13. TRANSIENT ANALYSIS OF CONTROL SYSTEMS (TACS)

Co-Author: S. Bhattacharya

The program part TACS (acronym for Transient Analysis of Control Systems) was developed 10 years ago by L. Dubé. In 1983/84, Ma Ren-ming did a thorough study of the code, and made major revisions in it, particularly with respect to the order in which the blocks of the control system are solved [187]. More improvements will be made in the future by L. Dubé and others. Because changes are expected anyhow, and because L. Dubé was not available for co-authoring this section, the general philosophy of the solution method in TACS and possible alternatives are emphasized more than details of implementation.

TACS was originally written for the simulation of HVDC converter controls, but it soon became evident that it had much wider applications. It has been used for the simulation of

(a) HVDC converter controls,

(b) excitation systems of synchronous machines,

(c) current limiting gaps in surge arresters,

(d) arcs in circuit breakers,

and for other devices or phenomena which cannot be modelled directly with the existing network components in the EMTP.

Control systems are generally represented by block diagrams which show the interconnections among various control system elements, such as transfer function blocks, limiters, etc. Fig. 13.1 is a typical example. A block diagram representation is also used in TACS because it makes the data specification by the user simple. All signals are assigned names which are defined by 6 alphanumeric characters (blank is included as one of the characters). By using the proper names for the input and output signals of blocks, any arbitrary connection of blocks can be achieved. Amazingly, there is no uniform standard for describing the function of each block in an unambiguous way, except in the case of linear transfer functions [189]. Users of the EMTP should be aware of this.

The control systems, devices and phenomena modelled in TACS and the electric network are solved separately at this time. Output quantities from the network solution can be used as input quantities in TACS over the same time step, while output quantities from TACS can become input quantities to the network solution only over the next time step. TACS accepts as input network voltage and current sources, node voltages, switch currents, status of switches, and certain internal variables (e.g., rotor angles of synchronous machines). The network solution accepts output signals from TACS as voltage or current sources (if the sources are declared as TACS controlled sources), and as commands to open or close switches (if the switch is a thyristor or a TACS controlled switch).

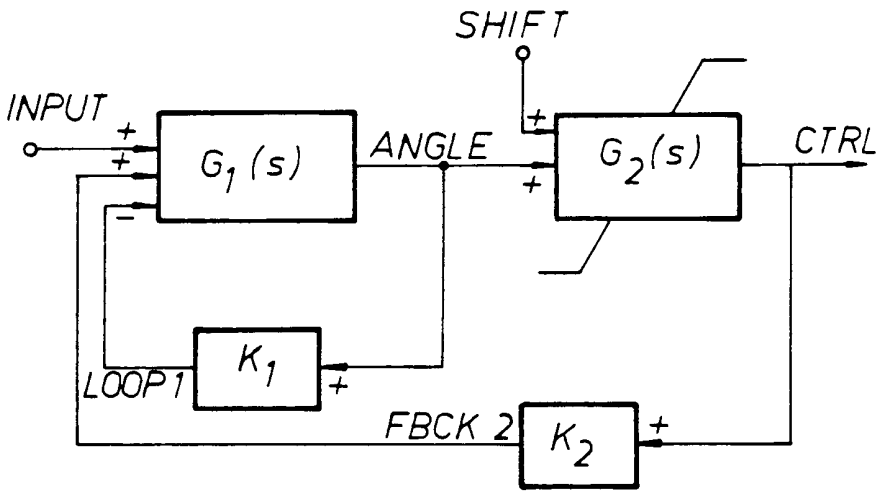


Fig. 13.1 - Typical block diagram representation of a control system

The present interface between the network solution and TACS, and possible alternatives to it, are explained first. The models available in TACS are described next, followed by a discussion of the initialization procedures.

13.1 Interface between TACS and the Electric Network

To solve the models represented in TACS simultaneously with the network is more complicated than for models of power system components such as generators or transformers. Such components can essentially be represented as equivalent resistance matrices with parallel current sources, which fit directly into the nodal network equations (1.8). The equations of control systems are quite different in that respect. Their matrices are unsymmetric, and they cannot be represented as equivalent networks.

Because of these difficulties, L. Dubé decided to solve the electric network (briefly called NETWORK from here on) and the TACS models (briefly called TACS from here on) separately. This imposes limitations which the users should be aware of. As illustrated in Fig. 13.2, the NETWORK solution is first advanced from $(t - \Delta t)$ to t as if TACS would not exist directly. There is an indirect link from TACS to NETWORK with a time delay of Δt, inasmuch as NETWORK can contain voltage and current sources defined between $(t - \Delta t)$ and t which were computed as output signals in TACS in the preceding step between $(t - 2\Delta t)$ to $(t - \Delta t)$. NETWORK also receives commands for opening and closing switches at time t, which are determined in TACS in the solution from $(t - 2\Delta t)$ to $(t - \Delta t)$. In the latter case, the error in the network solution due to the time delay of Δt is usually negligible. First, Δt for this type of simulation is generally small, say 50 μs. Secondly, the delay in closing a thyristor switch is compensated by the converter control, which alternately advances and retards the firing of thyristor switches to keep the current constant in steady-state operation. With continuous voltage and current source functions coming from TACS, the time delay can become more critical, however, and the user must be aware of its consequences. Cases have been documented where this time delay of Δt can cause numerical instability, e.g., in modeling the arc of circuit breakers with TACS [188].

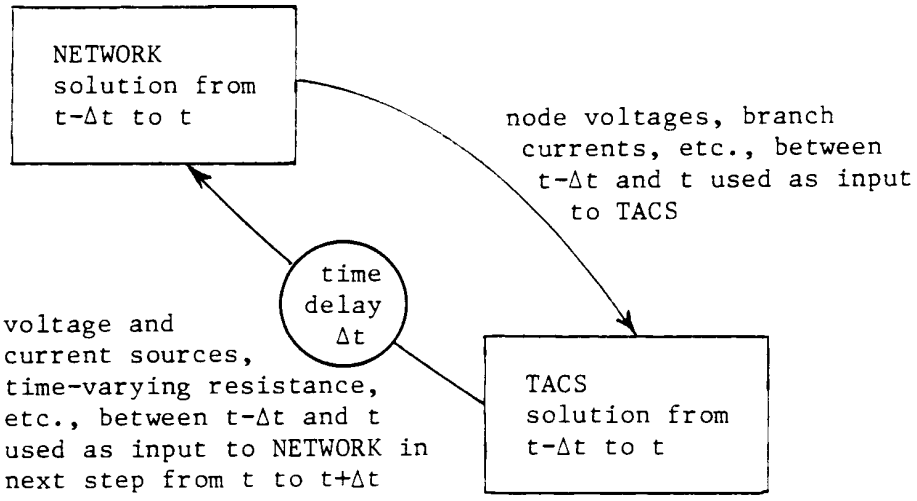
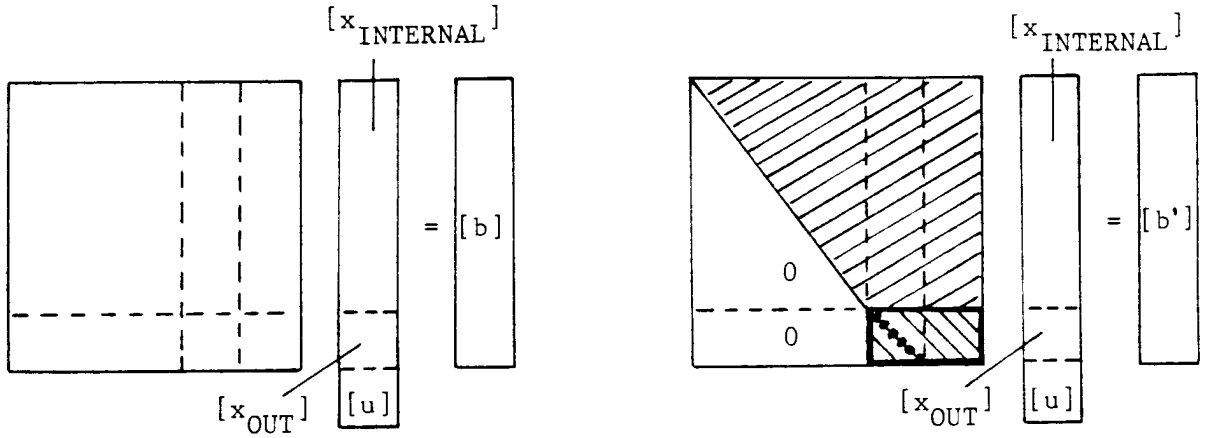


Fig. 13.2 - Interface between NETWORK and TACS solution

Once NETWORK has been solved, the network voltages and currents specified as input to TACS are known between (t - Δt) and t, and are then used to bring the solution of TACS from (t - Δt) to t. No time delay occurs in this part of the interface, except that TACS itself has built-in delays which may not always be transparent to the user.

If the EMTP is re-written some day to eliminate the time delay from TACS to NETWORK, two approaches (and possibly others) could be used:

(a) Predict the output from TACS at time t, and use the predicted values to solve the NETWORK and then TACS from (t - Δt) to t. Use the output from TACS as corrected values, and repeat the solution of the two parts again from (t - Δt) to t. If the differences between the predicted and corrected values are still larger than a specified tolerance, then do another iteration step with a repeat solution, until the values have converged to their final values. This approach is conceptually easy to implement, but its usefulness depends on the convergence behavior. Two or three iteration steps, on average, would probably be acceptable. This method would make it possible to add other corrections in NETWORK and TACS where only predictions are used now (e.g. correction of predicted armature currents in synchronous machines).



(a) before reduction

(b) after reduction

Fig. 13.3 - Form of difference equations for a control system. $[x_{INTERNAL}]$ = internal variables, $[x_{OUT}]$ = output signals, and $[u]$ = input signals

b) Do not solve the equations in TACS completely, but reduced them to an input-output relationship at time t, by eliminating variables which are internal to TACS. This approach has been used successfully in a stability program for the representation of excitation systems [72]. Assume that the trapezoidal rule of integration (or any other implicit integration method) is applied to the differential equations of the control system. Assume further that the variables are ordered in such a way that the internal variables $[x_{INTERNAL}]$ come first, then the output signals $[x_{OUT}]$ which become input to NETWORK (v_f in an excitation system) and finally the input signals [u] which come from the output of NETWORK ($v_{TERMINAL}$ in an excitation system). Then the equations would have the form of Fig. 13.3(a). By eliminating the internal variables $[x_{INTERNAL}]$ with Gauss elimination, the reduced system of equations in the bottom rectangle of Fig. 13.3(b) is obtained, which has the form

$$[A_{OUT}]\,[x_{OUT}] + [A_{IN}]\,[u] = [b'] \tag{13.1}$$

or in the case of an excitation system,

$$a_{OUT}\,v_f + a_{IN}\,v_{TERMINAL} = b'$$

In the latter case, this equation would have to be incorporated into the synchronous machine model of Section 8. Limiters can be handled as well with this approach, as explained in [72].

Method (b) could be implemented in a number of different ways. For control systems which can be represented by one transfer function, the implementation would be very simple, because TACS already produces an equation of the form of Eq. (13.1), as explained later in Eq. (13.7). For more complicated systems, the existing code of TACS could be used to solve the equations of system twice, e.g., in the case of the excitation system, for 2 predicted values of $v_{TERMINAL}$. The two solutions $v_{f1}(t)$ and $v_{f2}(t)$ would create 2 points in the $v_{TERMINAL}$ - v_F - plane, and a straight line through them would produce Eq. (13.1) indirectly.

13.2 Transfer Function Block with Summer

The transfer function block (Fig. 13.4) is used to describe a relationship between input U(s) and output X(s) in the Laplace domain,

$$X(s) = G(s)\,U(s) \tag{13.2a}$$

where the transfer function is a rational function of order n,

$$G(s) = K\frac{N_0 + N_1 s \ldots + N_m s^m}{D_0 + D_1 s \ldots + D_n s^n} \qquad with \quad m \le n \tag{13.2b}$$

The Laplace operator is replaced by jω to obtain the steady-state frequency response at any angular frequency ω, including dc. For transient solutions, s is replaced by the differential operator d/dt, which converts Eq. (13.2) into a linear n-th order differential equation

$$D_0 x + D_1 \frac{dx}{dt} \ldots + D_n \frac{d_n x}{dt^n} = K\left(N_0 u + N_1 \frac{du}{dt} \ldots + N_m \frac{d^m u}{dt^m} \right) \tag{13.3}$$

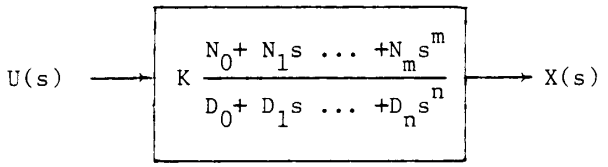


Fig. 13.4 - Transfer function block

In TACS, the n-th order differential equation is re-written as a system of n first-order differential equations by introducing internal variables for the derivatives of u and x

$$x_1 = \frac{dx}{dt}, \quad x_2 = \frac{dx_1}{dt}, \ \ldots \ \ldots \ x_n = \frac{dx_{n-1}}{dt}$$

$$u_1 = \frac{du}{dt}, \quad u_2 = \frac{du_1}{dt}, \ \ldots \ \ldots \ u_m = \frac{du_{m-1}}{dt} \tag{13.4}$$

With these internal variables, Eq. (13.3) becomes an algebraic equation

$$D_0 x + D_1 x_1 \ \ldots \ + D_n x_n = K(N_0 u + N_1 u_1 \ \ldots \ + N_m u_m) \tag{13.5}$$

To eliminate these internal variables again, the differential equations (13.4) are first converted into difference equations with the trapezoidal rule of integration,

$$x_i(t) = \frac{2}{\Delta t} x_{i-1}(t) - \{ x_i(t-\Delta t) + \frac{2}{\Delta t} x_{i-1}(t-\Delta t) \} \qquad for \ \ i = 1, \ldots \ n \tag{13.6a}$$

$$u_j(t) = \frac{2}{\Delta t} u_{j-1}(t) - \{ u_j(t-\Delta t) + \frac{2}{\Delta t} u_{j-1}(t-\Delta t) \} \qquad for \ j = 1, \ldots \ m \tag{13.6b}$$

where $x_0 = x$ and $u_0 = u$.

Expressing x_n as a function of x_{n-1} in Eq. (13.5) with Eq. (13.6), and then again expressing x_{n-1} as a function of x_{n-2} etc., until only x is left, and using the same procedure for u, produces a single output-input relationship of the form [189]

$$cx(t) = K\,du(t) + hist(t - \Delta t) \tag{13.7}$$

This is the equation which is used in the transient solution of the control system. After the solution at each time step, n history terms must be updated to obtain the single term "hist" for the solution over the next time step. If recursive formulas are used, then

$$\begin{aligned}
hist_1(t) &= Kd_1 u(t) - c_1 x(t) - hist_1(t-\Delta t) + hist_2(t-\Delta t)\\
&\ldots = \ldots\\
&\ldots = \ldots\\
hist_i(t) &= Kd_i u(t) - c_i x(t) - hist_i(t-\Delta t) + hist_{i+1}(t-\Delta t)\\
&\ldots = \ldots\\
&\ldots = \ldots
\end{aligned}$$

$$\begin{aligned}
&hist_n(t) = Kd_n u(t) - c_n x(t)\\
&with\\
&\quad hist = hist_1
\end{aligned} \tag{13.8}$$

The coefficients c_i, d_i are calculated once at the beginning from the coefficients N_i, D_i of the transfer function, with the recursive formula

$$c_i = c_{i-1} + (-2)^i \{ \binom{i}{i} (\frac{2}{\Delta t})^i D_i + \binom{i+1}{i} (\frac{2}{\Delta t})^{i+1} D_{i+1} \ldots + \binom{n}{i} (\frac{2}{\Delta t})^n D_n \} \tag{13.9}$$

where $\binom{i}{j}$ is the binomial coefficient, and where the starting value is

$$c_0 = \sum_{i=0}^{n} \left(\frac{2}{\Delta t} \right)^i D_i \,, \qquad with \quad c = c_0 \tag{13.10}$$

The formulas for d_i are identical, if D is replaced by N, and if the upper limit is m rather than n.

Instead of a single input signal u, TACS accepts the sum of up to five input signals $u_i, \ldots u_5$, as illustrated in Fig. 13.5 (subscripts 1,2,… are no longer used to indicate internal variables of a block from here on). To model a summer by itself, a zero-order transfer function block is used with $K = N_0 = D_0 = 1$. This zero-order transfer function is contained in Eq. (13.7) as a special case, with K = c = d = 1 and hist = 0.

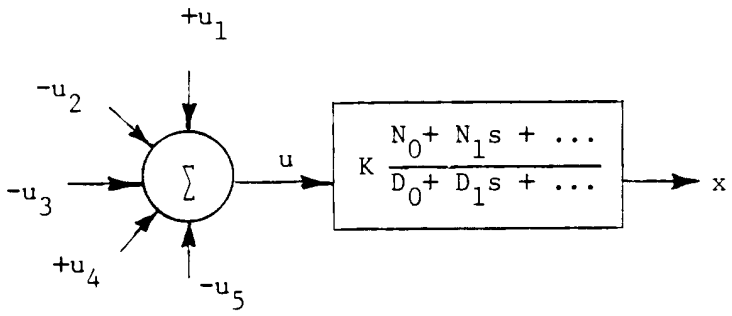


Fig. 13.5 - Transfer function with summer ($u = u_1 - u_2 - u_3 + u_4 - u_5$)

If the control system consists solely of interconnected transfer function blocks and summers, then the entire system is described by using an equation of the form (13.7) for each one of the blocks. For the example of Fig. 13.6, there would be four equations

$$\begin{bmatrix} c_a & -K_a d_a & -K_a d_a & 0 & K_a d_a & 0 \\ -K_b d_b & 0 & 0 & c_b & 0 & -K_b d_b \\ -K_c d_c & c_c & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_d d_d & c_d & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} hist_a \\ hist_b \\ hist_c \\ hist_d \end{bmatrix}$$

which is the same form as in Fig. 13.2(a), with

$$[x_{INTERNAL}] = [x_1\ x_2\ x_3]$$

$$[x_{OUT}] = [x_4]$$

$$[u] = [u_1\ u_2]$$

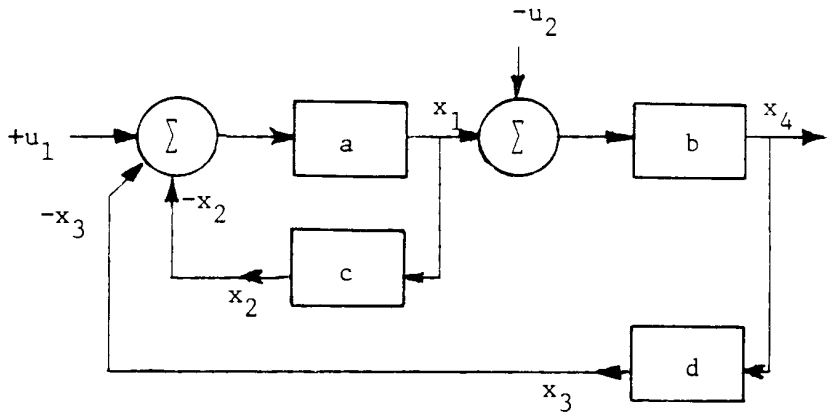


Fig. 13.6 - Control system with linear transfer functions

In TACS, this system of equations

$$[A_{xx}]\,[X] + [A_{xu}]\,[u] = [hist] \tag{13.11}$$

is solved by first performing a triangular factorization on $\{[A_{xx}].\ [A_{xu}]\}$ before entering the time step loop. In each time step, the unknown variables [x] are then found by

(a) assembling the right-hand side [hist],

(b) performing a downward operation on it,

(c) doing a backsubstitution to obtain [x], and

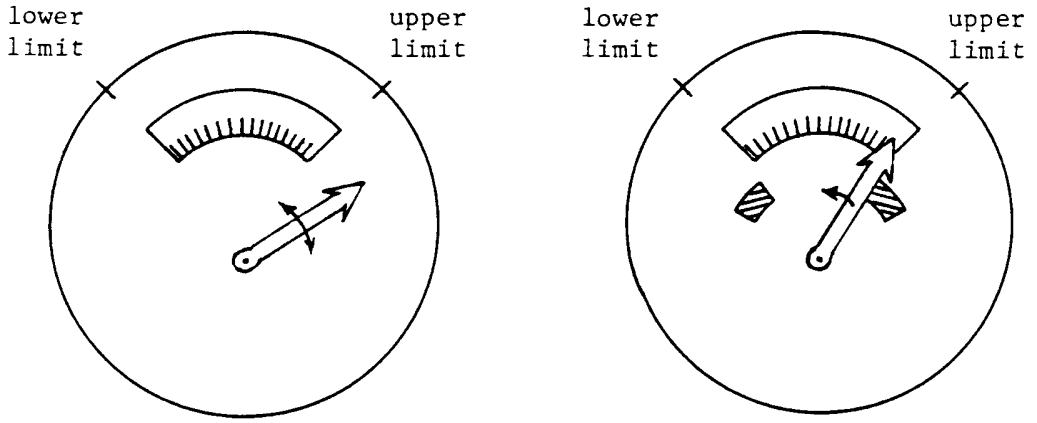
(d) updating the history terms of each block.

The solution procedure is very similar to the one indicated in Fig. 13.3(b), except that the elimination does not stop on the vertical line which separates the columns of $[x_{INTERNAL}]$ and $[x_{OUT}]$, but continues to the diagonal (indicated by dots in Fig. 13.3(b)). Since the matrix is unsymmetric here, both the upper and lower triangular matrix coming out of the triangularization must be stored, in contrast to the matrix in NETWORK where only the upper triangular matrix is stored. Since the matrix is sparse, optimal ordering techniques are used to minimize the number of fill-in elements. Only the nonzero elements are stored with a compact storage scheme similar to the one discussed in Appendix III. Whether pivoting is needed is unclear to the authors.

13.3 Limiters

There are two types of limiters, the windup limiter with clipped output ("static limiter" in the EMTP Rule Book) and the non-windup limiter with clamped output ("dynamic limiter" in the EMTP Rule Book). The windup limiter can be visualized as a measuring instrument in which the needle (position = output signal) can only be seen within a limited window, but the needle is allowed to move freely (wind up) outside the window (Fig. 13.7(a)). In the non-windup limiter, the needle is restrained from moving outside the window (Fig. 13.7(b)). In both cases, the movement of the needle is described by differential equations. The equation describing the limiting function has the

same form in both cases, but the criteria for backing off are different.



(a) Windup limiter (b) Non-windup limiter

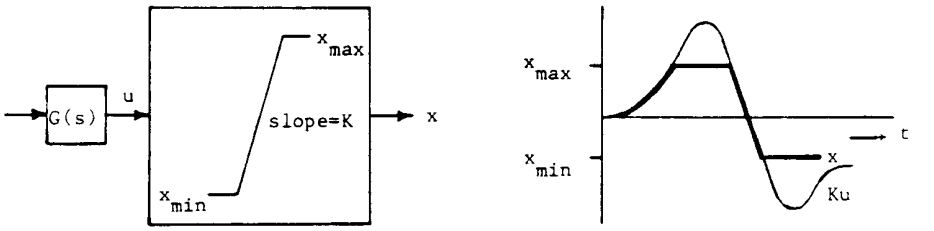
Fig. 13.7 - Limits in a measuring instrument

13.3.1 Windup Limiter

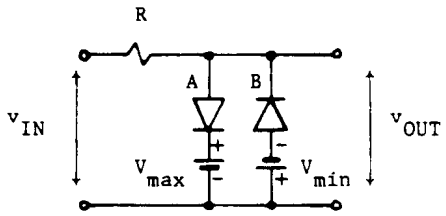
The output x of the windup limiter of Fig. 13.8 is

$$x = \begin{cases} Ku, & \text{if } x_{min} < Ku < x_{max} \\ x_{min}, & \text{if } Ku \le x_{min} \\ x_{max}, & \text{if } Ku \ge x_{max} \end{cases} \qquad (13.12)$$

Either one of the three equations is still a linear algebraic equation of the form of Eq. (13.7), with c = d = 1, hist = 0 inside the limits, and c = 1, d = 0, hist = (x_{max} or x_{min}) at the limit. The proper way of handling this limiter is to change the linear equations (13.11) at instants when x hits the limit and when it moves off the limit again. This requires occasional re-triangularizations, which are no different in principle from those required in NETWORK whenever switch positions change or when the solution in piecewise linear elements moves from one segment to another.



(a) Block diagram representation (b) Limiting action



(c) Clipping circuit implementation for K=1 (diode A conducts when $v_{IN} > v_{max}$, diode B conducts when $v_{IN} < v_{min}$

Fig. 13.8 - Windup limiter

If there are only a few limiters, one could also use the compensation method described in Section 12.1.2. Assume that the control system contains only two limiters which limit x_i and x_k. In that case, one could precalculate column i and column k of the inverse matrix of $[A_{xx}]$ with two repeat solutions before entering the time step loop. In each time step, the variables would first be calculated as if no limits exist. Call this solution $[x_{without}]$. If both $x_{i\text{-}without}$ and $x_{k\text{-}without}$ are outside their limits, then the necessary corrections $\Delta hist_i$ and $\Delta hist_k$ in the right-hand side of Eq. (13.11) to produce limited values are found by solving the two equations

$$\begin{bmatrix} x_{i-limit} \\ x_{k-limit} \end{bmatrix} = \begin{bmatrix} x_{i-without} \\ x_{k-without} \end{bmatrix} + \begin{bmatrix} b_{ii} & b_{ik} \\ b_{ki} & b_{kk} \end{bmatrix} \begin{bmatrix} \Delta hist_i \\ \Delta hist_k \end{bmatrix} \tag{13.13a}$$

If only $x_{i\text{-without}}$ is outside its limits, then

$$\begin{aligned} x_{i-limit} &= x_{i-without} + b_{ii}\,\Delta hist_i \\ \Delta hist_k &= 0 \end{aligned} \tag{13.13b}$$

The final solution is found by superposition,

$$[x] = [x_{without}] + \begin{bmatrix} b_{1i} & b_{1k} \\ b_{2i} & b_{2k} \\ . & . \\ . & . \\ . & . \end{bmatrix} \begin{bmatrix} \Delta hist_i \\ \Delta hist_k \end{bmatrix} \tag{13.13c}$$

The coefficients b in Eq. (13.3) are the elements of column i and k of $[Z_{ss}]^{-1}$.

At the time when TACS was first written, both the re-triangularization procedure and the compensation method were regarded as too costly, and the simpler method discussed in Section 13.4 was introduced instead. It suffered initially from unnecessary time delays, which have now been mostly removed with the recent code changes of Ma Ren-ming in version M36. Whether re-triangularization or compensation will be used in future versions to remove the remaining time delays remains to be seen.

In comparing Fig. 13.8(a) with the piecewise linear representation of network elements discussed in Section

12.1.3, one notices that the slope in the saturated region is always zero (hard limit) rather than finite (soft limit). In rewriting TACS, it may be worth considering whether soft limits would be a useful enhancement. In Fig. 13.8(c) the limits become soft if the internal resistances of the diodes and dc voltage sources are taken into account, or if resistors are specifically added for that purpose. The equation for soft limits, with the notation from Fig. 13.9, would be

$$X = \begin{cases} Ku & \text{if } x_{min} < Ku < x_{max} \\ x_{min} + K_{min}(u-u_{min}) & \text{if } Ku \le x_{min} \\ x_{max} + K_{max}(u-u_{max}) & \text{if } Ku \ge x_{max} \end{cases} \qquad (13.14)$$

These equations have again the form of Eq. (13.7), and soft limits can therefore be implemented in the same way as hard limits. As a matter of fact, the hard limit would become a special case of the soft limit of Eq. (13.14) by simply setting K_{max} or K_{min} to zero.

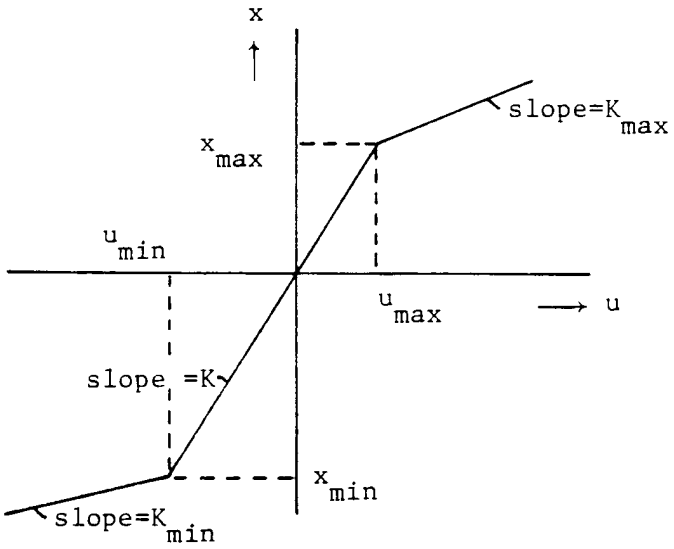


Fig. 13.9 - Soft limits

13.3.2 Non-Windup Limiter

In the windup limiter, the output of a transfer function block is just clipped, without affecting the dynamic behavior of the transfer function block on the input side itself. In a non-windup limiter, this is no longer true. Here, the dynamic behavior of the transfer function block is changed by the limiting action.

Before describing the limiting action with equations, it is important to understand that non-windup limiters should only be used with first-order transfer functions. For second and higher-order transfer functions, it is no longer clear which variables should be limited. Take a second-order transfer function G(s) = 1/ 2 as an example. It can easily be shown [190] that backing off the limit will occur in three different ways in this case, depending on whether the internal variables dx/dt or d^2x/dt^2, or both, are forced to remain at zero after the limit is hit. This ambiguity can only be removed if the user defines the problem as two cascaded first-order transfer function blocks,

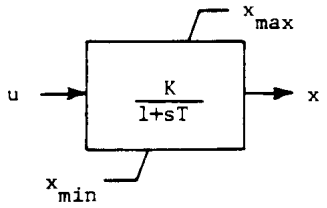
with the proper limits on each of them. Even for the first-order transfer function, the meaning of the limiting function is confused if it has any zeros ($N_1 \neq 0$) [191]. It is because of these ambiguities why the limiter in the current-controlled dc voltage source described in Section 7.6.2 may be incorrect.

To make the definition of non-windup limiters unique, they should only be allowed on first-order transfer functions with no zeros of the form

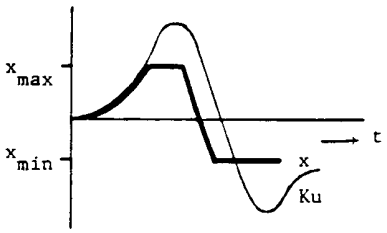
$$G(s) = \frac{K}{1 + sT} \tag{13.15}$$

The equations are

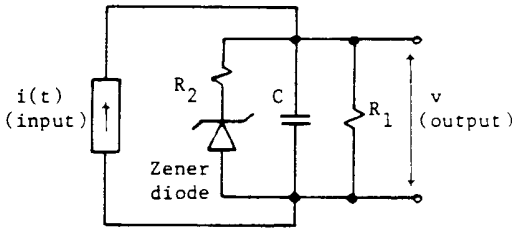
$$\begin{aligned} x + T\frac{dx}{dt} &= Ku \qquad if \quad x_{min} < x < x_{max} \\ x = x_{min} \qquad &if \quad x \le x_{min} \quad and \quad (Ku - x) < 0 \\ x = x_{max} \qquad &if \quad x \ge x_{max} \quad and \quad (Ku - x) > 0 \end{aligned} \tag{13.16}$$



(a) Block diagram representation



(b) Limiting action



(c) Circuit implementation

Fig. 13.10 - Non-windup limiter

For operation within the limits, the differential equation is converted to the algebraic equation (13.7). That equation, and the equations valid at the limit, are all linear algebraic equations, as in Eq. (13.12) for the windup limiter. The non-windup limiter can therefore be handled in exactly the same way as the windup limiter, either with re-triangularization or with the compensation method. While the windup limiter has the coefficients c = d = 1, and

hist = 0 inside the limits, the non-windup limiter has

$$d = K$$
$$c = \left(1 + \frac{2T}{\Delta t}\right)$$
$$hist = Ku(t - \Delta t) - \left(1 - \frac{2T}{\Delta t}\right) x(t - \Delta t)$$

For deciding when to back off, the derivative T dx/dt = Ku - x must be used, rather than Ku.

Changing from a hard to a soft limit would also be possible with the non-windup limiter. In the implementation of Fig. 13.10(c), the limits would become soft if $R_2 \neq 0$, where R_2 can either be the internal resistance of the Zener diode, or a resistor specifically added to create soft limits.

From the limited (unpublished) information available to the authors, it appears that TACS handles the non-windup limiters with a pseudo-compensation method, in which corrections are made to the right-hand sides [hist] in Eq. (13.11) a priori at the beginning of each time step. As explained above Eq. (13.13), a correct implementation of the compensation method requires a complete solution of the control system without limiters, followed by superposition of correction terms for which elements of $[A_{xx}]^{-1}$ are needed. This does not seem to be done in TACS, and the treatment of limiters is therefore somewhat suspicious. Since TACS does reset the variable to its limit value whenever it exceeds its limits, the answers are probably correct, except that the procedure is unable to eliminate the time delays in closed loops discussed in Section 13.4.

The pseudo-compensation method also seems to create subtle differences in the way it backs off the limit. It seems to use the equation

$$\frac{T}{\Delta t}\{2x(t) - x_{limit}\} = K\,u(t) - x(t)$$

in the first step after backing off, which would be the backward Euler formula if the factor 2 were missing, while Eq. (13.7) would back off with

$$\frac{T}{\Delta t}\{x(t) - x_{limit}\} = K\,\frac{u(t) + u(t-\Delta t)}{2} - \frac{x(t) + x_{limit}}{2}$$

13.4 Limiter Implementation with Possible Time Delay

With the code changes of Ma Ren-ming in 1983/84, the variables are now ordered in such a way that most of the time delays which were caused by limiters no longer exist in version M36 and later versions. For example, the open loop control system of Fig. 13.11 was originally solved in the sequence S_1, $G_2(s)$, $G_3(s)$, S_2, S_3 and finally $G_1(s)$, because of a rule that transfer function blocks feeding into special or supplemental device blocks S should be solved first. This has been changed, and the blocks are now solved in their functional order S_1, $G_1(s)$, $G_2(s)$, S_2,

$G_3(s)$, S_3. With this order, it is simple to observe the limits on the output x_1, without having to re-triangularize the matrix or without having to use the compensation method, because x_1 is limited first before any other variables are computed. In the system of equations (13.11), this means that the equation for $G_1(s)$ must be the last one, with enforcement of the limits on x_1 being done in the backsubstitution. Ma Ren-ming observes correctly [187] that there is no difference between n-th order and zero order transfer functions, or between windup and non-windup limiters in this simple ordering scheme.

A more complicated example for the new ordering rule is shown in Fig. 13.12, with 10 transfer function blocks of which three have limits. The first four blocks G_1, G_2, G_3, and G_4 form one set of equations which are disconnected from the others. This first set of equations is solved simultaneously, with rows in Eq. (13.7) ordered G_1, G_4, G_3, and G_2. With this order, the output of G_2 is the first variable to be found in the backsubstitution. By keeping it within its limits at that point, the properly limited value will be used in the rest of the backsubstitution in finding the outputs of G_3, G_4, and G_1. Using the known output of G_2, the output of G_5 is found from one single equation, and knowing the output of G_5, the output of G_0 is found from another single equation and then kept within its limits. Finally, the equations for G_7, G_8, G_9, and G_{10} for another independent set of equations, and if ordered in that sequence, the limits on the output of G_{10} can again be easily observed because it is the first variable found in the backsubstitution. So in spite of feedback loops and limiters, the control system of Fig. 13.12 is now solved simultaneously without the time delays observed in pre-M36 versions. Note that this ordering scheme developed for easy implementation of limiters may not completely minimize the fill-ins in the triangularization, but this is a small price to pay for the proper implementation of limiters.

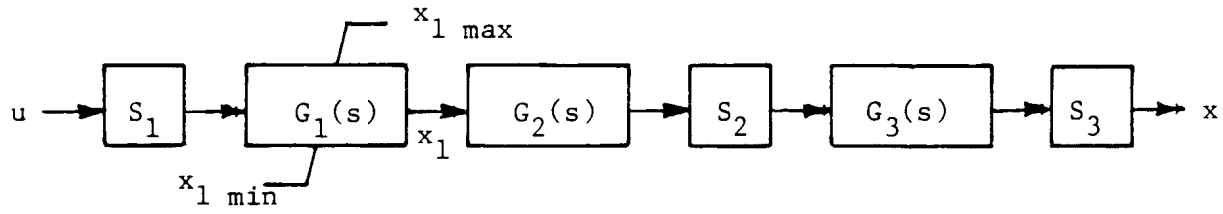


Fig. 13.11 - Simple open loop control system

Time delays cannot be avoided completely with the new ordering scheme. Fig. 13.13 shows an example where two limiters are within the same loop. In this case, TACS inserts a time delay of Δt (if not explicitly done so by the user) and the solution is then no longer simultaneous. Note that with re-triangularization or with the compensation method, the solution of that system would again become simultaneous.

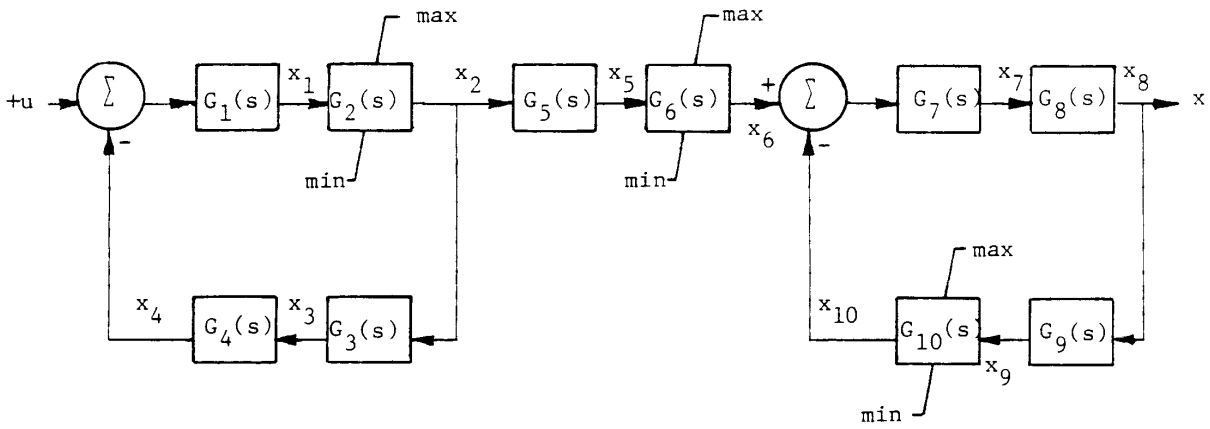


Fig. 13.12 - Control system with feedback loops

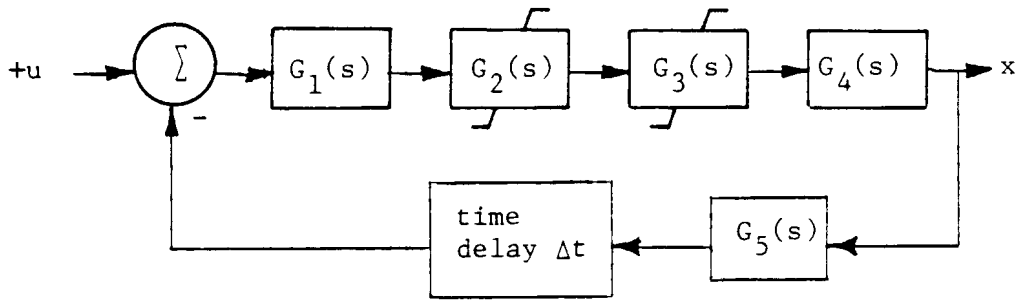


Fig. 13.13 - Two limiters within same loop

13.5 Signal Sources

TACS has signal sources built into it, similar to the voltage and current sources in NETWORK. They serve as input signals to transfer function blocks and other blocks. In the system of equations (13.11), they are handled as known values in vector [u].

Resident sources are signal sources with reserved names, which are available by simply referring to their names. Resident sources can also be used as voltage or current sources in NETWORK through the TACS-NETWORK interface. They are

TIMEX = simulation time in seconds (0 in steady state),

ISTEP = number of time step,

DELTAT = step size in seconds,

FREQHZ = network frequency in Hz of first sinusoidal source,

OMEGAR = 2πFREQHZ,

ZERO = 0.0,

MINUS1 = -1.0,

PLUS1 = +1.0,

INFTY = $+\infty$ (very large number which still fits computer system),

PI = π.

There are also signal sources with data specifications supplied by the user, such as

single rectangular pulse,

sinusoidal function,

repetitive pulse,

repetitive ramp function,

node voltage from NETWORK,

switch current from NETWORK,

special NETWORK variables (e.g., rotor angle of machines),

switch status,

any voltage or current source defined in NETWORK.

13.6 Special Devices

Transfer function blocks, limiters and signal sources are not enough to model realistic control systems. Other building blocks have therefore been added to TACS under the heading of "Special Devices" ("Supplemental variables and devices" in the EMTP Rule Book). They make TACS extremely versatile, but they do not fit neatly into the control system equations (13.7). They are therefore solved sequentially, rather than simultaneously as for the transfer function blocks, with the user controlling the sequence. In Fig. 13.11, the special device S_1 would be solved after G_2 has been solved, and S_2 would be solved after G_3 has been solved. The solution would still be simultaneous in this case. In general, the sequence of calculations is more complicated, with non-simultaneous solutions through time delays. For details, the reader should consult the EMTP Rule Book.

All special devices can either be designated as input devices, as output devices, or as internal devices by the user. To make the solution as much simultaneous as possible, the user should keep the number of internal devices as low as possible, and use input or output devices instead whenever possible. The rules for the designation are as follows:

(a) Input devices: All inputs must either be TACS signal sources or output from other input devices. They are essentially used to pre-process signals before they enter transfer function blocks (e.g., S_1 in Fig. 13.11).

(b) Output devices: Their output must not be used as input to any other block, except to other output devices. They are essentially used to post-process control system outputs for its own sake, or before passing them on as voltage or current sources or switching commands to NETWORK (e.g. S_3 in Fig. 13.11).

(c) Internal devices: They are inside the control system (e.g., S_2 in Fig. 13.11).

The behavior of the special devices is either defined through user-supplied FORTRAN expressions, or with built-in types.

13.6.1 FORTRAN-Defined Special Devices

The FORTRAN expression can be more or less as general as allowed by the FORTRAN-IV language itself (for details see the EMTP Rule Book). Algebraic operators (+-* etc.), relational operators (.EQ. etc.), logical operators (.AND. etc.), FORTRAN intrinsic functions (SIN, EXP, etc.) and special functions defined in the EMTP Rule Book can be used. In the example

ANGLE = DEG(AT AN(CNTRL - BIAS2)) + 36.2,

the output signal is ANGLE, while the input signals are CNTRL and BIAS2. AT AN is the arctangent function, while DEG is a special function for converting radians to degrees.

A FORTRAN expression of the form

VARIABLE = VARIABLE + {Arithmetic Expression}

is not allowed, because it gives rise to sorting problem within TACS.

13.6.2 Built-In Special Devices

There are 17 built-in special devices at this time, for which the user supplies the parameters only. They are

(a) accumulator and counter,

(b) controlled integrator,

(c) digitizer,

(d) frequency sensor,

(e) input-IF component,

(f) instantaneous min/max,

(g) level-triggered switch,

(h) min/max tracking,

(i) multi-operation time-sequenced switch,

(j) point-by-point user defined nonlinearity,

(k) pulse transport delay,

(l) relay-operated switch,

(m) RMS value,

(n) sample and track hold,

(o) signal selector,

(p) simple derivative (backward Euler),

(q) transport delay.

Details about their characteristics can be found in the EMTP Rule Book.

13.7 Initial Conditions

The ac steady-state solution for the electric network is found first, before TACS variables are initialized. All variables from NETWORK are therefore available for the automatic initialization in TACS, but not the other way

around. This may cause problems, e.g., if a TACS output defines a sinusoidal voltage source in NETWORK whose initial amplitude and phase angle, supplied by the user, could differ from the values coming out the TACS initialization. An iterative steady-state solution between NETWORK and TACS would resolve such discrepancies, but they are probably so rare that such an iteration scheme cannot be justified. An error message (possibly with termination) would be useful, however.

The automatic initialization of TACS variables is complicated and not foolproof at this time, and improvements are likely to be added in the future. The present initialization procedure is therefore only described in broad terms.

The input and output signals of transfer function blocks are usually dc quantities in steady state. For dc quantities, the output-input relationship of the transfer function (13.2b) becomes

$$x_{dc} = K \frac{N_0}{D_0} u_{dc} \tag{13.17}$$

which is the same form as Eq. (13.7) for the transient solution, with c = 1, d = N_0/D_0, and hist = 0. If the entire control system consists of transfer function blocks only, a system of equations can be formed, similar to Eq. (13.11), and solved for the unknown TACS variables [x_{dc}]. This is essentially what TACS does automatically now. The variables [x_{dc}] are not needed directly, but only indirectly for initializing [hist] in Eq. (13.11) before entering the time step loop.

Unfortunately, control systems are more complicated. Any sophisticated control system has integrators G(s) = K 1/ . Their steady-state output must now be supplied by the user, but these values are not always easy to find. For example, the output of an unbounded integrator with nonzero input is a continuously increasing ramp function. In practice, integrators are always bounded within upper and lower limits. Therefore, the steady-state output of a bounded integrator is either at its minimum or maximum value, which TACS could distinguish from the sign of the input signal. A realistic steady-state equation of a bounded integrator for nonzero input would therefore be

$$x_{dc} = \begin{cases} x_{\min} & \text{if } u_{dc} < 0 \\ x_{\max} & \text{if } u_{dc} > 0 \end{cases} \tag{13.18}$$

Evaluation of the steady-state output value of a bounded integrator with zero input or of an unbounded integrator is impossible from the knowledge of its input alone.

Further complications are introduced because TACS signal sources are not restricted to dc quantities in steady state. They could be pulse trains, sinusoidal functions, and other periodic functions. To automate the initialization procedure for all such eventualities is therefore still an unresolved issue. At this time, the user must supply initial conditions in complicated cases and for most special devices.