

Robotics

G. Sheppard, D. Thomas, J. Doering, J. Matthews, C. Li

University of Birmingham
Physics and Astronomy Department

Contents

HERE IS A LIST OF WORDS THAT WE SHOULD CHECK AT THE END TO MAKE SURE WE HAVE ALL USED THE SAME

Double pendulum not triple pendulum

Nao not NAO (or other way let's just keep it constant)

Posture not position

Start-up not startup

Large encoder not big encoder (sounds better I think)

Rotational not torque

Centre point not zero point (as now swings through the centre etc)

Set-up not setup (probably this way round who knows tbh) ((They mean slightly different things i think))

1 Abstract

[?].

2 Introduction

2.1 Background

2.2 Motivation

2.3 Theory

3 Connecting To Nao

3.1 Nao

Initial connections to Nao performed using choreographe good to familiarise with joint names and connections (graph of joint names, limited by implementation, switched to using the ALProxy and created a joint name model to better understand the joint and connections

Can connect to both virtual and simulated robots in order to check motions before implementing

Improved on positions from previous years to maximise each posture to extend as far as possible

Increased the speed of NAOs kicks as previous years limited by using the set range of the knee pitch as there max angle as they assumed this was the largest angle and so the one to maximise

instead based on the range of individual positions

Put in stops to prevent the angle aching to far, same with checking the initial posture when setting up the postures

communicate directly using python scripts

Include diagram of joints and talk about connecting and extracting sensor values and calling joints using python dictionaries: Normalising speeds, creating algorithms to ensure simultaneity. Slowing initial speed of NAO to prevent injury during initialisation of positions. Defining positions to ensure left and right side was symmetric and improved on previous years positions my maximising the range of motion with fail safes to ensure a correct position was reached by comparing joint data with expected position

Small amount about compatibility issues and that these were resolved by using the same python SDK and choreograph versions and running code via linux to prevent issues with compiling and installing:

Creating

3.2 Encoders

George Sheppard

There are two sets of hinge encoders used, the large/big encoder, and the small encoders. The large encoder records the angle from the vertical to the largest rod of the swing. The small encoders record the angle with respect to the previous rod, there are 2 on each side of the swing.

A large amount of time was invested into connecting to these hinge encoders from any computer other than the lab one. Unfortunately, due to the nature of the shared object files required by the encoders, the set up is complicated and it is advised to use the lab computer when using the hinge encoders.

A guide for setting up the hinge encoders is provided in the wiki found in ??.

3.3 Interface

The creation of each individual algorithms required the use of a substantial amount of common code, such as the logic required for: connecting to Nao, collecting values from the hinge encoders and Nao, handling of previous collected values, switching Nao's position, and storage. For this reason an interface was created that functioned as a base that all algorithms were built off, the logic of this base code is illustrated in figure ??. The interface starts by collecting all values required for the algorithm decision making process, it then passes these values through the algorithm, depending on the output of the algorithm it will either change Nao's position, switch to the next algorithm, or finish and store the data. It also fixes the sample rate to a specified frequency.

With this base structure in place, creating an algorithm involved one function that took a set of values, and decided how to react to them, the algorithm had access to all past data to help in decision making. As these algorithms were defined in the same way, it is easy to switch between different algorithms mid swing, such that the motion of NAO could be more stringently controlled. For example NAO could be told to increase to 30° , then maintain this amplitude for 10s, and then finally decrease back down to 0° .

With this base structure in place, creating an algorithm had two requirements: it took the set of values given to it by the interface, and it returned either 'switch', None, or any of the named positions that were defined in the Robot subsection of the interface e.g. 'Seated', 'Extended' etc.

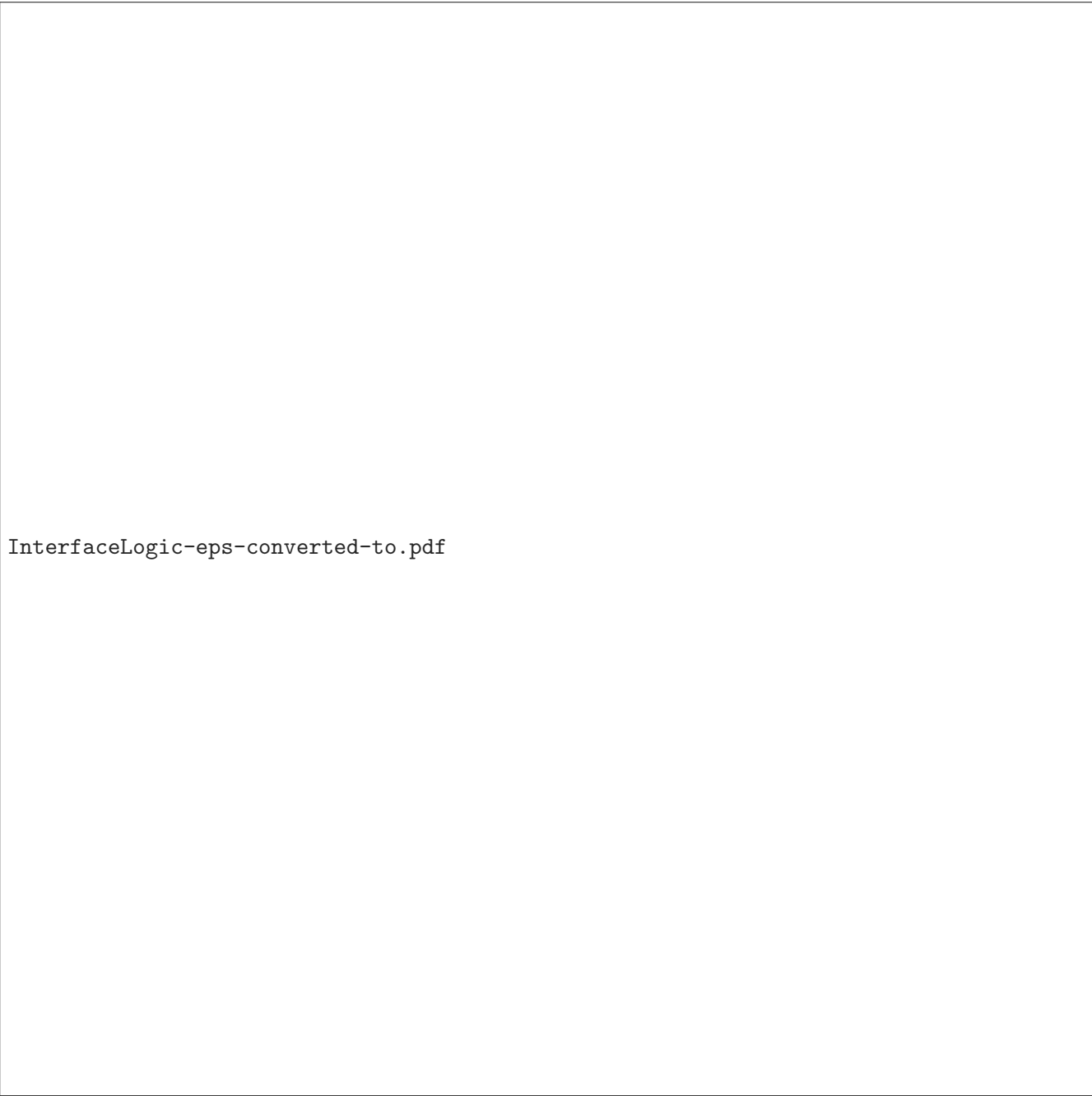
As the interface required the encoders and the robot to connect to, a series of mock classes were created that aimed to replicate their functionality. This meant that away from the lab these fake classes were used as a substitute for the real encoders such that the interface could be developed without errors.

3.4 Testing

One of the main advantages of this setup was the ability to test algorithms without connecting to either the robot or the hinge encoders. As all previous data collected in the lab were stored, a setup was developed that read the file in line by line and passed it through the algorithm. This is exactly what would happen in a real test except the real data is replaced with the pre-recorded data.

Properties such as the position of Nao at each cycle, and the current algorithm the interface was on is recorded in both the real mode and the testing mode. This meant how each algorithm reacted to different scenarios could be plotted away from the lab, given that pre-recorded data that mimicked that scenario existed.

One downside of this method was that the data being fed into the algorithm wasn't reactive to the previous decision of the algorithm, as all the data was pre-recorded. This meant that this method was in no way a



InterfaceLogic-eps-converted-to.pdf

Figure 3.1: The logic behind the interface, all that required changing was the algorithm logic that the decision making was based on.

full replacement for testing the limits of the algorithm in the lab, but instead was used as a way to check that the algorithm was logically correct before spending limited time with Nao on it.

3.5 Optimisations

As discussed in section ??, the interface would record any data relevant to the motion of the swing and Nao. This data could be used to effectively diagnose any problems with the motion, or investigate any other properties of Nao that could be useful in creating new algorithms. This amount of data, however, became a performance issue in certain situations, and this section is dedicated to discussing the

optimisations made regarding this.

3.5.1 Small Encoders

Jonathan Matthews

One of the obvious sources of delay in the system is the set of small encoders attached to the swing. These encoders are incredibly useful in modelling the double pendulum of the swing, however, they are irrelevant when the swing is setup in it's simple pendulum mode with all the joints fixed. For these reason there were two setups created for the interface, one that recorded from both the large and small encoders, and one that only recorded from the large encoder only.

A method that was investigated to avoid this problem was the introduction of multiprocessing into the interface.... TODO: JONATHAN FINISH THIS BIT, WHY DOESN'T IT WORK, WHAT IS THE SAMPLE RATE OF THE SMALL ENCODERS

3.5.2 Nao

George Sheppard

A second source of delay was the transmission of data to and from Nao, a small subset of this data was paramount to the success of the algorithms, such as the commands for changing the joint positions. A large amount, however, were redundant in certain situations. An example of this was the accelerometer values, initially these were included such that algorithms could be created using only values collected from Nao. A large majority of the time, however, these values were not being used in the algorithms or even looked at when investigating problems with Nao etc.

For this reason when creating an algorithm the user could input flags that allowed them to define whether they required these values, when they didn't require these values then fake values would be returned such that performance would increase.

Once the optimisations to both the small encoders and the calls to Nao, the interface was able to sample at a maximum of around 200Hz, at this point the large encoder was restricting any further increases. For most algorithms, however, the workload required by the individual algorithm would restrict this sampling rate before this point.

4 Swinging

4.1 Webots

James Doering

Webots is a program that can simulate robots in a virtual environment with accurate physics and sensory feedback. In the same way that the real Nao can be communicated with via the Python library NAOqi, the virtual Nao can be controlled via NAOqisim, which is available [PUT STUFF HERE]. Through this method, Python code can be run through the NAOqi library, and sent to the NAOqisim controller in Webots, meaning that code can be tested away from the lab. This was especially useful for defining Nao's postures, as described in figure ???. Webots automatically calculates the centre of mass of Nao - by removing the floor and setting gravity to be 0, the centre of mass relative to Nao's accelerometers could be found for each posture. This allowed for the 'raised' and 'lowered' postures to have the largest possible difference in heights for the centre of mass, optimising the parametric motion.

Webots also offered semi-accurate physics simulations, with some caveats. Using the swing modelled by the previous year revealed that the virtual Nao will easily assume it is falling, activating its fall

manager and changing the algorithm's current posture. If the computer running Webots was weaker, the simulation would quickly become physically inaccurate and the algorithm would desynchronise. Finally, Webots offered no simple way of extracting the hinge encoder data in real time - this forms the basis of most algorithms, and so it could only be used for very simple tests. More information on setting up Webots can be seen in [PLACE REF HERE]

I WILL REWRITE AND ADD IN FIGURES

4.2 Optimising Nao's Kicks

4.3 Rotational Method

$$\frac{\Delta E}{E} = 1 - e^{-\frac{2\pi}{Q}} \quad (1)$$

$$\text{At maximum angle,} \quad (2)$$

$$E = lmg(1 - \cos(\theta)) \quad (3)$$

$$\Delta E = E(\theta_a) - E(\theta_b) \quad (4)$$

$$\theta_a = \theta_b + d\theta \quad (5)$$

$$\frac{d\theta}{dt} = c - \frac{1 - e^{-\frac{2\pi}{Q}}}{dt} \frac{\theta_b}{2}, \text{ for small angles} \quad (6)$$

David: Say used previous defined positions, different methods for each algorithm (quarter period, integration of theory team calculation of max angle), results for each, try to fit maximum amplitude peaks to linear to see if proportional to distance rocked or not, how results varied on different parameters such as if he swings before peak, during, or after, limitations to each method. Calculations of offset parameters to maximise amplitude gain and comparison of each plot

4.4 Parametric Method

Jon: Talk about using the two extra positions defined for this motion from david, the idea behind the algorithm, results try to fit to exponential curve to see if it multiplies amplitude by fixed fraction like worksheet showed, any limitations, were two positions just different in vertical centre of mass or did horizontal distance change too

$$\theta_a = \theta_b \left(\frac{L_{squat}}{L_{stand}} \right)^{3/2} \quad (7)$$

$$\theta_a = \theta_b + \frac{d\theta}{dt} dt \quad (8)$$

$$\theta_b + \frac{d\theta}{dt} dt = \theta_b \left(\frac{L_{squat}}{L_{stand}} \right)^{3/2} \quad (9)$$

$$\frac{d\theta}{dt} = \frac{\theta_b}{dt} \left(\left(\frac{L_{squat}}{L_{stand}} \right)^{3/2} - 1 \right) \quad (10)$$

$$(11)$$

4.5 Damping

James Doering

I'll put stuff on how each posture changes his damping coefficient. I'm also going to talk about how the rotational and parametric methods might look with no damping. There's a part I need to talk about how rotational the parametric method actually is but IDK where to put it.

I'll talk about how much of the damping comes from the air resistance and how much comes from the bearing torque friction, then may calculate the maximum theoretical angle Nao should reach

5 Single Pendulum

5.1 Start-up

James Doering

There are multiple different ways to begin a swing from rest, the most intuitive being to "kick off" from the ground - due to Nao's short legs, this was not achievable. This was solved in previous attempts by using a weighted box to kick away from, or by manually pushing Nao, giving the initial amplitude required for normal swinging to be effective. In order to achieve our goals of a fully self-sufficient swinger, a dedicated start-up algorithm was developed. The method is simple - Nao kicks into the extended posture, waits a quarter period, kicks into seated, and then loops between waiting a half period and kicking into the next posture. The initial quarter period wait is used to account for Nao beginning in the centre of his swing, not at the edge.

Figure 5.1: HELP HELP HELP

As can be seen in figure ??, this method is blah blah blah... The period was calculated from previous data of nao swinging blah blah... notably, the period of the swing changes slightly depending on the posture, but this was not considered in the code due to the difference in timing being miniscule...

This script also offers a good comparison for mass vs no mass etc etc etc etc etc etc etc...

5.2 Timing via Angular Velocity

George Sheppard

To create algorithms that are efficient at applying Nao's energy into the motion of the swing requires an accurate estimate of the maximum angle of the swing, or the time at which this occurs. Too early a swing dissipates the energy, and too late doesn't take advantage of the boost in velocity caused by changing position.

For this reason a discussion of the best method for determining the maximum of the swing is outlined below.

Chenglong Li

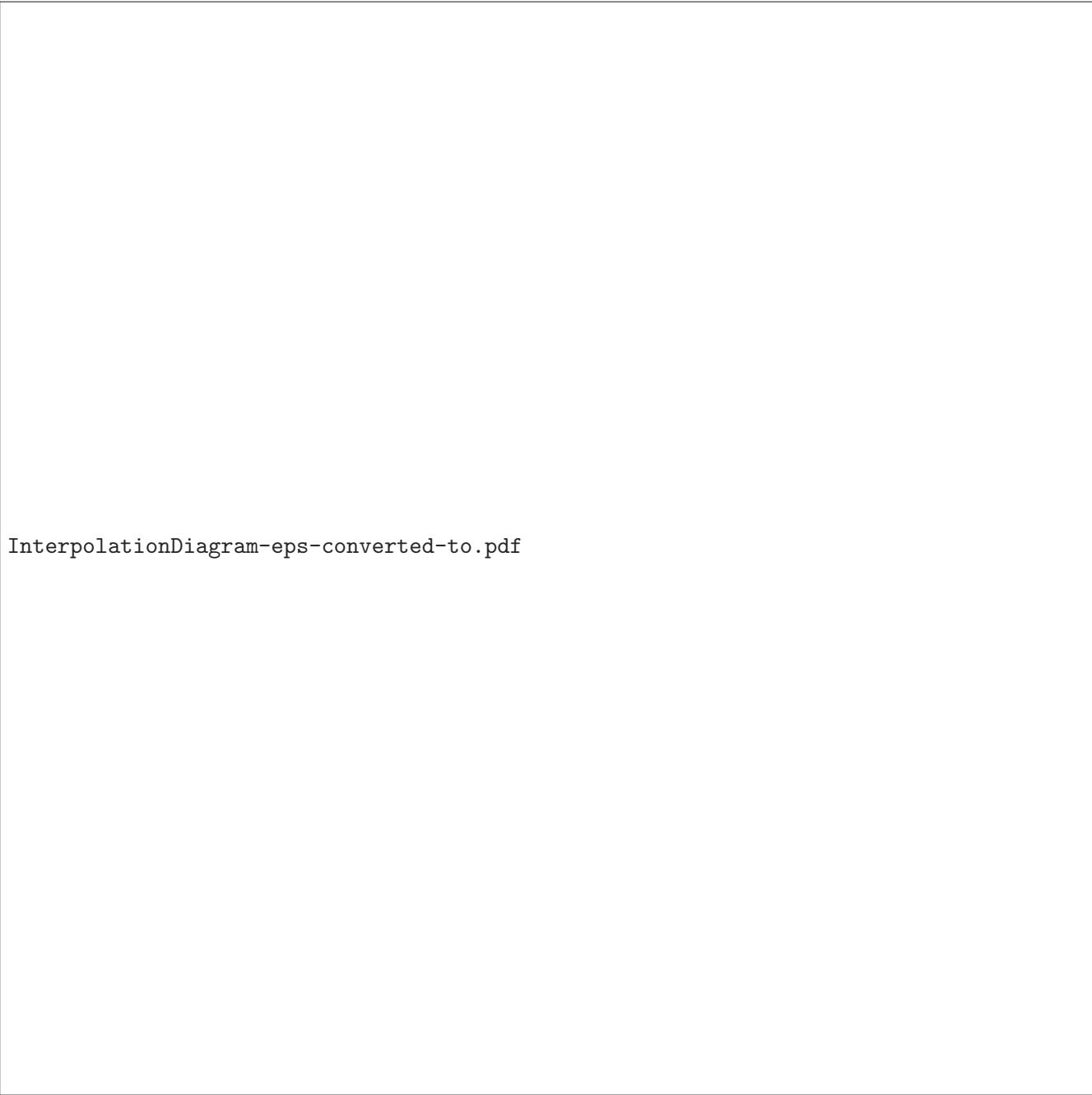
The angular velocity algorithm is developed based on the fact that the sign of the angular velocity will change just after the Nao hits its maxima. Therefore one can instruct the Nao to change its posture whenever the angular velocity changes its sign. This algorithm is very easy to implement and very reliable, which means that the Nao will never miss any kicks. However, there are two serious weaknesses for this algorithm. The first one is that this algorithm does not learn from old data, because this algorithm uses the instantaneous big encoder value to decide if Nao should kick or not. This indicates that the angular velocity method does not have the ability to improve itself. The second weakness of this algorithm is that one cannot set offset in this algorithm. Assuming that an algorithm predicts that Nao should be at the maxima at time T or angle A but one would like Nao to kick a little bit earlier or latter. Hence we set Nao to kick at $T + \text{offset}$ or $A + \text{offset}$.

5.3 Timing via Maximum Angle

5.4 Timing via Quarter Period

George Sheppard

The quarter period algorithm uses the time that the last quarter period of the cycle took to predict the time of the next maximum, and utilizes the fact that Nao's motion only increases in angle by a small amount per cycle. Whenever Nao swings through the centre of the swing, the first thing to calculate is an estimate of the time that Nao actually crossed the boundary. This is important as Nao swings the fastest through the centre, and therefore the first time recorded after he crossed may not be close enough to the true time. Shown in figure ?? is a diagram of this situation.



InterpolationDiagram-eps-converted-to.pdf

Figure 5.2: Illustration of the values of the swing, before and after the crossing point.

For angle and time (θ_1, t_1) before the crossing, and (θ_2, t_2) after the crossing, a linear interpolation can be applied to give a more accurate estimate of the true zero point crossing. This is calculated using

$$t_c = t_2 - (t_2 - t_1) \frac{|\theta_2|}{|\theta_2 - \theta_1|}. \quad (12)$$

As is expected for a higher sampling time this error on the true centre time is smaller. The second value to calculate is the time at which Nao was at the last angle maxima. This can be done simply as all previous values of the encoder are recorded. One small complication is the introduction of local maximas, these occur more often at low amplitude swings due to the jolted motion of Nao's movement, and need to be filtered out to allow a better estimate for the maximum angle time. To filter these local maxima out the moving average of the encoder values is calculated, this smooths the data, figure ?? shows a comparison between the moving average of some example data, for this example the latest local maximum would be the calculated max time, but after using the moving average the estimate is much closer to it's true point. Using the modified encoder values the time of the maximum can be determined, and therefore the quarter period of the swing can be calculated.

This process is repeated every half period, whenever Nao crosses the centre point, such that the next maxima time can be calculated before it is reached. An advantage of this algorithm is that the estimate of the period is constantly updated, such that the calculation of the maxima time remains accurate for all amplitudes (as long as there is sufficient build up time to calculate the previous period). From here the only decision that needs to be made is whether it is better to swing before the maxima, at the maxima, or after the maxima, for the greatest amplitude swing.

DAVID: OFFSET INVESTIGATION HERE

5.5 Timing via Accelerometer Values

George Sheppard

Nao's ability to swing in each previous timing method requires the use of external inputs, such as the hinge encoders. For Nao to truly swing unassisted an equivalent to the large encoder angle needed to be extracted from Nao's outputs only. It was decided that the accelerometer values, and more specifically the vertical accelerometer (Nao's Z accelerometer) would allow us to extract a sinusoidal curve.

A comparison between the Z accelerometer data for both a static and changing posture during swinging is shown in figure ??, this confirms that the Z accelerometer is the vertical direction, as for a static posture the accelerometer reading is of the order of g . It is sinusoidal due to the sinusoidal motion of the swing, and can be used to extract the maximas of the swing in the same way the large encoder is used, it is clear that the accelerometer frequency here is twice the frequency of the swing, as acceleration is maximum at both edges of the swinging motion. The introduction of a changing posture, however, causes the Z accelerometer to transition to something that resembles more of a square wave, and to be able to extract the maximas this data needed to be filtered, another interesting property that arises from switching posture is that the frequency is reduced to that of the swing, as the acceleration due to the swing is very small compared to the acceleration caused by changing posture.


Figure ?? shows a comparison between the frequencies present in the accelerometer data for a static posture, and for a switching posture, each dataset has been normalised by largest frequency peak. For the static posture dataset there are two spikes, the largest of which with a frequency of 0.8Hz. This frequency is twice that of the swing, as the acceleration goes from maximum at the edge, to minimum at the centre point, back to maximum at the other edge. The other peak has a frequency of 0.4Hz, this arises due to the asymmetry in Nao's seated position, as Nao's seated posture is slightly behind the seat, the

MovingAverageDiagram-eps-converted-to.pdf

Figure 5.3: Comparison of angle data before and after a moving average is applied.

maximum angle that he reaches will be asymmetric and will oscillate slightly higher and lower dependent on which side of the centre point he is on, this explains why it is at the frequency of the swing. To understand the frequency composition for the changing posture, it is worth looking at modelling Nao's motion. Nao's motion switches once at each maxima, and the switch point can be approximated as instantaneous, in other words Nao's motion is a square wave with the same frequency as the swing. The fourier transform for a square wave is

$$f(t) = \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{1}{2n-1} \sin(2(2n-1)\pi ft), \quad (13)$$




AccelerometerComparison-eps-converted-to.pdf

Figure 5.4: Z accelerometer readings for a static posture and a changing posture.

where f is the frequency of the swing, this means that there should be diminishing peaks expected at 0.4Hz, 1.2Hz, 2.0Hz etc due to Nao switching posture. The last remaining peak at 0.8Hz is for the same reason as before, however, it is now amplified due to the fact that the extended position has a moment of inertia further away from the line of the rod than the seated position. This does not appear so straight from the graph, however, remember that this is a relative plot and therefore multiplying by the scale factors shown in figure ?? will reveal this to be true.

To select only the frequency associated with the swing, the accelerometer data was passed through a band-pass filter, with a lowcut frequency of 0.30Hz, and a highcut frequency of 0.50Hz. With only the frequency of the swing selected the resulting curve can be inverted, and an offset can be added such that

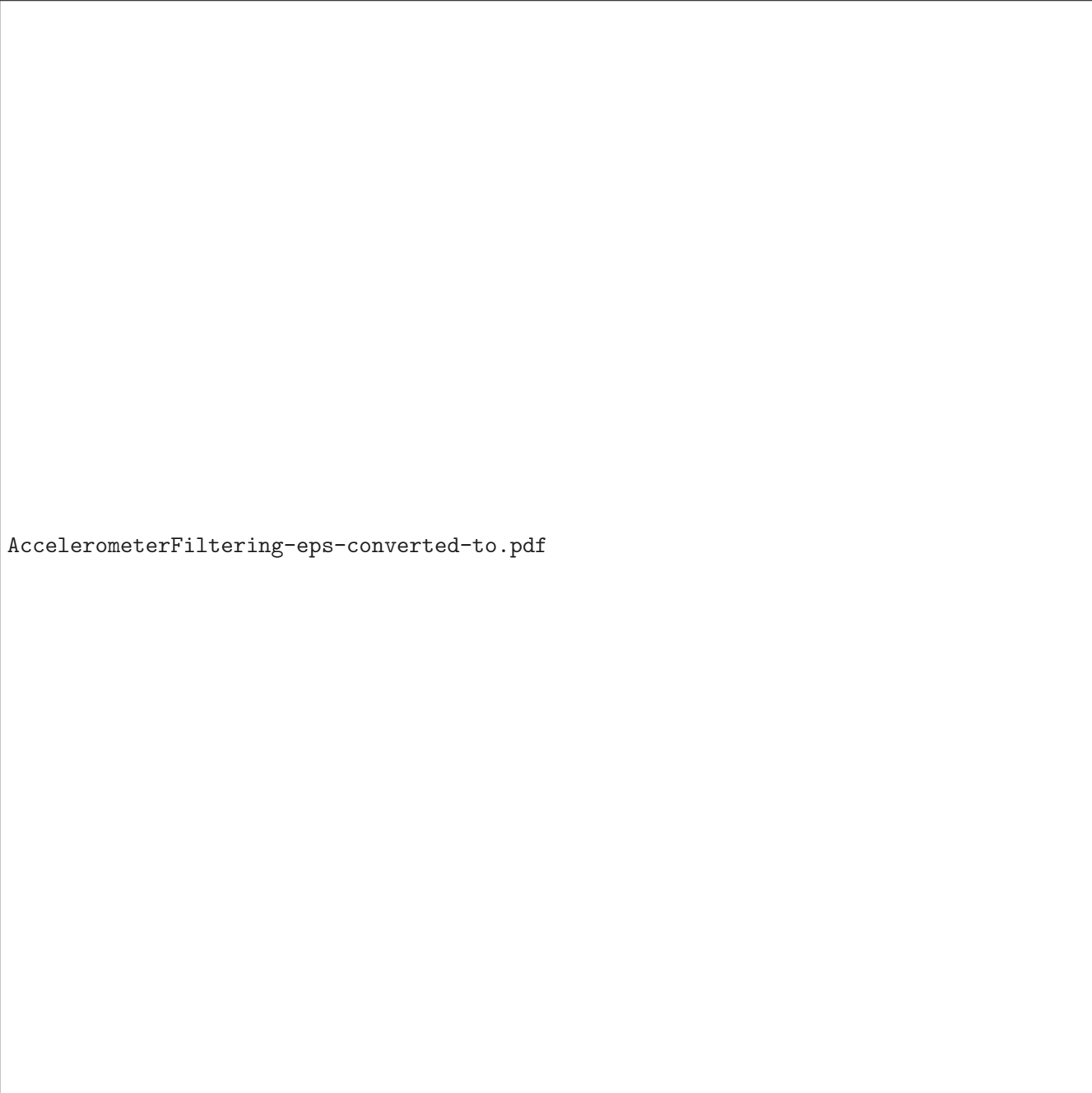


FrequencyDecomposition-eps-converted-to.pdf

Figure 5.5: Fourier transform of Z accelerometer data where Nao switches position at each maxima, and where he stays stationary throughout.

the final curve resembles that of the large encoder values, this process is shown in figure ?? . This method of filtering allows the Z accelerometer data to be 'converted' to the large encoder value, the magnitude of the curve sinusoidal motion is unimportant, as it is only the time at which the peaks occur that is used each of the previous timing methods. This means that the filtered Z accelerometer data can be used as a direct substitute for the large encoder value such that no modifications need to be made to any of the timing methods.

As this algorithm doesn't change the timing of the maxima, it will be subject to the same maximum angle that the individual timing method it is connected to has. One point of interest however is the reliability in

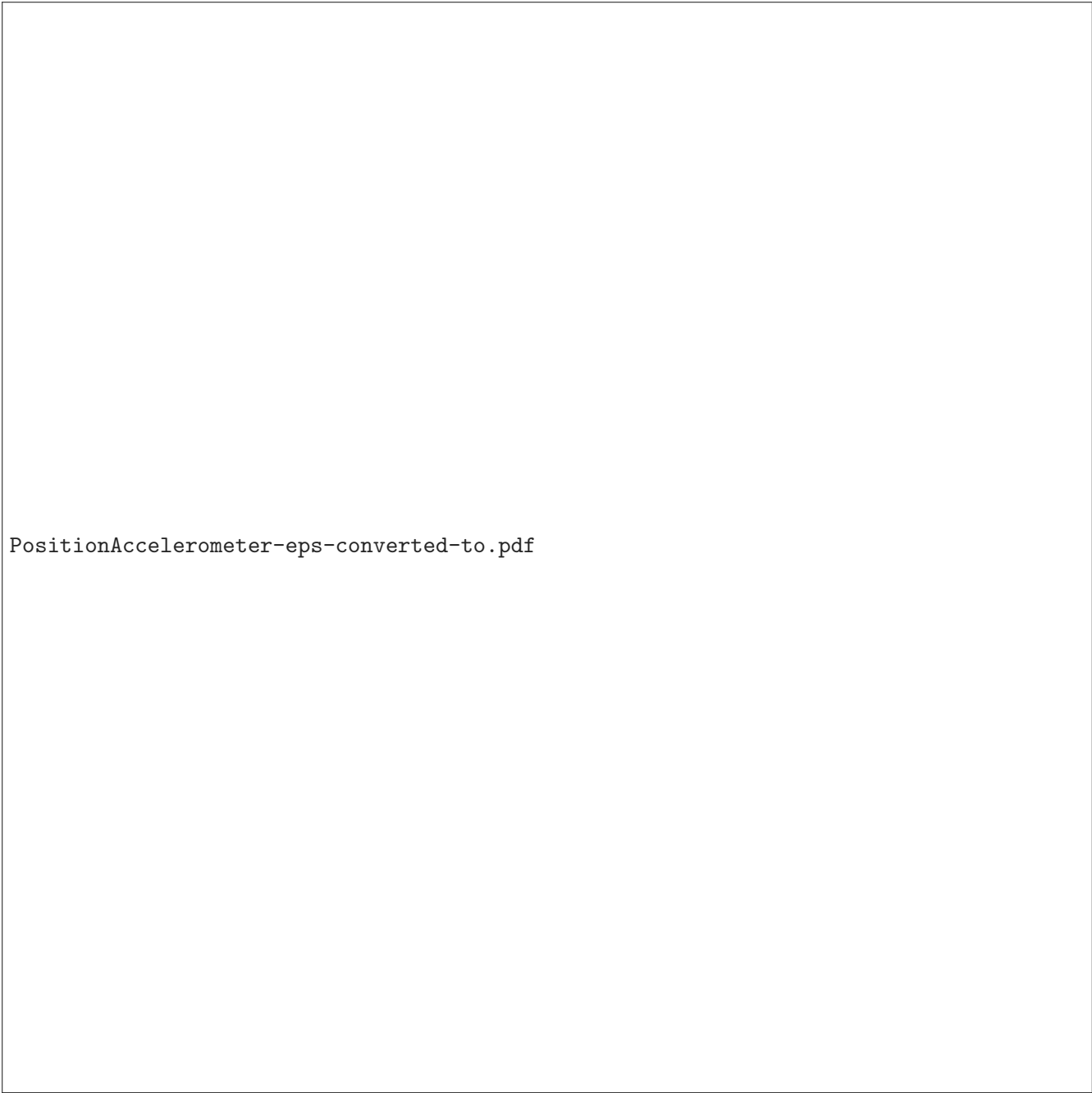


AccelerometerFiltering-eps-converted-to.pdf

Figure 5.6: Z accelerometer values are first passed through a bandpass filter, subsequently the values are flipped and offset to recreate large encoder values.

making this conversion from accelerometer to equivalent large encoder value. Figure ?? shows a sample of this algorithm reacting in real time. This algorithm is able to switch at the correct times for the majority of cases, unfortunately however, Nao occasionally fails to transmit the correct accelerometer value, as can be seen from (162 – 168)seconds. In this case the algorithm is unable to adapt and will not kick. The reason for this recording error is unknown due to time restrictions.

It is worth noting that the implementation of this algorithm as it currently stands hinges entirely on the asymmetry of the positions. This is a poor method to use and the proper way to implement this would be by extracting the 0.8Hz frequency corresponding to the accelerometer data, then halving the frequency



PositionAccelerometer-eps-converted-to.pdf

Figure 5.7: Response of algorithm to Z accelerometer data, doesn't react when poor values are recorded.

of the signal and using this as the substitute for the large encoder value.

5.6 Increasing QP Rotational

5.7 Increasing QP Parametric

5.8 Maintain

George Sheppard

TODO: THIS SECTION

5.9 Decreasing

5.10 Dicussion

6 Double Pendulum

6.1 Start-up

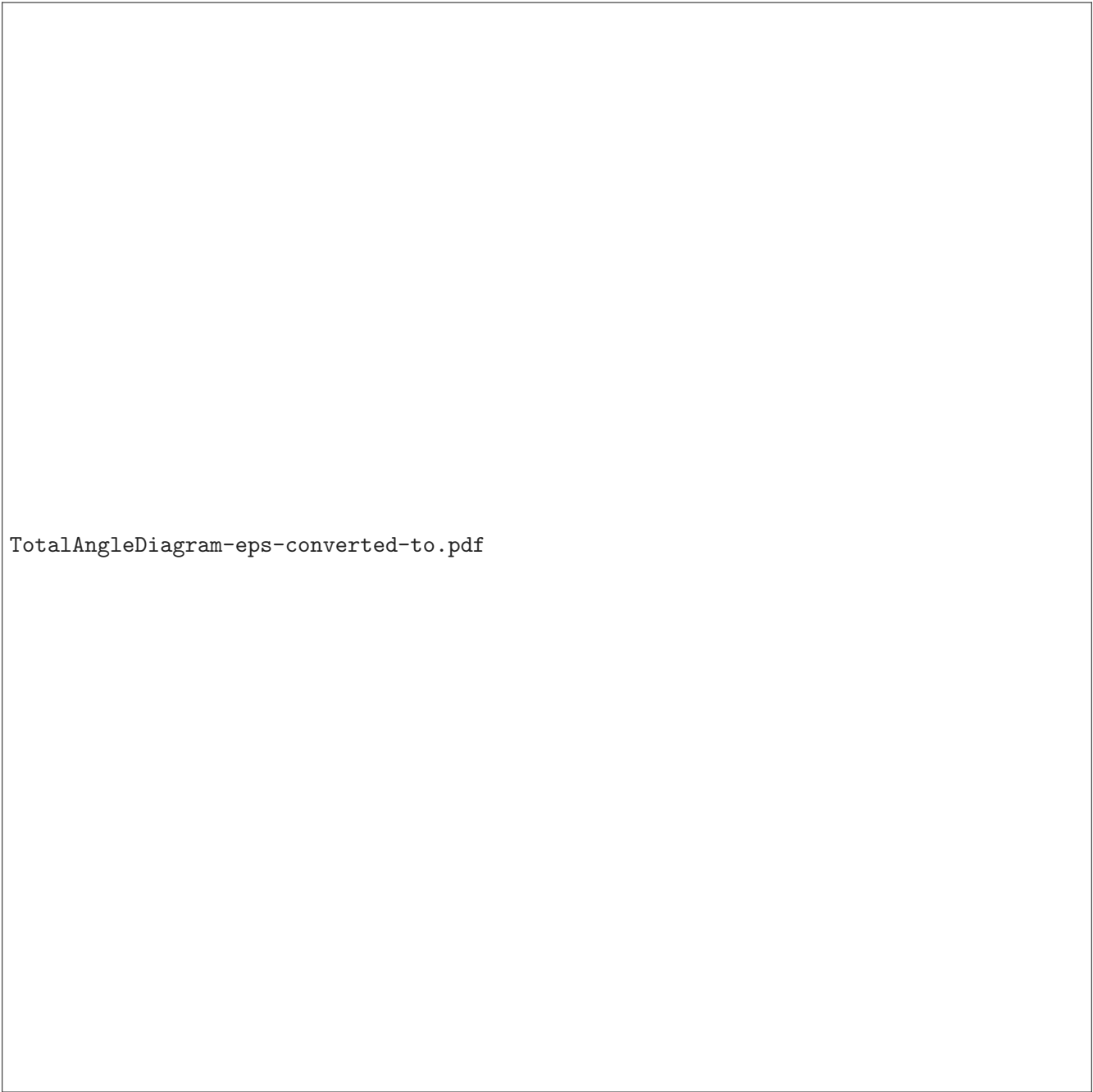
James Doering

How is this different to the single pendulum version? Need data on low angle oscillations. How should tactics change?

6.2 Adjusting Algorithms

George Sheppard

To adjust for the new setup of the swing, the single pendulum algorithms needed to be adjusted such that they would account for the motion of the swing below the hinges. For this reason a new coordinate system was defined that took into account the angle of the small encoders. This new coordinate system was defined by an angle named the 'total angle', and a diagram of this angle is shown in figure ?? . The large encoder value used previously in all single pendulum algorithms was then replaced with this new total angle to create the new algorithms.



TotalAngleDiagram-eps-converted-to.pdf

Figure 6.1: Adjustment made to the angle used in calculating the maximas, for single and double pendulum.

6.3 Increasing QP Rotational

From initial experimentation with the double pendulum, it was found that the period of the motion depended largely on the specific motion of the swing below the first now opened hinge. This motion is large when Nao implements the rotational motion seen in section ??, and as such the angular velocity timing method was decided to be optimum for a rotational motion, as the quarter period was difficult to determine and varied a large amount.

The adjusted angular velocity based on the total angle was used to decide when Nao has reached the maximum of his swing. For an efficient motion, the speed of Nao's motion had to be reduced. This is because when Nao was switching postures at the maximum speed, the motion of the rods below the first hinge became very chaotic, and ultimately caused a reduction in the total swinging angle. By reducing the posture switching speed to half of the maximum this motion was mostly under control, such that there was mostly a smooth swing.

Figure TODO: THIS FIGURE ?? shows the results for this method.
TODO: RESULTS OF THIS TODO: PROS AND CONS OF THIS

6.4 Increasing QP Parametric

6.5 Discussion

7 Future Suggestions

Code Team

There are several ideas that given a longer amount of time we would have liked to investigate, the first of which is an algorithm based on accelerometer and gyrometer values only. This would be interesting as it would remove the need for any external inputs such as the hinge encoders, and would allow Nao to be operate any swing without additional setup.

David: Improvements to the seat of the swing could be made in order to improve the stability of Nao when changing positions. During the current project tape was required to secure Nao but with a re-milled seat, grips could be made to attach to the rear of the hips and prevent any lifting motion. A CAD designed attachment could also be made for Nao's torso and feet in order to weights to be quickly added and removed. This would allow much greater amplitudes to be reached will securing the weights to Nao's body. It could also be beneficial in future projects to instigate machine learning teams to use internal sensors as inputs for the training, this could effectively mean Nao could be self sufficient and Not rely on any external sensor inputs.

The flexible shaft on the encoder adds a lot of sideways motion to the swing, and makes it difficult to maximise energy transferred to the swing, especially in the case where all the hinges are open.

A Appendix

A.1 Wiki

George Sheppard

Throughout this project, there were a large amount of issues found when trying to connect to: Nao, Webots, and the hinge encoders etc. For this reason a wiki was created to document how to set everything up correctly. This wiki is available at: <https://github.com/GeorgeSheppard/Robotics>. Alongside this wiki is the final code, including how to use it, the report, and any analysis files used to create plots. Feel free to clone this repository and add to it such that there remains a comprehensive guide for each year.

A.2 Webots

James Doering

Webots offers a substantial amount of uses - throughout this project Webots was used to test the interface and algorithms away from the lab, and to find accurate estimates for NAO's centre of mass in different positions. However, Webots did not offer much in useful physical simulations - the simulated swing would often fail, and physics would quickly go out of sync if the target frame-rate was missed momentarily. In order to interface with Webots from the NAOQI python library (supplied by Aldebaran/SoftBanks), the NAOQISIM controller was required - this acted as a 'virtual robot' to translate from Python script written with NAOQI to the NAOQISIM controller in Webots, which controlled the simulated NAO. This proved difficult to achieve, only being possible with certain software configurations. It is highly recommended that this is done on a linux operating system, otherwise there will be errors between 32 and 64 bit versions of python and NAOQI/NAOQISIM. The version of NAOQI required to connect to Webots is different to the version required to connect to the real NAO - as of writing, Webots/NAOQISIM requires NAOQI 2.1.4.13, whereas NAO requires NAOQI 1.14.5. Further explanation can be seen on the wiki decribed in section ??.