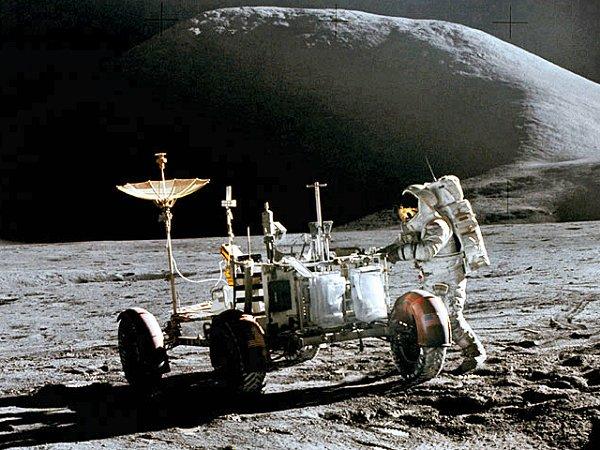
**Instrument Deployment Arm on Lunar Rover**

**2009**

Bill Pang Garrette Marcotte Justin Leung

**MAE 345 Midterm Project**



**Abstract**

* Major approach and findings

Table of Contents

[**I.** **Introduction** 3](#_Toc246787242)

[**II.** **Arm Configurations** 4](#_Toc246787243)

[**2.1** **Overview of Components** 4](#_Toc246787244)

[**2.2** **Designs of Individual Mechanical Parts (Justin)** 4](#_Toc246787245)

[**2.3** **A model of the Arm used in Simulation** 4](#_Toc246787246)

[**III.** **Materials (Justin)** 5](#_Toc246787247)

[**IV.** **Sensors and Actuators (Justin)** 7](#_Toc246787248)

[**V.** **Mechanism of Action** 7](#_Toc246787249)

[**5.1** **Overview** 7](#_Toc246787250)

[**5.2** **Calculating the Reference Position** 7](#_Toc246787251)

[**5.3** **Generating Angle Signals** 7](#_Toc246787252)

[**5.4** **Rocker Bogie** 7](#_Toc246787253)

[**VI.** **Control Logic (Garrette)** 14](#_Toc246787254)

[**VII.** **Arm in Action** 15](#_Toc246787255)

[**7.1** **Simulation** 15](#_Toc246787256)

[**7.2** **Power Consumption (Garrette)** 16](#_Toc246787257)

[**VIII.** **Major Pitfalls** 17](#_Toc246787258)

[**IX.** **Feasibility and Future** 17](#_Toc246787259)

[**X.** **Conclusion** 17](#_Toc246787260)

[Works Cited 17](#_Toc246787261)

[Appendix: Matlab code for Angle Calculator 18](#_Toc246787262)

1. **Introduction**
   1. Background and motivation
   2. High Level Logic Flow inside Robot



* 1. Organization of paper

1. **Arm Configurations**
   1. **Overview of Components**

Assumption on rover dimension and position of arm within rover:

Rover is about 1.2m wide, and length doesn’t matter…but probably 1.5 m

The wheel width is about 20cm. The important point is, we have about 70cm width between left and right wheels to work with. The chassis is lifted off ground by at least 10 cm, probably 15. This helps rover to get over obstacles, and also gives us space to place the arm.

Arm will be anchored beneath the vehicle, 5 cm to the **right** of **left** forward wheel, 5 cm from forward edge of rover, 5 cm from ground.

Actually the anchor is not stable because the rocker-bogie bounces, so the rover’s chassis bounces with it. But as a first approximation we assume the base of the arm to be stable, thus uses it as ground. Later we may need to model the rocker-bogie spring as well.

Model’s Orientations:

Forward: positive x.

Upward: positive y.

Right: positive z.

(left handed system, but Matlab uses it)

Joint 1: Anchor to neck piece

1 DOF, rotate about y-axis

Curled up position: theta = 90.

Forward position: theta = 0.

Range of motion: 90 ~ -60 (limited by the presence of left forward wheel)

Neck piece: shape in model is cylindrical, 10 cm, 5 cm diameter.

Joint 2: neck to upper arm

1 DOF, rotate about z axis

Curled up position: theta = 0.

Pointing straight forward: theta = 0

Pitching upward: theta > 0.

Range: 0 – 75

Upper arm: shape in model is cylindrical. 40 cm, diameter 5 cm

Joint 3: upper arm to lower arm

1 DOF, rotate about z axis

Curled up position: theta = 0.

Straight forward: theta = 0.

Bent down: theta < 0

Range: 0 -180

Lower arm: shape in model is cylindrical. 40 cm, diameter 5 cm

Joint 4: lower arm to manipulator

1 DOF, rotate about z axis

Curled up position: theta = 0.

Range: 360 deg rotation.

Manipulator: cylindrical, length 15 cm, center of mass attached to lower arm.

Material:

In reality: use 2 parallel beam instead of cylinder, to save mass. Apparently Darren said that saving just a little bit of mass saves a LOT of money in this context.

Maximum forward reach from underpinning of arm to reference position (tip of drill, when contracted) 127.5 cm

* 1. **Designs of Individual Mechanical Parts (Justin)**
  2. **A model of the Arm used in Simulation**

Links are hollow cylinders made of carbon fiber.

So the total mass of the arm is around 4.5kg + weight of sensors and actuators.

This is very light weight.

* 1. **Range of Action**

1. **Materials (Justin)**

Carbon fiber: Fiber-reinforced materials such as carbon, aramid and glass composites have the highest strength and stiffness-to-weight ratios among engineering materials. For demanding applications such as spacecraft, aerospace and high-speed machinery, such properties make for a very efficient and high-performance system. Carbon fiber composites, for example, are five times stiffer than steel for the same weight allowing for much lighter structures for the same level of performance. In addition, carbon and aramid composites have close to zero coefficients of thermal expansion, making them essential in the design of ultra-precise work stations.

1.78 g/cm^3.

1. **Sensors and Actuators (Justin)**
   1. Sensor Choice: Technical Specifications
   2. Sensor Choice: Advantages and Disadvantages
   3. Actuator Choice: Technical Specifications
   4. Actuator Choice: Advantages and Disadvantages

Detailed list of parameters and ranges

1. **Mechanism of Action**
   1. **Overview**
   2. **Calculating the Reference Position**

A core part of action mechanism is to figure out what angles each joint needs to be at each moment in time in order to reach the reference position for a given rock surface positioned at

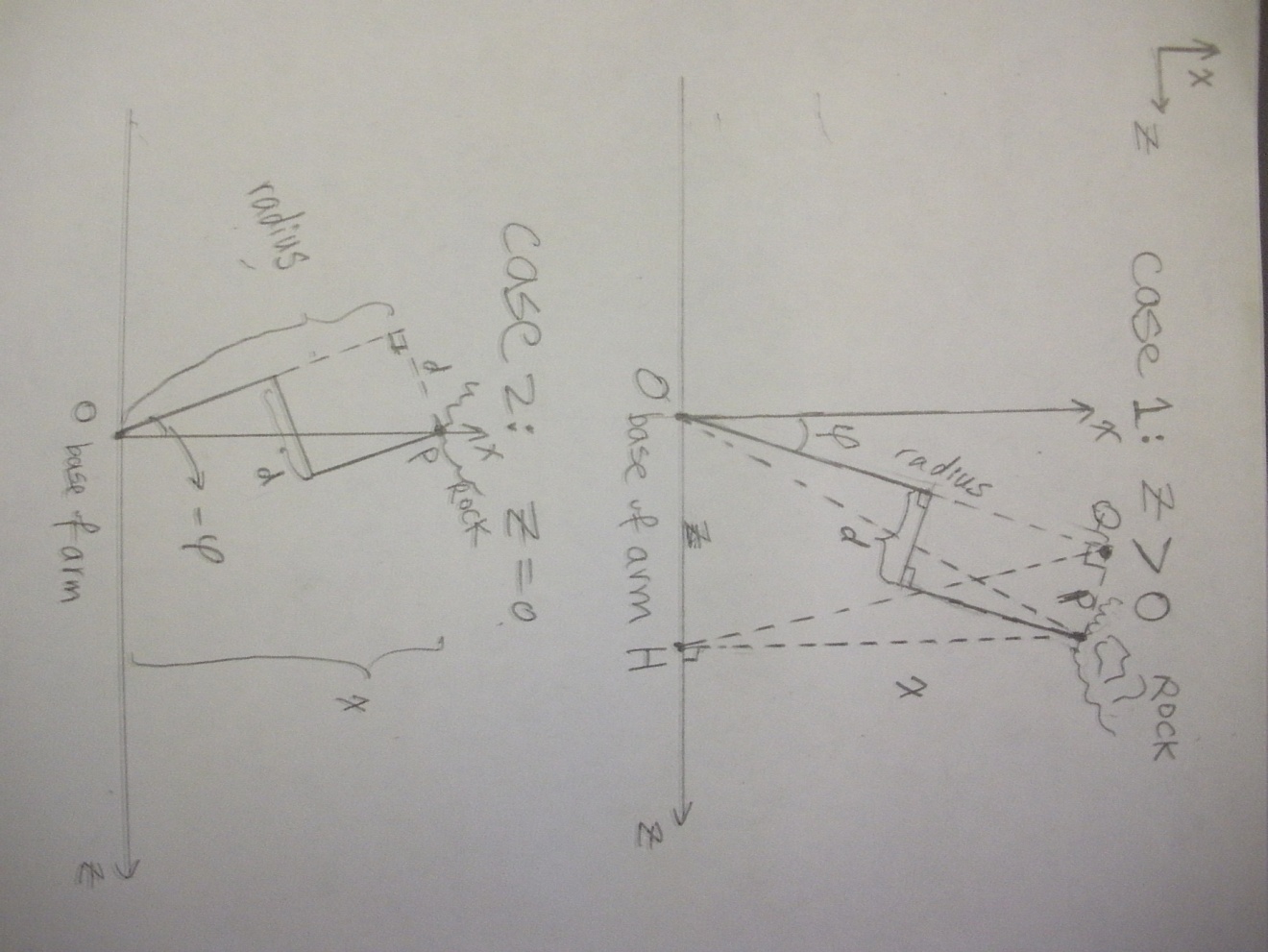
(x, y, z). The first step in this procedure is to figure out the FINAL angles we need each joint to have, when the arm is AT the reference position.

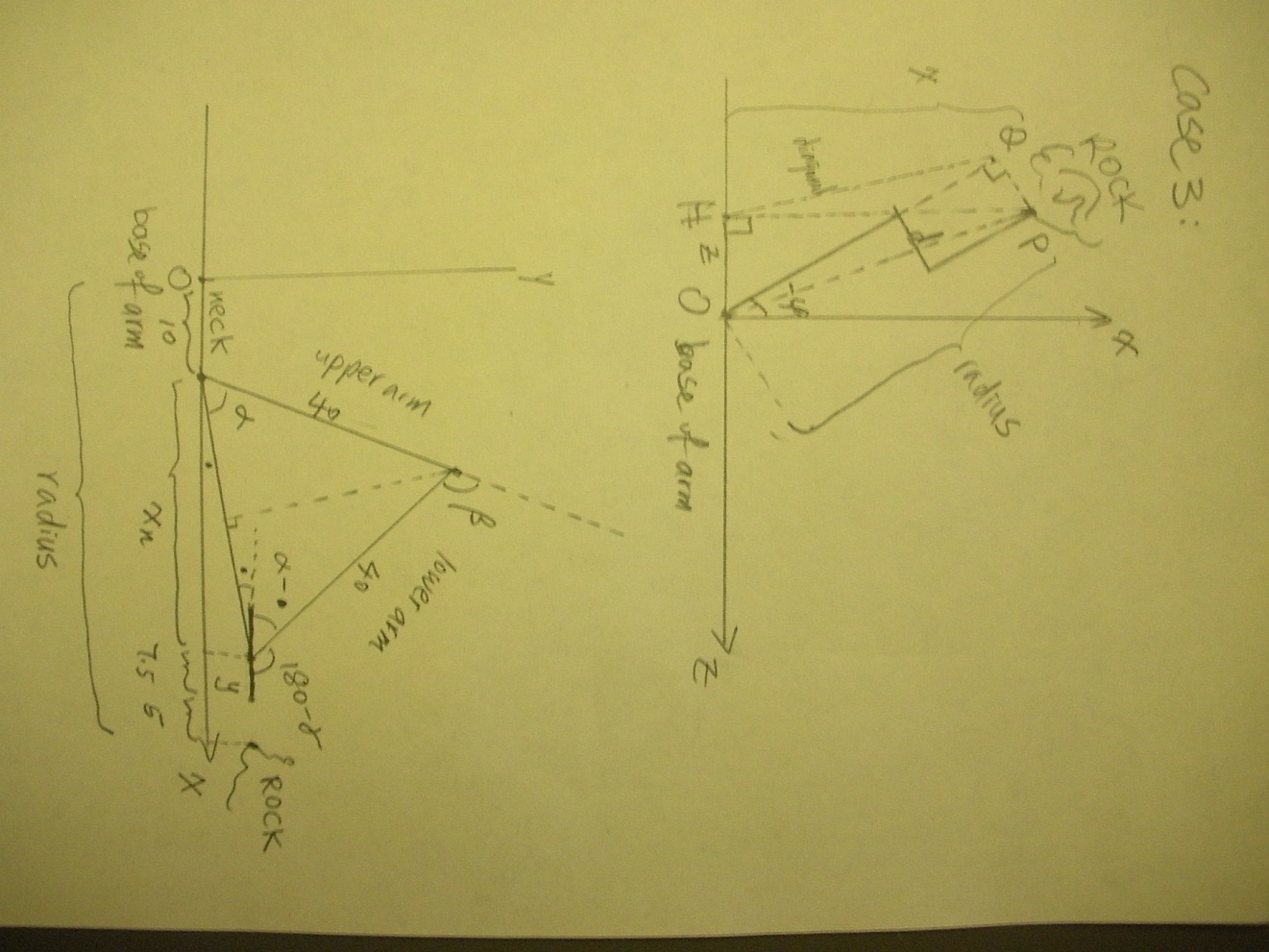
This part is non-trivial, and is analyzed below. For the matlab code implementing the ideas presented below, see Appendix A.

Step 1: Ignore y coordinate (vertical) for now. Using only the x and z coordinates of rock surface, we can already determine the neck yaw angle (φ) needed, as well as the “radius”, which is defined as the (axial) distance from base of arm to rock *as if there were no lateral displacement*. It is drawn in the diagrams below. Also, lateral displacement occurs in 2 parts, 1 on the connector between upper arm and lower arm, which has length 4, the other on the connector between lower arm and manipulator, which has length 8. So total lateral displacement = 12cm in this design. In the derivations to follow, we combine these 2 displacement into 1 equivalent lateral displacement, d.

Thus the current problem is: given x, z, d, find φ and radius.

We break the situation down into 3 cases because each case has a slightly different geometry: z >0 (case 1), z=0 (case 2), z<0 (case 3).



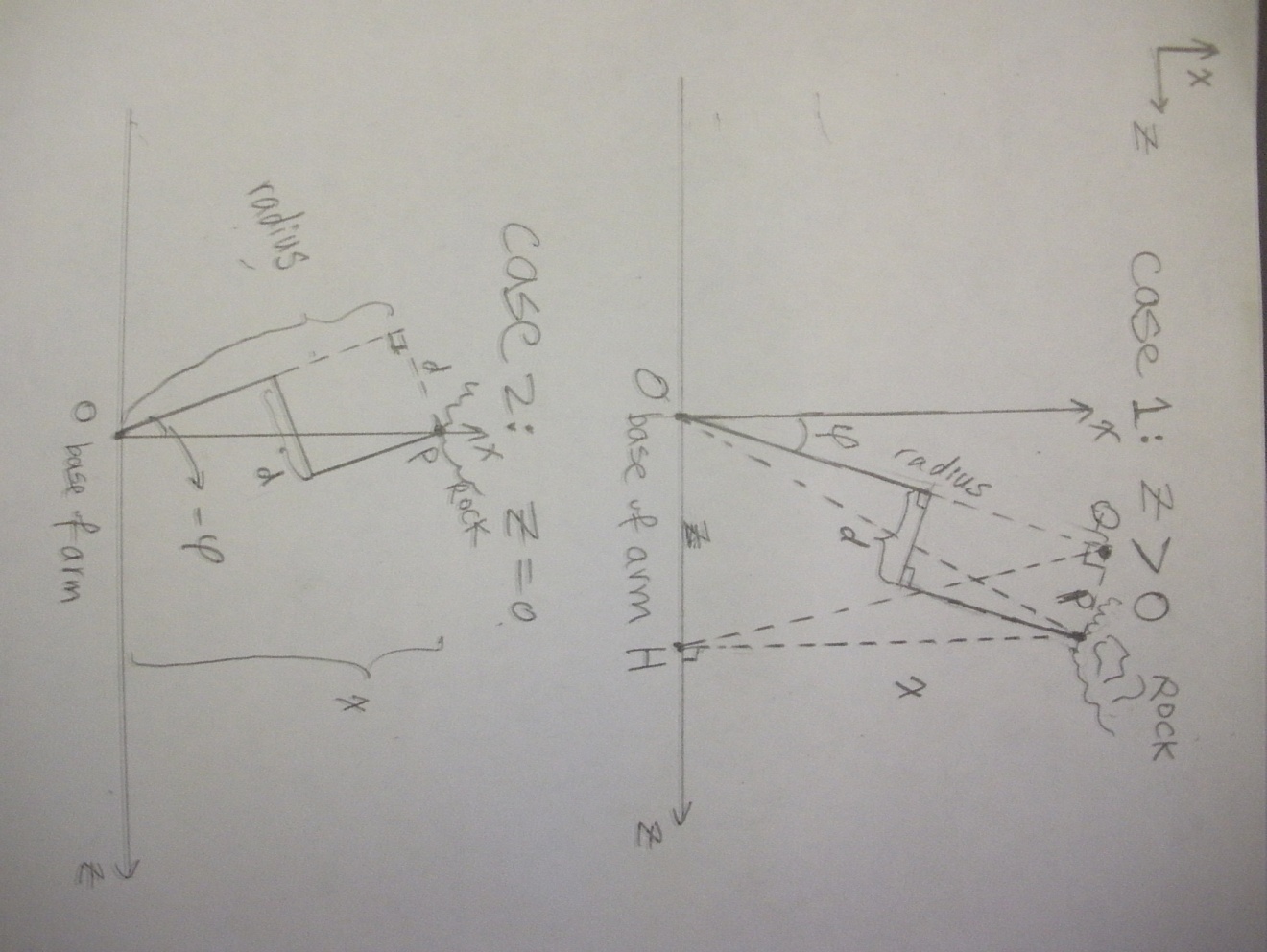


In case 3, we have z to the left of the x axis, so much of the calculation remains the same, but some signs we reversed:

Again we have OHPQ is a cyclic

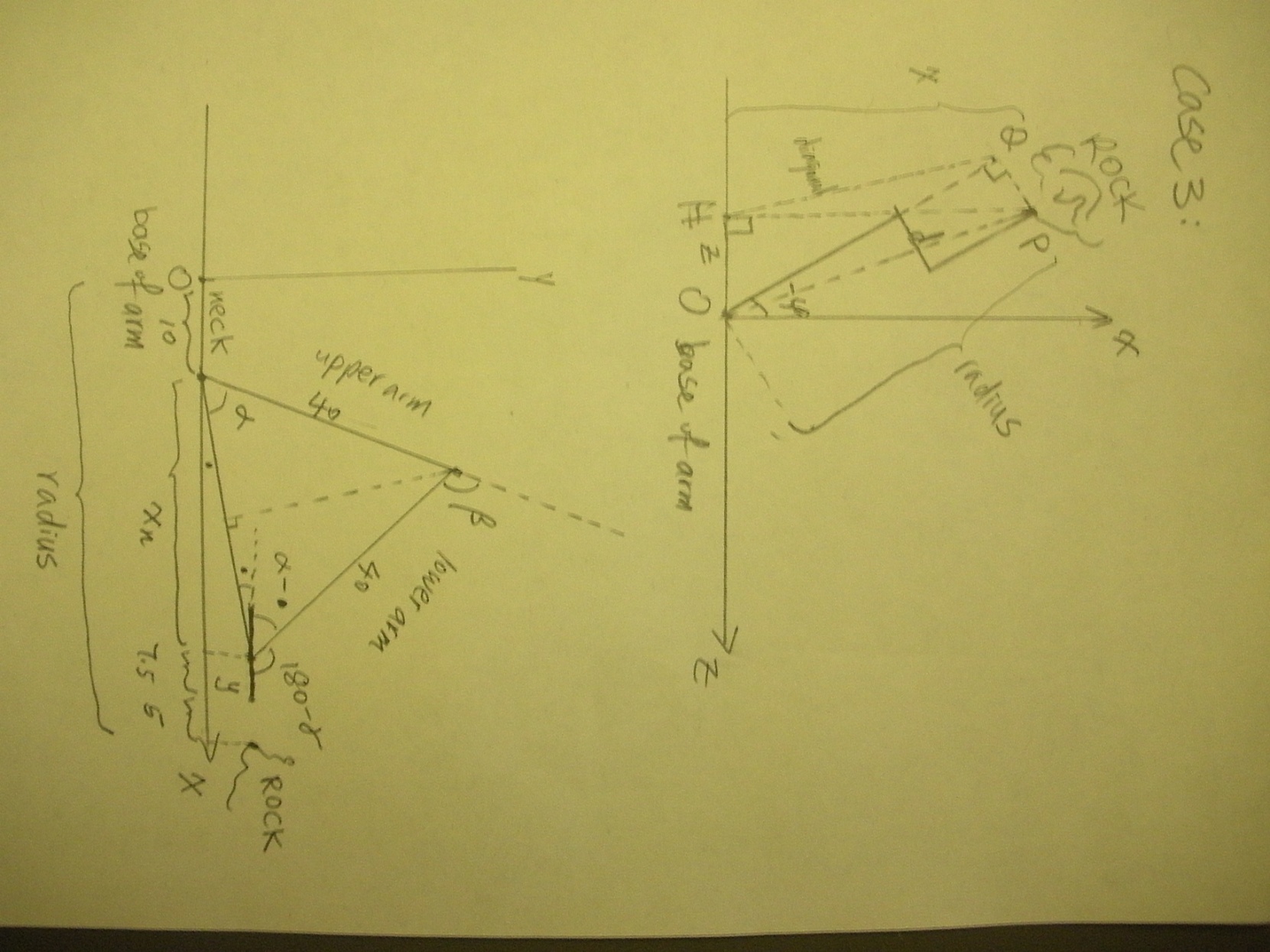
Quadrilateral, because of the 2 right angles at Q and H. We first find radius OQ, then apply Ptolemy’s Theorem and Cosine Law to find φ, which is negative, because we (or matlab) has defined the axis and rotation such that when the arm is pointing along x axis, φ is 0, and when it turns to the left, φ is negative. We have:

And radius is given by the same formula.

The second case is when z = 0. Notice that we can no longer divide by z, as is done in the formula above. The geometry is the following:

Step 2: Now having settled yaw angle and radius, we look at the vertical (x-y) plane at the yaw angle, and see what angles of the other joints are required to have the drill head reach reference position. Firstly, we recognize that the radius calculated is the distance “along” the arm in the forward direction, but includes the length of the neck piece, half the length of the manipulator, and the 5 cm from reference position to the rock. So we subtract these out to find the *net* x-distance, called xn: xn = radius – 10cm – (15/2)cm – 5 cm = radius – 22.5 .

Refer to diagram on the next page.



Note: in this diagram, the x axis is NOT the same x axis as before; it is simply the axis ALONG the direction of the arm. It is only equal to the x axis when neck yaw angle = 0. It is called the x axis here, perhaps confusingly, because most of the time we are dealing with arms pointing straight forward, and it is convenient to decompose motion into components of “along the arm” and “perpendicular to it”. As the reader will see, this different definition of x axis makes no difference in the calculations to follow.

We subtracted all these components off radius to get xn, because xn is what’s relavant in calculating angles of the upper arm, lower arm, and manipulator.

It is important to note how the radius in this diagram corresponds to the radius defined and calculated in the previous part. Convince yourself that they are indeed the same distance.

So the problem at this stage is: given xn, y, and arm lengths, find joint angles .

The result immediately follows from the diagram:

So

Note: is the angle labelled with a dot in the diagram.

* 1. **Generating Angle Signals**

So for given coordinate of rock position (x, y, z) relative to base of arm, we have caluclated the required joint angle at the end position. Now the question is how to reach there, from the curled up position.

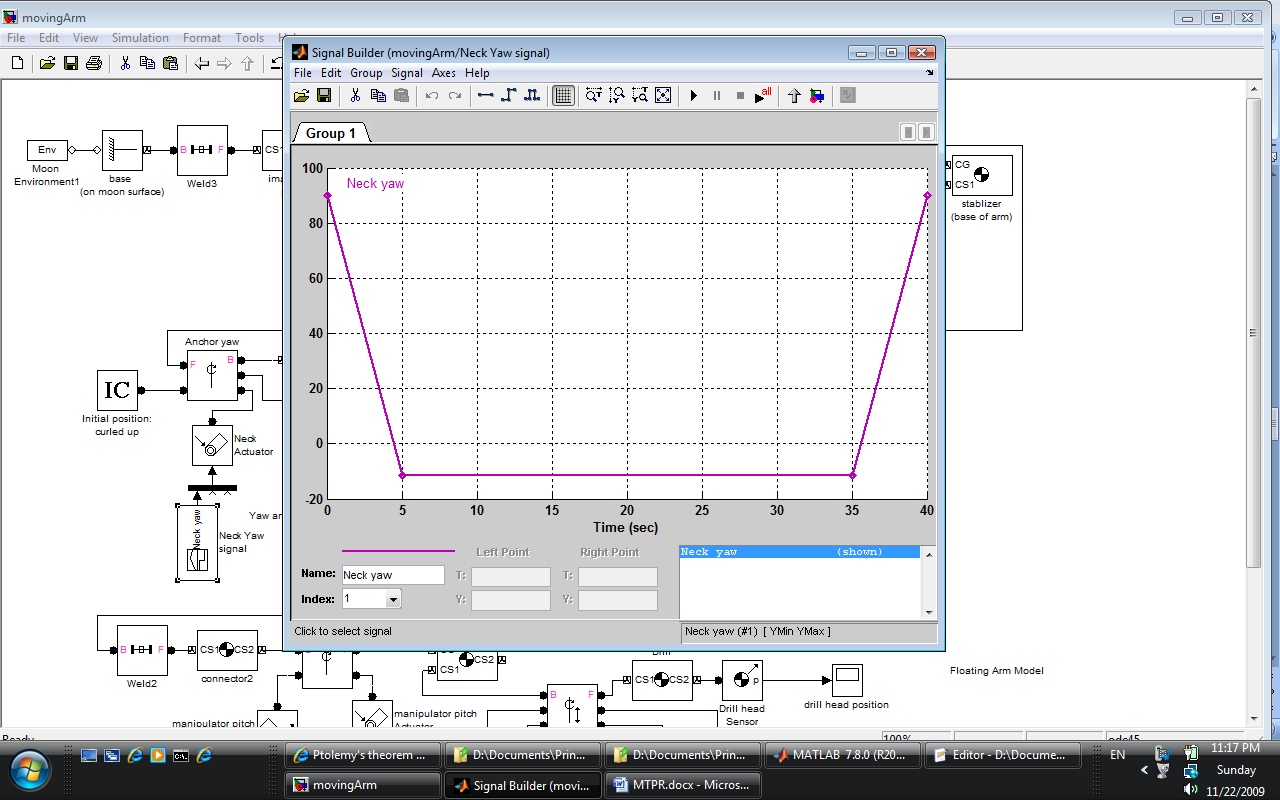
We need to take into account the phyiscal limit on movements of links. But even then, there are multiple feasible plans. We came up with the motion plan described below to transfer the arm from its curled up position to the reference position with angles just caluclated, with the following considerations in mind:

* This transfer plan is modular: each joint moves into position sepaerately to the extent possible, so easy to control.
* This transfer plan does not involve any part of the arm going too low in the y direction, which would mean hitting the ground.
* This transfer plan avoids hitting the vehicle too.

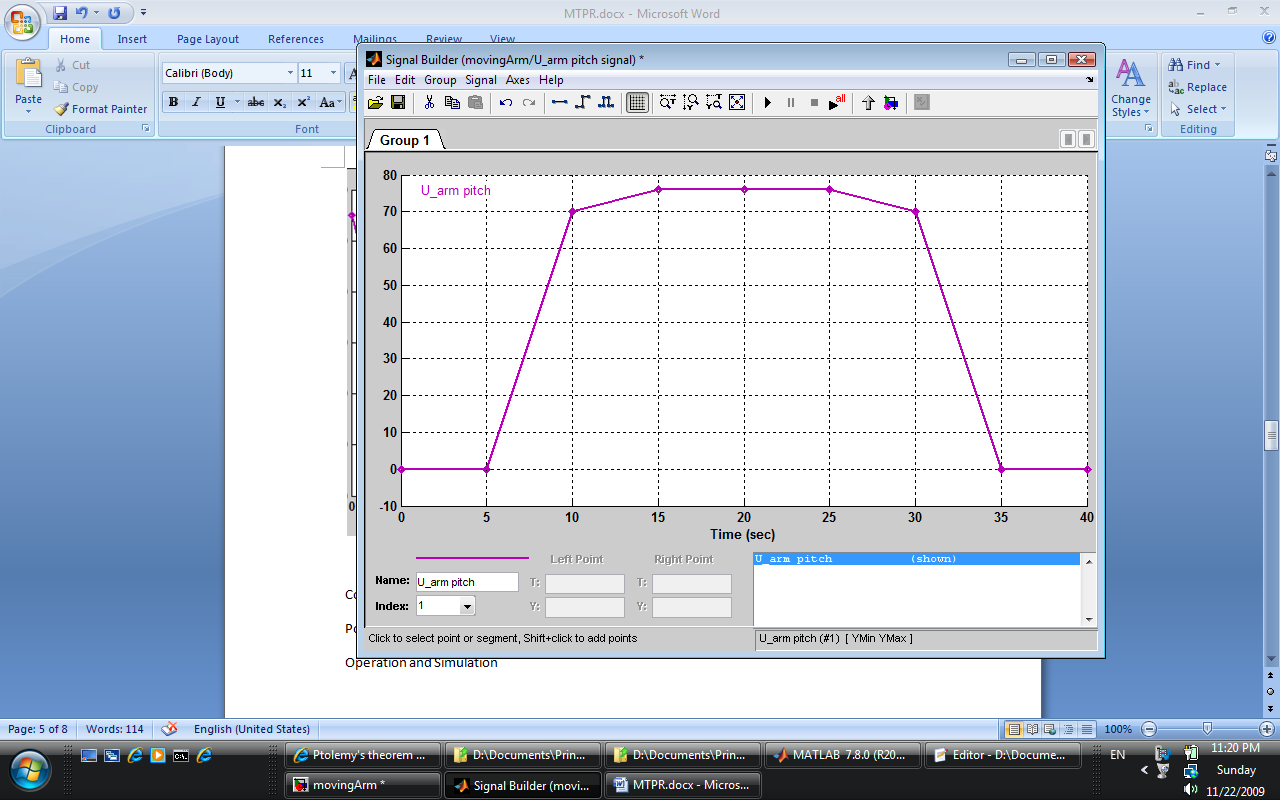
Transfer plan: The arm will start from the curled position, extend out to engage target, drill into rock, take pictures, then curl back up, in 40 seconds. This time interval can be easily changed as need arises.

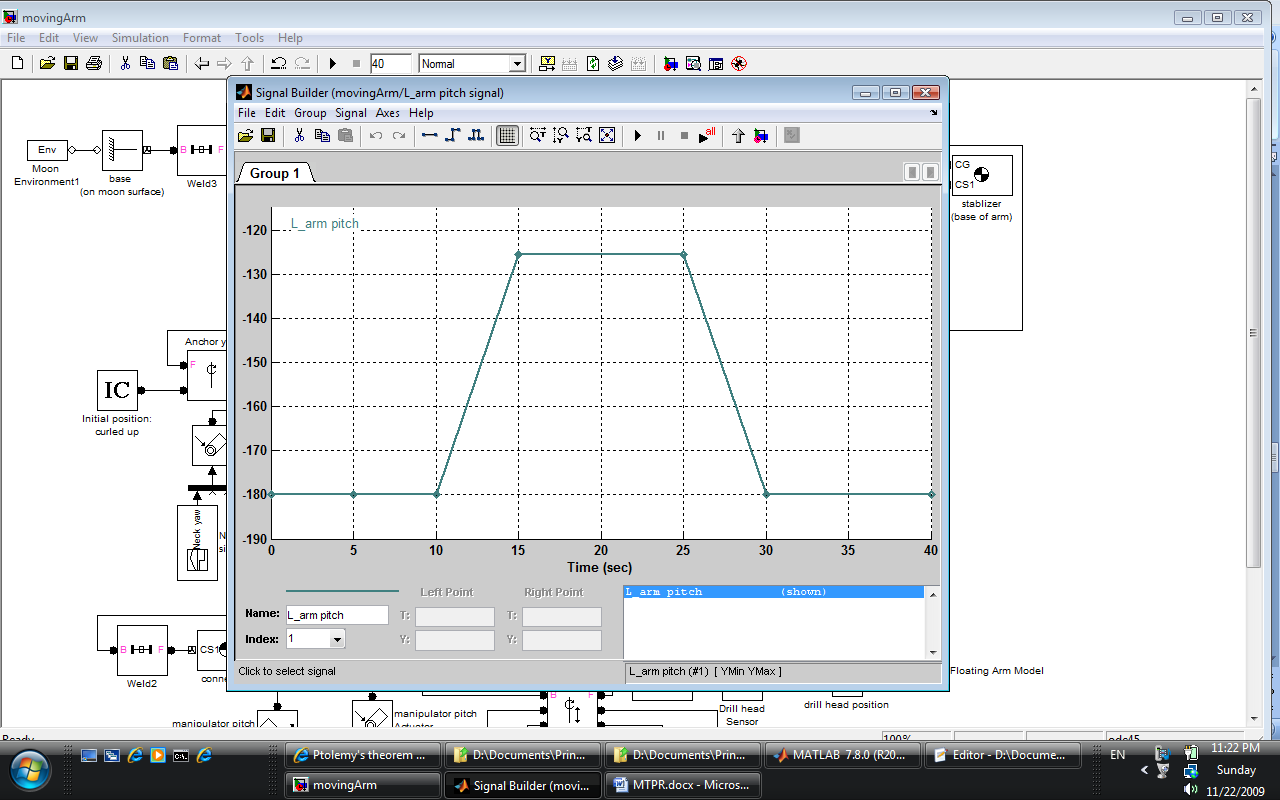
Stage 1: arm rotates outward to face forward, in the x-direction. This correpsonds to only a neck yaw movement form 90 degrees to , at uniform speed, from t = 0 to 5 sec.

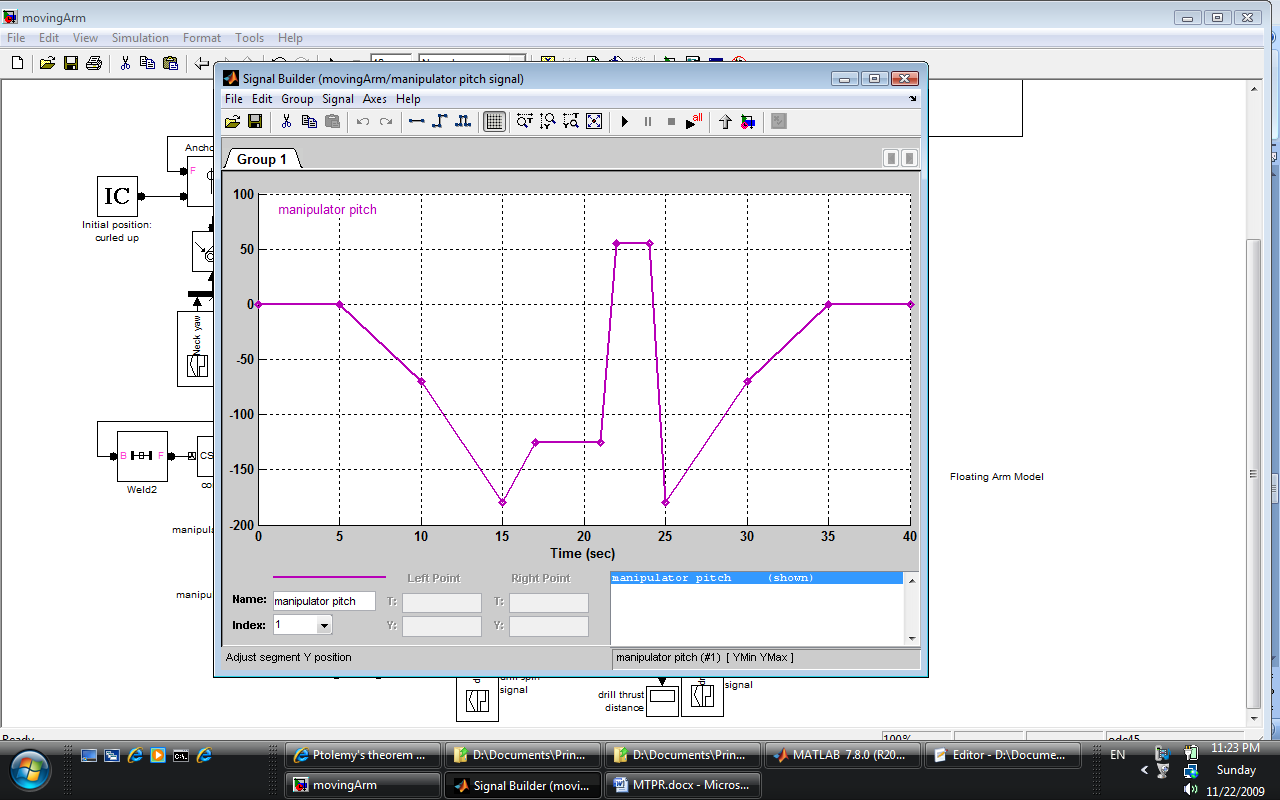
Stage last: this is the complement of stage 1; the arm rotates inward back underneath the rover. This means a neck yaw from to 90, at uniform speed, form t = 35 to t = 40 sec.

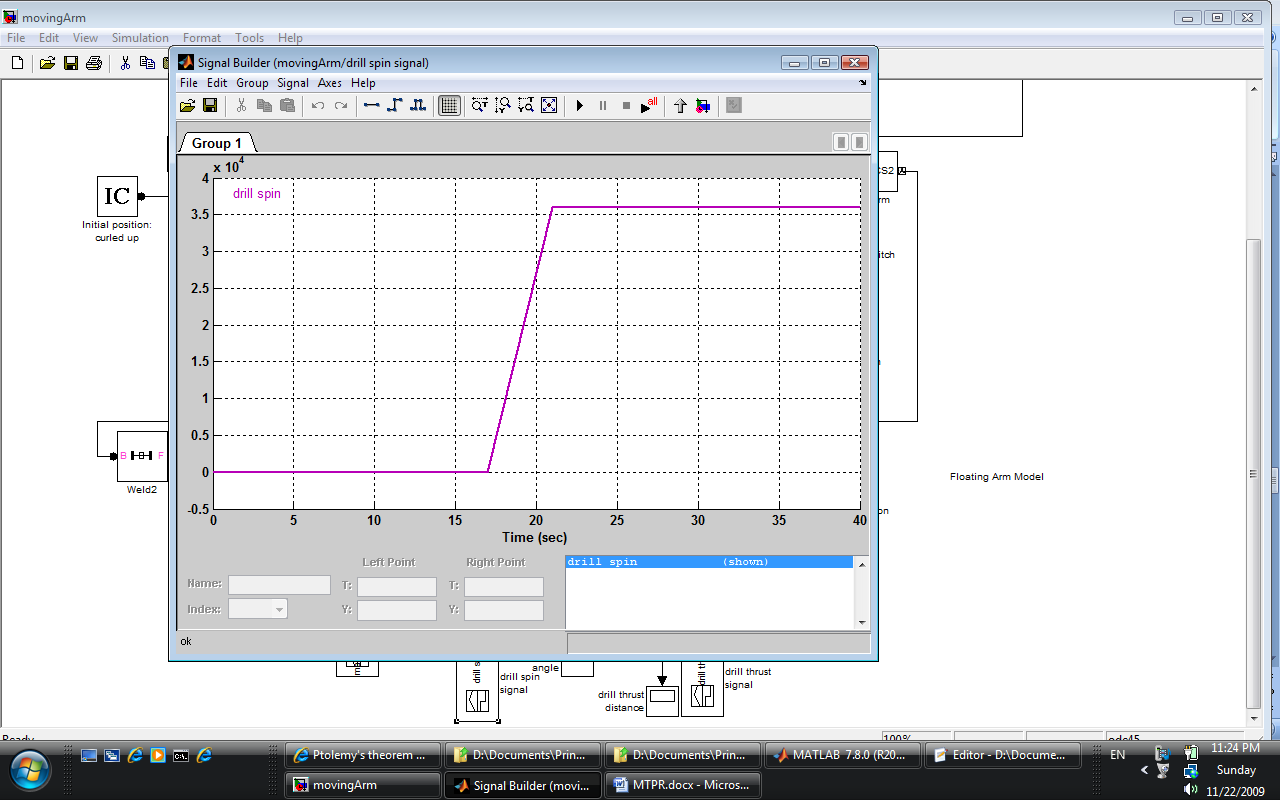
The required yaw angle for the neck yaw joint for the 40 sec duration is displayed below:

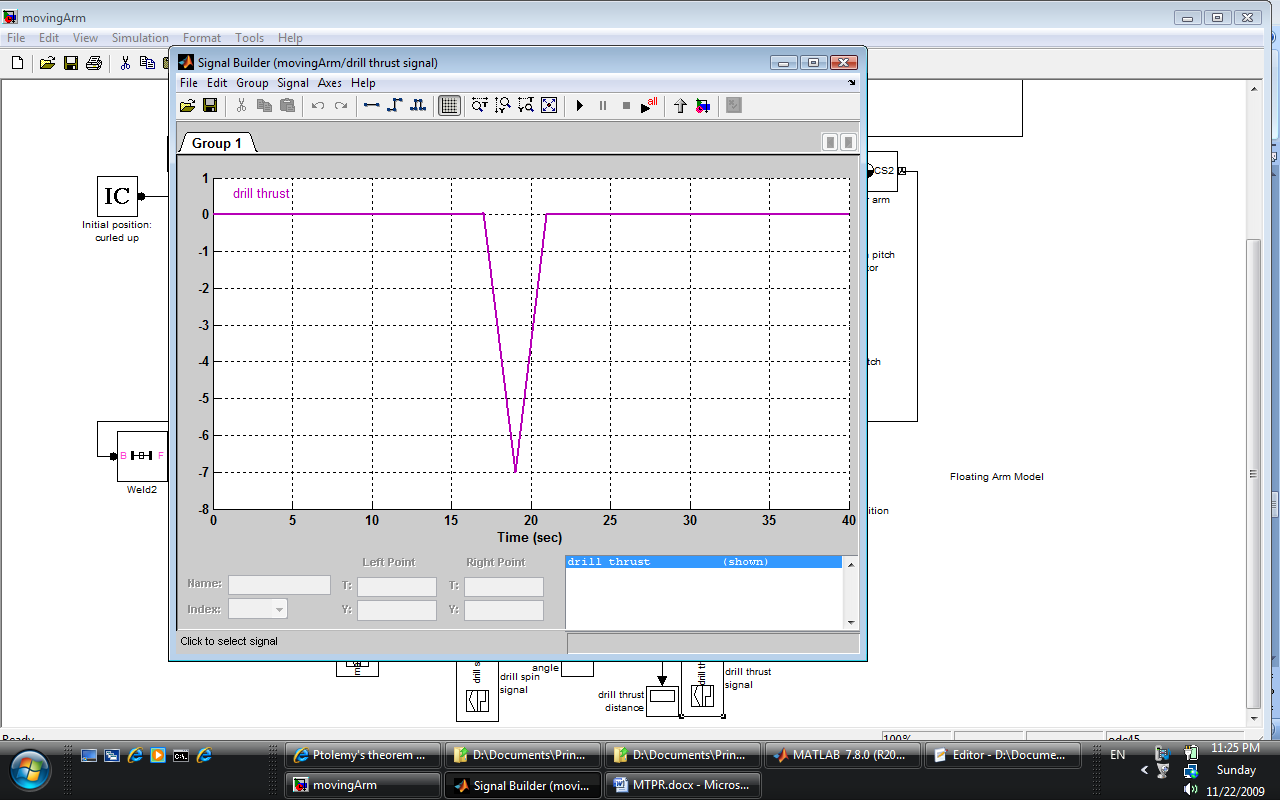
Stage 2: upper arm raises from flat to 70 degrees up, to facilitate later extension of lower arm (without hitting the ground). This occurs at uniform speed from time 5 to 10.



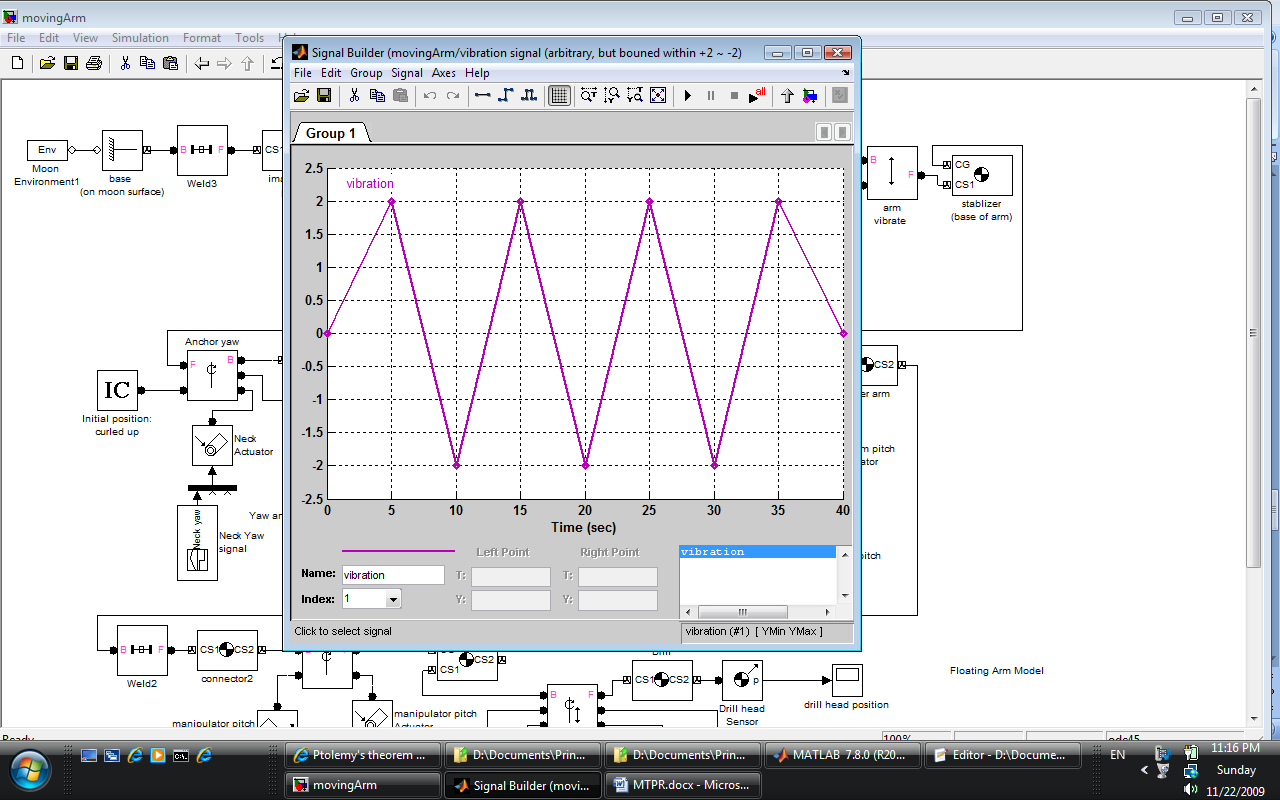


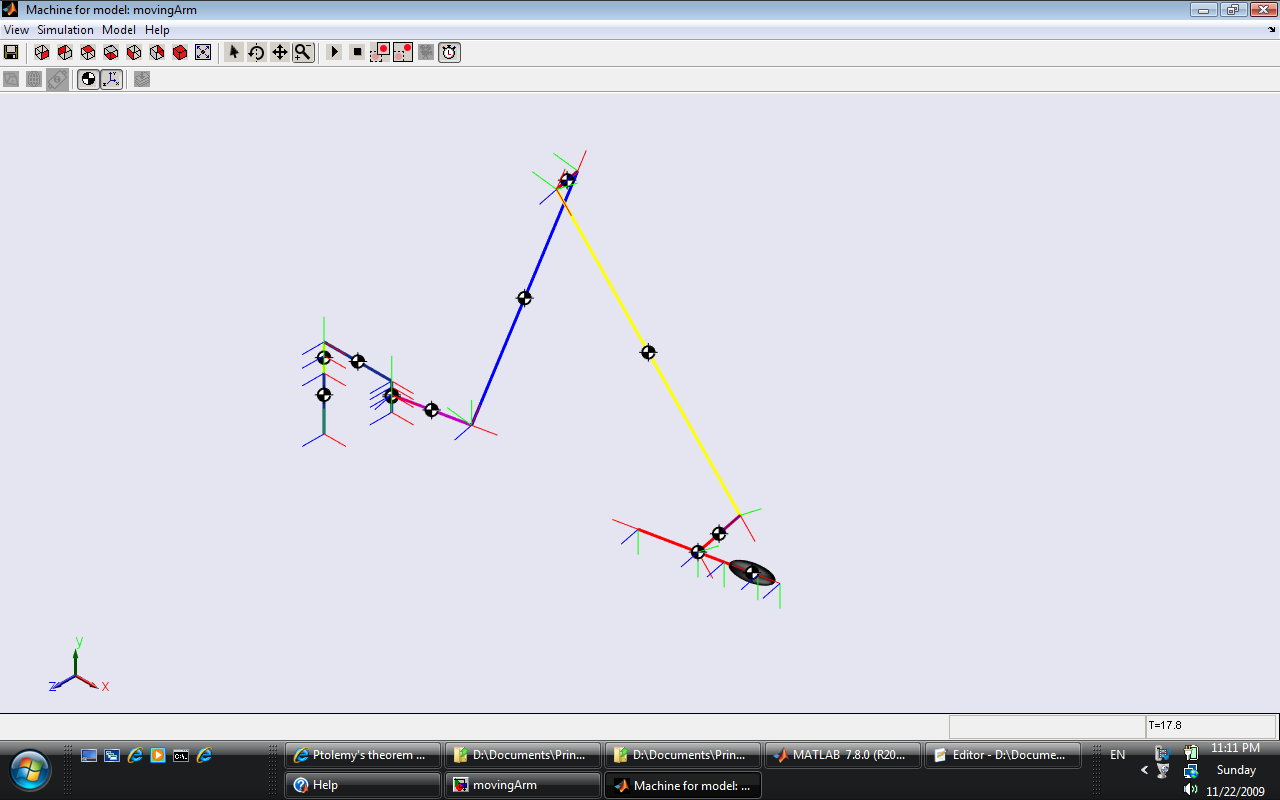


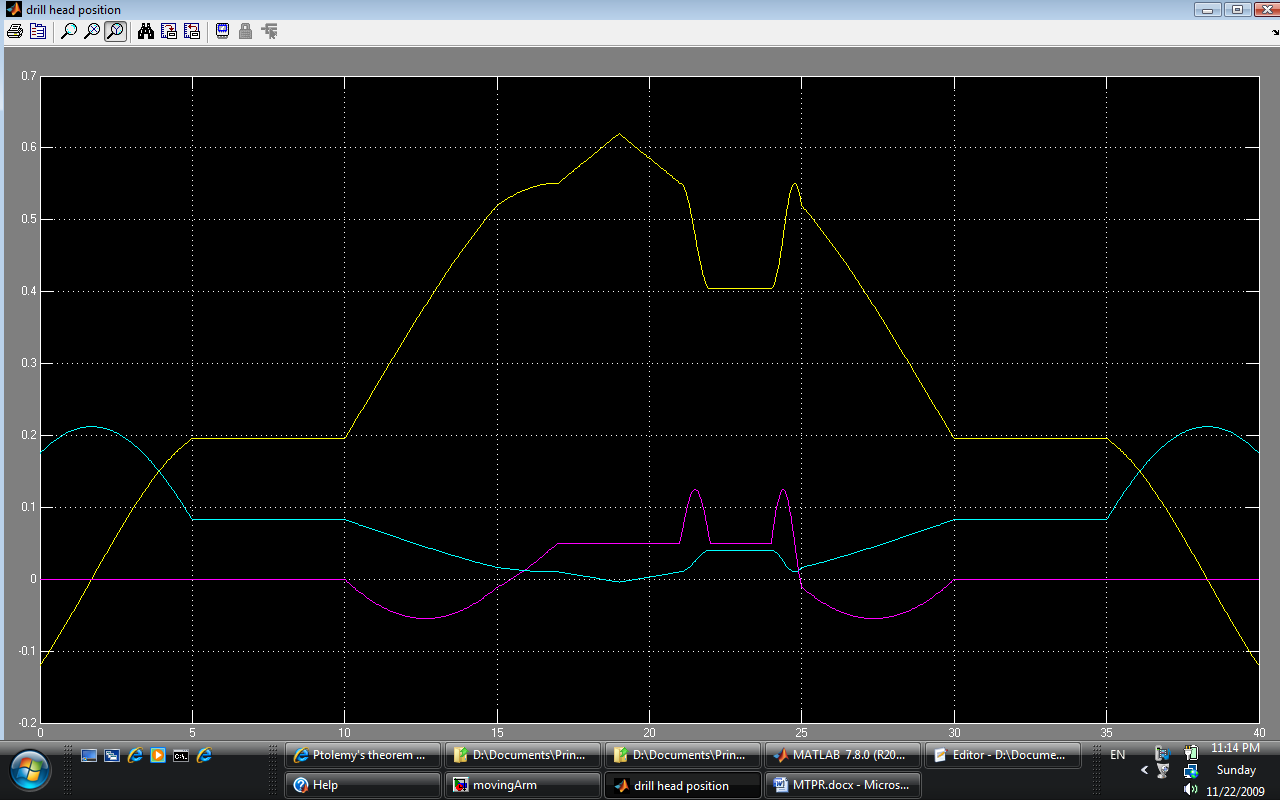




* 1. **Rocker Bogie**

****

1. **Control Logic (Garrette)**
   1. Algorithm
   2. Physical Aspect
   3. Treatment of Error
2. **Arm in Action**
   1. ******Simulation**

****

* 1. **Power Consumption (Garrette)**

1. **Major Pitfalls**
2. **Feasibility and Future**

(use equations, figures, tables, graphs, citations, and animations)

(cite all tables/equations/graphs borrowed. )

(format tables/graphs in appropriate manner to make information easy to understand. Explain their significance. )

1. **Conclusion**

Estimation of cost

# Works Cited

# Appendix: Matlab code for Angle Calculator

function [alpha beta gamma phi] = CalAngle(x, y, z)

%% x = forward distance from base of arm to rock surface

%% y = vertical distance from base of arm to rock surface

%% z = lateral distance from base of arm to rock surface

% alpha = inclination of neck-upperarm joint

% beta = bending down of upper-lower arm joint

% gamma = angle between lower arm and manipulator

% phi = yaw of anchor joint

% L = length of upper arm and lower arm

% N = length of neck

% R = distance from CG of manipulator to tip

% REF = 5 cm, specified in the problem

% OFFSET = lateral offset of the tip of drill from arm base

L = 40;

N = 10;

R = 7.5;

REF = 5;

OFFSET = 12;

radius = (x^2+z^2-OFFSET^2)^0.5;

xn = radius - N - R - REF;

beta = -180 + 2\*asin((xn^2+y^2)^0.5/2/L)/pi\*180

alpha1 = atan(y/xn);

alpha2 = acos((xn^2+y^2)^0.5/2/L);

alpha = (alpha1 + alpha2)/pi\*180

gamma = (alpha2 - alpha1)/pi\*180

diagonal = (x\*radius + OFFSET\*z)/(x^2 + z^2)^0.5;

if z > 0

phi = 90 - acos((radius^2 + z^2 - diagonal^2)/(2\*radius\*z))/pi\*180

elseif z == 0

phi = -asin(OFFSET/x)/pi\*180

else

phi = acos((radius^2 + z^2 - diagonal^2)/(2\*radius\*abs(z)))/pi\*180 - 90

end

end