DIAGRAME FOR A SOFTWARE PID

by
Valentin Casavela

Abstract.The software PID, shown below, is designated for to command an inverter, inverter used to vary the speed of an induction motor. The C++ program was descriebed in [3] and outs a byte on the parallel interface, that is the output data register. From this byte, six bits are used to command an interface, built in six channels, every channel commanding the gate of one of six IGBT-s. Every phase of the inverter needs two IGBT-s. The speed of the motor is read on the status register, of the same parallel interface.To enhance the performance, an algorithm with superior performance is adapted from the direct software Pulse Width Modulation (PWM) approach

Keywords.Pulse Width Modulation.Isolate Bipolar Transistors. Parallel interface.

Model for Software PID. The implementation of software PID control was performed in our laboratory. The beginning is the hardware PID, which scheme is designed in figure 1.

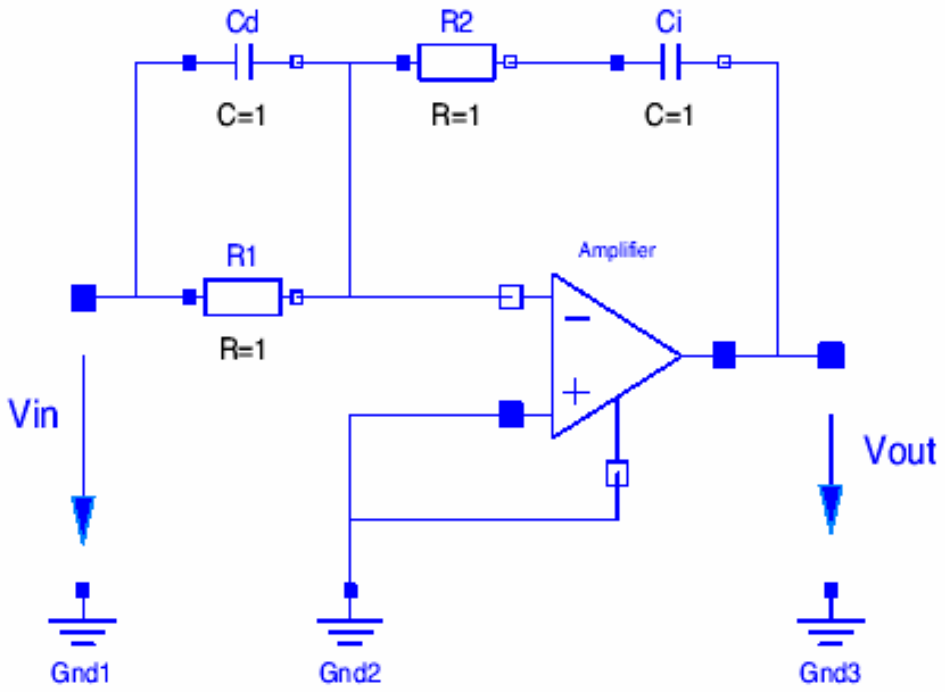


Fig. 1

Generally, the PID control has a potential ability to control the output of a process variable. If the control constants are chosen correctly, the variable approaches and maintains a desired value, its set point. This variable could be a temperature, a reactor conversion, the effluent conversion in a separation, etc. Our case is the speed of an induction motor.

These controllers can be implemented mechanically, by hard-wired electronics or within the software of a computer. Labview program may be used to accomplish control too. The purpose of this article is to show the general diagram, which may be implemented in Labview or in C++ software program. PID control starts by measuring the variable that is to be controlled. This is accomplished by an A to D (Analog to Digital) converter. This measures a voltage or current and outputs a numerical value to the computer, corresponding to this measurement. These measurements are typically 8,12 to 16 bits and are measured many times per second.

Based in the control operation, a voltage is set by the use of a D to A (Digital to Analog) converter. This sets the voltage or current to an electronic component, based on a variable chosen in software inside the computer program. The basic equation which reflects PID control uses the difference between a measured voltage, V_D, and the setpoint , S_P, to set a desired voltage, V_{out}. See fig1.

$$V_{in} = V_D - S_p$$

First, on calculate impedance:

$$Z_1(s) = \frac{R_1 * \frac{1}{C_d s}}{R_1 + \frac{1}{C_d s}};...Z_2(s) = R_2 + \frac{1}{C_I s};...H(s) = \frac{V_{out}(s)}{V_{IN}(s)} = -\frac{Z_2(s)}{Z_1(s)};.....$$

Finally,

$$H(s) = K_R\left(1 + \frac{1}{T_I s} + T_D s\right),....where:$$

$$K_R = -\frac{R_1C_1 + R_2C_2}{R_1C_2};...T_I = R_1C_1 + R_2C_2;..T_D = \frac{R_1R_2C_1C_2}{R_1C_1 + R_2C_2};...$$

$$V_{out}(s) = K_R\left(1 + \frac{1}{T_I s} + T_D s\right)V_{IN}(s);.....(1)$$

Leading that scheme, we'll obtain a software pid. First, the last expression above must be corrected with an anticipatory component $T_{\varepsilon,}$ which will introduce a

pole, near the zero, created by derivation time T_d. So, the excessive growing and noise, due the derivation, will be attenuated.

For our next proposes, it doesn't matter, the corresponding physical modification into figure 1. Expression 1 becomes:

$$V_{out}(s) = K_R\left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + T_\varepsilon s}\right)V_{IN}(s);.....(2)$$

The next step is passing to the digital form, using the sample theorem and Z transformation. For that, there are two possibilities: 1) rectangles method and 2) trapezium method.

1) For that,

$$e^{sT} = Z;..note..f(Z\} = \ln Z;...so,..f(Z) = f^{'}(z)(Z - Z_0) = \frac{1}{Z}(Z-1),..for..Z_0 = e^0 = 1;.$$

$$finally....s = \frac{Z-1}{T*Z}.....(3)$$

It will replace $"s"$, witch multiplies $T_D..and..T_\varepsilon$ in (2), and

2) Likewise, in trapezium method, $s = \frac{2}{T} * \frac{Z-1}{Z+1}$. It will replace $"s"$, witch multiplies T_I in (2).

T is sample time and is infinitesimal short. Trapezium method is used for slow variations, like integration (T_I), but rectangles method is used for speed variation, like derivation (T_D).

Now, *the digital transfer function* may be obtain:

$$D_{PID} = K_R[1 + \frac{T}{2T_I} * \frac{Z+1}{Z-1} + \frac{T_D}{T} * \frac{Z-1}{Z(1+\frac{T_\varepsilon}{T}) - \frac{T_\varepsilon}{T}}].....(4)$$

After some processing,

$$d_0 = \frac{K_R}{1+\frac{T_\varepsilon}{T}}(1 + \frac{T+T_\varepsilon}{2T_I} + \frac{T_D+T_\varepsilon}{T});...d_1 = \frac{K_R}{1+\frac{T_\varepsilon}{T}}(1 + \frac{T}{2T_I} - \frac{2(T_D+T_\varepsilon)}{T});...$$

$$d_2 = \frac{K_R}{1+\frac{T_\varepsilon}{T}}(\frac{T+T_\varepsilon}{T_I} - \frac{T_\varepsilon}{2T_I});.....c_1 = -\frac{T_\varepsilon}{T+T_\varepsilon});.....(5)$$

Our software PID regulator has input as *error* $e(k)=\Omega_P-\Omega_m$, where Ω_P is *imposed angular velocity* and Ω_m is *measured angular velocity*, for an induction motor.

The measurement for Ω_m is made with a transducer, a disk fixed on the motor ax, which disk has "n" halls and "n" full. Passing from a hall to a full determines a pulse, read by the system in a preset time. So Ω_m is measured.

Ω_P is the imposed angular velocity, by user program. To the inverter and, finally, to the motor comes, as a command, the *calculated angular velocity*: Ω_C, given by formula:

$$\Omega_C(K)=d_O e(K)+d_1 e(K-1)+d_2 e(k-2)+(1-c_1)\Omega_C(K-1)+c_1\Omega_c(K-2);.....(6)$$

d_I and c_I are given by (5) and e(k-i), Ω(K-i) are given by multiplying with z^{-I}

Ω_C is, in fact, V_{out} and "e" is V_{in}, in expression (2), but at discreet times.

$\Omega_C=2\pi*f_C$, where f_C is the resulting frequency for the stator rotating magnetic field.

The real (and measured velocity) is Ω_m, related Ω_c by expression:

$\Omega_m=(1-s)\ \Omega_C$

"s" is the slide for the induction motor. For according the software PID [2], with the system, formed by inverter-motor, the transfer functions of the last:

$K_s=\Omega_m/\ \Omega_C=1-s$,

must be equal with that given by (2) expression: V_{out}/V_{in} .

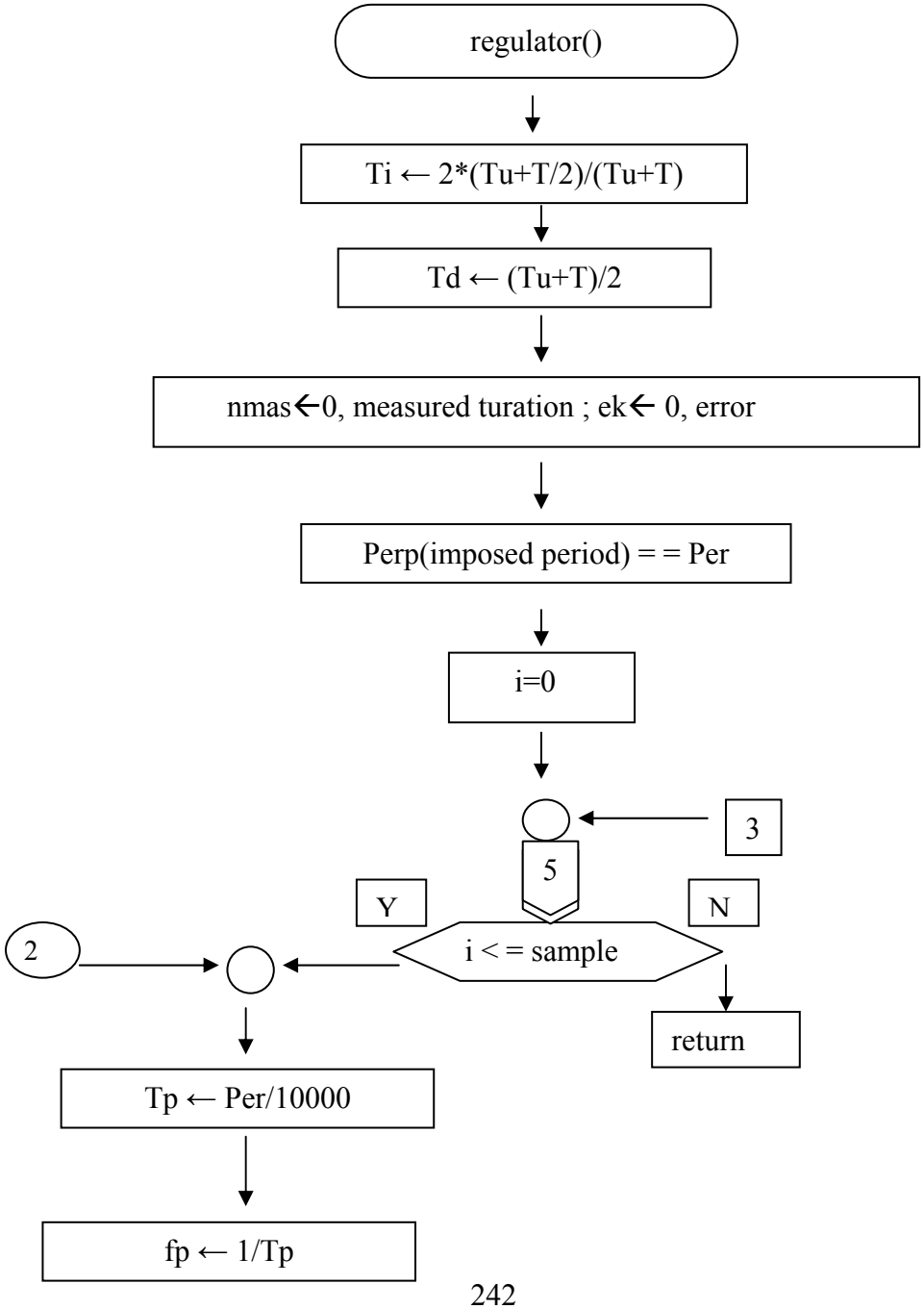
For the same purpose, the according, in (1) and (2) expressions, we must impose [2]:

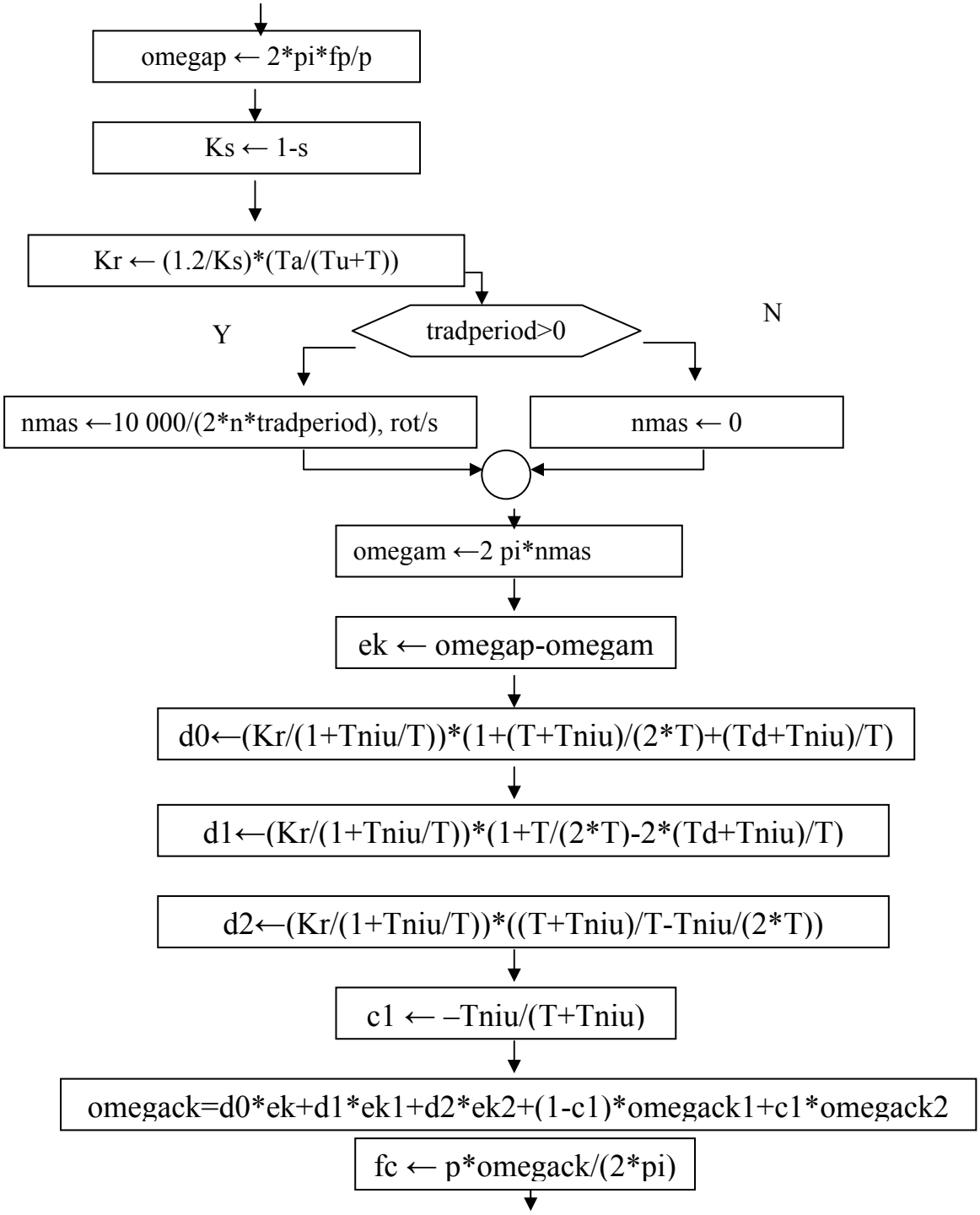
$$K_R=\frac{1.2}{K_S}*\frac{T_a}{T_u+T};.....T_I=2\frac{(T_u+\frac{T}{2})^2}{T_U+T};.....T_D=\frac{T_u+T}{2};.....(7)$$

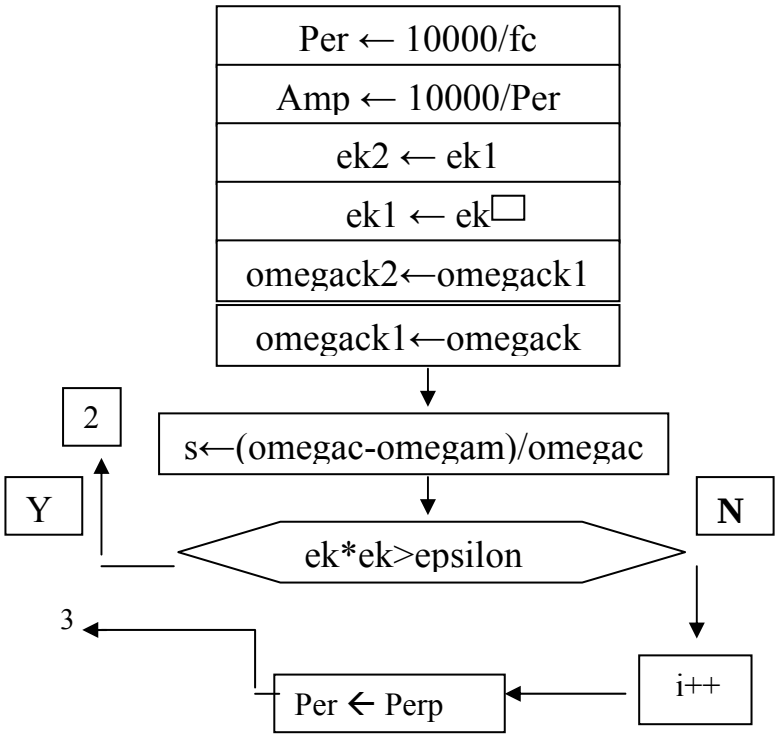
$T_u=L/R$, motor electrical constant (L,R—stator), $T_a=J\ \Omega_0/M_k$, motor mecanical constant (J-inertia moment , Ω_0-nominal velocity, M_k-nominal ax couple). Our case: $T_u=0.045$ s, $T_a=0$, $T_\varepsilon=0$.

Diagram for Software PID Implementation

The diagram is shown below.




omegap ← 2*pi*fp/p
Ks ← 1-s
Kr ← (1.2/Ks)*(Ta/(Tu+T))
Y
tradperiod>0
N
nmas ←10 000/(2*n*tradperiod), rot/s
nmas ← 0
omegam ←2 pi*nmas
ek ← omegap-omegam
d0←(Kr/(1+Tniu/T))*(1+(T+Tniu)/(2*T)+(Td+Tniu)/T)
d1←(Kr/(1+Tniu/T))*(1+T/(2*T)-2*(Td+Tniu)/T)
d2←(Kr/(1+Tniu/T))*((T+Tniu)/T-Tniu/(2*T))
c1 ← –Tniu/(T+Tniu)
omegack=d0*ek+d1*ek1+d2*ek2+(1-c1)*omegack1+c1*omegack2
fc ← p*omegack/(2*pi)



REFERENCES:

1) Teodor Pana: "Controlul sistemelor de actionare vectoriala cu motoare de inductie—Editura Mediamira, cluj-napoca, 2001

2) I. Cioc, C. Nica: "Proiectarea masinilor electrice"-Editura Pedagogica,1994

3) Valentin Casavela: P.W.M., Software P.I.D. and Command Transmission, Implemented in C++,for Induction Machine, International Conference of Automation, Quality and Tasting, Robotics, May 23-25, 2002, University of Girona.

Author:

Valentin Casavela, University of Petrosani, e-mai casavela@upet.ro, casavela@uvvg.ro