Biomedical Engineering Letters
(DOI 10.1007/s13534-017-0049-9)

PID controller design to generate pulsatile flow rate for in vitro experimental studies of physiological flows

Mohammad Reza Najjari · Michael W. Plesniak

Received: 13 July 2017 / Revised: 6 September 2017 / Accepted: 21 September 2017

Abstract *Purpose* Producing accurate pulsatile flow rates is essential for many *in vitro* experimental studies in biofluid dynamics research. A controller system was developed to control a flow loop to produce easily adjustable pulsatile flow rates with sufficient accuracy.
Method An Arduino board is used as a micro-controller to control a pump to produce various pulsatile flow rates, and an open-source proportional-integral-derivative (PID) control algorithm is developed for this purpose. Four non-trivial pulsatile waveforms were produced by the PID controller, as well as an iterative controller, and the performance of both controllers was evaluated.
Results Both the PID and iterative controllers were able to successfully produce slowly-varying signals (single and multi-harmonic low frequency sine waves), but for high frequency signals where the flow has strong acceleration/deceleration (e.g. for physiological waveforms) the iterative controller exhibited significant undershoot.
Conclusions The comparison of PID and iterative controllers suggests that if the desired flow rate is a low frequency, simple waveform then the iterative controller is preferred due to simplicity of implementation. However, if the desired signal is rapidly changing and more complicated then the PID controller achieves better results. This system can be implemented in many flow loops due to its simplicity and low cost, and does not require a mathematical model of the system.

Keywords Feedback PID controller · Arduino · Artery blood flow · Common carotid artery · Gear pump

Mohammad Reza Najjari
Biofluid Dynamics Laboratory, The George Washington University, 800 22nd St. NW, Washington, DC, USA
E-mail: mnajjari@gwu.edu

Michael W. Plesniak
Department of Mechanical and Aerospace Engineering, The George Washington University, 800 22nd St. NW, Washington, DC 20052, USA
E-mail: plesniak@gwu.edu

1 Introduction

In vitro experimental studies of physiological fluid flows are important because of complexity, expense, and limitations in *in vivo* experimental studies. Physiological flows are mostly unsteady and hence a general controllable system that can produce any desired pulsatile flow rate, can be useful for many *in vitro* studies. The challenges in generating the desired flow rate are mostly due to limitations introduced by pump and working fluid rheology, and if the pumping system has the potential to produce the desired flow rate, then only a comprehensive controller is required to control the pump.
Several studies employed enhanced experimental facilities for producing pulsatile waveforms for fluid mechanical researches. Chaudhury et al [1] for example, provided a design for a piston pump which can produce high flow rates as high as 850 mL/s with low turbulence intensities. Plewes et al [2] introduced a semi-real-time, iterative control loop to produce a pulsatile pressure signal in an elastic vessel model. Tsai and Sava [3] increased the accuracy of their flow loop by combining a centrifugal and a piston pump to produce physiological pulsatile flow rates. The centrifugal and piston pumps were responsible for producing the mean and oscillatory components of the pulsatile flow, respectively. They did not use feedback control system and

the input voltage to their pump required hand tuning of the signal. Similarly, Chodzyński et al [4] took advantage of using one gear pump along with a piston pump to increase the capability of their flow loop with the addition of proportional-integral-derivative (PID) controllers with feedback to produce highly accurate pulsatile flow. However, they used a commercial controller package (CompactRIO NI9024) which is expensive and also may not be available to all researchers. Liou and Li [5] used a PI feedback controller in parallel with a feedforward controller to increase the accuracy of the control loop. They achieved good accuracy with their controller, but because their study was focused on flow in an aneurysm model they did not provide details of their control loop.
The main aim of this study is to provide a design for a robust and low cost control system to produce a desired pulsatile flow rate. The controller system can easily be implemented and is compatible with many flow loops, and also will be easily available to researchers at a modest cost (approximately about 100 times less expensive than the controller in [4]).

2 Experimental apparatus and technique

2.1 Flow loop

The fluid flow loop consists of two straight acrylic pipes with a length of 1.2 m and inner diameter of $d = 12.7$ mm and a 180° rigid curved tube [1]. The flow is produced by a gear pump (Ismatec model BVP-Z) which requires a 0–5 V signal to produce various flow rates. An ultrasonic flowmeter (Transonic ME-12PXL) is used downstream of the curved tube to observe the real-time flow rate. The flowmeter provides a 0–5 V signal equivalent to flow rate of ∼0 to 5 l/min. The working fluid can vary based on the experiment and usually is a mixture of 40% glycerin and 60% water (by weight percent) with added sodium thiocyanate or sodium iodide for refractive index matching.

2.2 Iterative controller

In vitro experimental studies of physiological flows usually require variable flow rates. This requires a control system that can provide an appropriate input to the pump to produce a desired flow rate. Plewes et al [2] used a simple iterative controller to produce a variable pressure in their flow loop. In this study, we used the same method to produce pulsatile flow rates. Figure 1

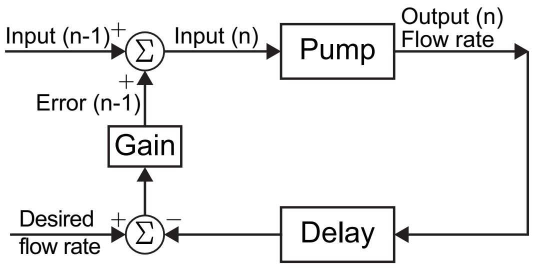

Fig. 1 Iterative controller method (modified from [2])

shows a block diagram of this semi-real time, iterative controller which is simple to implement by means of any data acquisition system (e.g. LabView, Matlab). In the first iteration, the input(n-1) and error(n-1) are set to be zero, which results in zero input to the pump. Iterations will automatically continue until the actual flow rate converges to the desired flow. To calculates the error between desired and actual flow rates, a delay function is used to shift the output in time to compensate the phase lag in the flow loop before calculating the error. The delay was found by a cross-correlation function. The gain affects the convergence speed — the larger the gain the faster the convergence, but the produced flow rate might have undesired over/undershoots. Therefore, smaller gains are usually preferred, which results in a smooth signal.

2.3 PID controller

The iterative method works well for linear systems. If there are non-linearities in the system or the iterative method fails to produce rapidly changing signals, then a real-time controller, i.e. PID controller, can be used to control the system. Here we provide a detailed explanation for implementing a PID controller using the feedback from the flowmeter sensor to control the pump. Equation 1 represents the PID equation, where the $V_{pump}(t)$ is the voltage sent to the pump and $e = v_d(t) - v_a(t)$ is the difference between desired flow rate (v_d) and actual flow rate (v_a, measured by flowmeter). K_p, K_i and K_d are the PID gains. In order to implement the PID controller, Eq. 2, the discrete form of Eq. 1, is used.

$$V_{pump}(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{1}$$

$$\begin{aligned} V_{pump}(t) = K_p e(t) + K_i \frac{\Delta t}{2} \sum_{t=0}^{t=T} (e(t) + e(t - \Delta t)) \\ + K_d \frac{e(t) - e(t - \Delta t)}{\Delta t} \end{aligned} \tag{2}$$

[1] This test section is mainly being used for particle image velocimetry experiments [6]

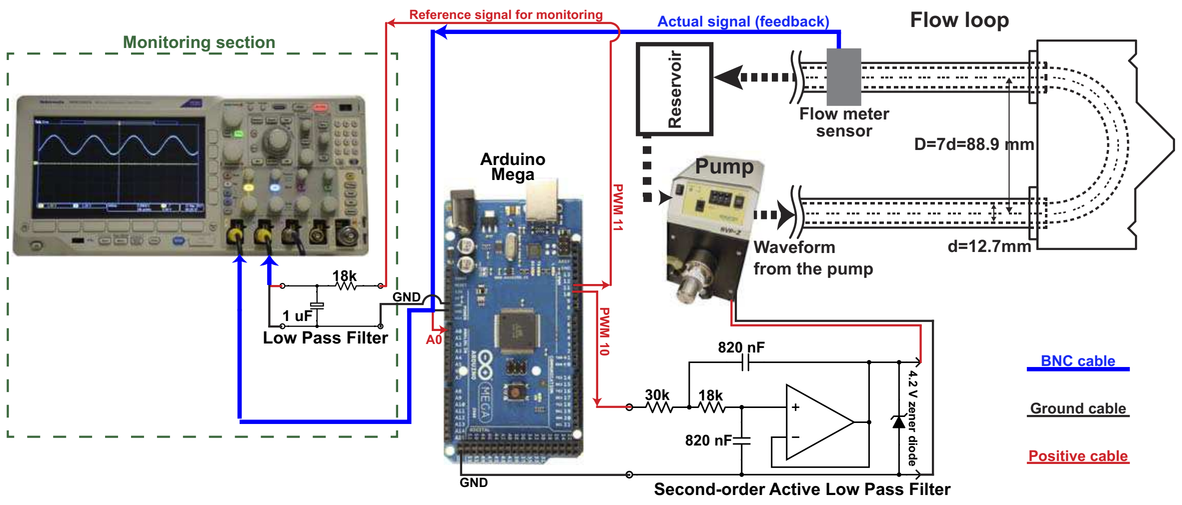

Fig. 2 Schematic of flow loop and electrical setup (scales are not real)

Here T is the waveform period and t is the discretized time and defined at $0 \leq t \leq T$.
An Arduino MEGA2560 micro controller (easily available for under 50 USD) was used to get input from the flow rate sensor, send output to the pump and perform control processing. The Arduino MEGA2560 is compatible with our flow loop setup because its operating voltage of 0–5 V matches the specifications of the pump and flowmeter sensor. The Arduino is a standalone board and does not require a computer to run. However, it does need a computer to upload the Arduino program to the board, which is done by the open-source, free Arduino software. In order to run the PID controller first the reference signal needs to be digitized (if it is not already). To do so, if the desired signal is available in image format, we suggest WebPlotDigitizer to extract the data followed by a curve fitting methods e.g. Fourier series to reconstruct the signal in uniform time-steps. If the desired waveform is available by an equation or an uniform time-step array then it can directly be used in the Arduino code[2]. An open-access Arduino PID controller program was developed by the authors to control the pump with high efficiency [7]. This code gets the reference signal and sends the required voltage to the pump to generate desired flow rate. However, this process needs the optimum values of PID gains.

2.4 PID gains

Selecting the appropriate gains is critical for optimum performance of the PID controller.If the desired flow rate is a simple sine wave with low frequency, or other slowly varying signal, then one set of gains for the entire period is typically sufficient. If the waveform consists of multiple frequencies with rapid accelerations and decelerations (e.g. our carotid artery waveform) then gain scheduling should be activated in the code. Gain scheduling enables the user to divide the desired waveform into multiple segments (e.g. acceleration, deceleration, stationary) and then set optimum gains for each segment separately. The gains in each segment can be constant or linearly varying. A combination of both fixed and variable gains within each segment is typically preferred to provide a smooth transition from one segment to another, and to eliminate sudden jumps in the actual flow rate at the boundary of each segment (see Arduino code for physiological waveform [7] for some examples). For gain tuning of the PID controller the following approach is used. First, a simple sine waveform is used to produce the desired flow rate. Proportional and integral gains (K_p, K_i) are initially set to zero, and the derivative gain (K_d) is set to a small value. Then K_p is gradually increased until the actual flow rate matches the desired waveform with minimal overshoot. The value of K_i is increased gradually until the offset between the magnitude of desired and actual flow rates is within an acceptable range. Afterwards, K_d is then increased until it damps high frequency oscillations. As K_i and K_d are increased, K_p must be decreased so that the system remains stable. After iteratively changing the gains, the optimum values can be found for a simple sine wave. To produce more complex waveforms, gains must be fine-tuned from their simple sine optimum values based on the shape of the desired signal. Experience with the apparatus is necessary to achieve optimum fine-tuning. For gain tuning of a waveform that requires gain scheduling, the aforementioned

[2] Due to limitations of the Arduino board SRAM memory the array usually cannot have more than about 1500 elements in it, or it should be stored in the program memory

steps to determine optimum gains should be followed first. Then, the waveform should be divided into proper segments, wherein all gains in all segments should initially be set to their baseline optimum values and then fine-tuned to achieve the desired flow rate waveform. The control code was equipped with a feature (looking ahead in the desired flow rate) to overcome the delay inherent into the flow loop system. This parameter also needs to be tuned for different settings which mainly accounts for a constant phase lag between actual and desired flow rates.

2.5 Experimental setup

Figure 2 shows the schematic of the experimental setup. The voltage sent to the pump was supplied through the PWM (pulse width modulation) pin on the Arduino which is similar to a digital signal with either 0 or 5 volts with different duty cycles. Therefore, if the PWM pin directly supplies power, the pump will have many undesirable oscillations. A second-order low pass filter (Fig. 2) was used to smooth the supplied voltage. Also an op-amp provides an active filter which increases filter reliability. The values of capacitors and resistors in the filter define the cutoff frequency, which is about 8.35 Hz for the shown values. These values can be chosen based on the highest required frequency in the reference signal. The pump power supply circuit was also equipped with a 4.2 V zener diode which limits the supplied power below 4.5 V to protect the pump from excess voltage. For observing the reference signal on an oscilloscope, another simple low-pass filter was used.

3 Results and discussion

For demonstrating the efficiency of the designed control system, four different nontrivial flow rates with Womersley numbers ($\alpha = \sqrt{\frac{2\pi f r^2}{\nu}}$) of 4.2–8.4 are produced using PID and iterative controllers. Figure 3-a shows that the PID and iterative controllers can both produce the simple 0.5 Hz sinusoidal flow with high accuracy. To produce this 0.5 Hz ($\alpha = 5.94$) *sine* flow rate, one set of gains were used in PID controller for the entire cycle. In order to produce a more rapidly varying signal (1.0 Hz *sine*) using one set of gains in PID for the whole cycle was impractical, so the waveform was divided to four sections and the K_i was changed gradually to produce the desired signal (Arduino code for *sine* signal [7]). Figure 3-b shows that gain-scheduling helped to produce the 1 Hz ($\alpha = 8.4$) signal despite some deviation from the reference flow. Because the main aim of the current study is to provide the desired flow for study of secondary flow morphology in a 180° curved pipe, these small deviations from the magnitude of reference signals will have negligible effect on the secondary flow as long as the phases match. The iterative controller, however, has a strong undershoot and a large phase lag around t=0.8 (s) which makes it unsuitable to produce this flow rate. Figure 3-c shows multi-frequency flow rate (i.e. summation of three *sine*s with 0.5, 1, and 1.5 Hz frequencies). Both PID (with gain-scheduling) and iterative controllers produced this 0.5 Hz signal with high accuracy. The common carotid artery flow rate [8] with 0.25 Hz frequency and $\alpha = 4.2$ is shown in the Fig. 3-d. This reference flow can be produced accurately by 30 modes of Fourier series and gain-scheduling was used in PID controller to generate the actual flow accurately. Similar to 1 Hz sine flow, the iterative controller had a strong undershoot near t=1.3 (s) and cannot generate this critical section of the flow rate accurately.
Therefore, if the desired flow rate has low frequency, the iterative controller is preferred due to its simplicity. But to generate higher frequency waveforms, using a PID controller is necessary. For investigating pulsatile flow morphologies, the real time controller might not be desired due to some slight cycle to cycle variations. Therefore, once the optimum gains are found and the accuracy of the PID controller to generate the required flow rate is established, the controller output (voltage sent to the pump) is saved from the serial command in Arduino program. Then, this signal can be directly sent to the pump without a control loop. It is important to note that, writing in Arduino serial command will slow down the controller loop speed and might lead to an undesired results. The controller loop speed should then be monitored to ensure that the loop is running fast enough with respect to the resolution of desired flow rate (see the comments in the Arduino code). An alternative way to record the PID output signal is to use a data acquisition system while the real-time PID is working. It should be kept in mind that recording the output of real-time PID, might lead to a signal attenuation, which may be resolved by an operational amplifier circuit.

4 Summary and conclusion

This paper provides a design and operation protocols for a real-time PID control system that controls a physiological flow loop with applications in fluid mechanics studies that require variable flow rates. Additionally, a semi-real-time, iterative controller method is evaluated to generate pulsatile flows. Four sets of pulsatile

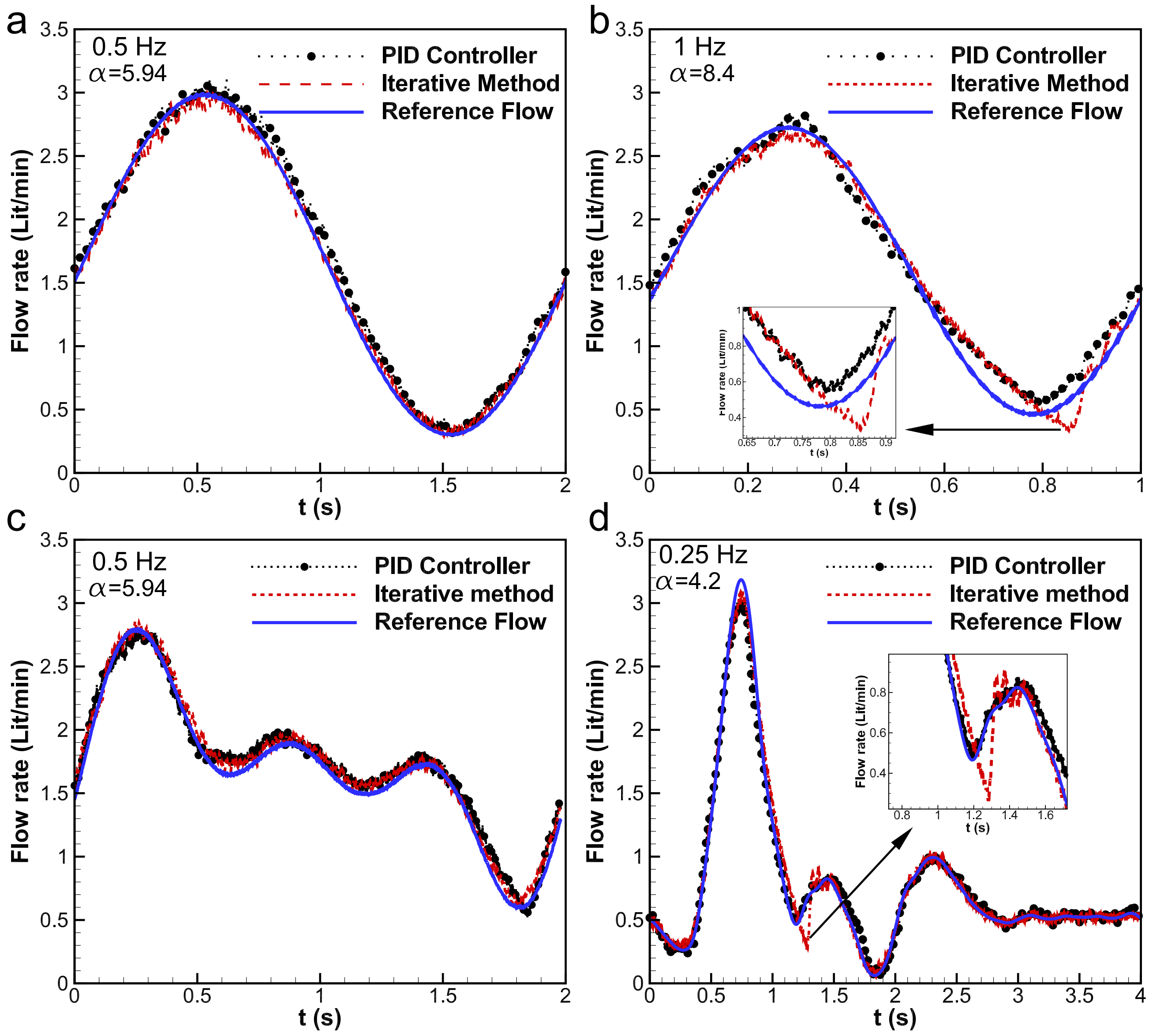

Fig. 3 Four desired (reference) flow and instantaneous actual flow rates produced by PID and iterative controllers. The Womersley number (α) is shown in each figure and the insets show regions where the iterative controller has large undershoot

flow rates were produced using PID and iterative controllers to demonstrate the feasibility of the method. If the desired flow rate is a low frequency (slowly varying) signal, both PID and iterative controllers can produce accurate results, in which case the iterative controller is preferred due to simplicity. But in case of a flow rate with high frequency and large amplitude variation, the PID controller is preferred because the iterative method cannot produce accurate flow rate; it exhibits strong undershoot and large phase lag.

The introduced PID controller method is robust and can be applied to a variety of flow loops to achieve different pulsatile flows. One example of an application to carotid artery flows was provided in this paper. Due to limitations imposed by the pump and flow momentum, the experimental apparatus was not able to achieve more rapid decelerations/accelerations than depicted in Fig. 3-d or much higher flow rates. This is not a limitation of the controller system, but rather a physical limitation that can be overcome by choosing an apparatus with the required physical performance characteristics. For example, to produce rapidly decelerating flow, a controllable needle valve (to stop flow impulsively) or a three-way valve (to direct a portion of flow through a short circuit loop) would increase the overall system capability. If higher flow rate is required, a higher capacity pump would be chosen. Appropriately selected experimental equipment will enable the PID controller

to produce various pulsatile flows found in the vasculature, e.g. in aortic and femoral arteries. The application of the described PID method is also not limited to controlling flow rates; it can be used for controlling other parameters such as time-varying pressure, or any other signal. An appropriate feedback signal is necessary, e.g. the output from a pressure transducer, with sufficient accuracy and sufficiently fast response time.
The PID controller system described herein is easily available to the community because of the low cost (total of under 80 USD, not including the flowmeter) and simplicity of setup. Also, all of the Arduino scripts are available as an open-source code, which can be used and modified by researchers. The importance of determining optimally tuned gains to maximize accuracy of the PID controller cannot be overemphasized. Tuning is a time-intensive, iterative process, but it is time saving in long run and well worth the initial time investment.

Acknowledgements This study was supported by the George Washington University Center for Biomimetics and Bioinspired Engineering. The authors would also like to thank Mahdis Bisheban and Kanishke Gamagedara for their contributions.

Conflict of interest

All the authors declare that they have no conflict of interest in relation to the work in this article.

References

1. R.A. Chaudhury, V. Atlasman, G. Pathangey, N. Pracht, R.J. Adrian, D.H. Frakes, Cardiovascular engineering and technology **7**(2), 148 (2016)
2. D.B. Plewes, S.N. Urchuk, S. Kim, I. Soutar, Medical physics **22**(7), 1111 (1995)
3. W. Tsai, Ö. Sava, Medical & biological engineering & computing **48**(2), 197 (2010)
4. K.J. Chodzyński, K.Z. Boudjeltia, J. Lalmand, A. Aminian, L. Vanhamme, D.R. de Sousa, S. Gremmo, L. Bricteux, C. Renotte, G. Courbebaisse, Biomedical engineering online **14**(1), 77 (2015)
5. T.M. Liou, Y.C. Li, Journal of biomechanics **41**(6), 1174 (2008)
6. M.R. Najjari, M.W. Plesniak, Experiments in Fluids **57**(6), 1 (2016)
7. M.R. Najjari, M.W. Plesniak. Pump Controller program to produce Pulsatile flow rate (2017). DOI https://doi.org/10.5281/zenodo.821599. URL https://github.com/mrnajjari/Pump-PID-Controller-Pulsatile-flow
8. D.W. Holdsworth, C.J. Norley, R. Frayne, D.A. Steinman, B.K. Rutt, Physiological measurement **20**, 219 (1999)