Software implementation for transfer functions of arbitrary order on managing controllers

F A Alekperli1 and I S Bayramova1
1 Sumgait State University, 43rd quarter, Sumgayit, Az5008, Azerbaijan

E-mail: fazlf_50@mail.ru

Abstract. Despite the great capabilities of controllers of the Simatic type, their library lacks a block for implementing the transfer functions necessary for programming complex control algorithms both in research by simulation and in industrial applications. This article discusses a method for software implementation of transfer functions of arbitrary order and delay in the TIA (Totally Integrated Automation) environment. The programs compiled in SCL and LAD languages and examples confirming the validity of the proposed method are given. A program is presented that implements the solution of a difference equation in the SCL language, as well as programs for implementing the factorial and the number of combinations. These programs are designed as functions. The fourth program, compiled in the LAD language, is designed to build a transient process for a given transfer function in real time. The created software blocks allow developing software for the study of fairly complex regulatory systems by simulating and adapting to real industrial conditions without much difficulty.

1. Introduction
One of the main advantages of digital control systems is the ability to implement sophisticated control algorithms [1]. Sophisticated control algorithms require the implementation of transfer functions. Currently, digital control systems are mainly implemented on the basis of programmable controllers. One of the widely used is Siemens type Simatic controllers, the library of which does not provide blocks for the implementation of transfer functions. In the Matlab package, there are functions tf, zpk and the Transfer Fcn block that implement transfer functions [2]. However, Matlab programs do not integrate with programs of controllers such as Simatic, which does not allow their use in industrial conditions. We have developed algorithms and programs for the implementation of transfer functions [3-5]. In particular, in [3], algorithms and programs for implementing transfer functions of the form

\[ W(s) = \frac{b_1 s + b_0}{a_2 s^2 + a_1 s + a_0} \]  \hspace{1cm} (1)

and lower orders are given. Algorithms are based on the method of solving differential equations on analog computers [6].

In [4, 5], algorithms and programs of transfer functions of the form

\[ W(s) = \frac{b_3 s^3 + b_2 s^2 + b_1 s + b_0}{a_2 s^2 + a_1 s + a_0} \] \hspace{1cm} (2)

and higher order are given. The implementation of the transfer functions (2) is based on (1) using the expression
\[
W(s) = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n} e^{-\tau s}, \quad m \leq r.
\]

where
\[
a_i^* = \frac{a_i}{a_2}, \quad a_0^* = \frac{a_0}{a_2}, \quad b_i^* = \frac{b_i}{b_2}, \quad b_0^* = \frac{b_0}{b_2} - a_0^*.
\]

The implementation of a higher-order transfer function is based on its decomposition into second-order and lower-order transfer functions using zeros and poles of the original transfer function.

As we can see, the implementation of the transfer functions (2) and higher order requires preliminary preparatory work for each specific case. This factor does not allow creating a universal algorithm on the one hand, and to implement transfer functions with variable parameters on the other. Although in the environment of Step 7, where the programming languages LAD, FBD and/or STL are used, this method is the only way to implement transfer functions.

2. Transfer functions and their software implementation

Let implementation of the transfer function be required in the following form:
\[
W(s) = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n} e^{-\tau s}, \quad m \leq r.
\]

Those or other coefficients of the transfer function can be constant or variable. The implementation of the transfer function (3) is proposed using the difference equation [7]
\[
A_0 y[nT] + A_1 y[(n-1)T] + \ldots + A_{r-1} y[(n-r+1)T] + A_r y[(n-r)T] = B_0 x[(n-k)T] + B_1 x[(n-1-k)T] + \ldots + B_{m-1} x[(n-m+1-k)T] + B_m x[(n-m-k)T].
\]

The parameters of difference equation (4) are determined by the expressions:
\[
k = \left\lfloor \frac{\tau}{T} \right\rfloor, \quad B_j = \sum_{i=j}^{m} (-1)^i C_i^j b_{m-i} T^{-r-i}, \quad j = 0, m, \quad A_j = \sum_{i=j}^{r} (-1)^i C_i^j a_{r-i} T^{-r-i}, \quad j = 0, r.
\]

Here \(\left\lfloor \frac{\tau}{T} \right\rfloor\) is the rounded value of the ratio \(\frac{\tau}{T}\) to the integer value, \(T\) is the discretization interval, \(C_i^j\) is the number of combinations from \(i\) elements by \(j\), which is calculated by the expression
\[
C_i^j = \frac{\binom{i}{j} (i-j)!}{j!}.
\]

It is known that the difference equation gives an adequate result with a small value of the discretization interval [7]. Controllers that perform regulation typically operate at short discretization intervals. It should be noted that in the library of the Simatic controller there is a PID controller Cont_C, which is called continuous. This provides a prerequisite for the legitimacy of the implementation of the transfer function using the difference equation. The solution of difference equation (4) has the form
\[
y[nT] = \frac{\sum_{i=0}^{m} B_i x[(n-i-k)] - \sum_{i=1}^{r} A_i y[(n-i)]}{A_0},
\]
the universal implementation of which in the STEP 7 environment on LAD, FBD and / or STL is not possible, since indexed addressing of array elements in these languages is not provided. For the software implementation of expression (6), we used the SCL language, which is convenient for working with arrays and is a part of the TIA library.
Table 1. The first program “Software for transfer functions implementation”

| Input          | Description                               |
|----------------|-------------------------------------------|
| m              | Int numerator order                       |
| n              | Int denominator order                     |
| Tau            | Real time-lag                             |
| T              | Real discretization interval              |
| Bf             | Array [0..10] of Real numerator coefficients (3) |
| Af             | Array [0..10] of Real denominator coefficients (3) |

| Output         | Description                               |
|----------------|-------------------------------------------|
| Bd             | Array [0..10] of Real input coefficients (4) |
| Ad             | Array [0..10] of Real output coefficients (4) |
| y              | Array [0..10] of Real sequential output values |
| InOut          | Array [0..20] of Real sequential input values |

FOR #j := 0 TO #n DO
#Ad[#j] := 0.0;
FOR #i1 := #j TO #n DO
#i := #j+#n-#i1;
"Block_2"(m:=#i,n:=#j,y=>#kom);
#Ad[#j] :=#Ad[#j]+(-1**#j)*#kom*#Af[#n-#i]*(#T*(#n-#i));
END_FOR;
END_FOR;
FOR #j := 0 TO #m DO
#Bd[#j] := 0.0;
FOR #i1 := #j TO #m DO
#i := #j+#m-#i1;
"Block_2"(m:=#i,n:=#j,y=>#kom);
#Bd[#j] :=#Bd[#j]+(-1**#j)*#kom*#Bf[#m-#i]*(#T*(#n-#i));
END_FOR;
END_FOR;
#kr := #Tau/#T+0.5;
#k := REAL_TO_INT(#kr);
#y[0] := 0.0;
FOR #i := 0 TO #m DO
#y[0] := #y[0]+#Bd[#i]*#x[#i+#k];
END_FOR;
FOR #i := 1 TO #n DO
#y[0] := #y[0]-#Ad[#i]*#y[#i];
END_FOR;
#y[0] := #y[0]/#Ad[0];
FOR #i := 1 TO #m+#k DO
#x[#m+#k-#i+1] := #x[#m+#k-#i];
END_FOR;
FOR #i := 1 TO #n DO
#y[##n-#i+1] := #y[##n-#i];
END_FOR;
The following are four programs. Table 1 shows the first program that implements expression (6). Table 2 a, b shows two programs: in the first column - the second program for calculating the factorial of a given number; in the second column - the third program for calculating the number of combinations from \( m \) elements by \( n \) (expression (5)). These programs are designed as functions (FCs). The fourth program, compiled in the LAD language, is designed to build a transient process for a given transfer function in real time (figure 1). It is necessary to note that, due to the impossibility of dynamic allocation of the controller memory [8], the presented programs are designed to implement transfer functions up to the 10th order with a delay of up to 20 T. If necessary, it is not difficult to introduce the corresponding changes.

Figures 2-5 show the transient characteristics calculated for various transfer functions using the above-described programs and the Simulink application of the Matlab package. The scales of the coordinate axes are the same. As can be seen from the figures, the transient characteristics practically coincide.
The four program

Ad  Array [0..10] of Real
Bd  Array [0..10] of Real
x   Array [0..20] of Real
y   Array [0..10] of Real

a)

| Name  | Data_type         | Offset | Start Value | Retain | Visible In | Setpoint | Comment |
|-------|-------------------|--------|-------------|--------|------------|----------|---------|
| b[0]  | Array [0..10] of Real | 8.0    | 40.0        | ✓      | ✓          | ✓        |         |
| b[1]  | Real              | 0.0    | 30.0        | ✓      | ✓          | ✓        |         |
| b[2]  | Real              | 8.0    | 20.0        | ✓      | ✓          | ✓        |         |
| b[3]  | Real              | 12.0   | 5.0         | ✓      | ✓          | ✓        |         |
| b[4]  | Real              | 16.0   | 0.0         | ✓      | ✓          | ✓        |         |
| e[0]  | Array [0..10] of Real | 52.0  | 400.0       | ✓      | ✓          | ✓        |         |
| e[1]  | Real              | 0.0    | 480.0       | ✓      | ✓          | ✓        |         |
| e[2]  | Real              | 8.0    | 294.0       | ✓      | ✓          | ✓        |         |
| e[3]  | Real              | 12.0   | 95.0        | ✓      | ✓          | ✓        |         |
| e[4]  | Real              | 16.0   | 17.0        | ✓      | ✓          | ✓        |         |
| e[5]  | Real              | 20.0   | 1.0         | ✓      | ✓          | ✓        |         |
| e[6]  | Real              | 24.0   | 0.0         | ✓      | ✓          | ✓        |         |

b)

Figure 1. The program, compiled in the LAD language and designed to build a transient process for a given transfer function in real time: a – description of datasets; b, c, d and e – structural diagrams of LAD-implementation.
Figure 2. Transient characteristics for transfer function $W(s) = \frac{5}{90s+1} e^{-5s}$: (a) – received according to programs 1-4; (b) – obtained using Simulink application of Matlab package.

Figure 3. Transient characteristics for transfer function $W(s) = \frac{5}{2000s^2 + 90s + 1} e^{-5s}$: (a) – received according to programs 1-4; (b) – obtained using Simulink application of Matlab package.

Figure 4. Transient characteristics for transfer function $W(s) = \frac{5}{250s^2 + 16s + 1} e^{-5s}$: (a) – received according to programs 1-4; (b) – obtained using Simulink application of Matlab package.

Figure 5. Transient characteristics for transfer function

$$W(s) = \frac{40s^3 + 30s^2 + 20s + 5}{400s^3 + 480s^2 + 294s^3 + 95s^2 + 17s + 1} e^{-10s};$$

(a) – received according to programs 1-4; (b) – obtained using Simulink of Matlab package.
3. Conclusions

The examples shown in figures 2-5 confirm the efficiency of the proposed method for software implementation of transfer functions of arbitrary order and delay in the TIA (Totally Integrated Automation) environment [9, 10]. Programs written in SCL and LAD are designed as functions. A program that implements the solution of a difference equation in SCL was developed. Another program, written in the LAD language, calculates the transient process for a given transfer function in real time. The created software blocks allow developing software for the study of fairly complex regulatory systems by simulating and adapting to real industrial conditions without much difficulty.

References

[1] Iserman R 1984 *Digital control systems: trans. from English* (Moscow: Publishing House “Mir”)
[2] Porshnev S V 2006 *Matlab 7. Fundamentals of work and programming. Textbook* (Moscow: Publishing House “Binom Press LLC”)
[3] Alekperli F A and Askerova S F 2018 *Bulletin of computer and information technology* 9 39-48
[4] Alekperli F A and Bayramova I S 2019 *Bulletin of computer and information technology* 8 3-14
[5] Alekperli F A and Bayramova I S 2019 *High performance computing systems and technologies* 3(1) 204-209
[6] Urmaev A S 1978 *Basics of modeling on analog computers* (Moscow: Publisher “Science”)
[7] Kuzin L T 1962 *Calculation and design of discrete control systems* (Moscow: Publisher “Mashgiz”)
[8] Siemens 2017 *Simatic Programming s7-1200 / controllers. System manual* p 1028
[9] Sujoldzic S and Watkins J M 2006 *Proc. of the 2006 American Control Conference* (Minneapolis, MN) pp 2427-2432
[10] Tan N 2005 *ISA Trans.* 44 213-223