EasyChair Preprint

№ 5320

# PI Controller for Solving Simple Equations

Pasi Airikka



April 13, 2021

**Pasi Airikka***

# PI controller for solving simple equations

**Abstract:** PID control is a true workhorse for industrial process control. Its widespread use in various control applications is highly contributed by its astonishing property of removing a control error between setpoint and controlled variable. The very same feature, however, can also be utilized for other purposes where zero error is required. Solving an equation is good example of such dilemma. In this paper, the PI controller is presented as a solver e.g. for computing square roots, solving quadratic equations with real-valued solutions and linear equations of two variables.

**Keywords:** PID control, linear, equation, solution.

***Corresponding Author:** Senior lecturer, Tampere University of Applied Sciences, Tampere FINLAND, E-mail: pasi.airikka@tuni.fi

## 1 Introduction

A PID controller (Proportional-Integral-Derivative) is a backbone of industrial process control. Typically, it is implemented using a ready-made PID controller function block available on a process control system. And once commissioned, it constantly regulates a process variable as specified by removing the control error between setpoint and controlled variable if an integrating controller is used.

The PID controller has an integral controller for guaranteeing a zero error between setpoint and process output. The integral controller is based on computing an integral of the control error with respect to time. Receiving the control error as an input, it gives the time-integrated control error as an output. Combined with feedback, the simple mathematical operation of integration has an astonishing capability of driving its input inevitably to zero. The mathematical proof can be found e.g. in [6].

The amazing property of the integral controller can be used for driving any feedback signal to zero. As reported in [1], a PI controller can be used for identifying process model parameters or it might be used for matching true process measurements with simulated process model outputs as explained in [3] and [4]. In all these applications, the integrating controller plays a significant role in removing an error between two signals by feedback. One of the latest reported non-control related application of a PID controller is given in [5] where an integrating controller is used for searching prime number orders.

In this paper, the PI controller is first introduced as a calculator for computing some scientific functions such as square roots and logarithms. Then, the method of zero removal by the PI controller is extended for solving single variable quadratic equations with real roots. And, finally, the application of solving linear equations of two variables with real-valued solutions is explained. In all above cases, examples are given to highlight the way the PI controller acts as a numerical routine for solving equations or computing scientific functions.

It is worth pinpointing, that the methods given in this paper are far from being competitive alternatives for numerical routines to the existing, well-known and efficient methods for solving equations or computing functions. Numerical operations and computing time needed for computations is far too big for taking the proposed method seriously. Instead, the purpose of the paper is to show how the integral controller can be used for solving many kinds of problems, not only process control issues.

## 2 PI controller as a solver

A PI (Proportional-Integral) controller receives control error $e$ as an input and returns control $u$ as an output. The PI controller can be given as

$$u(t) = k_p(e(t) + \frac{1}{t_i}\int_0^t e(\tau)d\tau) \qquad (1)$$

with proportional gain $k_p \neq 0$ and integral time $t_i > 0$ for any time instant $t \geq 0$. The control error is a deviation between setpoint $r$ and controlled variable $y$

$$e(t) = r(t) - y(t) \qquad (2)$$

In process control, the controlled variable $y$ is typically an on-line measured physical variable such as volume flow, pressure, temperature, level or consistency. The setpoint $r$ is the targeted value for the measurement. The controller output $u$ is typically taken to a physical control device such as control valve or pump. Often, the

control is a percentage value in the range of 0–100 %.

In this paper, the PI controller does not act on a physical control device nor receives its feedback signal from any physical sensor or transmitter. Instead, the controlled variable $y$ is a cost function value for given controller input $u$. Thus, there is a true feedback closing the control loop and allowing a PI controller to act. The biggest difference to real PI controller applications is that there are no signals to be transmitted between sensors, computers and actuating devices. Instead, all the computation takes place in a computer.

Another significant difference is that there is no time-domain dynamics such as dead time or time constants involved as in true process control applications for regulating physical variables. The only relation between process input $u$ and output $y$ is static by nature but it can be linear or non-linear. When being the latter, it may unfortunately pose some difficulties on tuning and using PI controllers.

## 3 PI controller implementation

In distributed control systems, there is typically a function block for implementing a PID controller. If, however, some other platform is used, then some programming might be required to implement a PID controller for execution. In this paper, a simple discrete variant of an analogue PI controller is applied.

The PI controller (1) can discretized using a forward Euler approximation and given as

$$e_k = r_k - y_k \tag{3a}$$
$$u_k = k_p\left(e_k + \frac{1}{t_i} I_k\right) \tag{3b}$$
$$I_{k+1} = I_k + h \cdot e_k \tag{3c}$$

where a sub-index $k$ is an integer counter ($k = 1, 2, 3, \dots$) and $h$ is a sampling period or, in this context, a computation interval.

The PI controller can be initialized to $I_1 = 0$ but it is recommended to initialize it to a reasonable initial value for a faster convergence to a numerical solution. The reasonable initial value would be any non-zero value which would be closer to a final solution than pure zero.

In real applications, the discretized PI controller (3a-c) should be equipped with many practical features such as anti-windup but in this context those practicalities are neglected on purpose without compromising achievable results.

The PI controller always has a control direction which is either direct (positive) for $k_p > 0$ or reverse (negative) for $k_p < 0$ but cannot be both at the same time. This limitation needs to be carefully considered in solving numerical equations by a PI controller.

## 4 PI controller for computing simple scientific functions

The PI controller can be used for calculating many of the scientific functions such as square root, exponential functions and logarithms.

### Case 1. Square root.

Compute a square root $u = \sqrt{a}$ for a given real-valued $a > 0$. Squaring both sides of the square root gives

$$u = \sqrt{a} \;\Leftrightarrow\; u^2 = a \tag{4}$$

The process (function) to be controlled by a PI controller is $y = u^2$ with controlled variable $y$, controller output $u$ and setpoint $r = a$.

**Example**. For computing a square root $u = \sqrt{51}$, the function $y = u^2$ is plotted for $u = 0 \dots 10$ in figure 1.

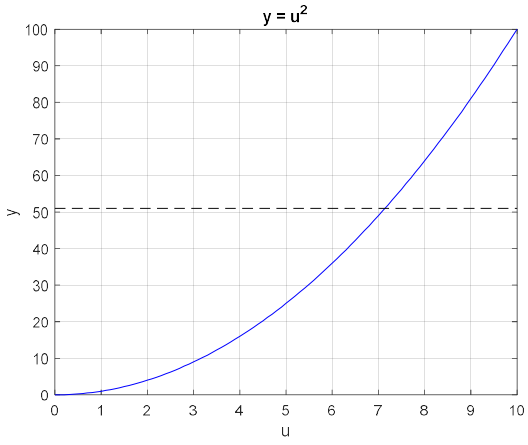


**Fig. 1.** Function $y = u^2$ for computing a square root $u = \sqrt{51}$.

The discrete PI controller (3a-c) with $k_p = 0.05$, $t_i = 1$, $h = 1$ is implemented for controlling the process $y = u^2$ with a setpoint $r = 51$. With $I_0 = 0$, the PI controller output converges to a solution $u \approx 7.1414$ by making the control error $e = r - y = 51 - u^2$ to zero in 21 iterations (solid in figure 2). For comparison, the integral term is initially set to $I_0 = \frac{t_i}{k_p} \cdot \frac{a}{10} = \frac{1}{0.05} \cdot \frac{51}{10} = 102$ providing with a faster convergence to a solution (dashed in figure 2).

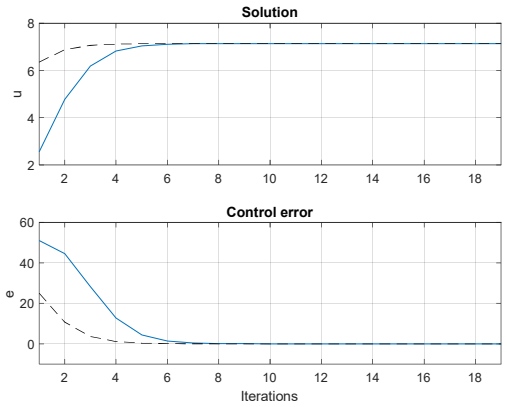


**Fig. 2.** PI controller responses for solving $u = \sqrt{51}$. *Upper*: PI controller output converging to a solution $u = \sqrt{51}$. *Lower*: control error $e = 51 - y$ converging to zero for $y = u^2$.

### Case 2. 10-base logarithm

Compute a 10-base logarithm $u = \log_{10} a$ for a given real-valued $a > 0$. By applying exponentiation on both sides of the logarithm gives

$$u = \log_{10} a \;\Leftrightarrow\; 10^u = a \qquad (5)$$

Now, the process (function) to be controlled by a PI controller is $y = 10^u$ with controlled variable $y$, controller output $u$ and setpoint $r = a$.

**Example**. For computing a 10-base logarithm $u = \log_{10} 51$, the function $y = 10^u$ is plotted in figure 3.

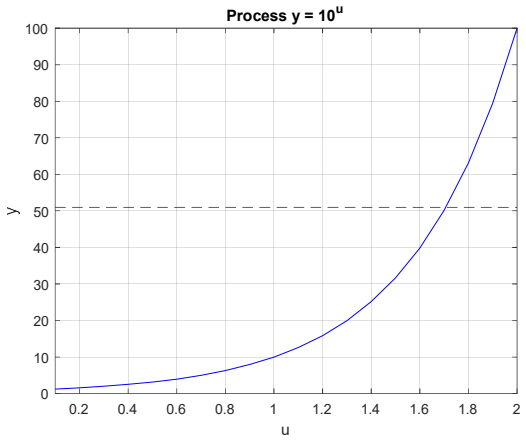


**Fig. 3.** Function $y = 10^u$ for computing a 10-based logarithm $u = \log_{10} 51$.

The discrete PI controller (3a-c) with $k_p = 0.01$, $t_i = 1$, $h = 1$ is implemented for controlling the process $y = 10^u$ with a setpoint $r = 51$. Without any initialization of $I_k$, the PI controller output converges to a solution $u \approx 1.7076$ by making the control error $e = r - y = 51 - 10^u$ to zero in 17 iterations (solid in figure 4). For comparison, the integral term $I_k$ is initially set to $I_0 = \frac{t_i}{k_p} \cdot \frac{a}{10} = \frac{1}{0.01} \cdot \frac{51}{10} = 501$ enabling a quicker convergence to a solution (dashed in figure 4).

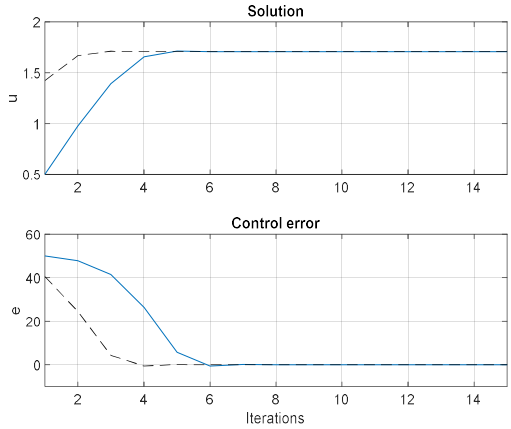


**Fig. 4.** PI controller responses for solving $u = \log_{10} 51$. *Upper*: PI controller output converging to a solution $u = \log_{10} 51$. *Lower*: control error $e = 51 - y$ converging to zero for $y = 10^u$.

## 5 PI controller for solving quadratic equations

The PI controller can be used for solving a quadratic polynomial

$$au^2 + bu + c = 0 \qquad (6)$$

with $a \neq 0$ for real-valued solutions (roots) of $u$ only, the condition of which limits the usage of a PI controller only for cases when $b^2 > 4ac$.

The process (function) to be controlled by a PI controller is $y = au^2 + bu + c$ with controlled variable $y$, controller output $u$ and setpoint $r = 0$.

**Example**. Solve (6) for $a = 1, b = -6, c = -16$. The condition of $b^2 = 36 > 4ac = -64$ is satisfied for existence of a real-valued solution for $u$. The polynomial is plotted for $u = -4 \ldots 10$ in figure 5. The roots of the polynomial are $u_1 = -2$ and $u_2 = 8$ and they are denoted by red circles whereas the turning point $-\frac{b}{2a}$ (minimum) of the polynomial is denoted by a red cross.

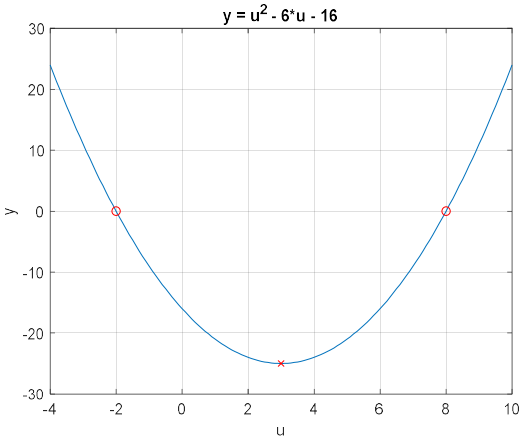


**Fig. 5.** Polynomial $y = u^2 - 6u - 16$ for solving $y = u^2 - 6u - 16 = 0$.

Due to the turning point of a polynomial (figure 5), the polynomial either decreases or increases for increasing $u$. However, the PI controller has a fixed, predetermined control direction. Therefore, two PI controllers are required, each assigned with one solution of the polynomial.

Two discrete PI controllers (3a-c) with $k_p = \pm 0.1$, $t_i = 1$, $h = 1$ are implemented for controlling the process $y = u^2 - 6u - 16$ with a setpoint $r = 0$. The other PI controller has a negative sign for the other solution. Both controllers are initialized to a turning point $u_0 = -\frac{b}{2a} = -\frac{-6}{2 \cdot 1} = 3$ from where they start to operate to opposite directions due to opposite proportional gain signs.

PI controllers' outputs converge to solutions $u_1 = -2$ and $u_2 = 8$ by making both control errors $e_i = r_i - y_i = 0 - (u_i^2 - 6u_i - 16)$ for $i = 1, 2$ to zero in 8 iterations (figure 6). The convergence is visualized with the polynomial function in figure 6 (right) by red circles.

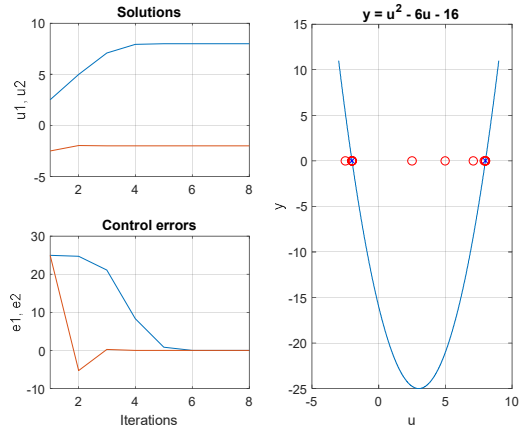


**Fig. 6.** PI controllers' responses for solving $u^2 - 6u - 16 = 0$. *Upper left*: PI controllers' outputs converging to solutions $u_1 = -2$ and $u_2 = 8$. *Lower left*: control errors during computation. *Right:* Polynomial with converging estimates (red circles).

The PI controllers could be used for solving polynomial of higher degrees, as well. However, the number of PI controllers required correspond the degree of polynomial as each controller is assigned for solving one root of the polynomial. In addition, turning points of the polynomial are to be computed initially making the method much too complicated for practical usage.

# 6 PI controller for solving a set of two linear equations

The PI controller can be used for solving a simple set of two equations with two unknown real variables $u_1$ and $u_2$

$$\begin{aligned} a_{11}u_1 + a_{12}u_2 &= b_1 \\ a_{21}u_1 + a_{22}u_2 &= b_2 \end{aligned} \qquad (7)$$

assuming the condition for the existence of the solution is satisfied. In linear algebra, the set (7) can be formulated using a matrix syntax

$$A\boldsymbol{u} = \boldsymbol{b} \qquad (8)$$

where $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, $\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ and $\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$. The condition for the existence of the solution can be formulated as $rank([A \mid \boldsymbol{b}]) = 2$ saying that the matrix has a full rank.

Now, there are two processes (functions) to be controlled with two controlled variables $y_i = a_{i1}u_1 + a_{i2}u_2 - b_i$, two controller outputs $u_i$ and two setpoints $r_i = 0$ for $i = 1, 2$.

As with any multivariable control application, the input-output pairing is a concern to tackle with for designing controllers. A method often used is based on a simple RGA analysis (Relative Gain Array) as given in [2]. The same method is recommended here for selecting appropriate input-output pairs for solving the linear equations using PI controllers (7).

**Example**. Solve a linear set of equations

$$\begin{aligned} u_1 + 9u_2 &= 70 \\ 2u_1 + 8u_2 &= 8 \end{aligned} \qquad (9)$$

The equation set (9) has a feasible solution as the matrix $A = \begin{bmatrix} 1 & 9 \\ 2 & 8 \end{bmatrix}$ has a full rank.

The RGA method with a matrix $RGA = \begin{bmatrix} -0.8 & 1.8 \\ 1.8 & -0.8 \end{bmatrix}$ recommends pairing $u_2$ with $y_1$ and $u_1$ with $y_2$ for process outputs $y_1 = u_1 + 9u_2 - 70$ and $y_2 = 2u_1 + 8u_2 - 8$. Both processes (functions) have positive gains (9 and 2) allowing to use the PI controllers with positive (direct) control directions.

Two discrete PI controllers (3a-c) with $k_p = 0.1$, $t_i = 1$, $h = 1$ are implemented for regulating the controlled variables $y_i$ with setpoints $r_i = 0$ for $i = 1, 2$. The controllers are initialized to zero $u_i = 0$.

The PI controllers' outputs converge to solutions $u_1 = -48.8$ and $u_2 = 13.2$ by making both control errors $e_1 = r_1 - y_1 = 0 - (2u_1 + 8u_2 - 8)$ and $e_2 = r_2 - y_2 = 0 - (u_1 + 9u_2 - 70)$ to zero in 211 iterations For visualization, only the first 50 iterations are plotted in figure 7.

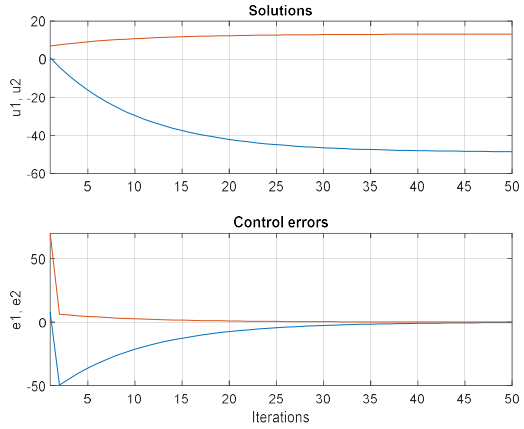


**Fig. 7.** PI controllers' responses for solving a set of two linear equations (9). *Upper*: PI controller outputs converging to solutions $u_1 = -48.8$ and $u_2 = 13.2$. *Lower*: control errors during computation.

# 7 Conclusion

The PI controller (Proportional-Integral) has an astonishing capability of eliminating a feedback error between targeted and measured variable. This amazing feature contributes to numerical integration of the control error in an integral part of the PI controller.

In process control, integral control is used for eliminating permanent errors between setpoints and measured, controlled variables. However, the method can be extended for other purposes as well. Literature shows that it has been used for model identification or searching prime numbers even. In this paper, the integral control has been used for solving simple scientific functions and equations.

A PI controller can be used as a solver for equations. However, computational effort and numerical operations required to set up a PI controller is a bit too much for practical use. In this paper, examples given were such that they are traditionally solved using algorithms and methods that do not require iterations. For example, an equation of second order has a well-known solution formula. Higher-order polynomial equations are efficiently solved using e.g. eigenvalue decomposition.

Most probably, to allow a fair comparison between traditional solvers and a PI controller, more complicated numerical problems requiring iterative approach should be introduced. Yet, increasing complexity in problems would necessarily demand more of the PI control design, the operation direction of a PI controller being just one of the issues to tackle with.

An interesting topic to study would be stability of the PI controller as a solver. During a vast number of simulations carried out for the problems presented in this paper alone, oscillations and even unstable performance was detected. Tuning parameters of a PI controller make a big difference even though there is no time-domain dynamics such as dead time and time constants involved.

An apparent reason for the stabilization issue of using a PI controller for solving equations is due to a non-linear nature of problems. Already a second-order polynomial has a non-linear element due to squaring of a variable. Nonlinearities might require nonlinear behavior or adaptation at least from a PI controller to work more reliably and without compromising stability.

# References

[1] Airikka, P. Continuous-time parameter identification using PI controllers. Nordic Process Control Workshop, Oulu, Finland, 2013.

[2] Bristol, E. H. On a new measure of interaction for multivariable process control. IEEE Transactions on Automatic Control, 1: 133–134, 1966.

[3] Friman, M., Airikka, P. Tracking simulation based on PI controllers and autotuning, 2nd IFAC conference on advanced PID control, Brescia, Italy, 2012.

[4] Friman, M., Airikka, P. Tracking simulation method, Patent US20130116802A1, 2010.

[5] Klán, P., Optimized integral controller for searching

prime number orders.

[6] Åström, K.J. and Hägglund, T. PID controllers: theory, design and tuning. page numbers. Instrument Society of America, United States of America, 1995.