Medical Engineering and Physics 000 (2017) 1–5

Contents lists available at ScienceDirect

Medical Engineering and Physics

journal homepage: www.elsevier.com/locate/medengphy

Technical note

Arduino control of a pulsatile flow rig

S. Drost[a], B.J. de Kruif[b], D. Newport[a,*]

[a] Bernal Institute, School of Engineering, University of Limerick, Limerick, Ireland
[b] Design Factors, Faculty of Science and Engineering, University of Limerick, Limerick, Ireland

ARTICLE INFO

Article history:
Received 12 December 2016
Revised 15 September 2017
Accepted 8 October 2017
Available online xxx

Keywords:
Pulsatile flow pump
In-vitro
Hemodynamics
Arduino
Feed forward control

ABSTRACT

This note describes the design and testing of a programmable pulsatile flow pump using an Arduino micro-controller. The goal of this work is to build a compact and affordable system that can relatively easily be programmed to generate physiological waveforms. The system described here was designed to be used in an in-vitro set-up for vascular access hemodynamics research, and hence incorporates a gear pump that delivers a mean flow of 900 ml/min in a test flow loop, and a peak flow of 1106 ml/min. After a number of simple identification experiments to assess the dynamic behaviour of the system, a feed-forward control routine was implemented. The resulting system was shown to be able to produce the targeted representative waveform with less than 3.6% error. Finally, we outline how to further increase the accuracy of the system, and how to adapt it to specific user needs.

1. Introduction

In-vitro experiments are a useful tool in hemodynamics research, offering a wide range of applicable experimental techniques, combined with good accuracy and reproducibility. To simulate physiologically realistic pulsatile flow, a dedicated pumping system is used in these experiments. This can either be a mock loop representing the full circulatory system [8,9], or a pump that reproduces a waveform measured locally in-vivo [4–7,12].

For both cases, pumps are commercially available. However, researchers often employ in-house built systems, for example because of cost considerations, or to add specific functionality. Examples of pumping systems that were developed to reproduce physiological waveforms are a gear pump driven by a stepper motor or servomotor, the rotation rate of which is controlled by a microcomputer [5] or personal computer [7], a controlled piston pump [4], or a combination of both: a gear pump to deliver the steady flow component, and a piston pump to generate the pulsatile component [12].

With so many options already available, the goal of our current work is not so much to develop a superior system as to design a system that is affordable and relatively simple to set up, using off-the-shelf components. With this in mind, the full source code for the control of the pump is included as a supplement to this note.

The system that is presented in this technical note is designed to reproduce a waveform typical for an arterio-venous fistula for hemodynamics, more specifically, a waveform with a mean flow of $\mathcal{O}(1)$ l/min, a frequency content up to 4 Hz, and a pulsatility index (peak-to-peak flow over mean flow) of 0.4. We consciously chose to use the simplest possible control strategy, first identifying the system, and subsequently using a feed-forward controller in combination with a run-to-run controller to achieve the target waveform. The present system is aimed to be used in an in-vitro flow rig to test different modalities for vascular access for hemodynamics, but with suitable parts selection, the approach we used is expected to be useful for a wider range of cardiovascular flows as well.

2. Materials and method

2.1. Set-up

The experimental set-up is shown schematically in Fig. 1. The flow loop that was used for the experiments consists of a straight poly(methyl methacrylate) (PMMA) tube with an inner diameter, ID, of 5 mm and a length of 500 mm, connected to an open reservoir ($\pm$500 ml) and to the pump by flexible polyethylene tubing (ID 6 mm). This is the simplest possible set-up to study fully developed steady or pulsatile laminar flow, up to a Reynolds number of 2000, while the ID of the straight tube is representative for vascular access. Tap water at room temperature was used as a test liquid. An in-line flow sensor (Transonic TS410 ME 6PXN), placed around 50 cm from the pump outlet, was used to monitor the flow rate. Its signal was both sent directly to the Arduino micro-

* Corresponding author.
E-mail address: david.newport@ul.ie (D. Newport).

https://doi.org/10.1016/j.medengphy.2017.10.006
1350-4533/

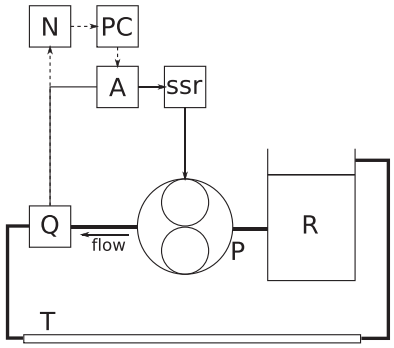

Fig. 1. Experimental set-up (not to scale): Flow loop, consisting of reservoir (R), flexible tubing, perspex tube (T), Transonic flow sensor (Q), and gear pump (P). Control loop, consisting of an Arduino (A) with a Solid State Relay (SSR). For monitoring and identification a data acquisition card (N) and PC are used.

controller (see below), and to a data acquisition card (National Instruments, NI USB-6001), which in turn was read out using Matlab.

The pumping system consists of a gear pump (excess from the medical manufacturing and spare parts for a Gambro C3/CS3 dialysis machine, equivalent to a Diener Extreme Series 1000 ml/min with 9 mm gear set), magnetically driven by a core-less DC motor (Premotec, 30 V DC). A 24 V DC power supply is used, in combination with a solid-state relay (SSR) to control the voltage supplied to the motor. Even though, when used in the prescribed range of rotation rates, gear pumps produce negligible ripple, care was taken to place the flow sensor sufficiently far from the pump outlet to not be affected by any possible ripple.

An Arduino micro-controller (Genuino MEGA 2560) sends a pulse width modulation (PWM) signal to the SSR to achieve the required waveform. The Arduino has a resolution of 8 bits, which means that the PWM signal can be set to integer values between 0 and 255, to achieve a motor voltage between 0 and 24 V (i.e. the duty cycle of the PWM signal determines the voltage output of the power supply). As the power supply only delivers positive voltages, this setup results in a one quadrant controller that can only apply a positive torque with a positive velocity.

To be more flexible during the design phase of the system, we used a PC with Matlab to send the calculated pump voltages to the Arduino, and to monitor the waveform generated by the pump. To emphasise that this communication only happens during the design phase, the connection in Fig. 1 is dashed. Arduino offers the option of a Secure Digital (SD) card module, so that eventually the waveform can be stored on an SD-card and the Arduino can operate as a stand-alone controller (the flow sensor signal should still be read out directly by the Arduino in that case, to maintain the run-to-run controller functionality as discussed in Section 2.2.2).

The total cost of the system (excluding the Transonic system) was approximately € 350 (≈ $ 393: ± € 250 for the pump and motor, € 50 for the Arduino, and € 50 for the remaining parts), which is considerably less than the typical cost for a commercial system (± $ $\mathcal{O}(10^4)$, estimate based on personal communication with suppliers).

2.2. Method

The targeted velocity waveform for our experiments is representative for the waveforms encountered at the proximal arterial side of an arterio-venous fistula used for hemodialysis [2,3,10,11]. It is constructed as a Fourier series prescribing the flow rate, $Q(t)$, in l/min:

$$Q(t) = \sum_{n=-N}^{N} c_n \exp\left(i\frac{2\pi nt}{T}\right), \tag{1}$$

Table 1
Coefficients for Fourier series to construct waveform (c_n^* denotes the complex conjugate of c_n).

n	c_n
0	0.9
1	−0.015−i
2	−0.036 − 0.03i
3	$-6\cdot10^{-3}+1.2\cdot10^{-3}$i
4	$-0.012+6\cdot10^{-4}$i
c_{-n}	c_n^*

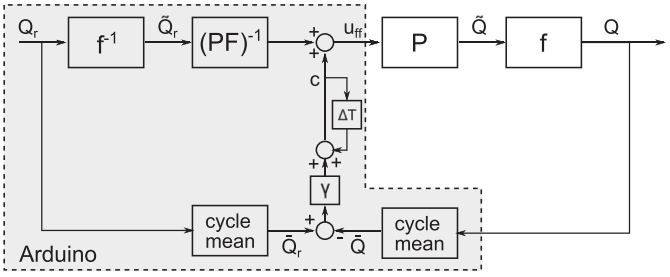

Fig. 2. Control scheme of the system, consisting of the linear transfer function of the system (P), a low pass filter (F), static non-linear output function (f), a constant regulating the run-to-run controller (γ) and a delay of one pulse cycle (ΔT). The signals are defined in the text.

with a period, T, of 1 s, $N = 4$, and coefficients, c_n given in Table 1 (for a graphical representation see Fig. 6).

To be able to reproduce this waveform, the non-linear and dynamic behaviour of our system are identified, and a controller is designed based on this.

2.2.1. System identification

The system is considered as a combination of a linear dynamic system with a static non-linear output function. The equations that describe this behaviour are:

$$\tilde{Q} = P(s)u, \quad Q = f(\tilde{Q}). \tag{2}$$

In this, $\tilde{Q}$ denotes a virtual flow (in arbitrary units), while Q represents the true flow in litres per minute, s is the Laplacian variable, and P the linear transfer function of the system. The input u is the voltage to the pump. Although a first principles physical model is not made of the system, the non-linearity expected to be dominant is the flow – gear wheel relation due to back flow ("slip": not all fluid on the intake side is transported to the outlet side by the gears. This effect is particularly noticeable at higher differential pressure, and for low viscosity working fluids).

In order to identify the relation in (2) two dedicated experiments are performed. The output non-linearity is identified by applying a series of different input voltages to the pump and measuring its steady state output. A smooth function is fitted through these measurement points, minimising the weighted sum-of-squares error. The weighting in this fitting procedure is done to optimise the fit in the range of flow rates present in the target waveform.

To identify the transfer function, a set of step functions with different amplitudes is applied. Based on the response, the Matlab function 'tfest' is used to find the linear dynamic response. The number of zeros and poles is varied to get a minimal system with good response.

The system identification procedure is carried out semi-automatically, by running first the Arduino script `Identify.ino` and the Matlab script `nidaq_identify.m` (to control the pump and read the resulting flow rates, respectively), and then the Matlab script `IdentifyPump.m` for the actual identification (see supplementary material for the code).

Please cite this article as: S. Drost et al., Arduino control of a pulsatile flow rig, Medical Engineering and Physics (2017), https://doi.org/10.1016/j.medengphy.2017.10.006