$ and the vector of state variables is $\boldsymbol{x}^s=\left[x_1^s,x_2^s,\dots x_n^s\right]^T$.

Components of this vector are unknown and they could be computed by solving of equations of the model in steady-state:

$$\boldsymbol{f}=\left(\boldsymbol{x}^s,\boldsymbol{u}^s\right)=0 \tag{23}$$

where $\boldsymbol{u}^s=[u_1^s,\ u_2^s,\dots\ u_m^s]^T$ is vector of assigned (known) input variables which comes from basic steady-state. Unknown variables in the equation (23) are components of the state vector $\boldsymbol{x}^s$, which creates n state variables in basic steady-state. Computation of the initial conditions $\boldsymbol{x}^s$ is for nonlinear system important not only for computation of the dynamics but also it is used for creating of a linearized mathematical model of the process in working point. It is known that parameters of this model depend on values of state variables in the working point.

Equation (23) can be now rewritten to

$$\boldsymbol{f}(\boldsymbol{x})=0 \tag{24}$$

for unknown values of x_i for $i = 1,2,\dots n$.

Next step in the solution is following. The equivalent set of equations to the set (24) is

$$\boldsymbol{x}=\boldsymbol{\varphi}(\boldsymbol{x}) \tag{25}$$

where $\boldsymbol{\varphi}$ is nonlinear vector function $\boldsymbol{\varphi}=[\varphi_1,\ \varphi_2,\dots\ \varphi_n]^T$ and which leads to iterative equation in the form of

$$\boldsymbol{x}^{k+1}=\boldsymbol{\varphi}\left(\boldsymbol{x}^k\right)\ \text{for}\ k=0,1,\dots \tag{26}$$

The iterative method leads to the exact solution only if it converges. *The convergence condition* of the iterative process (26) then can be formulated as:

Let the vector function φ is defined in the closed convex region $D\subset\ \Re^n$ and if $\boldsymbol{x}\in D$ so $\boldsymbol{\varphi}\in D$ too. Moreover, let functions φ has continuous partial differential derivations of all variables $x_1\div x_n$ in the region D, then there exists matrix

$$\boldsymbol{\varphi}'(\boldsymbol{x})=\frac{d\boldsymbol{\varphi}}{d\boldsymbol{x}}=\begin{pmatrix}\frac{d\varphi_1}{dx_1} & \frac{d\varphi_1}{dx_2} & \cdots & \frac{d\varphi_1}{dx_n}\\ \frac{d\varphi_2}{dx_1} & \frac{d\varphi_2}{dx_2} & \cdots & \frac{d\varphi_2}{dx_n}\\ \vdots & & \ddots & \\ \frac{d\varphi_n}{dx_1} & \frac{d\varphi_n}{dx_2} & \cdots & \frac{d\varphi_n}{dx_n}\end{pmatrix} \tag{27}$$

If matrix (27) complete condition $\left\|\boldsymbol{\varphi}'(\boldsymbol{x})\right\|<1$ for any $\boldsymbol{x}\in D$, there are only one solution $\boldsymbol{x}^*\in D$ of the equation (26).

There could be of course thousands of iterations during the computation but from practical point of view is convenient to stop the computation in the case that difference between values of actual and previous iteration is sufficiently small, i.e. condition

$$\left\|\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k-1)}\right\|<\varepsilon_{ss} \tag{28}$$

is fulfilled for accuracy $\varepsilon_{ss}>0$, value of which depends on supposed absolute dimension of computed variables.

B. *Standard Runge-Kutta's Method*

The second main group of the numerical solving is solution of the ordinary differential equations (ODE) which can be found mainly in the dynamic analysis as the numerical solution of the set of ODE (1) but there are used in the adaptive control as it is mentioned above too.

We supposed the general differential equation be in the form of

$$\boldsymbol{y}(t)=\boldsymbol{g}\left[\boldsymbol{x}(t),\boldsymbol{u}(t)\right] \tag{29}$$

with the initial condition

$$y(t_0)=y_0 \tag{30}$$

There are a lot of methods which can be used for numerical solving of this problem. General division is into the two main groups – one-step and multi-step methods.

The popular Runge-Kutta's standard method was used in this work. This method is very often used because of its simplicity. Runge-Kutta's methods belong to the class of high-order methods, they can be used for computation of the initial values or for the final result and they are easily programmable. The (standard) fourth-order Runge Kutta's method uses first four parts of the Taylor's series:

$$y(k+1)=y(k)+\frac{1}{6}\cdot\left(g_1+2g_2+2g_3+g_4\right) \tag{31}$$

where coefficients $g_{1\text{-}4}$ are computed from:

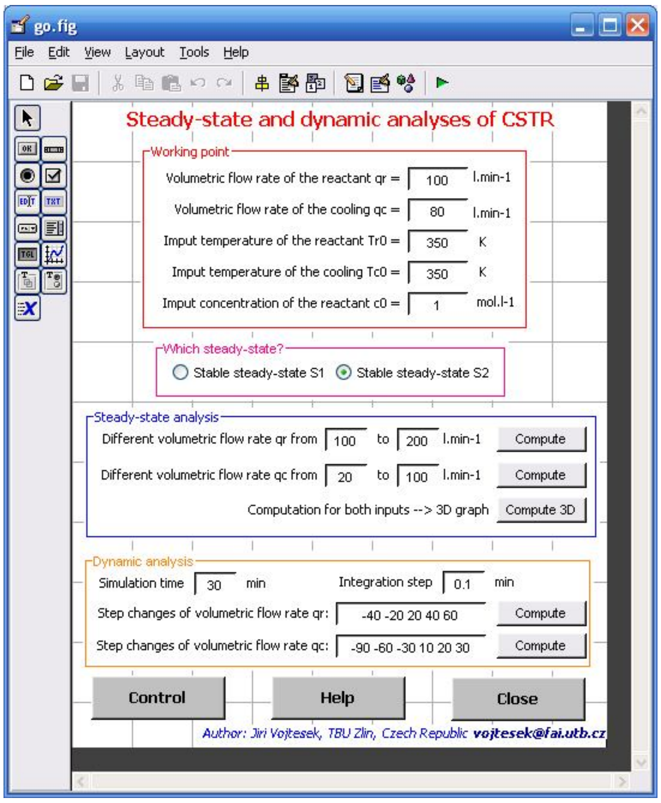
$$\begin{aligned}g_1&=h_i\cdot f\left(x_n,y(x_n)\right)\\ g_2&=h_i\cdot f\left(x_n+\frac{h_i}{2},y(x_n)+\frac{g_1}{2}\right)\\ g_3&=h_i\cdot f\left(x_n+\frac{h_i}{2},y(x_n)+\frac{g_2}{2}\right)\\ g_4&=h_i\cdot f\left(x_n+h_i,y(x_n)+g_3\right)\end{aligned} \tag{32}$$

The Runge-Kutta's methods are in some cases build-in functions in mathematical softwares. For example in MATLAB, which is used for simulation in this work, are Runge-Kutta's methods in functions `ode23` (the second order Runge-Kutta formula) or `ode45` (the fourth order Runge-Kutta's formula described above). One of advantages of these methods is that they have flexible integration step h_i, which recomputes every step according to the actual computation error. The standart Runge-Kutta method has a several modifications like *Runge-Kutta-Fehlberk* method, *Runge-Kutta-Nyström* method etc.

VI. Simulation Program

The simulation program which deals with the simulation of the steady-state, dynamics and of course adaptive control of the CSTR was made in mathematical software MATLAB (MATrix LABoratory), version 7.0.1 from Mathworks [10]

using Graphical User Interface (GUI). The use of this tools enable programmer to make program user-friendly and close to the users who do not know or do not like programming. They can use all features of Matlab as a simulation tool by just changing of the most important variables and pressing buttons for computing.
The MATLAB has special tool for creating of programs with GUI. This tool is called simply by typing of the command **`guide`** in the Command window and the sample window is shown in Fig. 4.



Steady-state and dynamic analyses of CSTR

Working point
Volumetric flow rate of the reactant qr = 100 l.min-1
Volumetric flow rate of the cooling qc = 80 l.min-1
Imput temperature of the reactant Tr0 = 350 K
Imput temperature of the cooling Tc0 = 350 K
Imput concentration of the reactant c0 = 1 mol.l-1

Which steady-state?
Stable steady-state S1 [✓ Stable steady-state S2]

Steady-state analysis
Different volumetric flow rate qr from 100 to 200 l.min-1 Compute
Different volumetric flow rate qc from 20 to 100 l.min-1 Compute
Computation for both inputs --> 3D graph Compute 3D

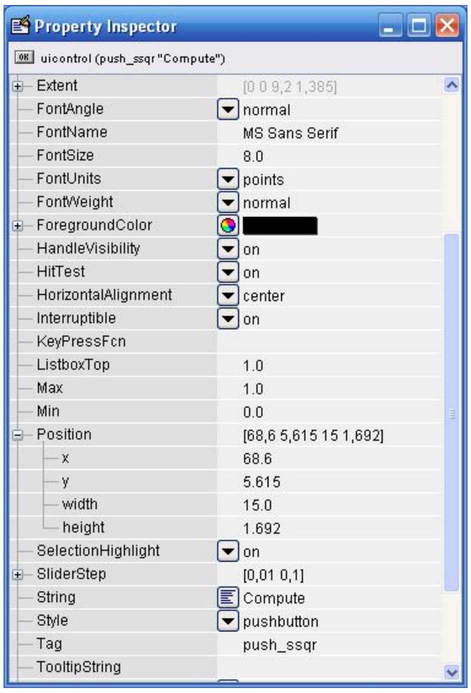
Dynamic analysis
Simulation time 30 min Integration step 0.1 min
Step changes of volumetric flow rate qr: -40 -20 20 40 60 Compute
Step changes of volumetric flow rate qc: -90 -60 -30 10 20 30 Compute

Control Help Close

Author: Jiri Vojtesek, TBU Zlin, Czech Republic vojtesek@fai.utb.cz

Fig. 4 The MATLAB's tool GUIDE

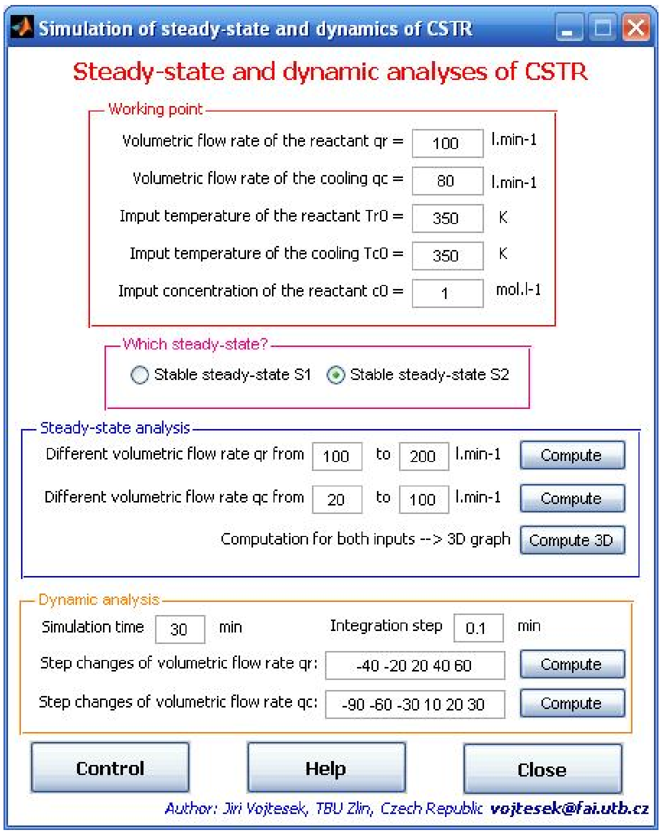
The usage of this tool is very simple and understandable. It has two main parts – I. workspace for program's sketch and II. toolbar on the left side which provides all common objects used in the program design like text and edit boxes, buttons, radio buttons, list boxes etc.
Each object could be of course edited via property inspector where you can edit color, font, size, position etc. of the object.
The output from the GUIDE are two files – e.g. *sample.fig* where the sketch of the program and all object is saved and *sample.m* where are defined actions to individual objects mainly procedures to the buttons.
The program can be start by the typing the command **`go`** in the program's directory. It is divided into two main windows mainly because of the space. The first window (Fig. 6) involves simulation of the steady-state and dynamics of the system. The user can set the working point of the reactor which is defined by volumetric flow rates of the reactant and the cooling, q_r and q_c, input temperatures of the reactant and the cooling, T_{r0} and T_{c0}, and input concentration of the reactant, c_0. The next part gives user choice between two stable states S1 or S2 which closely described in chapter III.



uicontrol (push_ssqr "Compute")

Extent	[0 0 9,2 1,385]
FontAngle	normal
FontName	MS Sans Serif
FontSize	8.0
FontUnits	points
FontWeight	normal
ForegroundColor	■
HandleVisibility	on
HitTest	on
HorizontalAlignment	center
Interruptible	on
KeyPressFcn	
ListboxTop	1.0
Max	1.0
Min	0.0
Position	[68,6 5,615 15 1,692]
x	68.6
y	5.615
width	15.0
height	1.692
SelectionHighlight	on
SliderStep	[0,01 0,1]
String	Compute
Style	pushbutton
Tag	push_ssqr
TooltipString	

Fig. 5 Property inspector in GUIDE



Steady-state and dynamic analyses of CSTR

Working point
Volumetric flow rate of the reactant qr = 100 l.min-1
Volumetric flow rate of the cooling qc = 80 l.min-1
Imput temperature of the reactant Tr0 = 350 K
Imput temperature of the cooling Tc0 = 350 K
Imput concentration of the reactant c0 = 1 mol.l-1

Which steady-state?
Stable steady-state S1 [✓ Stable steady-state S2]

Steady-state analysis
Different volumetric flow rate qr from 100 to 200 l.min-1 Compute
Different volumetric flow rate qc from 20 to 100 l.min-1 Compute
Computation for both inputs --> 3D graph Compute 3D

Dynamic analysis
Simulation time 30 min Integration step 0.1 min
Step changes of volumetric flow rate qr: -40 -20 20 40 60 Compute
Step changes of volumetric flow rate qc: -90 -60 -30 10 20 30 Compute

Control Help Close

Author: Jiri Vojtesek, TBU Zlin, Czech Republic vojtesek@fai.utb.cz

Fig. 6 GUI for the simulation of the steady-state and dynamics of CSTR

The third part is dedicated to the steady-state analysis where two analyses could be done – the steady-state analysis for different volumetric flow rate of the reactant q_r and different volumetric flow rate of the coolant q_c where the starting and end values could be set in the edit boxes. The steady-state analysis for both input variables together is represented by the push-button "Compute 3D" and results in 3D graphs. The sample results of the steady-state analysis are shown in Fig. 7.

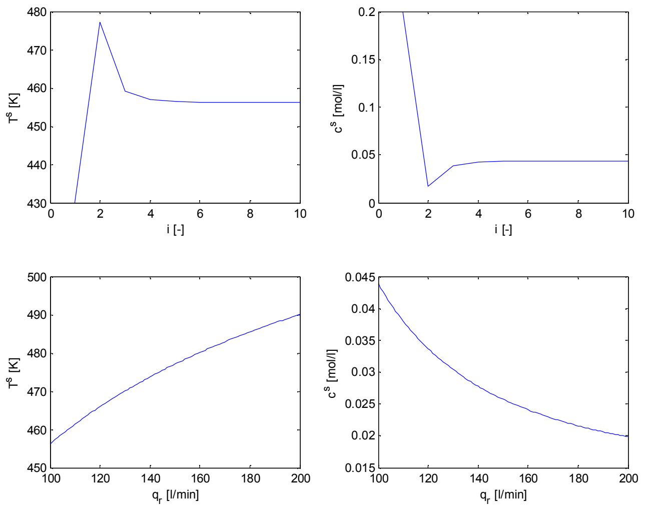


Fig. 7 Sample results of the steady-state analysis

The last part is focused on the dynamic analysis which could be done for step changes of both input variables q_r and q_c. Both dynamic analyses can be done for more step changes (e.g. six step changes -60, -40, -20, 20, 40 and 60 as it can be seen in Fig. 6). The simulation time and the integration step in Runge-Kutta's method could be set via appropriate edit boxes. Again, the sample results of the dynamic analysis are shown in Fig. 8.

The buttons in the bottom of the window used for opening the next window for control (push-button "Control"), displaying the help to the program (push-button "Help") and closing of this window and all graphs (push-button "Close").

The second sub-program called by the pressing of the push-button "Control" or by the command `go_control` from the Matlab's command window deals with the simulation of the adaptive control. The window is displayed in Fig. 9.

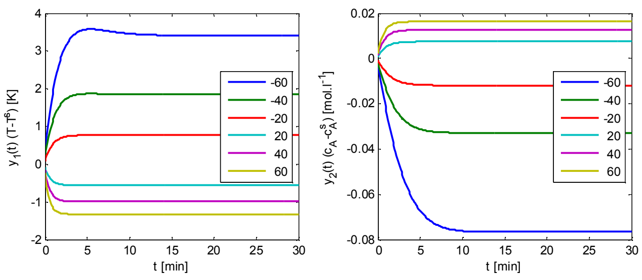
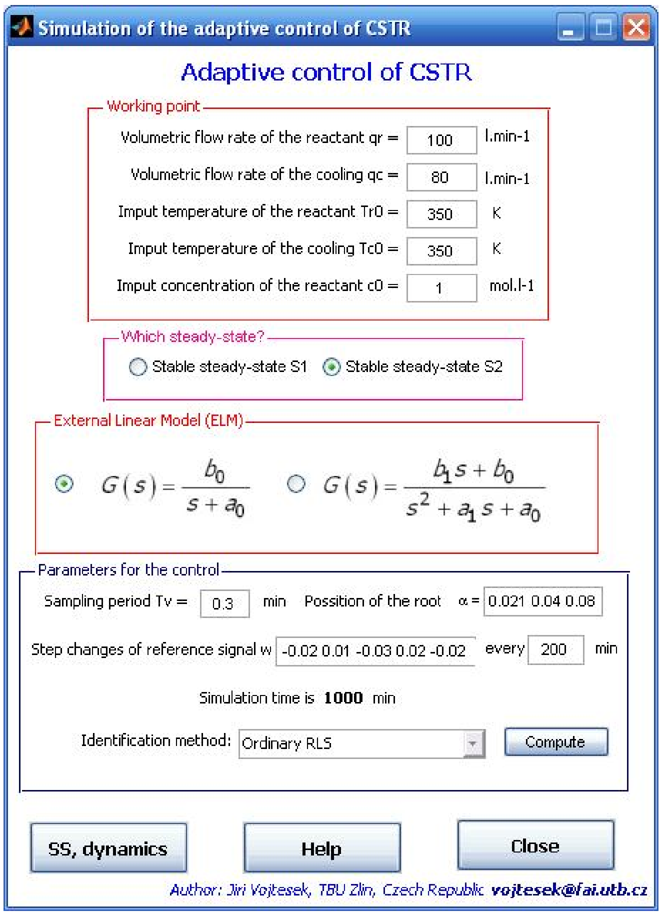


Fig. 8 Sample results of the dynamic analysis

The first two parts related to the working point and the choice of the steady-state are the same as in previous case. The new part here is the choice of the external linear model and settings for the control where the user could set the position of the root α, sampling period T_v, definition of step changes of the reference signal $w(t)$ which represents wanted value and time when they occur.


Simulation of the adaptive control of CSTR

Adaptive control of CSTR

Working point

Volumetric flow rate of the reactant qr =	100	l.min-1
Volumetric flow rate of the cooling qc =	80	l.min-1
Imput temperature of the reactant Tr0 =	350	K
Imput temperature of the cooling Tc0 =	350	K
Imput concentration of the reactant c0 =	1	mol.l-1

Which steady-state?

○ Stable steady-state S1 [✓ Stable steady-state S2]

External Linear Model (ELM)

[✓ $G(s)=\dfrac{b_0}{s+a_0}$] ○ $G(s)=\dfrac{b_1 s+b_0}{s^2+a_1 s+a_0}$

Parameters for the control

Sampling period Tv = 0.3 min Position of the root α = 0.021 0.04 0.08

Step changes of reference signal w -0.02 0.01 -0.03 0.02 -0.02 every 200 min

Simulation time is **1000** min

Identification method: Ordinary RLS Compute

SS, dynamics Help Close

Author: Jiri Vojtesek, TBU Zlin, Czech Republic **vojtesek@fai.utb.cz**

Fig. 9 GUI for the simulation of the adaptive control of CSTR

The final simulation time is recomputed according to the number of steps and time for each step. The last part in this sub-window is dedicated to the choice of recursive methods (see Fig. 10) for identification.

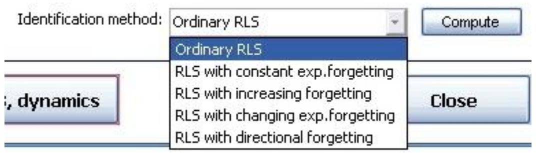

Identification method:
Ordinary RLS
Compute
Ordinary RLS
RLS with constant exp.forgetting
RLS with increasing forgetting
RLS with changing exp.forgetting
RLS with directional forgetting
, dynamics
Close

Fig. 10 The choice of the identification method

The buttons below has the same functions as in previous case except the first push-button on the left which will call the program for simulating of the steady-state and dynamics in this case. Simulation results of the adaptive control can be seen in Fig. 11.

As a result of the simulation, program shows the final value which indicates what computation was done and in which

MAT-file are data saved in. The name of this file differs with the computation – see Fig. 12.

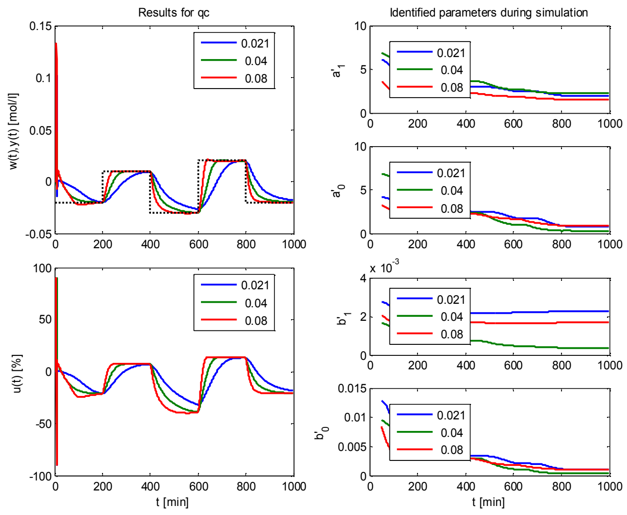
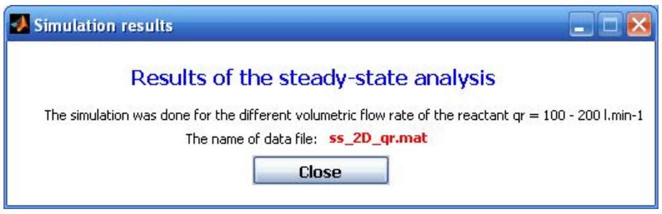


Fig. 11 Sample results of the adaptive control


Simulation results

Results of the steady-state analysis

The simulation was done for the different volumetric flow rate of the reactant qr = 100 - 200 l.min-1

The name of data file: ss_2D_qr.mat

Close

Fig. 12 GUI with the results of the computation

VII. Conclusion

This contribution shows procedure which is connected with the modeling and simulation of the system's behaviour that usually precede the design of the controller. The steady-state and dynamic analyses uncover nonlinearity of the examined continuous stirred tank reactor. The designed adaptive controller is based on the recursive identification of the external linear model as a linear representation of the originally nonlinear system. This controller could be tuned via the choice of the parameter α where increasing value of this parameter results in quicker output response but more shaking course of the input variable. The main goal of this contribution is to show usability of the mathematical software MATLAB for creating simulation programs which could help users to investigate the behaviour of the nonlinear system represented by the CSTR. The resulting program has two main windows – the first provides the simulation of the steady-state and dynamics of the system for different values of input quantities. Results are displayed in the separate figures and the data were also saved in the MAT-files. The second program deals with adaptive control of this system and user can again set different input variables and choose various computations. The benefit of this program can be found in the GUI which provides changing of the most important values by the edit windows instead of the change of the program code. The program is available also for free at the author email.

References

[1] P. Dostal, V. Bobal, F. Gazdos, Simulation of nonlinear adaptive control of a continuous stirred tank reactor, *International Journal of Mathematics and Computers in Simulation* ,Volume 5, Issue 4, 2011, Pages 370-377

[2] J. Ingham, I. J. Dunn; E. Heinzle, J. E. Přenosil, *Chemical Engineering Dynamics. An Introduction to Modeling and Computer Simulation*. Second. Completely Revised Edition. VCH Verlagsgesellshaft. Weinheim, 2000

[3] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Society for Industrial & Applied, 2003

[4] F. L. Severance, *System Modeling and Simulation: An Introduction*. John Wiley & Sons 2001

[5] J. H. Mathews, K. K. Fink, *Numerical Methods Using Matlab*. Prentice-Hall 2004

[6] L. Pekar, R. Prokop, Stabilization of a delayed system by a proportional controller,*International Journal of Mathematical Models and Methods in Applied Sciences* 4 (4), 2010, pp. 282-290

[7] D. Samek, D. Manas, Artificial neural networks in artificial time series prediction benchmark, *International Journal of Mathematical Models and Methods in Applied Sciences* 5 (6), 2011, pp. 1085-1093

[8] V. Bobal, J. Böhm, J. Fessl, J. Machacek, *Digital Self-tuning Controllers: Algorithms. Implementation and Applications*. Advanced Textbooks in Control and Signal Processing. Springer-Verlag London Limited, 2005.

[9] V. Kucera, Diophantine equations in control – A survey. *Automatica*. 29. 1361-1375, 1993

[10] *MathWorks - MATLAB and Simulink for Technical Computing,* official webpage. Available: http://www.mathworks.com (URL)

[11] Matusu R. A software tool for algebraic design of interval systems control, *International Journal of Computational Science and Engineering* 5 (3-4), 2010, pp. 262-268

[12] Brancik, L., Sevcik, B. Time-domain simulation of nonuniform multiconductor transmission lines in Matlab, *International Journal of Mathematics and Computers in Simulation* 5 (2), 2011, pp. 77-84

[13] R. Gao, A. O'dywer, E. Coyle, A Non-linear PID Controller for CSTR Using Local Model Networks. Proc. of *4th World Congress on Intelligent Control and Automation*. Shanghai. P. R. China. 3278-3282, 2002.

[14] J. Vojtesek, P. Dostal, Effect of External Linear Model's Order on Adaptive Control of CSTR. In: *Proceedings of the IFAC workshop on Adaptation and Learning in Control and Signal Processing ALCOSP 2010*. Antalya, Turkey.

[15] D. L. Stericker, N. K. Sinha, Identification of continuous-time systems from samples of input-output data using the δ-operator. *Control-Theory and Advanced Technology*, vol. 9, 1993, 113-125

[16] S. Mukhopadhyay, A. G. Patra , G. P. Rao, New class of discrete-time models for continuos-time system. *International Journal of Control*, vol.55, 1992, 1161-1187

[17] G. P. Rao, H. Unbehauen, Identification of continuous-time systems. *IEEE Process-Control Theory Application.*, 152, 2005, 185-220

Jiri Vojtesek (Ph.D.) was born in Zlin, Czech Republic in 1979 and studied at the Tomas Bata University in Zlin. where he got his master degree in chemical and process engineering in 2002. He has finished his Ph.D. focused on Modern control methods for chemical reactors in 2007. He now works as a Senior Lecturer at Department of Process Control, Faculty of Applied Informatics, Tomas Bata University in Zlin.

Petr Dostal (prof.) studied at the Technical University of Pardubice. He obtained his PhD. degree in Technical Cybernetics in 1979 and he became professor in Process Control in 2000. His research interest are modeling and simulation of continuous-time chemical processes. polynomial methods. optimal. adaptive and robust control. He works as a professor and head of the Department of Process Control, Faculty of Applied Informatics, Tomas Bata University in Zlin.