COURSEWARE FOR CONTROL ENGINEERING EDUCATION

Jan John

Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Automatic Control

Abstract: The paper treats MATLAB and SIMULINK courseware for education of the first two subjects in control engineering curriculum. The courses are based on simple SISO control using classical controllers, including simple nonlinear elements. Description of basic functions and programs and several examples of their use comprise the main part of the paper. The software is freely available on the Czech Technical University web pages.

Keywords: Control education, Teaching, Computer software, Pole assignment, Nonlinear analysis, Nonlinear control

1. INTRODUCTION

The Department of Control Engineering of the Faculty of Electrical Engineering of the Czech Technical University in Prague is responsible for teaching the subjects of Control Engineering about 200 students of its own study branch. Moreover, there are several tenth of students of other study branches like Power Engineering, Electric Drives and Traction, Computer Engineering and others. Teaching Control Engineering subjects has on the Czech Technical University more than 100 years tradition, of course, under various names and at different University departments. In the last 20 years, the laboratories of the subjects were totally changed and equipped by the series of physical models of controlled plants like coupled water tanks, electric furnace, ball-and-beam apparatus, water turbine with long feeder, helicopter, inverse pendulum, etc. The laboratories are managed as open ones, that is, students have free access to them, however, for obtaining the laboratory credit the student must completely cope with two tasks of identification and control of such plants, supposing that one task is formulated as linear and the other as nonlinear control.

During these years of education emerged a very interesting fact: If the lectures begun with "modern" control theory, the students were not able to apply it for control of the said plants. To enhance the students' motivation it was decided to return in the first basic subjects to the "old" control theory, which is based on SISO access, frequency responses, simple methods of pole positioning etc. The state space access is discussed concurrently with these methods, but with less emphasis. After passing such organized basic subjects students are well prepared to continue studying the state space theory, algebraic methods, robust and fuzzy control. This access to the basic subjects of control engineering is now common in several other schools and very popular among autodidacts. In consequence, the subsidized textbooks published by the Department were always sold very rapidly out. These facts led the teachers to create an Internet textbook (John and Fuka, 2002) for the mentioned subjects, in the first stage only for the Faculty of Electrical Engineering of the Czech Technical University in Prague, and offer this text-book to the general public.

Both Czech and English version of the textbook is now under construction and the most up-and-coming part of it is the auxiliary educational software pre-sented here. The software is based on MATLAB and SIMULINK and a part of it has also perspective use in industrial control.

2. DESCRITION OF THE PROGRAMS

Only the most important programs are described here with presumptive goal to show the main ideas of the

software. More details and possibilities to copy the software can be found on http://dce.felk.cvut.cz/sari/.

2.1 System Identification

Function *parameters*=**minel**(T_s,*T*,*staircase*,*data*,*c*) is a multiple integration procedure (John and Rauch, 1999, John, 2001) for estimation of parameters of a system characterized by a chain of integrators

$$\begin{aligned} \dot{x}_1 &= f_1(u,y) + x_2 \\ \dot{x}_2 &= f_2(u,y) + x_3 \\ &\vdots \\ \dot{x}_{n-1} &= f_{n-1}(u,y) + x_n \\ \dot{x}_n &= f_n(u,y) \end{aligned} \qquad (1)$$

and an output function

$$y = g_y^{-1}[g_u + x_1; u] \qquad (2)$$

where y is a system (scalar) output and u is a system input (generally vector). There must exist an inverse function g_y of g_y^{-1}, so that

$$x_1 = -f_0(u,y) = g_u(u) - g_y(u,y) \qquad (3)$$

The (generally nonlinear) functions

$$f_i(u,y), (i = 0,1,\cdots,n) \qquad (4)$$

are known and linear in the estimated parameters. The procedure input variables are:

- T_s: data sampling period
- *T*: vector of integration periods. The integration periods affect filtration properties of the algorithm.
- *staircase*: (*m*) vector of data interpreting indicators.
  - For *staircase*(*i*) = 0 the data column *i* is integrated as continuous, by trapezoid method.
  - For *staircase*(*i*) = 1 the data column *i* is integrated as stepwise with valid right value of the sampling period.
  - For *staircase*(i) = -1 the data column *i* is integrated as stepwise with valid left value of the sampling period.
- *data*: (:,*m*) matrix of the (measured) data. The columns of this matrix contain the data which are measured and/or calculated from the measured ones. This data represent the outputs of the known (generally nonlinear) functions f_i. (The number of columns *m* must be equal in the matrices *staircase*, *data* and *c*.)
- *c*: (:,*m*) matrix of auxiliary integers. Every column of c characterizes the use of the corresponding column of the matrix *data*. The absolute value of a number in a given column of *c* corresponds to (1 + the number of integrators on the path between the corresponding function block and the system output *y*). Its sign corresponds to the supposed sign of the function contribution and its column position corresponds to the position of the coefficient in the resulting vector of coefficients. Tailing zeros (with no significance to the identified model structure) must be added to shorter columns of *c* to fill the matrix.
  The element *c*(1,1) corresponds obligatorily to the known unitary coefficient of the identified system (future right sides of the linear equation system).

The function displays relative error of the least squares procedure (used for parameters calculation) and singular decomposition of the corresponding matrix. These data help to estimate the error of the parameters and conditionallity of the least squares matrix.

Example:
Parallel dynamo in idle mode is characterized by

$$\dot{\Phi} = \frac{u - R \cdot i(\Phi)}{N}; i(\Phi) = a \cdot \Phi + b \cdot \Phi^3; u = \omega \cdot \Phi \quad , (5)$$

where Φ is machine flux, *i* is excitation current, *R* is excitation resistance, *u* is armature voltage, ω is rotor angular velocity and *N* is excitation constant. Measured variables are: *u*,*R*,ω. Putting machine flux Φ=*u*/ω as output variable and the combinations of the measured variables (i.e. *R* and ω) which appear in the describing equation as inputs results in
data=[*u*./ω, *u*, *R*.**u*./ω, *R*.*((*u*./ω).^3)];
c=[-1,2,-2,-2];
and supposing that all the measured variables are continuous also
staircase=[0,0,0,0].
Identified parameters will be 1/*N*, *a*/*N*, *b*/*N* .

Function *S*=**mult**(*u*,*y*,*m*,*n*,T_s,*staircase*,ω_c,*astat*) is a multiple integration procedure (John and Rauch, 1999, John, 2001) for estimation of parameters of a linear continuous SISO system *S*

$$S = \frac{b_{n-m}s^m + b_{n-m+1}s^{m-1} + \cdots + b_{n-1}s + b_n}{s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_{n-1}s + a_n} \quad , (6)$$

where $m \le n$ are orders of numerator and denominator, respectively. The state diagram of the system is characterized by a chain of integrators

$$\begin{aligned} \dot{x}_1 &= -a_1 \cdot y + x_2 + b_1 \cdot u \\ \dot{x}_2 &= -a_2 \cdot y + x_3 + b_2 \cdot u \\ &\vdots \\ \dot{x}_{n-1} &= -a_{n-1} \cdot y + x_n + b_{n-1} \cdot u \\ \dot{x}_n &= -a_n \cdot y + b_{n1} \cdot u \end{aligned} \qquad (7)$$

and an output function

$$y = x_1 + b_0 \cdot u \qquad (8)$$

where *y* is system output and *u* is system input.

The input variables of the function **mult** are:

- *u*: time vector of system input variables
- *y*: time vector of system output variables
- *m*: order of the transfer function numerator
- *n*: order of the transfer function denominator
- T_s: data sampling period
- *staircase*: vector of data interpreting indicators – see description of **minel** (default value [0,0]).
- ω_c: critical frequency (characterized by maximum filter gain). For frequencies higher then $1.5\omega_c$ the system input and output variables *u*, respectively *y* are attenuated (default value $\omega_c = \pi/(12 \cdot T_s)$, substitution $\omega_c = 0$).
- *astat*: order of system astatism (default value *astat* = 0)

2.2 Stability Criterions

Mansour (1988) published notable consecutive proof of classic stability criterions for characteristic polynomials of continuous transfer functions (Cremer - Leonard - Michailov, Hermite - Bieler, Routh, Hurwitz). Function **stab**(*polynomial*,...) is based on his work and displays the results of all the above mentioned criterions.

Function **cauchy**(*numerator, denominator, centre, radius*,...) demonstrates Cauchy's theorem graphically and can be used for illustration of the proof of Nyquist criterion.

Function **nqst**(*G*,...) offers five aspects of Nyquist criterion for the open loop transfer function *G*(*s*), namely

- Bode plot,
- linear Nyquist plot,
- logarithmic Nyquist plot (modules of the corresponding vectors in dB),
- classic Nichols plot (transposed: imaginary part on the horizontal axis, real part on the vertical one) and
- non-transposed Nichols plot (maintains the frequency response graph sides in accordance with the Nyquist plot).

Function **freq**(*F*,...) offers the same five aspects of frequency response for transfer function *F*(*s*) as **nqst**, however, without complex conjugated frequency plots and other complements of the Nyquist curve. The function can be used for frequency analysis.

2.3 Controllers and Control Circuits

Program **satx** is a demo program for simulation of a linear plant with a saturated controller. The saturation element

- is either surrounded by an auxiliary feedback loop with a transfer function K/R_{fb} which gives the controller a quasi-R_{fb} performance,
- or is connected in series with linear controller R_{dp}
- or the circuit is composed with both above functions.

The code is composed as an open program with several prepared experiments. To call one of them, a simple statement line

ex = <experiment number>; satx

can be used, however, the user can build very simply his own experiment by a slightly more complicated command line. The program calls one of two SIMULINK schemes, depending on above mentioned circumstances.

Programs **twox, trix** and **trindx** are demo programs for simulation of a linear plant with two- or three-level controller, respectively. They have similar structure and use as **satx**. The hysteresis of any of the elements is eligible, the dead zone of the three-level elements is ±1. The programs **twox** and **trix** have output signal ±1, **trindx** has freely eligible output levels.

2.4 System Synthesis by Frequency Methods

Functions **frpd**(*P*,Δφ,...) and **frpid**(*P*,Δφ,...) are intended for PD (PI) and PID controller adjustment for plant *P*, given the specified phase margin Δφ. The synthesis is followed by analysis of closed loop behaviour both in time and frequency region.

2.5 System Synthesis by Pole Placement

Function **pkpp**(*P*,*a*,...) is intended for classic PD, PI, PID, PIDD^2 etc. controllers adjustment for plant *P* and given relative damping *a*. The predominant poles of the closed loop transfer function are positioned in vertical line, which results in optimum time response of the system. The algorithm is based on comparison of unknown closed loop characteristic polynomial with its ideal form and is, of course, limited to only several forms of *P*. On the other hand, the function can be used as initial condition finder for more complicated cases.

Function [*M*,*N*,*T*] = **sri_pp**(B^+,B^-,*A*,*D*',*C*,l_i,*q*,*d*); calculates a controller by means of the general pole positioning method (Astrom, Wittenmark 1990) for a SISO plant $S = A/B$.
The function inputs are:

- B^+,B^-: plant numerator decomposition (B^+ monic polynomial which will be eliminated by closed-loop system poles, B^- appears in model numerator *D*).
- *A*: plant denominator
- *D*': model numerator $D = D' \cdot B^-$
- *C*: model numerator
- l_i: number of free integrators in the controller to obtain the desired steady-state precision of the control

- *q*: the function finds the observer order *k* automatically. The observer denominator is put to (*p*=*s* for continuous systems and *p*=*z* for the discrete ones):
- (*p*+*q*) for *k*=1
- (*p*^2+2*qp*+(2*q*)^2) for *k*=2
- (*p*^2+2*qp*+(2*q*)^2)·(*p*+*q*)^(*k*-2) for *k*>2
- For continuous system *q* could be elected as a relatively big positive number with respect to the model dynamics, for discrete systems usually *q*=0.
- *d* is a descriptive string. If this string begins by 'd' or by 'D', the system is treated as a discrete one, otherwise as a continuous one.

The function outputs are:

- *M*: direct path numerator of the controller
- *N*: feedback numerator of the controller
- *T*: common denominator of the controller

ppx is a MATLAB program for demonstration of the pole-placement procedure (sri_pp) for a SISO system. A typical pole-positioning experiment can be achived by writing a command line

ex=<experiment number>; ppx

To create user's own experiment, data must be written as a command line in the MATLAB command window:

ex=0; Bplus=...; Bminus=...; A=...; Dprime=...;
C=...; l=...; q=...; description='...'; ppx

(see help sri_pp for more information). The program displays the root locus diagram and calls one of two SIMULINK programs for control circuit simulation. The SIMULINK programs include possibility of limitation the controller output by changing the variable *lim*.

dcppx is a demo program for synthesis of the discrete controller for a continuous plant by the minimum steps number criterion and simulation of the circuit model. As by **ppx**, there is a possibility of limit the controller output. The code is composed as an open program with several prepared experiments. To call one of them, a simple statement line

ex = <experiment number>; dcppx

can be used, however, the user can build very simply his own experiment by a command line

ex=0; num=...;den=...;l=...;h=...;zeromax=...; dcppx

The input data are:

- *num*, *den*: plant transfer function numerator and denominator
- *l*: number of additive integrators (for obtaining desired steady state precision)
- *h*: sampling period
- *zeromax*: maximum permitted absolute value of the discretized plant transfer function zero to be eliminated by a closed loop pole

Function **sri_dcpp** is similar to **sri_pp** and calculates the discrete controller for control of continuous plant.

2.6 Stability of Nonlinear Systems and Synthesis of Nonlinear Control Circuits

Function **popov**(*G*,...) displays the modified frequency response of the system *G* for Popov's stability criterion.

Function [d_n,*d*]=**desc**(*F*,F_n,...) displays describing function (Kochenburger, 1960) d_n together with a linear frequency response *d* in in several ways.

The function input data are:

- *F* : linear system (is supposed with astatism order > 0 because of symmetry of the oscillations)
- F_n : nonlinear element, described by a vector [*type*, *hysteresis*], where
  - *type* = 1 or -1 for linear characteristic with unity gain and unity saturation
  - *type* = 2 or -2 for two-level element with unity amplitude and hysteresis
  - *type* = [3,*h*] or[-3,*h*] for three level element with unity amplitude, dead zone ±1 and hysteresis *h* (*h* optional, default 0).
  - *type* = 4 or -4 for two-level element with unity amplitude, without hysteresis.

For *type* < 0 the output tables will be displayed on the screen.

The following input parameters are optional:

- *stepsn* - number of steps of Fn; default: 12
- *steps* - number of steps of frequency for F.
- *omegamin*, *omegamax* - minimum and maximum circular frequency. Defaults of steps and frequencies given by the **bode** function of MATLAB.
- *Ainit* - initial value of the amplitude (default *Ainit* = max(1e-6,min(|*F*(*i*·ω)|)) for |*type*| = 4, otherwise *Ainit* = 1).

Outputs:

- d_n - describing function matrix
  [step No., A, phase(-1/F_n), 20*log|-1/F_n|, |-1/F_n|]
- *d* - linear frequency response matrix
  [step No., ω, phase(*F*), 20*log|*F*|, |*F*|]

If no outputs are invoked, the function plots both curves as linear Nyquist plot, logarithmic Nyquist plot and Nichols plot.

3. EXAMPLE

A drying oven is heated by an electrical heating spiral and cooled by a ventilator. Both inputs are planned as on – off.

3.1 Plant identification

Experiment was planned as several on-off impulses of heating and after being heated the plant was cooled by similar impulses of ventilation. Typical heating data with sampling time T_s = 0.25s are displayed on Fig. 1. For identification was used the method of multiple integration – function **mult.** Several model

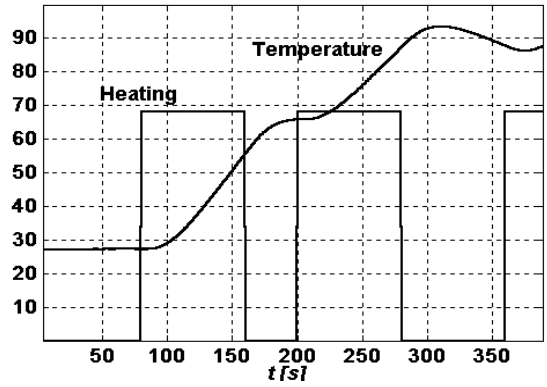

Fig.1. Measured data - heating

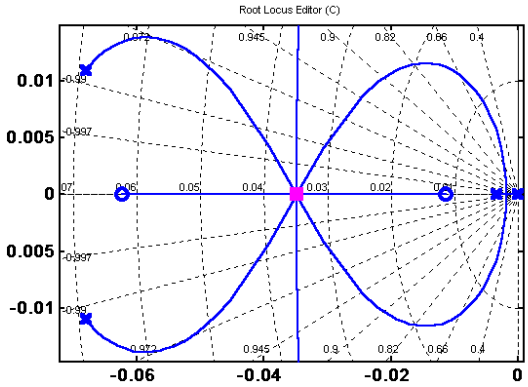

Fig. 2. Control circuit root locus

types were tested and the most appropriate was the third order without transport delay ($m = 0$, $n = 3$)

$$S(s) = \frac{0.029}{(s+0.0035)(s^2+0.14s+0.0049)} = \frac{0.029}{s^3+0.14s^2+0.0054s+1.7\cdot10^{-5}} \quad (9)$$

With default value of critical frequency $\omega_c = \pi/3 \approx 1$ the function calculated 1750 equations from 25 periods of integration, the relative rms error was less than 7% and the singular values of the LS matrix were $1.2\cdot10^9$, $1.1\cdot10^7$, $4.3\cdot10^6$ and $4.5°10^4$. Conditionallity number $1.2\cdot10^9/4.5\cdot10^4 \approx 2.7\cdot10^4$ was relatively high, but at least all the tested models had very similar frequency response around the plant critical frequency, i.e. lead to very similar behaviour with the same controller.

3.2 Controller Design

For sake of control static precision and dynamic enhancement PID controller was elected. To obtain adjustment on aperiodicity limit, function **pkpp** was called by command line **pkpp**(*S*/*s*), resulting in controller

$$R(s) = \frac{0.751s^2+0.0553s+0.000534}{s} = \frac{0.751\cdot(s+0.0622)(s+0.011)}{s} \quad (10)$$

and closed loop transfer function

$$F(s) = \frac{0.0021\cdot(s+0.062)(s+0.011)}{(s+0.035)^4} \quad (11)$$

The resulting root locus (Fig. 2) corresponds to (11).

3.3 Control Circuit Simulation

Because the plant *S* and the controller *R* are already defined from the previous calculations, the only statements necessary for calling simulation program are

ex=0;w=70;Tmax=500;K=400;h=0.7;trix

and the program **trix** calculates necessary data and calls the universal SIMULINK program **tripd_sim** for control circuit simulation (Fig. 3). All blocks of the simulation programs are open to eventual changes.

3.4 Simulation results

The three-level controller with inverse PID feedback transfer function maintains the dead zone of the three-level element, divided by the constants in the direct path of the scheme, in our case

$$\Delta = \frac{\pm1}{r_{-1}\cdot K} = \frac{\pm1}{0.000534\cdot400} = \pm4.7^{o}\mathrm{C} \quad . (12)$$

As the consequence of the dead zone, the temperature reaches the desired value 70°C only with notable error – see Fig. 4.

This disadvantage can be eliminated by dividing the PID controller into two parts: continuous PI controller (R_{dp}) in series with impulse PD controller designed as feedback structure (R_{fb}) of the three-level element. Controller (10) in that case will be decomposed to

$$R(s) = R_{dp}(s)\cdot R_{fb}^{-1}(s) = \frac{0.751\cdot0.0622\cdot(s+0.011)}{s}\cdot\frac{(s/0.0622+1)}{1} \quad . (13)$$

Assigned the transfer functions from (13) to the SIMULINK scheme from Fig. 3, the results on Fig. 5

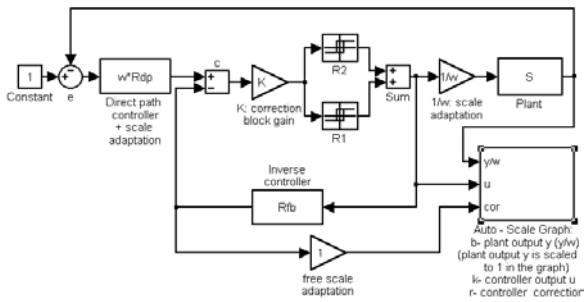

Fig. 3. Control circuit simulation program (tripd_sim)

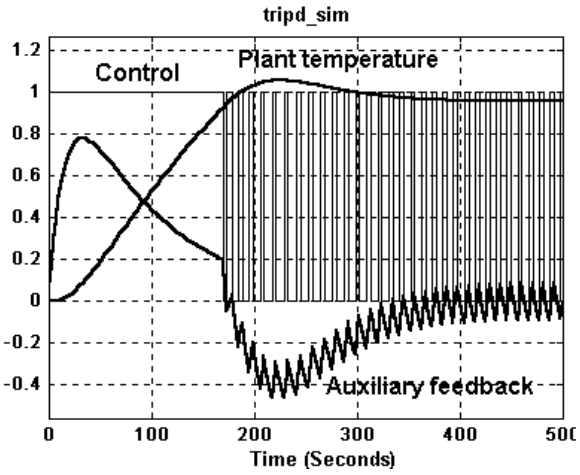

Fig. 4. Control circuit simulation - results

can be reached. Control is now precise, however, results in big overshot, caused by windup effect of the linear PI controller.

This phenomenon is another interesting task for the students – the PI part of the controller must be windup free. This can be reached for example by limitation the integrator output to the limitation of the impulse PD part, that is ±1.

3.6. Real Control of the Physical Plant

The controller with continuous PI part and integral channel limitation was installed with the physical plant and works without problems. The dynamic behaviour of the real control circuit is very similar to the model. Typical transients are displayed on Fig. 6. A group of the Department teachers and technicians works nowadays on an industrial version of the nonlinear controllers and auxiliary programs.

4. CONCLUSION

A freeware for teaching control engineering is

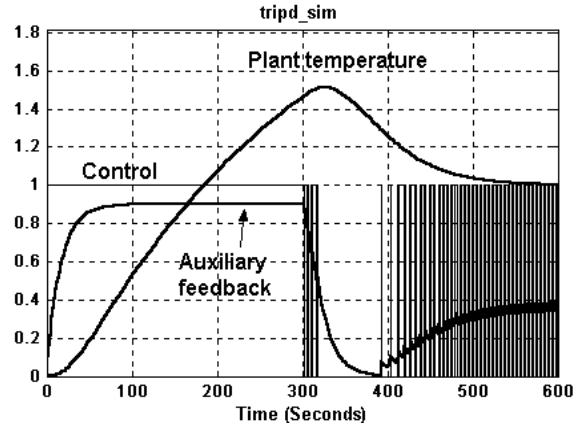

Fig. 5. Control circuit simulation - results

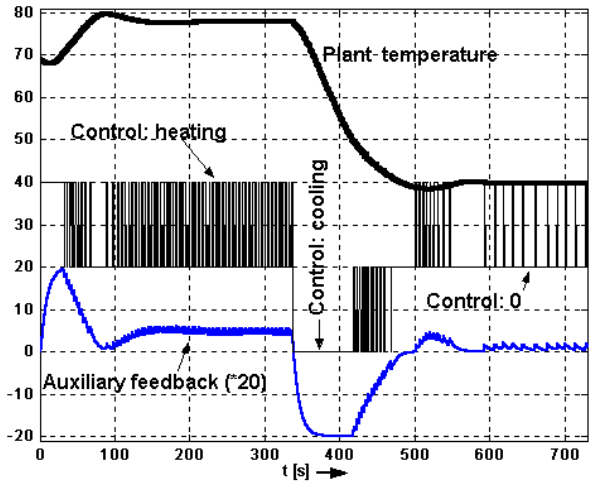

Fig. 6. Control of real plant by means of continuous PI in series with impulse PD controller.

available on web site of the Department of Control Engineering. The teachers' experience with it is very promising. A specialised internet textbook is under construction. It will help users to take advantage of the courseware. The programs and their description are already available there. Industrial control system is being derived, based on the ideas of the software.

REFERENCES

Astrom, K.J. and B. Wittenmark, (1990). *Computer Controlled Systems - Theory and Design*, Prentice Hall, Englewood Cliffs.

John, J. and J. Fuka, (2002). Innovation of Basic Subjects in Automatic Control In: *Preprints of FIE 2002 2nd International Conference on Automatic Control 2002* (F.Chang (Ed)). University of Oriente, Santiago de Cuba. (See also http://dce.felk.cvut.cz/sri2/sam_sri).

John J. and V. Rauch (1999). Nonlinear continuous system identification by means of multiple integration. *Acta Polytechnica* **39**, No.1, pp.55-62.

John J. (2001). Nonlinear continuous system identification by means of multiple integration II. *Acta Polytechnica* **41**, No.1, pp.64-67. (Also http://dce.felk.cvut.cz/sri2/mien/mien.htm

Kochenburger, R.J. (1960).A Frequency Response Method for Analyzing and Synthesizing Contactor Servomechanisms, *IEEE Trans. Aut. Control*, **AC-5**, No. 2, pp. 135-141.

Mansour, M. (1988). *A Simple Proof of the Routh–Hurwitz Criterion*. Report No. 88–04, Institute of Automatic Control & Industrial Electronics, Swiss Federal Institute of Technology, ETH–Zentrum, CH–8092 Zürich, Switzerland.