Assistive Writing device for Parkinson's patient.

Member names: Jagatpreet Singh Nir, Mingda Ju, Xuyang Sun, Zimou Gao

March 7, 2024

1 Introduction

1.1 Parkinson's disease

Parkinson's disease (PD) is a long-term degenerative disorder of the central nervous system that mainly affects the motor system. The symptoms generally come on slowly over time. Early in the disease, the most obvious are shaking, rigidity, slowness of movement, and difficulty with walking [**1**]. Thinking and behavioral problems may also occur. Depression and anxiety are also common, occurring in more than a third of people with PD [**2**]. Other symptoms include sensory, sleep, and emotional problems. In 2015, PD affected 6.2 million people and resulted in about 117,400 deaths globally [**3**, **4**].

1.2 Our goal

Through the understanding of the disease, we see that the daily life of Parkinson's patients has been greatly affected. They are suffering from both physical and psychological attacks. Therefore, we decided to design a kind of wearable device that assists them to write. The device's function is to cancel out the tremors of Parkinson's patients and stabilize their hands for writing task, hoping to alleviate some problems in their lives.

1.3 Overview of this report

In section 2, we state the problem we are going to solve. In section 3 we discuss the requirements and methodology of the device. In section 4, a mathematical framework and a methodology for design and development of the wearable device are presented. In section 5, we present our simulation and experimental results. Finally, in section 6 we present a conclusion and future work that needs to be done.

2 Motivation and Problem statement

2.1 Motivation

Our motivation to design this product comes from two aspects. The first aspect is the huge demand. PD affects approximately seven million people globally and one million people in the United States. [**5**, **6**, **7**] The number of new cases per year of PD is between 8 and 18 per 100,000 person–years. [**8**]. Therefore, such high incidence rates will inevitably cause a large burden to the patient's families and to the society as a whole. The second aspect is the economic factor. Although there are current devices that exist in the market and reported by media to help Parkinson's patients, their high price has discouraged many patients from buying them. So, we aim to build a wearable device using open source platforms and simple components, so that the device can be made available to a more number of patients.

2.2 Analysis of the writing action of normal people and Parkinson's patients

In order to understand the difference between a normal human being and a parkinson's patient, we imitated writing activities of the Parkinson patients and the normal's. During this experiment we placed an EMG sensor on different area of arms to detect muscular activities data. The figure 1 shows that, specific muscles near the forearm and wrist are activated which produce the shaking hand movement. A patient having parkinson's disorder exhibits such muscle activity whenever they write and it is not in

their control.

For the normal people, when tremors occur during writing, they get the visual feedback from their eye,

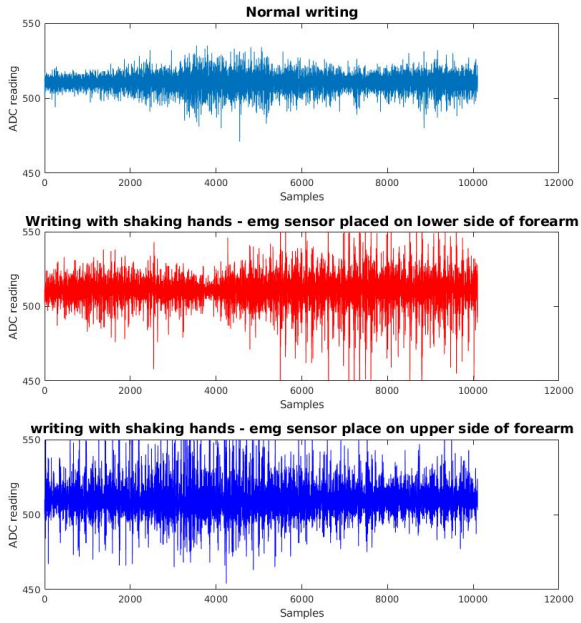


Figure 1: Normal people Parkinson patients

setting up a negative feedback loop that can control their shaking.

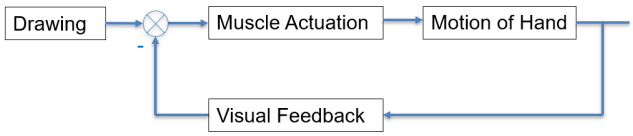


Figure 2: Normal people

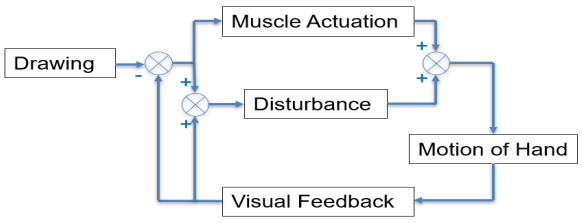


Figure 3: Parkinson's patient

But for the patient with a PD, the visual feedback does not help them. In fact, when patients perform any activity such as writing or sketching, involuntary muscle activities start to shake the hand. Getting visual feedback of shaking hands, the patients really want to control the disturbance, but this kind of behavior sets up a positive feedback loop which will continuously amplify the hand tremors of patients, intensifying the hand movements, making it very difficult for the patient to write.

If we can cancel the involuntary movement produced by the patients' forearm, we can help them to write. So, our team's problem statement is :

Develop a wearable device to cancel the movements caused by tremors while writing in a patient with Parkinson's disorder.

3 Methodology and Design Requirements

3.1 Methodology

This section introduces the system components of the device and converts the contents of the previous block diagram into the form of mathematical expressions.
Our approach models the patient's forearm as a one link pendulum. The wrist of the forearm is the free end of the one link pendulum. The wearable device is mounted at the wrist. So, in our case, it will be placed at the top of the pendulum. The patient tries to control muscles voluntarily and in this attempt creates a involuntary tremors which are modeled as disturbance.

$$\theta = f(x) + U_{voluntary} + D_{forearm} - S_u \tag{1}$$

$f(x)$ is the dynamic model of the forearm
$U_{voluntary}$ is the torque exerted by the patient to control the hand and forearm.
$D_{forearm}$ is the disturbance torque generated from other muscles of the patient.
S_u is the item we want to calculate to cancel out the disturbance.

3.2 Requirements

(a) The device cancels out the tremors of the patient to make motion of hand stable.

(b) The device must be light weight. It should not interfere with normal function of the limb.

(c) The device should be easy to wear and easy to switch on/off.

(d) The device must be operated on battery power, it needs to be energy-efficient.

4 Math Works and Model Physical Process

In this section, we will introduce two physical models we have tried in this semester. One is the Acrobot model, the other is one link pendulum model.

4.1 Acrobot Model

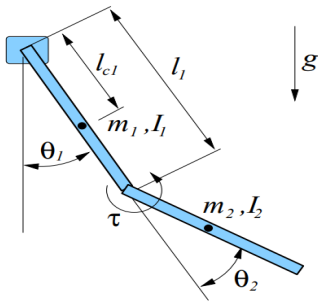


Figure 4: The Acrobot Model

We started by assuming human arm and hand as an Acrobot model because this model is extremely similar to the configuration of human's arm. We are considering a 2D planar case. The link 2 in the serial chain is our hand, and link 1 in the serial chain is the forearm. The numbering starts from the link attached to the ground in figure 4. We consider the upright position is our desired position, any deflection of the desired position will be pulled back by an actuator (modeling our wearable) which is

mounted at the second joint(wrist joint). In our case, when people write, people's arm is fixed on the desired position. Parkinson's tremors is like deflection of the desired position. [**9**].

First, we will demonstrate the typical Acrobot model which simplifies the model and emphasizes the most relevant parts. In the below figure, the Acrobot is a planar two-link robot arm in the vertical plane which works against gravity with only an actuator at the second joint (It is in the position of joint τ).

4.1.1 Dynamic modeling

Figure 4 illustrates all model parameters used in our analysis. θ_1 is the joint angle on the elbow, θ_2 is the joint angle on the wrist, and we will use state vectors $\mathbf{q} = [\theta_1, \theta_2], \mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]$ to present the position and state of the arm. The zero state $\mathbf{q} = [\theta_1, \theta_2]^T$ is the with both links pointed directly down.[**10**] Our aim is to stabilize the Acrobot at the fixed point at $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}] = [\pi, 0, 0, 0]^T$. From the equation of Lagrangian, we can yield certain equations of motion:

$$(I_1 + I_2 + m_2 l_1^2 + 2m_2 l_1 l_{c2} c_2)\ddot{q_1} + (I_2 + m_2 l_1 l_{c2} c_2)\ddot{q_2} + 2m_2 l_1 l_{c2} s_2 \dot{q_1}\dot{q_2} -$$
$$m_2 l_1 l_{c2} s_2 \dot{q_2^2} + (m_1 l_{c1} + m_2 l_1) g s_1 + m_2 g l_2 s_{1+2} = 0$$
$$I_2 + m_2 l_1 l_{c2} c_2 \ddot{q_2} + I_2 \ddot{q_2} + m_2 l_1 l_{c2} s_2 \dot{q_1^2} + m_2 g l_2 s_{1+2} = \tau$$

These equations when expressed in standard manipulator form are :

$$\mathbf{H}(\mathbf{q}) = \begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + 2m_2 l_1 l_{c2} c_2 & I_2 + m_2 l_1 l_{c2} c_2 \\ I_2 + m_2 l_1 l_{c2} c_2 & I_2 \end{bmatrix}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -2m_2 l_1 l_{c2} s_2 \dot{q_2} & -m_2 l_1 l_{c2} c_2 \\ m_2 l_1 l_{c2} s_2 \dot{q_1} & 0 \end{bmatrix}$$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} (m_1 l_{c1} + m_2 l_1) g s_1 + m_2 g l_2 s_{1+2} \\ m_2 g l_2 s_{1+2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

4.1.2 Balancing

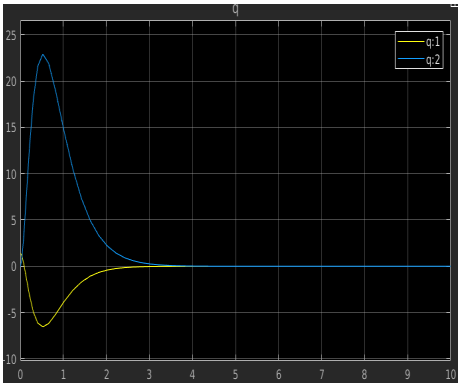


Figure 5: Joint Angle vs time

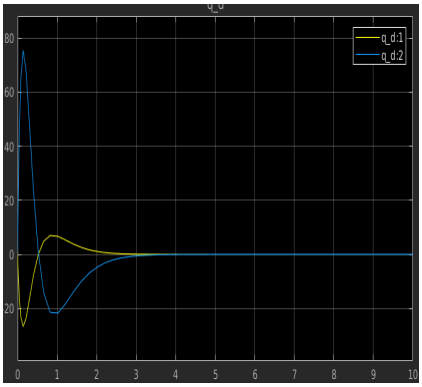


Figure 6: Joint velocities vs time

The Acrobot model is responsible for swinging up the two link system to the vertical point. We linearized above nonlinear equations about our desired position.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{H}^1(\mathbf{q})[\mathbf{B}\mathbf{u} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})] \end{bmatrix} \approx \mathbf{A}(x - x^*) + \mathbf{B}(\mathbf{u} - \mathbf{u}^*)$$

$$\dot{\bar{\mathbf{x}}} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}}$$

We found out the $\mathbf{A}$ and $\mathbf{B}$ for our acrobot model by using Taylor series expansion.

We got some preliminary simulation results with Two Link Pendulum Model. The simulation shows that our two-link pendulum simulation can be balanced when it is deflected from the desired position with LQR controller. Both joint angles and joint velocities reached to the desired values. The linear feedback matrix $\mathbf{K}$ used as

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t),$$

$$\mathbf{J}(x_0 = \int_0^\infty [\mathbf{x}(t)^T \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)\mathbf{R}\mathbf{u}(t)]dt$$

), $\mathbf{x}(0)$=X_0, $\mathbf{Q}$=$\mathbf{Q}^T$>0, $\mathbf{R} = \mathbf{R}^T$>0.

In this system. we use the torque of motor to control the state. However, to control the the desired torque of an ordinary DC motor we require a current controlled drive. This will make our device impractical for demonstration as the cost of the current controlled drive which suited our requirements was high. Therefore, we changed our design to one link inverted pendulum, trying to handle the problem by controlling the speed of a vibrating motor (a motor with an unbalanced mass attached to its shaft).

4.2 One Link Pendulum Model

In this section, we present our final version of the model which is based on one link pendulum, which is shown in figure 7. Realizing the difficulties of implementing two-link pendulum for demonstration, we considered to balance the forearm of the patient by a vibrating motor in a wearable device. This is a valid assumption as the origin of the tremors generally results from muscles in forearm and not from the muscles in the hand. The muscles in the forearm are big and it is from these spots uncontrolled muscles can produce high amount of forces to shake hands.

4.2.1 Dynamic modeling of one link pendulum model

We used Newton Euler approach to develop the dynamic model of the one link pendulum. Figure 8 illustrates all model parameters used in our analysis. ϕ is the angle from the horizontal position to the pendulum position and $\theta(\theta = \omega t)$ is the angle of the unbalancing motor rotates. The vibration motor produces 2 forces: Tangential acceleration force and inward radial force and we consider the CG lies at $\frac{l}{2}$. Below is the diagram analyzing the dynamics of this system:: F_R

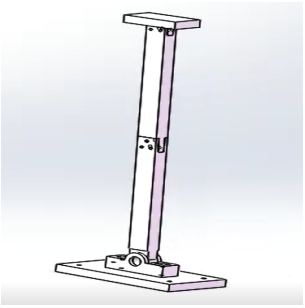


Figure 7: CAD model of one link pendulum

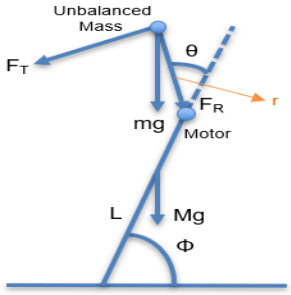


Figure 8: FBD of one link Pendulum

$$F_R = m\omega^2 r = m\dot{\theta}^2$$
$$F_T = m\dot{\omega} r = m\ddot{\theta} r$$
$$\vec{F_R} = F_R[-\sin(\theta + \phi)\hat{\jmath} - \cos(\theta + \phi)\hat{\imath}]$$
$$\vec{F_T} = F_T[-\sin(\theta + \phi)]\hat{\imath} + cos(\theta + \phi)\hat{\jmath}]$$

$$I\ddot{\phi} = \frac{l}{2}(\cos\phi\hat{\imath} + \sin\phi\hat{\jmath}) \times Mg(-\hat{\jmath}) + l(\cos\phi\hat{\imath} + \sin\phi\hat{\jmath}) \times \vec{F_T} +$$

$$l(\phi\hat{\imath} + \sin\phi\hat{\jmath}) \times \vec{F_R} + l(\cos\phi\hat{\imath} + \sin\phi\hat{\jmath}) \times mg(-\hat{\jmath})$$

$$I\ddot{\phi} = -[\frac{Mgl}{2} + mgl]\cos\phi + F_T l\cos\theta - F_R l\sin\theta$$

$$\ddot{\phi} = -A\cos\phi + B\cos\theta - C\sin\theta$$

where:

$$A = \frac{1}{I}[\frac{Mgl}{2} + mgl]$$

$$B = \frac{1}{I} m\ddot{\theta} rl = \frac{1}{I} m\dot{\omega} rl$$

$$C = \frac{1}{I} m\dot{\theta}^2 rl = \frac{1}{I} m\omega^2 l$$

We try to control the motion of the pendulum about an equilibrium point by controlling the angular velocity ω of the vibration motor. So our control input is ω and state vector is $x = [\phi, \dot{\phi}]$. Let $x_1 = \phi, x_2 = \dot{\phi}, u = w$, linearizing about the state x_o and ω_o, we get the equations below:

We got:

$$\begin{gathered} \dot{x_1} = x_2 \\ \dot{x_2} = -A\cos\phi + B\cos\theta - c\sin\theta \\ x_0 = (\tfrac{\pi}{2}, 0) = (\phi_0, \dot{\phi_0}) \end{gathered}$$

$$F(x,u) = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} (x - x_0) + \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix} (\omega - \omega_0)$$

From above, when $\ddot{\phi} = 0, (\phi, \dot{\phi}) = (\frac{\pi}{2}, 0)$, and then we get:

$$B\cos\omega_0 t_0 = C\sin\omega_0 t_0$$

$$\tan\omega_0 t_0 = \frac{B}{C} = \frac{\dot{\omega}}{\omega^2} = 0$$

$$\omega_0 = \frac{\pi}{t_0}, x_0 = \begin{bmatrix} \frac{\pi}{2} \\ 0 \end{bmatrix}$$

t_0 is the time we set to balance the pendulum to desired position when it deflects from the desired position, ω_0 is the angular velocity we need to control when we aim to balance it within the set time.

$$\begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix} (\omega - \omega_0) = \begin{bmatrix} 0 \\ -\frac{m\omega_0^2 rt}{I}\cos\omega_0 t - \frac{2\omega_0 mr}{I}\sin\omega_0 t \end{bmatrix} (\omega - \omega_0)$$

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ A & 0 \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} H\sin(\omega_0 t + \gamma)(\omega - \omega_0)$$

$$H = \sqrt{a_1^2 + a_2^2}, \gamma = \tan^{-1}\frac{a_2}{a_1}, a_1 = -\frac{m\omega_0^2 rt}{I}, a_2 = -\frac{2\omega_0 mr}{I}$$

$$\omega = \frac{-K_p\tilde{x_1} - K_d\tilde{x_2}}{H\sin(\omega_0 t + \gamma)} + \omega_0$$

5 Simulation and experimental validation

In this section, we present the results of the simulations done in Matlab to validate our models and controllers. We also built a prototype to test the validity of our controller and understand the limitations of our simulation. Figure 9 shows the block diagram of major components of our prototype:

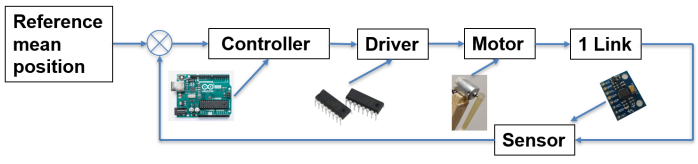


Figure 9: System Block Diagram

Figure 10 shows our prototype and a table below lists out mass and inertia about the pin joint. The pendulum is pinned to the 3D printed base linkage. The base linkage is mounted to the wooden base using 4 screws. In the front, we have Arduino Uno which is our single board computer where control algorithms are run. The motor driver is mounted on the 400 pin breadboard. At the back, we have 9v battery to power the Arduino and 2×3V battery to power up the motor.

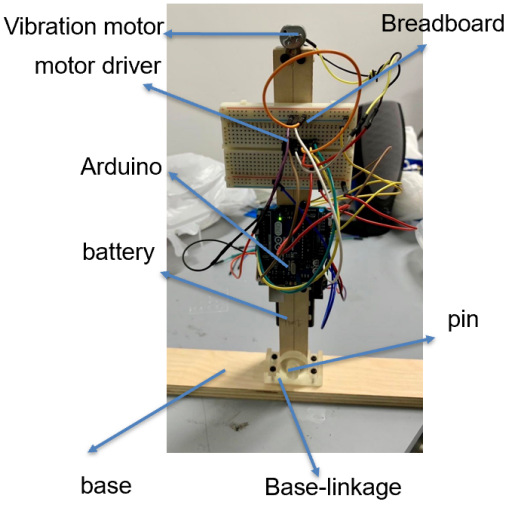


Figure 10: prototype

Parts Included In The Prototype

Item	Mass(gram)	Inertia(kg·m$^2 \times 10^{-4}$)
Arduino	27.22	4.6
Breadboard	86.18	36.22
MPU-6050	18.14	3.56
Motor	19.54	14.24
Battery(9v) and holder	43	2.1
Button cell and holder	5.27	0.64
Total	241.6	68.50

Following simulations were done in Matlab. The plots are shown alongwith them.

(a) A Simulation of one link pendulum that can be balanced by tuning the gains of the PD controller. Using pole placement method in linear control design, we computed the upper limits of k_p and k_d of our controller which keeps the poles of the system in negative half of s-plane. As the simulation shows, the system stabilizes to the desired position asymptotically in time.

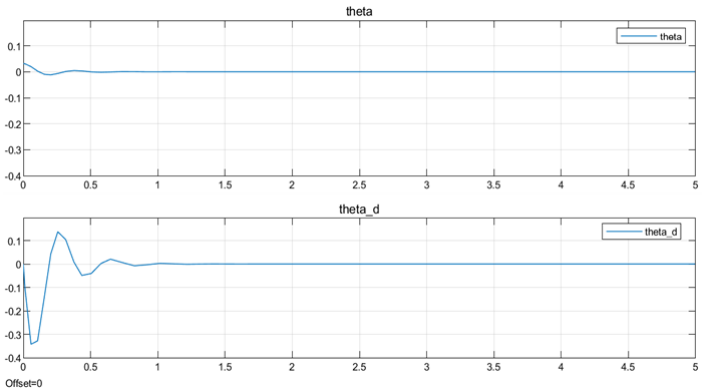


Figure 11: PD controller simulation

(b) A simulation of one-link pendulum that optimizes the use of power by the motor. A good controller for the wearable will minimize the consumption of electric power by vibration motor. This prolongs the use time of the device. We used LQR control technique and penalized those control policies which tries to accelerate the speed of the motor. Increasing or decreasing the speed of the motor draws more current from the battery and thereby reduces the use time of the wearable.

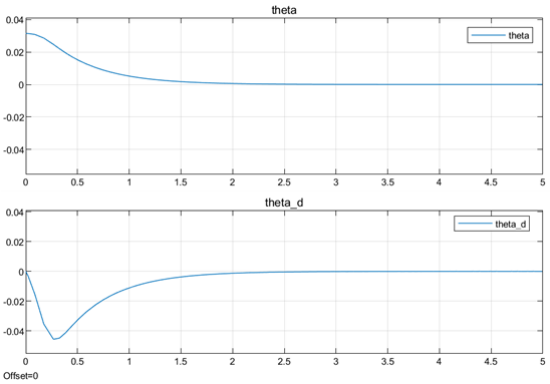


Figure 12: simulation of LQR

(c) The patient who will be wearing the device has tremors due to which random unpredictable movements are caused. The task of the controller must be to reject such random disturbances while it tries to stabilize the forearm at desired position. A simulation of the single link pendulum that can cancel out disturbance of the patients with Linear-Quadratic-Gaussian control (LQG) is presented.

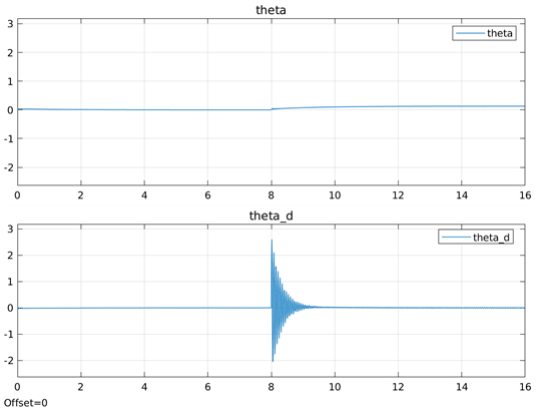


Figure 13: simulation of LQG

5.1 A controller that was actually realized in hardware

During our experimental realization and implementation, we found out that the linearized control technique requires a fast controller and fast and responsive motors. The motors we procured did not had such characteristics. So, we were unable to implement our control algorithm on the hardware we built which successfully demonstrates the canceling of tremors. In order to fix this problem, we devised another control strategy. In one link pendulum model, when hands have tremors, they deflect from the desired mean (the upward vertical position in our case), back and forth continuously. Therefore, in our assumption, if the pendulum is on the left side of the desired position, we need to control the angular velocity ω of the vibration motor so that a net average force pulls it back to the mean position. When our motor is responsive enough, we can actually control the pendulum to pull it in the opposite direction back to the desired position, which is upward vertical position. We have analyzed this situation shown by the figures below and developed a control strategy using it.

$$I\ddot{\phi} = F_T l\cos\theta - F_R l\sin\theta - (\frac{Mgl}{2} + mgl)\cos\phi$$

Since $\theta = \omega t$ we have

$$\ddot{\phi} = \frac{m\dot{\omega}r}{I}\cos\omega t - \frac{m\omega^2 r}{I}\sin\omega t - \frac{1}{I}(\frac{Mgl}{2} + mgl)\cos\phi$$

As you can see in the diagram, when the pendulum is on the right side of the desired position, we need to run the motor to create a net force pointing at the left direction and vice versa. In our specific case, since we experiment the pendulum in the horizontal plane, we decided to dismiss the gravity,and equation becomes like as below:

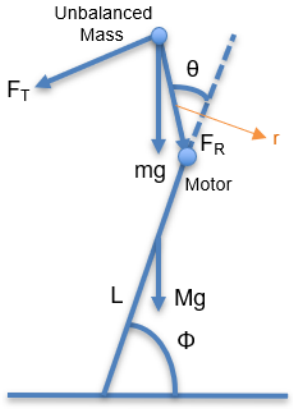


Figure 14: analysis of one pendulum model

$$\ddot{\phi} = \frac{m\dot{\omega}r}{I}\cos\omega t - \frac{m\omega^2 r}{I}\sin\omega t$$

So you can see from the diagram that when $\phi > \frac{\pi}{2}$, we need to make sure that the net force created by motor needs to point to the right direction. Vice versa, if $\phi < \frac{\pi}{2}$, we need to generate a net force pointing to the left direction.

If the pendulum is on the left side of the desired position, our objective is to control the angular velocity of the vibration motor ω to generate a net force over some time with direction to the desired position to pull it back to the upright position. besides, in this situation, we aim to increase the angle ϕ to be close to $\frac{pi}{2}$, so we need to give the pendulum an acceleration pointing to the left:

$$\ddot{\phi} = \frac{m\dot{\omega}r}{I}\cos\omega t - \frac{m\omega^2 r}{I}\sin\omega t < 0$$

Where $\omega = K\cos t$(K is a tuning parameter) Then we have: $\dot{\omega} = -K\sin t$, and we have below equation to prove after integration, we can control to generate a net force to further control our pendulum that they can move to the desired position.

$$\Delta t\frac{\ddot{\phi}}{mr} = -\frac{1}{I}\int_0^{\Delta t}[K\sin(t)\cos(Kt\cos t) + K^2\frac{1+\cos 2t}{2}\sin(Kt)]dt$$

The Δt time must be chosen as per the control frequency which was 33 hz in our case. We chose *Deltat* to be 200 ms. This led to a good enough estimate through the numerical integration technique.

To get the angle of inclination of the one-link pendulum, we used a MPU-6050 sensor which is a 6 axis accelerometer+gyroscope sensor. It uses I2C protocol to send data to its master. We implemented the I2C protocol of communication using Arduino Wire library as we used Arduino Uno as our Single board computer for the control implementation. For filtering, we recorded the sensor at averaged 10 values of the gyro data. This acts like a low pass filter removing the high frequency noise of the sensor. Every time the code runs the program gathers 500 data points of sensor values to calibrate the sensor bias. The sensor bias is dependent on environmental conditions such as temperature at which data is recorded. So, this is an important step for accurate measurements. We also computed the sensor's drift as it will alplay a role in the accuracy of measurement if the device is used for a long time. This is actually desired for a

wearable that will aid in writing. In that case, due to sensor drift, the errors accumulate and may result into undesired control actions from the wearable. The motor was interfaced to the Arduino Uno using L293 motor driver whose maximum current capacity from output pins is 600 mA. Arduino controls the duty cycle of the PWM output (5V signal). In order to control ω of the motor, we were sending duty cycle commands every 33 ms. In practise, motors do not operate for full range of the duty cycle. There is a saturation operating voltage below which it does not operate. In our case, this happened below 40% duty cycle. Our device needs to be powered by batteries, but we found that the internal resistance of the battery limits the output voltage. So when the current increases, the voltage drops and we can't get a constant voltage output and the desired angular velocity. Therefore, we switched to using a constant voltage power supply.

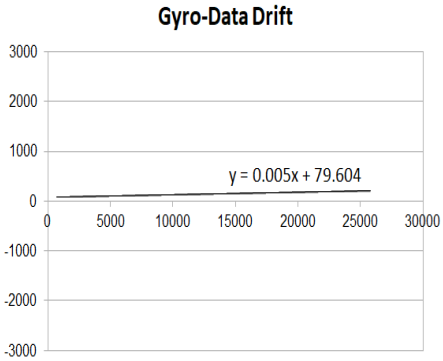
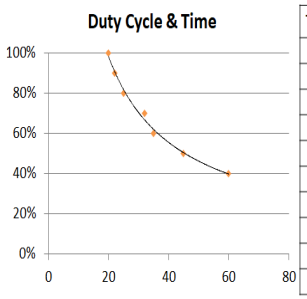


Figure 15: Filtering the Noise of the Sensor

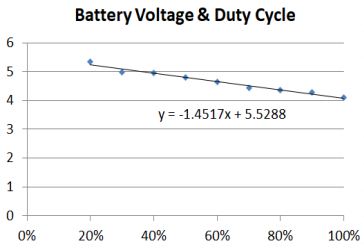
	Acc-X(m)	Acc-Y(m)	Gyro-Z(rad)
Bias	0.967779675	-9.870905209	0.000490287
Standard Deviation	0.088415429	0.060230496	0.028707556

Figure 16: Calibration of the Bias and Drift



Time(ms)	Duty Circle	Rotation
-	10%	no rotation
-	20%	no rotation
-	30%	no rotation
60	40%	2π
45	50%	2π
35	60%	2π
32	70%	2π
25	80%	2π
22	90%	2π
20	100%	2π

Figure 17: Duty Cycle Change Due to the load



Duty Cycle	Battery Voltage(V)
20%	5.35
30%	4.99
40%	4.96
50%	4.8
60%	4.64
70%	4.45
80%	4.36
90%	4.27
100%	4.1

Figure 18: Battery Voltage vs Duty Cycle

6 Conclusions

In this project we have defined a design process for designing a parkinson's assistive device which is based out of strong mathematical framework supported by simulation and experimental validation. Although the modeling and experimental validation is done on 2D planar case of the forearm and the motor, the project clearly brought out many practical difficulties that were not taken into account during simulation of the models and were encountered during implementation of the device on hardware. The simulations can be improved by removing assumptions which do not exist in real world situation such as no biases in sensors, full operating voltage range etc. For the actual implementation of the wearable, it should first filter out the disturbances from the hand and the device should not cancel out the voluntary movement of the hand. The models in the planar case must be extended to 3D case since the human joint at the elbow and wrist allow motions with multiple degrees of freedom. This naturally will increase the number of the states of the model. Another aspect is the requirement to use additional vibration motors in 3D case. To implement it for 3D case and make it in a wearable form, we need to improve the design and reduce the weight for the wearable device with the use of On-board electronics. An extension to the 3D case requires more rigorous simulations, experimental testing and data collection with the actual PD patients.

References

[1] Parkinson's Disease Information Page, https://www.ninds.nih.gov/Disorders/All-Disorders/Parkinsons-Disease-Information-Page

[2] Sveinbjornsdottir S(October 2016). clinical symptoms of Parkinson's disease. Journal of Neurochemistry. 139 Suppl 1: 318–324. Bibcode:2006JNeur..26.9606G. doi:10.1111/jnc.13691. PMID 27401947.

[3] Disease Injury Incidence Prevalence Collaborators (October 2016).Global, regional, and national incidence, prevalence, and years lived with disability for 310 diseases and injuries, 1990–2015: a systematic analysis for the Global Burden of Disease Study 2015. Lancet. 388 (10053): 1545–1602. doi:10.1016/S0140-6736(16)31678-6. PMC 5055577. PMID 27733282.

[4] Mortality Causes of Death Collaborators (October 2016). Global, regional, and national life expectancy, all-cause mortality, and cause-specific mortality for 249 causes of death, 1980-2015: a systematic analysis for the Global Burden of Disease Study 2015. Lancet. 388 (10053): 1459–1544. doi:10.1016/s0140-6736(16)31012-1. PMC 5388903. PMID 27733281.

[5] Yao, S.C.; Hart, A.D.; Terzella, M.J. (May 2013). An evidence-based osteopathic approach to Parkinson disease. Osteopathic Family Physician. 5 (3): 96–101. doi:10.1016/j.osfp.2013.01.003

[6] de Lau LM, Breteler MM (June 2006). Epidemiology of Parkinson's disease. The Lancet. Neurology. 5 (6): 525–35. doi:10.1016/S1474-4422(06)70471-9. PMID 16713924

[7] R T. Mhyre, J T. Boyd, R W. Hamil, K A. Maguire-Zeiss (2012). Sub-Cellular Biochemistry. Springer, Dordrecht. pp. 389–455. ISBN 978-94-007-5415-7.

[8] Elbers RG, Verhoef J, van Wegen EE, Berendse HW, Kwakkel G (October 2015). Interventions for fatigue in Parkinson's disease. The Cochrane Database of Systematic Reviews (10): CD010925. doi:10.1002/14651858.CD010925.pub2. PMID 26447539

[9] Tedrake, Russ Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for MIT 6.832, 2009

[10] Spong, Mark W, Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on

Supplementary

(a) Block diagram code

(b) Arduino code

```
#include<Wire.h>

const int MPU_addr=0x68;
const int GyrZAddr = 0x47;
const float convFactor = 131.0; //gyrocope
const byte N = 10;
const byte ZeroN = 5;
float noiseBand = 0.24;
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
int i = 0;//02
byte countS = 0;//03
float zeroOmegaI = 0.0f;// zero bias of omega
float recOmegaI[10];//05
float omegaI = 0.0;//06
float thetaI = 0.0;//07

//const float kAngle = 5.0;// Kp constant in PD control
//const float kOmega = 0.0;//Kd constant in PD control
```

```
float powerScale;//14
float power=0.0;//15
int deltaT; // in milliseconds
int currentTime=0;
int lastTime=0;

const float k = 100;
const float c = 1;
int currentTime_1=0;
int lastTime_1=0;
int deltaT_1; // in milliseconds
int dir;
float adjustA = 210;
int adjustT = 20;
int adjustD = 3000;

void setup ()
{
  // Initialise an array with zero values
  for ( i = 0 ; i < N ; i++ )
  { recOmegaI[i] = 0; }//25

  // Setup the pins for output
  pinMode(3, OUTPUT); // enable Pin
  pinMode(7, OUTPUT); // Motor output pin 1
  pinMode(8, OUTPUT); // Motor output pin 2
  ConfigureMPU6050();
  Serial.begin(9600); //19
  delay(300);
  training();
}

void loop ()
{
  chkAndCtl();

  delay(10);
}

void ConfigureMPU6050()
{
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up he MPU-6050)
  Wire.endTransmission(true);
}

int GetRawGyrZ()
{
  int GyZ;
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x47 (gyr_H)
  Wire.endTransmission(true);
  Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
```

```
  Tmp=Wire.read()<<8|Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  return GyZ;
}


// Computes zero error bias of the sensor before start
void training()
{
  delay (1000);
  float sensorData;
  for ( i = 0 ; i < 500 ; i++ )
  {//50
    sensorData = -GetRawGyrZ()/convFactor;
    //Serial.print("Training = ");
    //Serial.println(sensorData);
    zeroOmegaI = zeroOmegaI + sensorData;
  }
  zeroOmegaI = zeroOmegaI / 500.0;
  Serial.print("Inside training - ZeroOmegaI=");
  Serial.println(zeroOmegaI);
}


void chkAndCtl()
{
  float sensorData;
  omegaI = 0;
  for ( i = 0 ; i < N ; i++ )
  {
    sensorData = -GetRawGyrZ()/convFactor;
    //Serial.print("Inside chkandctrl= ");
    //Serial.println(sensorData);
    omegaI = omegaI + sensorData - zeroOmegaI;
    //Serial.print(omegaI);
    //Serial.println();
    delayMicroseconds(10); //NL6
  }
  omegaI = omegaI / N;
  //Serial.print(omegaI);
  //Serial.println();
  if (abs( omegaI ) < noiseBand )
  {
    omegaI = 0;
  }
  recOmegaI[0] = omegaI;
  currentTime = millis();
  deltaT = currentTime - lastTime;
  thetaI = thetaI + omegaI*float(deltaT)/1000.0;
  lastTime = currentTime;

  if (thetaI > 0)  {dir = -1;}
  else if (thetaI < 0)  {dir = 1;}
  else  {dir = 0;}

  //Serial.print(thetaI);
```

```
  //Serial.println();
  countS = 0;
  for ( i = 0 ; i < 10 ; i++ )
  {
    if (abs( recOmegaI[i] ) < 9*noiseBand )
    {
      countS++;
    }
  }
  if ( countS > ZeroN )
  {
    thetaI = 0.0;
  }
  for ( i = 9 ; i > 0 ; i-- )
  {
  recOmegaI[ i ] = recOmegaI[ i-1 ];
  }

  for ( i = 0 ; i < adjustT ; i++ )
  {
    currentTime_1 = millis();
    deltaT_1 = currentTime_1 - lastTime_1;
    powerScale = float(dir) * ( k * cos(c * float(i) * float(deltaT_1) +
    float(adjustD)) + adjustA) + 102.0; //Still has some problems
    power =int (max (min (95*powerScale/100 , 255 ) , -255 ));

    if ( power > 0 )
    {
      analogWrite( 3, power );
      digitalWrite( 7,HIGH );
      digitalWrite( 8,LOW );
    }
    else
    {
      analogWrite( 3, -power );
      digitalWrite( 7,LOW );
      digitalWrite( 8,HIGH );
    }
    delayMicroseconds (adjustD);
   }
  //powerScale = ( kAngle * thetaI  ) + ( kOmega * omegaI );
  //power =max (min (95*powerScale/100 , 255 ) , -255 );
  //Serial.print("power ="); Serial.println(powerScale);
}
```