Design and Control of a Semi-Passive, Heavy-Duty Paired Mobile Robot System with Application to Aircraft Wing Assembly

by

Manas Chandran Menon

B.S. Bio-engineering, The University of California, Berkeley (2003)
S.M. Mechanical Engineering, Massachusetts Institute of Technology (2008)
S.M Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2008)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

Author .....
Department of Mechanical Engineering
June 25, 2010

Certified by .....
H. Harry Asada
Ford Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
David Hardt
Graduate Officer, Professor of Mechanical Engineering

Design and control of a heavy duty paired mobile robot system with application to aircraft wing assembly

By

Manas Menon

Submitted to the Department of Mechanical Engineering
on June 25, 2010, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

We describe the development of a robotic system capable of performing a class of manufacturing operations. An example of such an operation is commonly found in aircraft assembly - this demonstrates the immediate applicability of this research.

The system utilizes a unique concept – a pair of mobile robots acting on opposite sides of a thin wall. The robots interact with one another through the use of magnetic fields that penetrate this wall. The 'inner' robot is untethered and is controlled by the 'outer' robot. Despite the significant mass of the outer robot, it operates without the aid of physical external supports.

Full modeling of the system is presented. We include calculations for forces and torques produced by sets of permanent magnets for any system state. Simplified, tractable versions of this model for the purpose of control are also described.

The system is designed to execute closed loop fine position control and large scale locomotion. Experimental results from a functional prototype verify the effectiveness of the design as well as the robustness of a position controller. Numerical optimal control results have been developed for high speed point to point trajectory motion.

This 'pair of robots' paradigm could be applicable to a variety of tasks. This work outlines analysis techniques that are useful for such a system at most scales.

Thesis Supervisor:

H. Harry Asada
Ford Professor in Mechanical Engineering

Thesis Committee

Steven Leeb
Professor, Department of Electrical Engineering and Computer Science

Maria Yang
Professor, Department of Mechanical Engineering

Acknowledgement

I'd like to extend my thanks to the Boeing Company for sponsoring my research. It is because of their financial support and flexibility that I have been able to work on such an interesting and stimulating project, and I greatly appreciate it.

I would never have come to MIT, and certainly would not have been able to survive here, were it not for all of the educators and experiences leading up through my undergraduate education. Though they are too many to name here, I must thank the teachers that inspired me through grade school and UMTYMP and my time at UC Berkeley. I would like to acknowledge especially Professor Ron Fearing and his lab in the EECS department at Berkeley for a great undergraduate research experience.

My advisor, Professor Harry Asada, has been an incredible instructor and mentor to me. His creativity is invaluable when I get stuck and inspirational when I'm discouraged. His practicality, doled in measured doses, brings me back to earth when I need it most. He never allows me to settle for mediocrity and has helped me to become a better engineer and researcher, and more importantly, a much more careful and critical thinker. The other members of my thesis committee, Professors Steve Leeb and Maria Yang have provided excellent advice in their areas of expertise, helping to round out my work in a way that would have been impossible on my own. Professor Marcus Zahn has also set aside time to meet with me and contributed significantly to my understanding of magnetic fields and forces.

My fellow lab members are a constant source of wisdom and insight as well as an often needed comic relief. I have never had the opportunity to work with so many selfless, creative, motivated and intelligent people as I have these past few years and I'm honored to be considered their peer. In particular I would like to recognize Geoffrey Karasic, Anirban Mazumdar and Binayak Roy, who were my colleagues on the Boeing project for several years, as well as the undergraduate researchers I supervised: Albert Wang, Matt Robertson and Todd Zusman.

Last, but certainly not least, I'd like to thank my family, especially my parents. It is difficult to put into words the gratitude I have for the way they have raised me. They are constant role models and teachers. They have given me an appreciation for math and science, food and art. They have taught me the importance of compassion and work ethic. They have given me the opportunity to live a full, happy life and I am forever indebted to them for it.

Table of Contents

List of Figures

II – 16: This figure shows the location of the electromagnet and steel plate used to demonstrate the clamping functionality of the system.

II – 17: CAD drawing of the single degree of freedom prototype showing the location of all components in the prototype

II – 18: Single degree of freedom prototype with labeled parts. The electromagnet and wire coils are hidden behind a mask.

II – 19: Plot showing the fine positioning ability of the single degree of freedom system. This demonstrated the effectiveness of using a Lorentz force for fine positioning.

II – 20: Diagram showing the forces on the unsupported system

II – 21: Forces on the moving unsupported system – this shows the nonlinear forces that result from a non-zero velocity

II – 22: Side view of the chosen design. The inner and outer robot are separated by a thin aluminum skin. From this view it is possible to see the alignment of permanent magnets between the inner and outer robots as well as the alignment between the magnets on the inner robot and the wire coil on the outer robot

II – 23: Top view of the chosen design. This view shows the layout of the permanent magnets and coils relative to one another.

II – 24: Effect of number of magnets on holding stability. This figure shows the difference in safety between having three magnet banks (minimum number necessary) and four magnet banks (allows for more safety). We chose to have four magnet banks on the prototype for this additional safety

II – 25: Halbach array. This figure shows the location of flux concentration as well as the location of reduced flux concentration for a Halbach array

II – 26: CAD model of 'magnet bank.' This is the basic magnetic building block used in this system. Aluminum housings were used for strength and rapid prototyped plastics were used for alignment. Assembly of these magnet banks was relatively straightforward.

II – 27: Lorentz force direction given a direction of current and a direction of flux.

II – 28: Lorentz force proof of concept demo. This simple setup was built to test that the Lorentz force we would be able to generate would be of appropriate order of magnitude for the application. The left side of the picture shows a fixed coil of wire. The right side shows a magnet mounted on a vertical linear rail. By driving the coil with an oscillating current we were able to observe oscillations in the magnet on the right side of the photo.

II – 29: Image taken from an FEA to find the flux in the coil. This was used only as a rough order of magnitude estimate to chose coil and core size.

II – 30: Use of laser to align robotic system. This figure shows how a robot mounted position sensitive device (PSD) can be used to locate a laser (the laser in this case would be mounted on a fastener, or fastener location).

II – 31: Beam splitter and mirror setup used to create a low form factor bi directional laser. This device was very useful for testing with our prototype.

II – 32: CAD model of functional prototype. Components are marked.

II – 33: Outer robot prototype suspended upside down. It is held in place through the skin due to magnets on the inner robot.

II – 34: Inner robot prototype.

III – 1: Inner and outer robots across the skin

III – 2: Inner and outer robot represented as rigid bodies. Note that the outer robot rear wheel contact is not marked – this is instead considered constrained to a plane. Inertia of the wheel on the outer robot is ignored.

III – 3: Definition of several components of the state

III – 4: Magnetic charge model description. Magnetic north and south pole faces are represented by areas with some 'magnetic charge.'

III – 5: Left image shows the fields due to single magnet. The top and bottom (north and south) poles of the magnet are shown as squares. Right image shows the fields due to a Halbach array. Again, north and south poles of the three magnets in the array are shown as squares. In both cases, to help visualization, the fields are only shown on a plane just above and below the magnet.

III – 6: For Maxwell's stress tensor, we must create a surface around one Halbach array and evaluate the magnetic fields at all locations at this surface. This figure shows the surface automatically chosen by the code and the fields at that surface.

III – 7: CAD showing setup in the Admet machine to find normal forces between a pair of Halbach arrays.

III – 8: Normal force measurements as a function of displacement for two different magnet grades.

III – 9: Comparison of simulated forces for a pair of Halbach arrays to the experimental data. Note that the simulation data was scaled to fit the experimental data at the minimum distance location (this corresponded to around 3 mm). After this scaling, however, the two curves matched well. This suggests that a reasonable design method is to measure the forces between a pair of magnets, use this

information to set the 'assumed magnet grade' (basically use this as a scaling factor) and then run analyses with this assumed magnet grade.

III – 10: Lateral restoring force versus robot misalignment, found in simulation.

III – 11: Expected response for a system with linear viscous and nonlinear coulomb damping

III – 12: Experimental results in the two cases (1) with eddy current (2) without eddy current

III – 13: Experimental data compared to an exponential curve fit (which would be expected in the case of linear viscous damping). The curve fits very well.

III – 14: Drawing showing the possible effect of eddy currents caused by motion of the *inner* robot affecting the force on the *outer* robot.

III – 15: Transfer function between velocity of inner magnet to force on outer magnet (forces due to trans-skin eddy effects) showing little difference between acrylic skin and aluminum skin. This suggests we can safely ignore these trans-skin effects.

III – 16: Diagram labeling dimensions on the outer robot.

III – 17: Diagram labeling dimensions on the outer robot.

III – 18: Components of forces on magnet banks.

III – 19: Components of torques on magnet banks.

III – 20: Reaction forces on outer robot. Note that reaction forces occurring at the rear axles signify that we ignore inertia of the wheels.

III – 21: Free body diagram showing the effect of ignoring inertia on the rear wheels of the outer robot.

III – 22: Outer robot chassis tilt found using assumed compliance in wheel elements. This is applied in the method of deformations.

III – 23: Total deflection of compliant elements, outer robot. This is applied in the method of deformations.

III – 24: instantaneous center of rotation of the outer robot is along the axels of the rear wheels. This is due to the applied no slip condition at these wheels.

III – 25: Diagram labeling dimensions on the inner robot.

III – 26: Diagram labeling dimensions on the inner robot.

III – 27: Magnet force and torque components, inner robot.

III – 28: Reaction forces inner robot.

III – 29: Inner robot chassis tilt found using assumed compliance in wheel elements. This is applied in the method of deformations.

III – 30: Total deflection of compliant elements, outer robot. This is applied in the method of deformations.

IV – 1: Strategy for point to point control of outer robot. First the pair of robots turns to align with the desired endpoint. Next, the pair of robots drives towards this endpoint in a straight line.

IV – 2: Reaction forces and their relationship to no slip and no fall conditions, outer robot.

IV – 3: Example optimal trajectory for point to point optimal control. Note that the outer robot initially leads the inner robot, until, near the end of the trajectory, it slows to let the inner robot pass. This allows both robots to quickly reach the desired endpoint with zero velocity.

IV – 4: No fall and no-slip conditions during execution of this optimal trajectory (in simulation). Note the safety factory added to each condition. Also note that the no-slip condition is far more strict.

IV – 5: Maximum torque the outer robot can apply before slippage or falling. The white region is the region in which, even at zero torque, the robot will fall. At (a) and (b), the robot falls due to reduce magnet holding forces because of gross misalignment between the inner and outer robots. At (c) and (d), the velocity is large enough to cause significant eddy current forces on the outer robot, applying a torque that peels it off the surface.

IV – 6: This figure describes in more detail how high velocities can cause eddy forces causing a torque that helps peel the outer robot from the surface.

IV – 7: Allowable motor torques before slippage or falling for different coefficients of eddy current damping.

IV – 8: Allowable motor torques before slippage or falling for normalized position and velocity. This is independent of eddy current.

IV – 9: Region in which not only torque can be applied, but *direction* of acceleration can be influenced, for two different values of eddy current damping.

IV – 10: Description of inverse kinematics that must be solved for to get some desired velocity direction.

IV – 11: Lumped parameter modeling for fine positioning.

IV – 12: Root locus for fine positioning of inner robot.

IV – 13: Closer view of root locus for fine positioning of inner robot.

IV – 14: Inner robot following a ladder – type reference trajectory.

IV – 15: Coupling between axes. Top plot shows x direction motion while bottom plot shows y direction motion. Note that a step in one axis causes a disturbance in the other.

List of tables

List of Abbreviations and Symbols

Gross motion analysis

x_o, y_o, θ_o : x position, y position, orientation of outer robot in outer robot frame

x_d, y_d, θ_d : relative x, y positions and orientation of inner robot in outer robot frame

$\alpha_1, \alpha_2, \alpha_3$: coordinate frame fixed to outer robot

$\beta_1, \beta_2, \beta_3$: coordinate frame fixed to inner robot

$v_{ow1}, v_{ow2}, v_{ow3}, v_{ow4}, v_{iw1}, v_{iw2}, v_{iw3}, v_{iw4}$: velocities of outer, inner wheels in outer robot frame

$\dot{\mathrm{X}}$: a vector containing $\dot{x}_o, \dot{y}_o, \dot{\theta}_o, \dot{x}_d, \dot{y}_d, \dot{\theta}_d$

X_d : a vector containing x_d, y_d, θ_d

$m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9$: outer robot dimensions

$n_1, n_2, n_3, n_4, n_7, n_8$ outer robot dimensions

m_i, m_o : mass inner , outer robot

I_C : 2nd moment around center of mass

g : gravity

$R_{o1x}, R_{o4x}, R_{o1y}, R_{o2y}, R_{o3y}, R_{o4y}, R_{o1z}, R_{o2z}, R_{o3z}, R_{o4z}$: reaction forces outer robot

$\underline{R}_{o1}, \underline{R}_{o2}, \underline{R}_{o3}, \underline{R}_{o4}$: reaction force vectors outer robot

$R_{i1x}, R_{i2x}, R_{i3x}, R_{i4x}, R_{i1y}, R_{i2y}, R_{i3y}, R_{i4y}, R_{i1z}, R_{i2z}, R_{i3z}, R_{i4z}$: reaction forces inner robot

$\underline{R}_{i1}, \underline{R}_{i2}, \underline{R}_{i3}, \underline{R}_{i4}$: reaction force vectors inner robot

$M_{o1z}, M_{o2z}, M_{o3z}, M_{o4z}, M_{o1x}, M_{o2x}, M_{o3x}, M_{o4x}, M_{o1y}, M_{o2y}, M_{o3y}, M_{o4y}$: outer magnet forces

$\underline{M}_{o1}, \underline{M}_{o2}, \underline{M}_{o3}, \underline{M}_{o4}$: outer robot magnet vector forces

$M_{i1z}, M_{i2z}, M_{i3z}, M_{i4z}, M_{i1x}, M_{i2x}, M_{i3x}, M_{i4x}, M_{i1y}, M_{i2y}, M_{i3y}, M_{i4y}$: inner magnet forces

$\underline{M}_{i1}, \underline{M}_{i2}, \underline{M}_{i3}, \underline{M}_{i4}$: inner robot magnet vector forces

$\tau_{o1z}, \tau_{o2z}, \tau_{o3z}, \tau_{o4z}, \tau_{o1x}, \tau_{o2x}, \tau_{o3x}, \tau_{o4x}, \tau_{o1y}, \tau_{o2y}, \tau_{o3y}, \tau_{o4y}$: outer magnet torques

$\underline{\tau}_{o1}, \underline{\tau}_{o2}, \underline{\tau}_{o3}, \underline{\tau}_{o4}$: outer magnet torque vectors

$\tau_{motor2}, \tau_{motor3}$: motor torques

$\tau_{i1z}, \tau_{i2z}, \tau_{i3z}, \tau_{i4z}, \tau_{i1x}, \tau_{i2x}, \tau_{i3x}, \tau_{i4x}, \tau_{i1y}, \tau_{i2y}, \tau_{i3y}, \tau_{i4y}$: inner magnet torques

$\underline{\tau}_{i1}, \underline{\tau}_{i2}, \underline{\tau}_{i3}, \underline{\tau}_{i4}$: inner magnet torque vectors

Note: all magnet torques forces w.r.t. coordinate frame located at outer robot

$\underline{r}_{om1}, \underline{r}_{om2}, \underline{r}_{om3}, \underline{r}_{om4}$: vector from outer robot COM to outer magnets

$\underline{r}_{im1}, \underline{r}_{im2}, \underline{r}_{im3}, \underline{r}_{im4}$: vector from inner robot COM to inner magnets

$\underline{r}_{ow1}, \underline{r}_{ow2}, \underline{r}_{ow3}, \underline{r}_{ow4}$: vector from outer robot COM to outer robot wheel contact points

$\underline{r}_{iw1}, \underline{r}_{iw2}, \underline{r}_{iw3}, \underline{r}_{iw4}$: vector from inner robot COM to inner robot wheel contact points

$\underline{r}_{bm1}, \underline{r}_{bm2}, \underline{r}_{bm3}, \underline{r}_{bm4}, \underline{r}_{bw1}, \underline{r}_{bw2}, \underline{r}_{bw3}, \underline{r}_{bw4}$: vectors from outer robot rear axle to magnets and wheels

$\underline{r}_{B1\_A}, \underline{r}_{B2\_A}$: vectors from outer robot wheels to tool location

Note: wheel, magnet location vectors in coordinate frame attached to corresponding robot

c : rolling resistance coefficient

c_r : coefficient of friction for rubber wheels

$\Delta_{o1}, \Delta_{o2}, \Delta_{o3}, \Delta_{o4}$: small deflection of the outer robot wheels in the upward (+z) direction

$\Delta_{i1}, \Delta_{i2}, \Delta_{i3}, \Delta_{i4}$: small deflection of inner robot wheels in the downward (-z) direction

Δ_{oc}, Δ_{ic} : offset deflection for the entire chassis (see figure)

a_o, b_o, a_i, b_i : coefficients describing outer and inner chassis planes

$P_{o1}, P_{o2}, P_{o3}, P_{o4}, P_{i1,} P_{i2}, P_{i3}, P_{i4}$: points on chassis corresponding to wheels for outer and inner robot

$k_{o1}, k_{o2}, k_{o3}, k_{o4}$: stiffness's of outer robot wheels

$k_{i1}, k_{i2}, k_{i3}, k_{i4}\ k_{i1}, k_{i2}, k_{i3}, k_{i4}\ k_{i1}, k_{i2}, k_{i3}, k_{i4}$: stiffness's of inner robot wheels

$C_{ox}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right), C_{oy}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right), C_{o\theta}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right)$: coulomb friction in x, y, theta directions, outer robot

$C_{ix}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right), C_{iy}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right), C_{i\theta}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right)$: coulomb friction in x, y, theta directions, inner robot

$D_{ox}\left(\dot{\mathrm{X}}\right), D_{oy}\left(\dot{\mathrm{X}}\right), D_{o\theta}\left(\dot{\mathrm{X}}\right)$: damping force in x, y, theta directions, outer robot

$D_{ix}\left(\dot{\mathrm{X}}\right), D_{iy}\left(\dot{\mathrm{X}}\right), D_{i\theta}\left(\dot{\mathrm{X}}\right)$: damping force in x, y, theta directions, inner robot

$VR_{ox}\left(\mathrm{X}_d\right), VR_{oy}\left(\mathrm{X}_d\right), VR_{o\theta}\left(\mathrm{X}_d\right)$: variable reluctance force in x, y, theta directions, outer robot

$VR_{ix}\left(\mathrm{X}_d\right), VR_{iy}\left(\mathrm{X}_d\right), VR_{i\theta}\left(\mathrm{X}_d\right)$: variable reluctance force in x, y, theta directions, inner robot

$\Sigma F_{ox}, \Sigma F_{oy}, \Sigma \tau_o$: sum of forces in x, and y, sum of moments in theta, outer robot

$\Sigma F_{ix}, \Sigma F_{iy}, \Sigma \tau_i$: sum of forces in x, and y, sum of moments in theta, inner robot

Fine positioning analysis

$(x_1, y_1), (x_2, y_2)$: position of outer, inner robots while in 'fine positioning' mode

k_1, k_2: stiffness of outer robot's chassis, linearized stiffness due to magnet force between robots

b_1, b_2: viscous damping on outer, inner robot

k_p, k_d, k_i: proportional, derivative, and integral control gains for PID controller

F_x, F_y: force in x, y direction from Lorentz force

I. Introduction

I.1. Motivation

I.1.1. Future factory

Dealing with volatility is one of the greatest challenges in the aircraft manufacturing industry [21,24]. Tremendous resources are expended maintaining empty facilities during recessions and building temporary assembly stations in regions experiencing transient economic growth. This problem demonstrates a clear need for factories that can quickly and cheaply be constructed and taken down, or transported. Such a task is difficult because current facilities use large, heavy fixtures such as scaffolding as an integral part of the manufacturing process.

A future factory concept devised in collaboration with our research sponsor is that of a manufacturing facility devoid of such bulky components. Such a factory would start as a large empty hangar outfitted with a sensor array. Raw materials and partially assembled components would be brought in and assembled into more finished parts such as complete wings. This assembly would be carried out by self-supported, mobile robotic systems that could be easily transported to any such facility on the globe. Our aim is the development of a type of robotic system that enables such a paradigm shift in aircraft manufacturing.

I.1.2. Fastener installation

Fastener installation is an assembly operation that could benefit greatly from automation, and is therefore an excellent starting point for our efforts towards this future factory. Fasteners are pieces of hardware used to join two or more components such as sheet metal or flanges. There are on the order of several hundred thousand fasteners on common aircraft wings. Proper installation of these fasteners

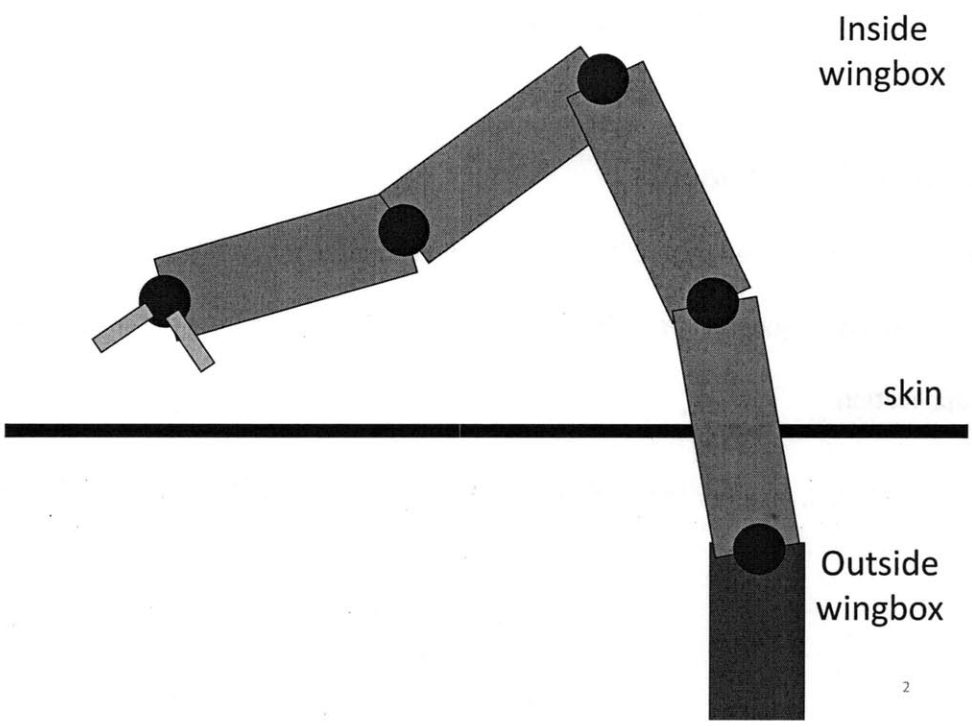
is a two sided operation – on one side a hole is drilled, deburred, and the fastener inserted, while on the opposite side the hardware is braced against the drill and a nut or sleeve is attached to the inserted fastener.

The nature of this operation requires a thin skin of material separating the tooling on either side of the operation (in our case, the aluminum skin of the airplane wing). In order to maintain structural integrity of the wing, the fasteners must be precisely located. The tooling that performs this operation must be capable of undergoing a long stroke – on the order of the size of an airplane wing. Part of the fastener installation procedure requires a clamping force across the skin. In addition, some of the existing tooling for this operation is heavy, and must be supported.

As there is a great deal of existing tooling for the *tasks* in fastener installation (drilling, deburring, etc), we are interested in providing a robotic platform that enables these tools to perform their tasks in an autonomous manner. We are interested in positioning, locomotion and load bearing. Our overarching goal is to design a system that can enable fastener installation within the non-supported, mobile robotic framework necessitated by the 'mobile factory' paradigm.

I.2. Previous work

Fastener installation is currently performed by hand. A worker crawls inside the wingbox and uses a tool to slip nuts or sleeves over fasteners, mating with a tool on the outside of the wing. This operation is uncomfortable and dangerous for the worker. Previous attempts at automation have consisted mainly of variations on an articulating snake-like robot arm that could enter the wingbox through an access hole [9,34,35].



I - 1: Snakelike robotic arm

II. Design

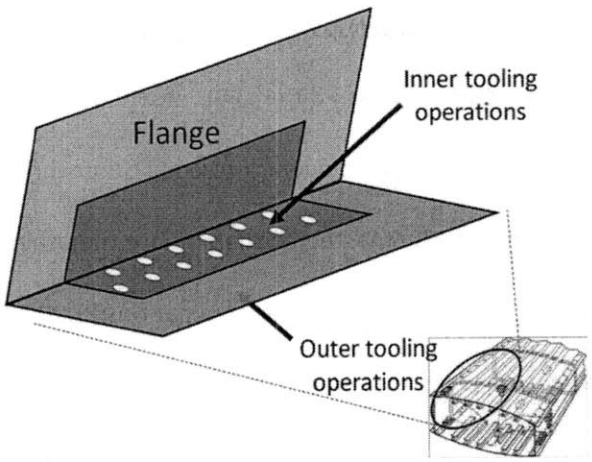
II.1. Functional Requirements

This robotic system is motivated by a real need in the aircraft industry. While we are interested in developing a system with general applicability, it is important to first address the specific needs of the motivating application.

As previously mentioned, fastener installation is a two sided process. The steps involved in the process are:

- Position tooling
- Clamp skin to flange
- Drill and countersink hole while removing chips
- Visually inspect the hole
- Insert fastener
- Seat fastener
- Add nut or sleeve to fastener (this depends on which type of fastener is being used)
- Release clamping

This process requires that tooling is present on both the inside and outside of the wingbox.



II - 1: Flange location

This pair of tools requires some clamping functionality across the skin. For fastener installation, accuracy on the order of 100 um is needed. Additionally, the tooling should be able to move from hole to hole, potentially traversing large distances. The holes are located at distances on the order of several cm. The tooling may be called upon to travel the length of a wing, on the order of up to 30 m. The system needs to be able to deal with heavy tooling – current systems on the outside of the plane weigh on the order of 100 kg; 1000 N holding force is required. Whatever positioning system we use must be capable of overcoming static friction effects caused by 1000 N of magnetic clamping force. Later, we show that a force of approximately 50 N is required for fine positioning. Finally, the robotic system on the inside of the wing should be able to operate within a cluttered environment.

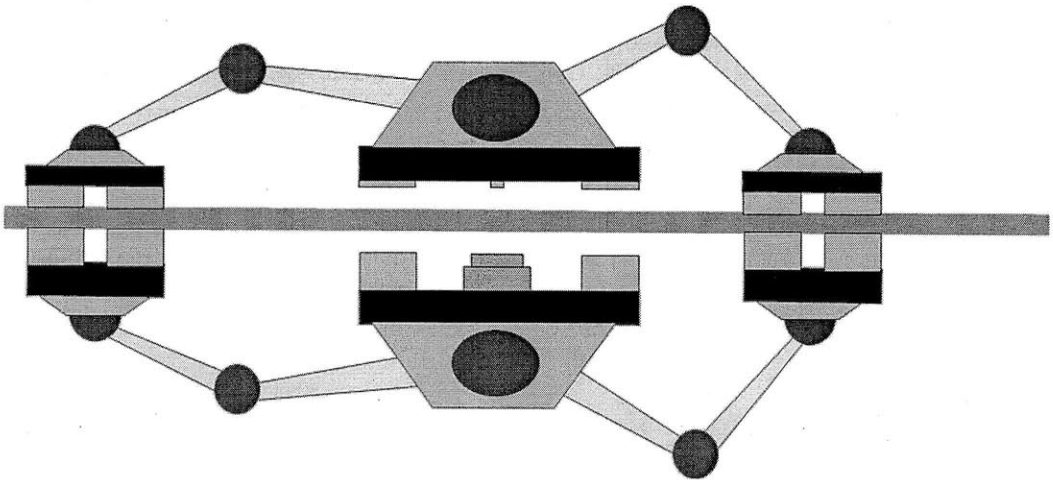
It is worth noting that the tooling to perform the fastener installation operation already exists. It only remains that we develop a system capable of delivering the tooling precisely to some location and supporting it while it works. This robotic system is in a sense a generic tool delivery system for operations of this kind. It is also easily able to perform auxiliary tasks, such as clamping.

II.2. Previous Designs

As previously mentioned, there are several groups attempting to automate this process by developing articulated snake-like robot arms. Our approach differs in that we intend to use a pair of robots that work together across the skin. These robots would stick to each other through the skin using magnets. Furthermore, we would like one robot (the master) to manipulate the other (slave). Non contact power transmission has been shown effective in several other applications [8,10,12,18,38]. This section describes some of the alternative designs considered within this framework of a pair of mobile robots utilizing magnets.

II.2.1. Spider robots

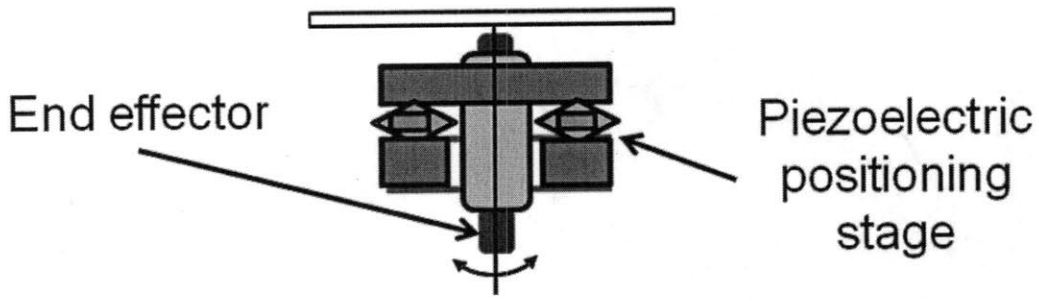
Our initial approach was the development of a pair of fully articulated robots acting across the skin as shown in Fig. II-2. Each leg would have an electromagnet in order to be able to clamp / release as desired to facilitation locomotion.



II - 2: Pair of spider robot

Gross position could be achieved by articulation of the legs, but fine positioning with such a system would likely be difficult. Several schemes for fine positioning were considered, including the addition of

a fine positioning stage to the end effector of the robot. High stiffness, short stroke piezoelectric actuators could be used to precisely position the tooling as desired.

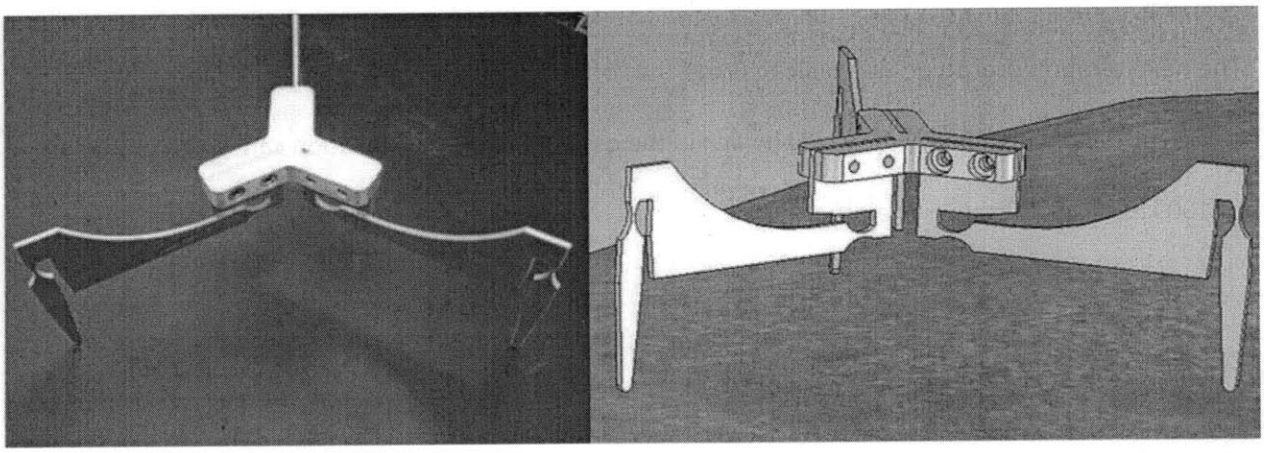


II - 3: Positioning stage for fine alignment of end effector

Unfortunately, it became quickly apparent that such a system was needlessly complex and would likely be too heavy to support itself against gravity.

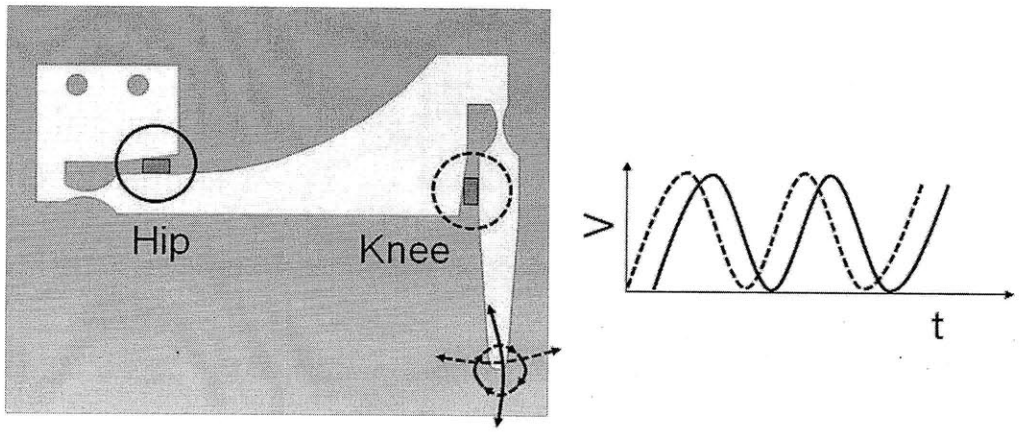
II.2.2. Piezo crawler

Some development was made in the direction of a mobile robot powered by piezo electric actuators. Fig. II-4 shows a CAD model and photograph of the finished prototype.



II - 4: Piezo crawler prototype and CAD model

This three legged robot had two actuators per leg. By running these actuators out of phase or asymmetrically, net work could be created at the output of each leg, as shown in Fig. II-5.

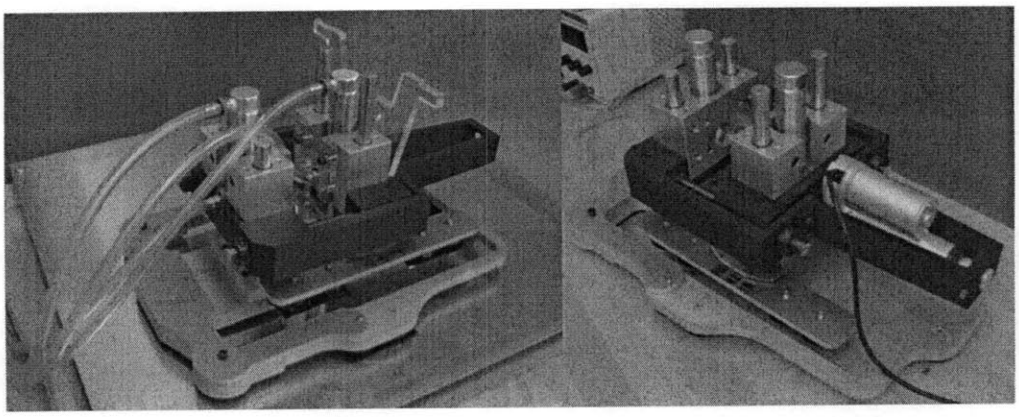


II - 5: Two degree of freedom foot

We were successful in achieving precise positioning control with this prototype. Unfortunately it would be unable to handle heavy tooling, and the functionality of the piezoelectric actuators do not scale up well to deal with forces present in the real system.

II.2.3. Inchworm walker

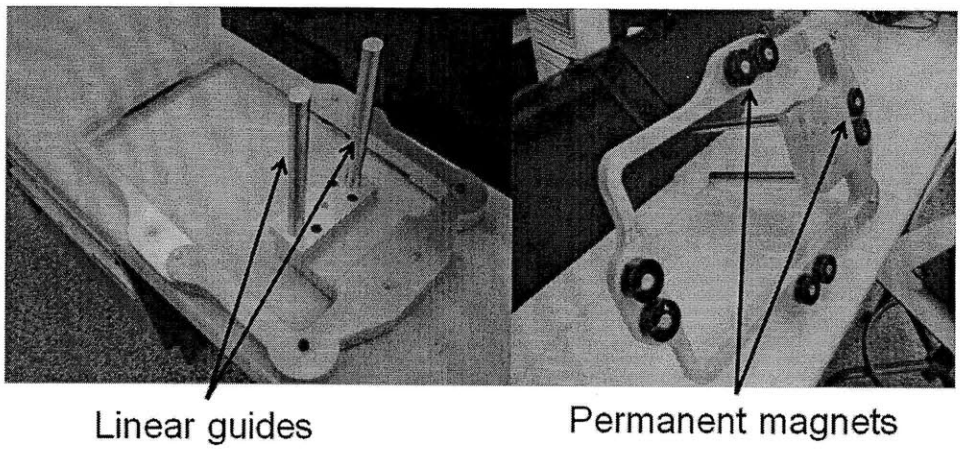
The next iteration prototype was built to prove that off the shelf permanent magnetic forces would be sufficient to hold up a heavy duty tool (which in the case of fastener installation can weigh on the order of 100 kgs). Fig. II-6 shows the robot.



II - 6: Inchworm walker prototype

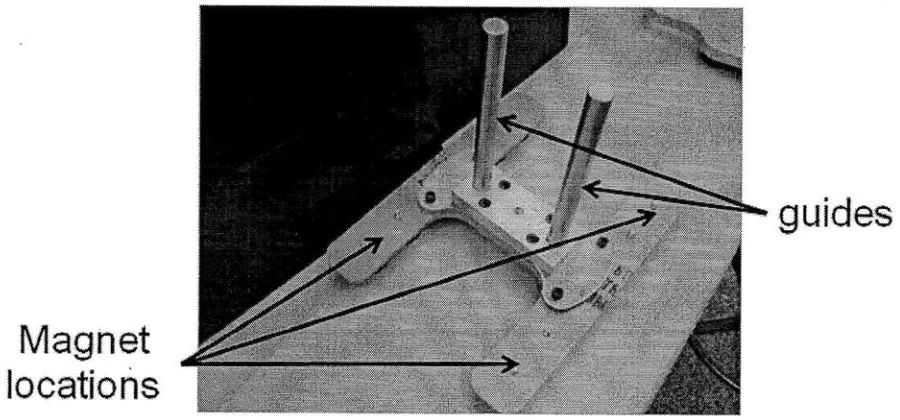
The robot was composed of two main sections. Each section had a 'foot' with a series of permanent magnets on its underside (see figures below). The robot hung upside down, supporting its own weight due to an attraction force between its magnets and blocks of steel placed inside the wingbox mock-up. This holding force was induced over an aluminum skin thickness of 1/8'' – a common value in airplane wings.

A lead screw was used to actuate the relative displacement of these feet. By lifting the 'outer' foot, displacing it forward, and then lowering the foot, the robot effectively took a step forward. This inchworm locomotion demonstrated a safe, albeit slow, method of locomotion for such a system.



II - 7: Inchworm walker outer foot

The inner foot is nested within the outer foot – this allows for the robot to remain stably attached when either foot is in contact.

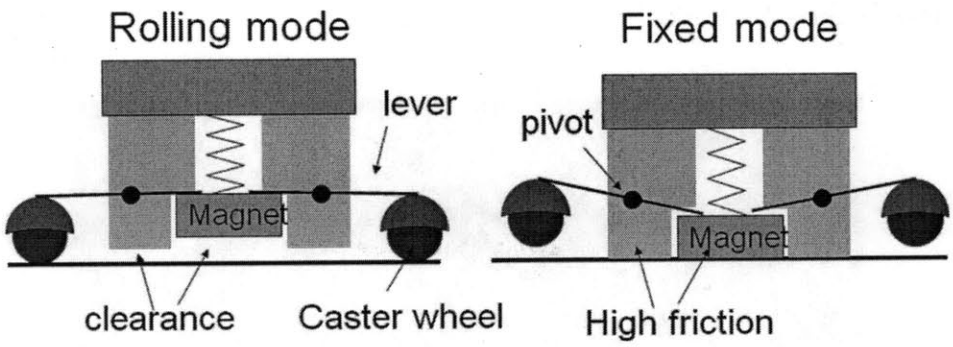


II - 8: Inchworm walker inner foot

Having demonstrated the effectiveness of magnets to hold such a system against gravity, we were interested in making the mobile robots more nimble.

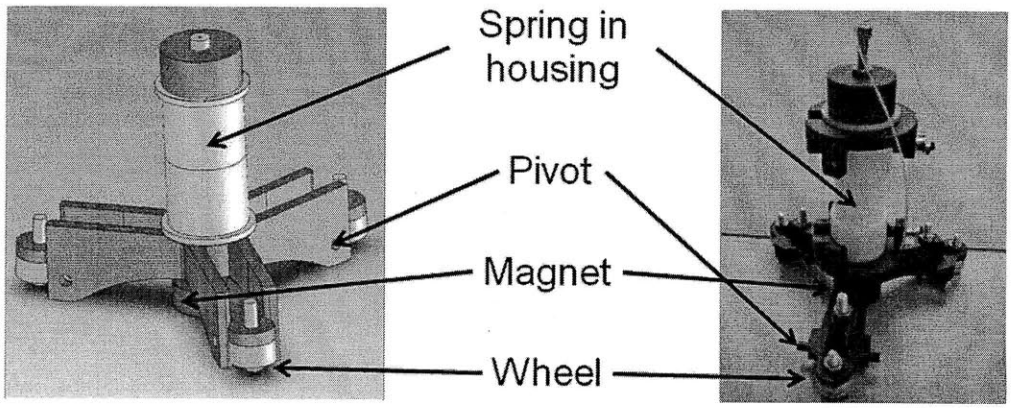
II.2.4. Multifunctional foot and resulting designs

A multifunctional foot was developed. It has the ability to clamp strongly to a surface or release and roll with very little friction [27]. Fig. II-9 shows the concept. A spring loaded magnet has the ability to engage or release a set of wheels. The position of the magnet is dependent on the pulling force on it (through the skin) as well as the force the spring exerts. This means that by modulating the attractive force with an electromagnet only slightly, we can change the state of the foot while still maintaining substantial normal forces. This allows us greatly modulate frictional properties of the foot (rolling coefficients of friction are often two orders of magnitude higher than sliding friction coefficients) while maintaining a relatively constant attractive normal force.



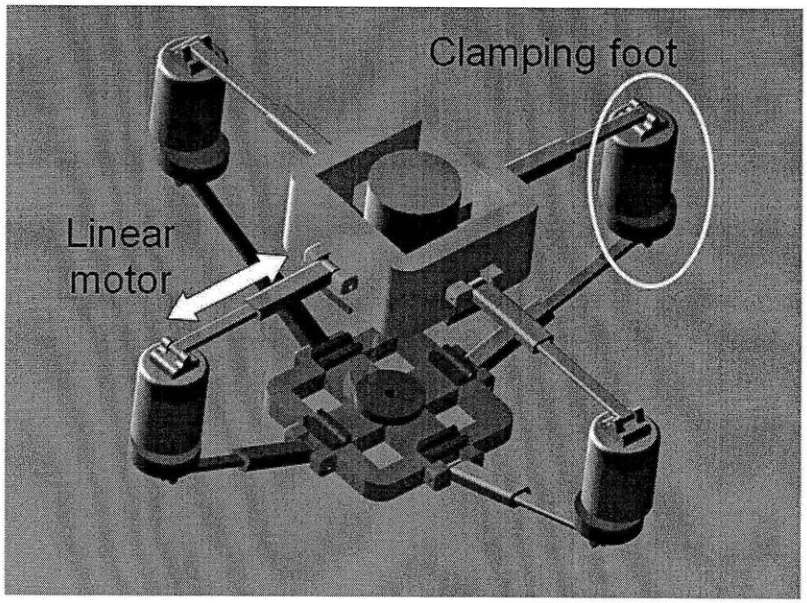
II - 9: Multifunctional foot demonstrating clamping and release

Fig. II-10 shows the CAD drawing and functional prototype foot.



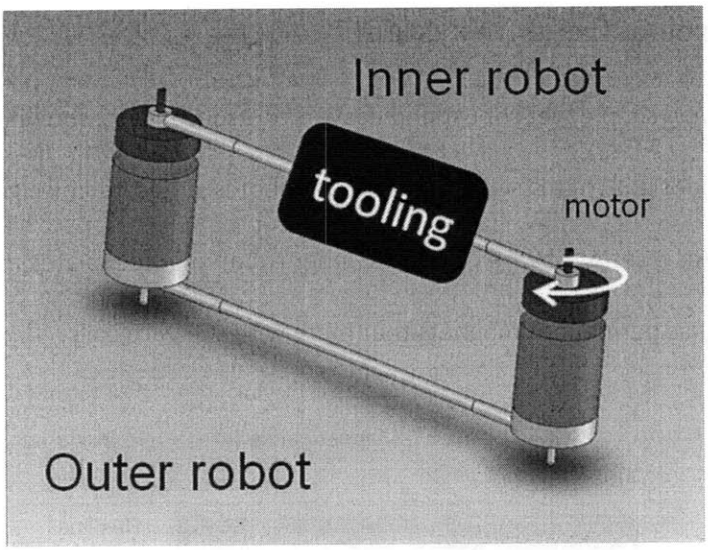
II - 10: CAD model and functional prototype of multifunctional foot

This attaching / detaching foot concept allowed spurned the development of several new designs. We imagined a pair of robots in which two linear motors could control the X Y position of the tooling, while the feet clamped or unclamped as desired for locomotion.



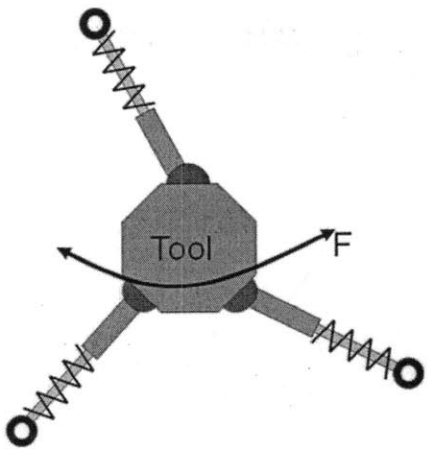
II - 11: Two DOF linear motor pair of robots

A similar concept was a single pair of feet with rotary motors as the joints. Positioning would be more difficult with this design, but the robot would be quite simple.



II - 12: Single DOF pivoting foot system

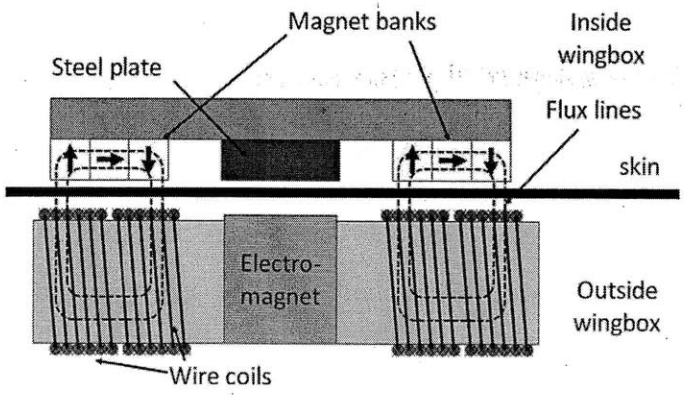
A more complex design utilizing these multifunctional feet involved a series of legs, potentially with linear springs, allowing us to exploit the dynamics of the system. These dynamics would change depending on which feet were attached, potentially leading to interesting behavior.



II - 13: Multi-foot swinging robot utilizing compliance

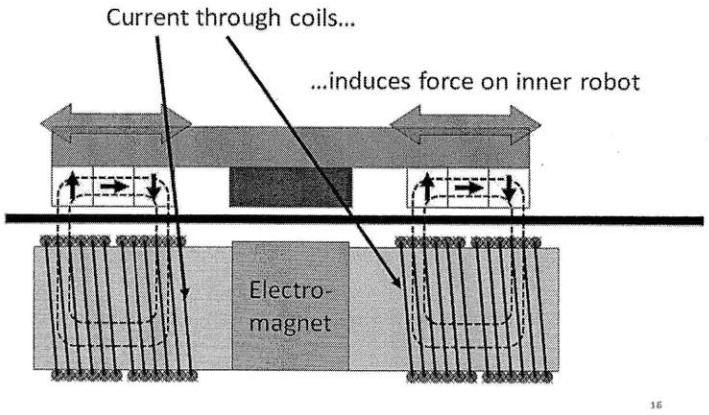
II.2.5. Single degree of freedom system

We settled on a relatively simple, yet robust and easy to control concept, shown in Fig. II-14, as a precursor for the final design. In this system, the outer robot is supported by a gantry and has an electromagnet as well as a series of wire coils. The inner robot has steel plates mating with the outer electromagnet, as well as permanent magnets that mate with the wire coils.



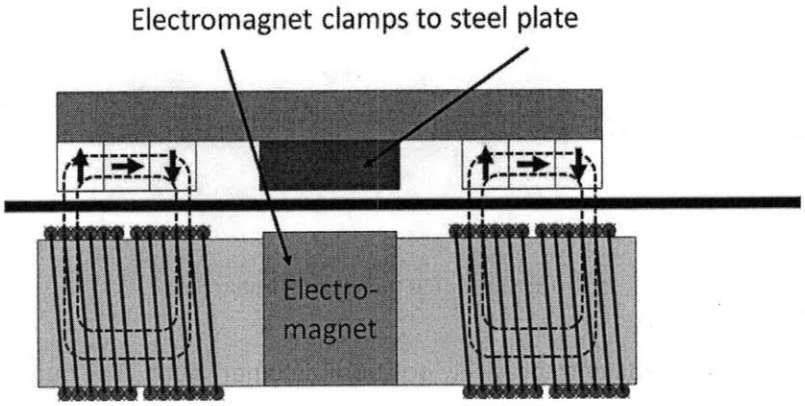
II - 14: Architecture of single DOF system

By running a current through the coils, equal and opposite Lorentz forces are induced on the pair of robots. This allows us to perform fine positioning.



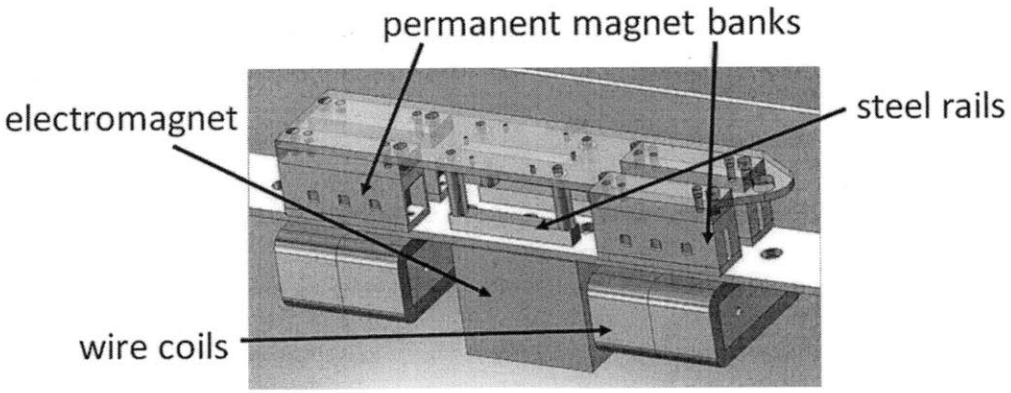
II - 15: Lorentz force used for fine positioning

This system also demonstrated clamping functionality, as the steel plates were able to clamp to the electromagnet through the skin.



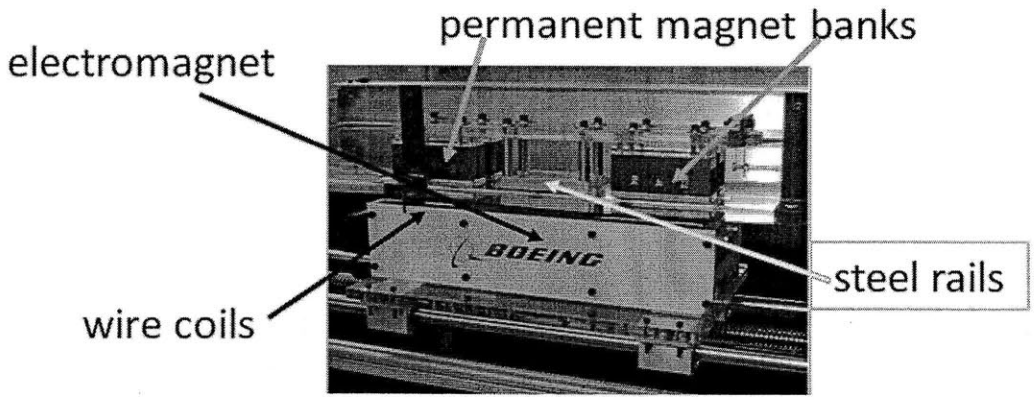
II - 16: Clamping ability

Fig. II-17 shows a CAD model of the system.



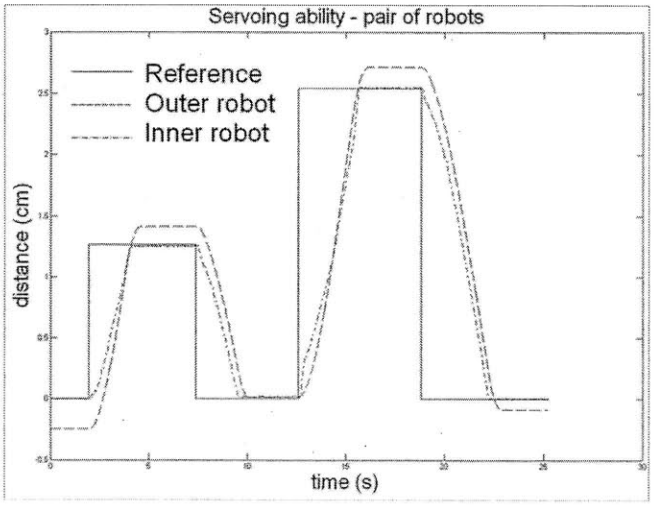
II - 17: CAD drawing of prototype

Fig. II-18 shows the working prototype.



II - 18: Single DOF functional prototype

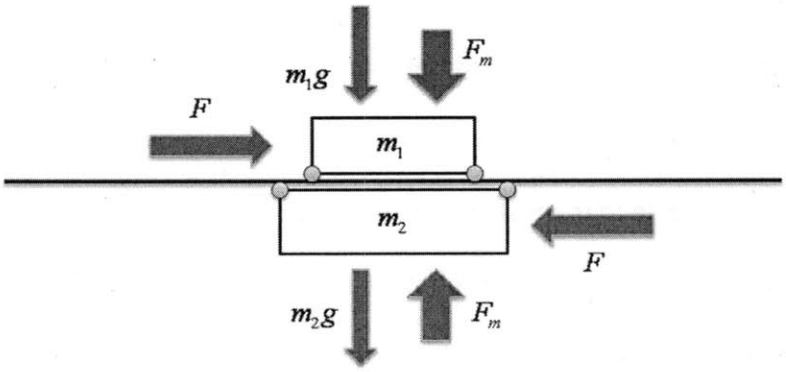
This system showed excellent ability to precisely position the inner robot. The outer robot is affixed to a gantry to support its weight and controlled by a leadscrew. At this point, we were not interested in servoing the outer robot, as control of the inner robot using the Lorentz force coil from across the skin is a much more interesting and challenging problem.



II - 19: Fine positioning ability of Lorentz force actuators

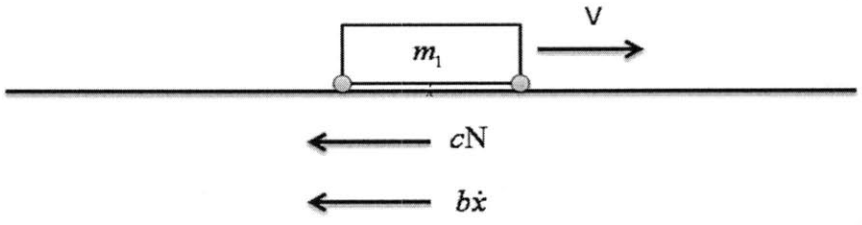
II.2.6. Unsupported system – asymmetric oscillation

Removing the support gantry used in the previous prototype is extremely advantageous in terms of flexibility and cost in manufacturing. In order to remove this support, it is important to apply a high magnetic holding force (normal to the skin) to support the outer robot against gravity. This translates to a high normal force between the inner robot and the wing skin. Frictional forces scale with normal forces – if we used a sliding contact (as was the case in the previous iteration), coloumb friction would make positioning using Lorentz forces difficult (it is difficult to produce Lorentz force of sufficient magnitude). For this reason, we want the inner and outer robot to be wheeled. A pair of wheeled robots feels forces due to gravity, magnetic forces, and a force we can apply due to the coil of wire.



II - 20: Forces on stationary unsupported system

Additionally, as the system moves, it feels forces due to friction. Some of these forces are nonlinear.



II - 21: Drag forces on moving system

Initial simulations suggested it was possible to exploit this nonlinear force to get locomotion. By applying an asymmetric current input (such as a sawtooth wave), net motion of the system was observed. While this was an interesting result, further exploration in this direction was halted for two reasons. First, it was not immediately clear how well this could be applied to systems with more than one degree of freedom. Second, a direct alternative to this scheme exists. In a manufacturing environment, utilizing straightforward solutions as much as possible is generally preferred.

II.3. Design of choice

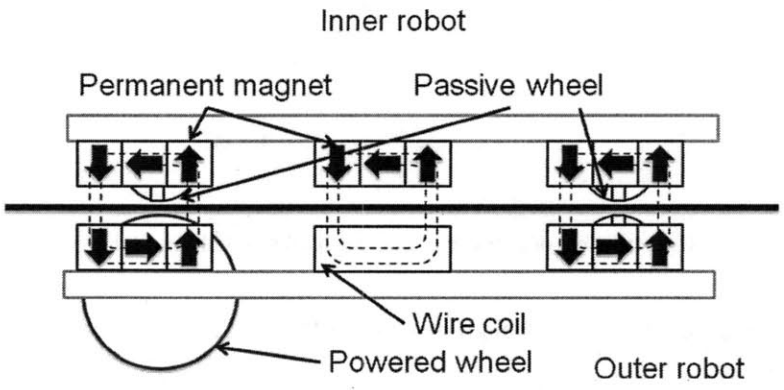
II.3.1. General description

Our design of choice utilizes the concept of a pair of mobile robots described in the previous section. The system should remain fully self-supported; sets of magnets on the inner and outer robots hold the pair together, through the skin, and create enough attractive force to bear the weight of the heavy duty tooling. The outer robot is able to traverse the wing skin due to a set of powered wheels. Due to the attractive force between magnets, the inner robot follows the outer robot as it moves. Fine positioning of the inner robot is performed using a coil on the outer robot that mates with a magnet on the inner robot. A current through this coil induces a Lorentz force on the inner robot. Note that this allows for the inner robot to remain passive (tetherless); only the outer robot is powered. This feature could be enormously useful for a variety of other applications. The following Pugh chart shows a comparison between this design and the others considered.

	Spider robots	Piezo crawler	Inchworm walker	Multifunctional foot designs	Single DOF system	Asymmetric oscillation	Final design
load bearing	+	-	+	+	+	+	+
clamping	+	-	+	+	+	+	+
gross positioning	+	+	+	+	+	0	+
fine positioning	0	+	+	0	+	+	+
simplicity	-	+	+	0	+	-	+
3 DOF motion	0	+	-	+	-	0	+
tetherless inner robot	0	0	+	+	+	+	+

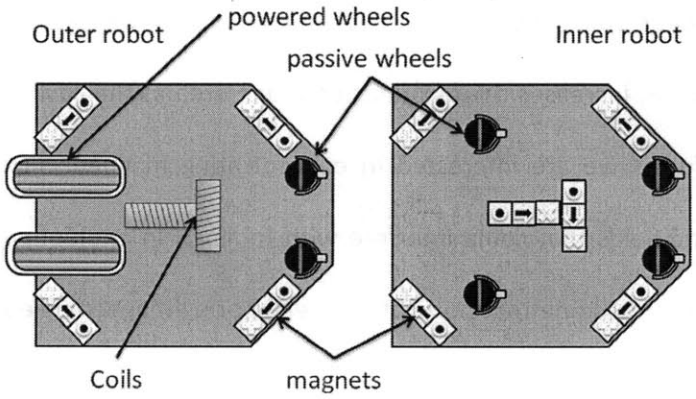
II - 1: Pugh chart

The system looks like this from the side:



II - 22: Side view

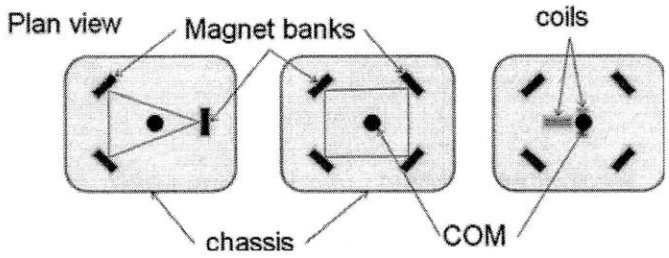
And like this from the top:



II - 23: Top view

II.3.2. Architecture

For the system to be self supporting, it is helpful to keep the center of mass of the outer robot within a polygon defined by the magnet locations. This requires a minimum of 3 Halbach arrays. For safety, we use 4 arrays.

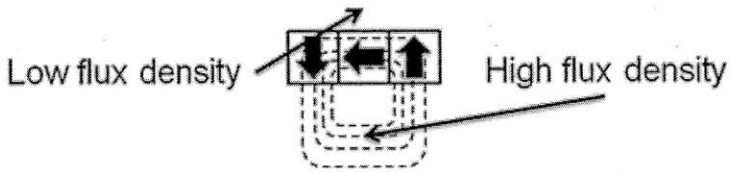


II - 24: Effect of number of magnet banks on holding stability

The tooling used in this operation must align in x and y to the desired fastener location; orientation is not important. In Fig. II-24, two coils are placed such that their axis of force coincides with the center of mass of the system. This allows us to have 2 DOF position control of the tool location.

II.3.3. Magnet selection

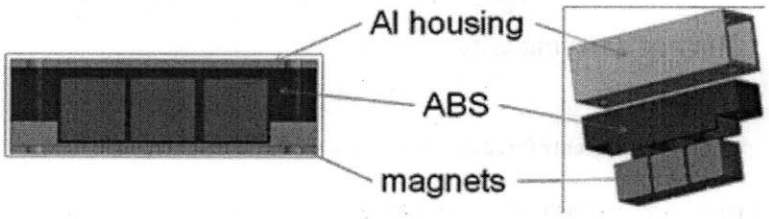
We select a basic magnetic building block: a halbach array as shown in Fig. II-25. This configuration (of three magnets in our case) increases the flux density in one area at the cost of reduced flux density at another. This is useful as we are interested in concentrating magnetic flux across the skin, while reducing stray magnetic fields that could interfere with tooling. In addition, analysis of the system is greatly simplified if we can assume that none of the neighboring magnetic elements interfere with one another.



II - 25: Halbach array showing region of flux concentration

Magnet design, while a potentially rewarding direction of research is outside the scope of this work – we have chosen to stay within the confines of the myriad of off the shelf permanent magnets. This means

we have a finite set of magnet sizes / strengths available for selection. We chose to use Neodymium Iron Boron magnets for several reasons. These rare earth magnets are more robust to impact demagnetization than Samarium cobalt magnets. They are also cheaper and generally have higher flux densities. Happily, these magnets are often modeled as linear in the operating region of their BH curve, which simplifies our analysis. Using permanent magnets instead of electromagnets is preferred as permanent magnets do not require continuous power input. Also, to generate the types of forces a small Neodymium magnet is capable of would require an extremely large and heavy electromagnet.

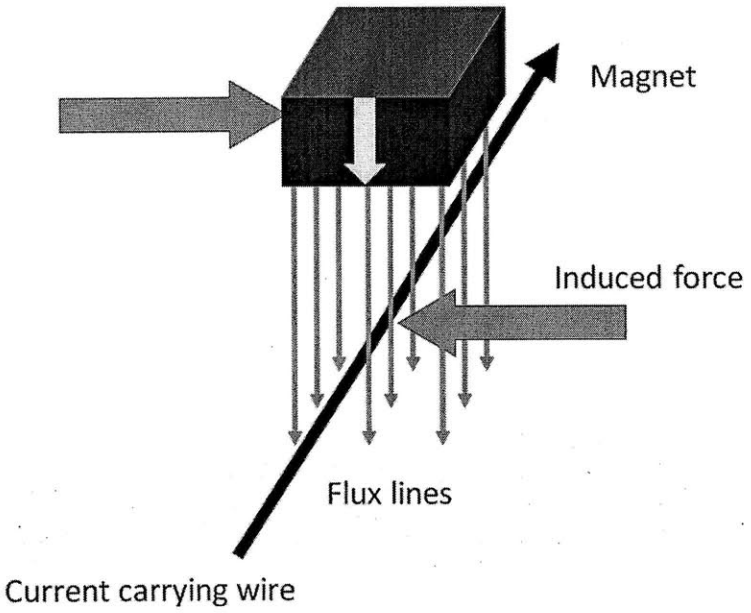


II - 26: CAD drawing of magnet bank

3 commercially available 1'' (2.54 cm) cube neodymium magnets were arranged into a halbach array as shown in Fig. II-25. An aluminum extrusion was used to provide structural support and a 3d printed ABS piece used to align and position the magnets. A pair of these 'magnet banks' showed attractive force >70 lbf (318N) at a distance of 1/8'' (3.2 mm), a standard wing skin thickness. Four such magnet banks can easily provide the required 1000 N holding force.

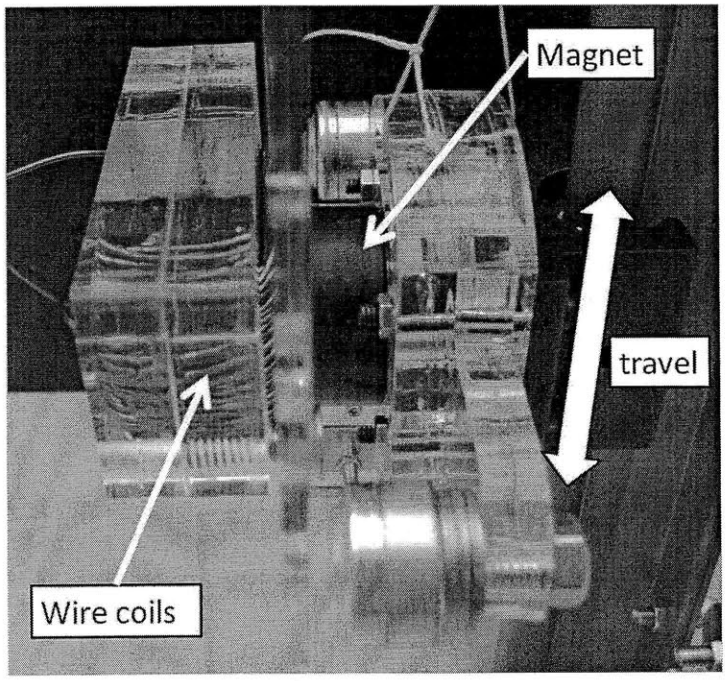
II.3.4. Coil selection

A set of coils is mounted to the outer robot to make with a set of magnets on the inner robot. Running a current through these coils induces equal and opposite Lorentz forces on the inner and outer robot.



II - 27: Lorentz force direction for a wire coil in a magnetic field

An initial proof of concept demo was run to see the effectiveness of using a Lorentz force at these scales.



II - 28: Lorentz force proof of concept demo

Initial tests showed the ability to induce reasonable forces at high bandwith.

The coils on the outer robot must be able to provide a strong enough Lorentz force to overcome friction. However, large, heavy coils are undesired. We would like to find the smallest coil size possible that can induce the required forces for fine positioning.

Without the aid of a three dimensional FEA software package capable of dealing with ferromagnetic components, such analysis is difficult. We settled for the use of a 2 dimensional package to get an order of magnitude estimate on required coil size.

The robots are clamped to the skin with a force of 1000 N. Rolling friction coefficients are on the order of 0.01 – 0.001. For safety, we choose the high value of .01 and add a safety factor of 5. This gives a rolling friction force of 50 N; we require a coil design that can generate 50 N of lateral force. The force generated by a coil in a magnetic field is given by:

$$F = iBl \tag{II-1}$$

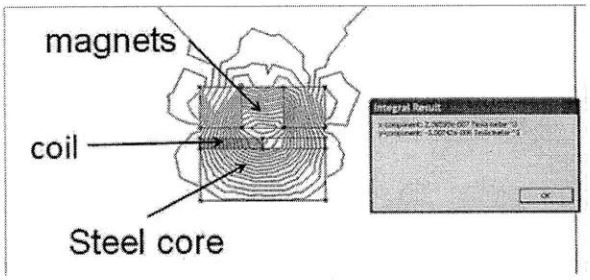
If we consider a volume with a current rather than a single line (in our case, the volume taken by the coil), we can find the force by:

$$F = \rho_i \cdot \int_V B dv \tag{II-2}$$

Where ρ_i is current density and $\int_V B dv$ is the total flux over a volume.

The maximum current density is nearly independent of wire gauge. We chose wire gauge of 18 based on the maximum current our hardware is capable of sourcing (around 10 amps). This corresponds to a current density of around 9.7e6 A/m. The integral volume of flux is estimated by running an FEA

simulation. The total flux in the coil region is a function of not only the geometry of the wire section, but a function of the geometry of its flux concentrating core.



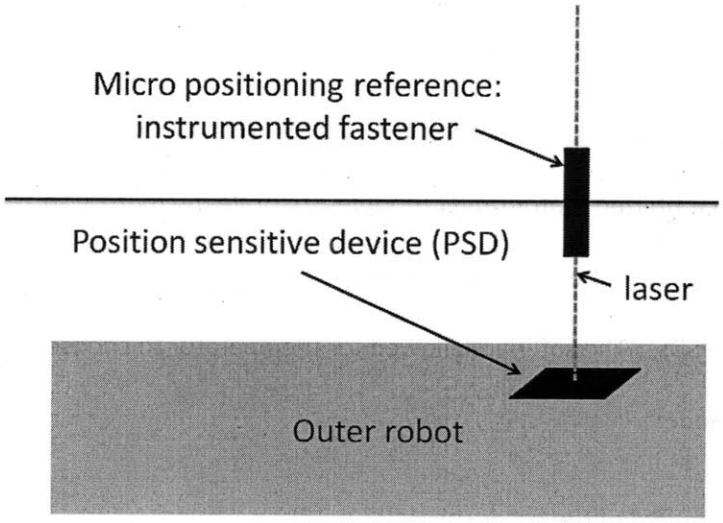
II - 29: Image from FEA for fields in coil

Initial FEA tests suggested a halbach array of 1'' (2.54 cm) cube magnets would easily provide the required flux to generate the desired 50 N of force (given reasonable coil and core dimensions) while this would be unfeasible with 0.5'' (12 mm) magnets. This is convenient, as we can use the same magnet bank array for holding / rough alignment as we do for Lorentz force positioning, helping to reduce unique part count. Reasonable coil dimensions included a steel core at least 3/8'' (9.5 mm) as well as coil area of about the same size.

II.4. Prototype

A working prototype was built. Caster wheels were used for passive rolling elements, mounted to standoffs for appropriate spacing. The Halbach arrays were also spaced from the chassis as desired. Two harmonic drive DC motors were used to control the wheels on the outer robot. These were mounted to the chassis using 3d printed ABS parts. The powered wheels on the outer robot were off-the-shelf components with a layer of rubber serving as a contact surface. The chassis for the robots were waterjet aluminum.

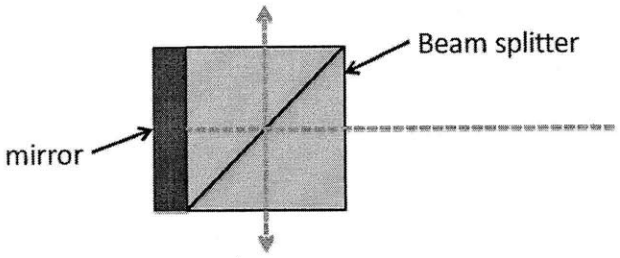
A position sensitive detector (PSD) was mounted to the robots. This device returns the location of the centroid of light impinging onto its detection surface – it can be used to precisely locate a laser beam. The aircraft industry is interested in exploring the idea of instrumenting fasteners, for example with lasers. In practice, these lasers could serve as the alignment reference for our robotic system.



II - 30: Use of laser to align robotic system

For the purpose of experiments, a PSD was mounted to both the inner and outer robots, although in practice if tetherless operation was required, no PSD is needed on the inner robot. An optical sensor at the fastener could be used to locate the inner robot.

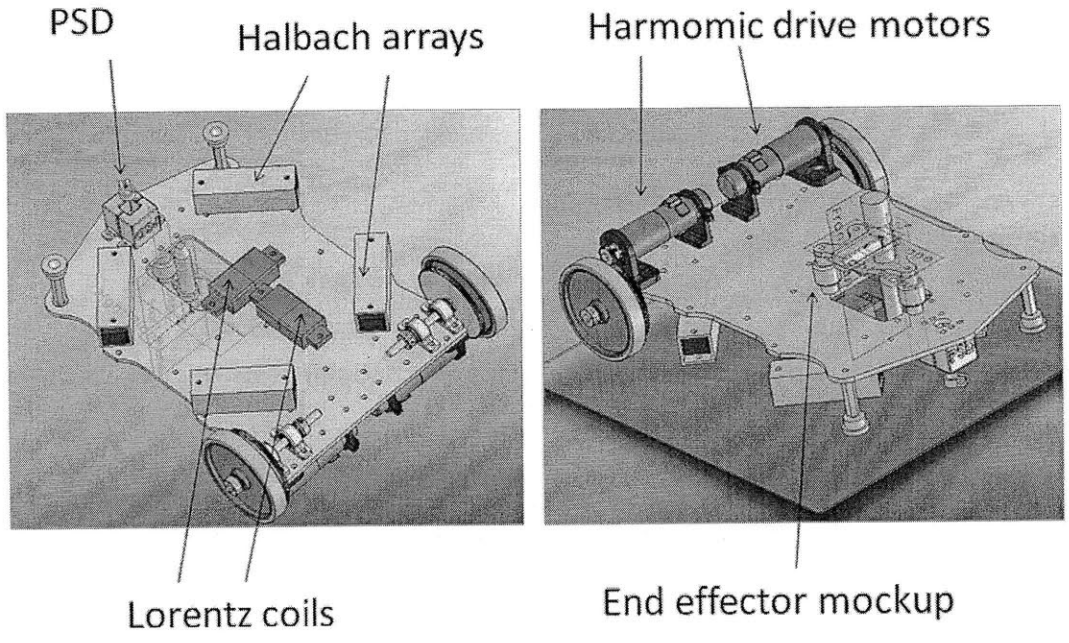
We developed a bi-directional laser with small form factor to use during testing. This component utilized a half mirror (beam splitter) as well as a regular mirror. Components were assembled and aligned with a rapid prototyped part.



II - 31: Bi directional low form factor laser

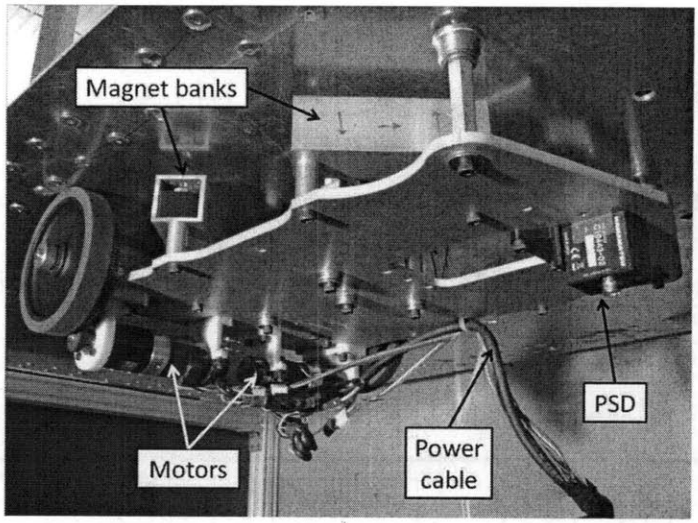
Labview realtime software was used to control the system. This software was implemented on a compact RIO (cRIO) that utilized motor driver modules (to drive the motors and read encoder information as well as to drive the wire coils) as well as an analog input module (to get position information from the PSDs). The software allowed for the operator to choose between a 'driving' and an 'auto-alignment' mode.

Fig. II-32 shows a CAD model of the outer robot.



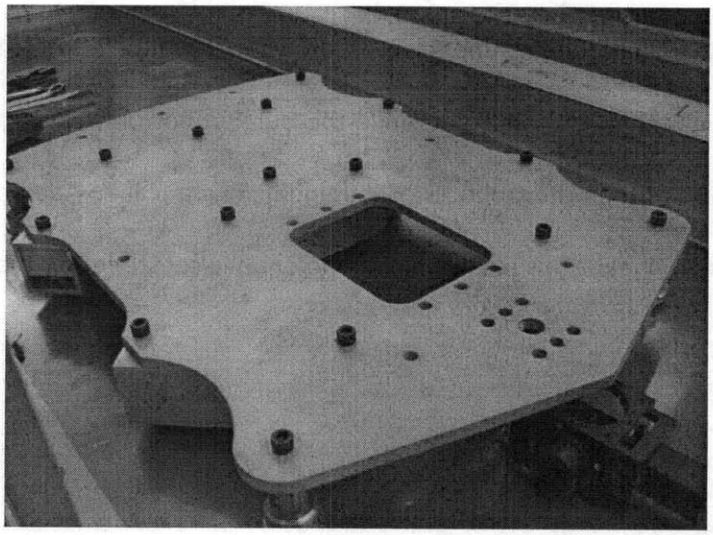
II - 32: CAD model of functional prototype

Fig. II-33 shows this outer robot suspended upside down by its magnet banks.



II - 33: Outer robot suspended

Fig. II-34 shows the inner robot.



II - 34: Inner robot

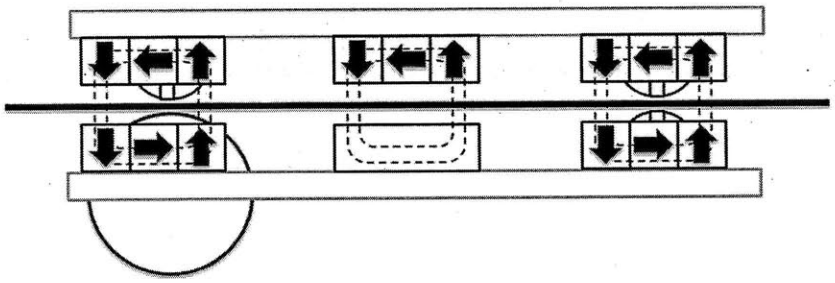
This prototype was successfully able to demonstrate required functionality for the fastener installation procedure.

III. Modeling

III.1. Full system model

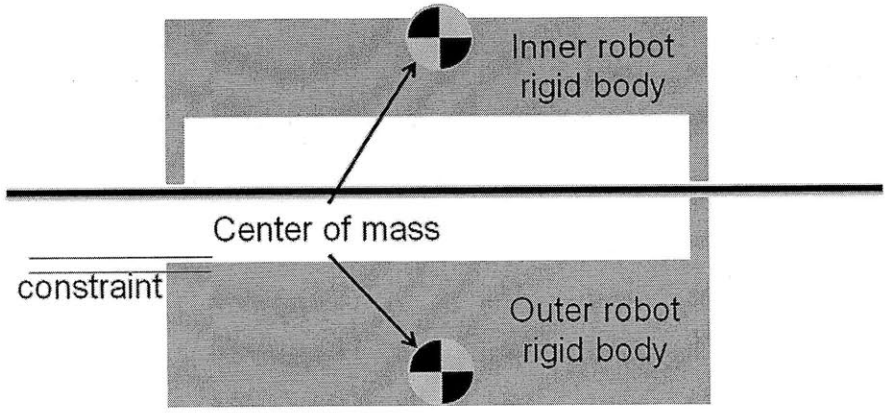
III.1.1. Assumptions

Fig. III-1 shows the inner and outer robots mated with one another across the skin.



III - 1: Inner and outer robots across skin

We treat these robots as rigid bodies, modeled as shown in Fig. III-2. Note that the inner robot has contacts with the skin at the front and rear while the outer robot only contacts the skin at the front wheel. The rear section of the outer robot is constrained along a plane. Additionally, we impose a no-slip condition at the rear wheel – this results in a nonholonomic constraint for the outer robot.

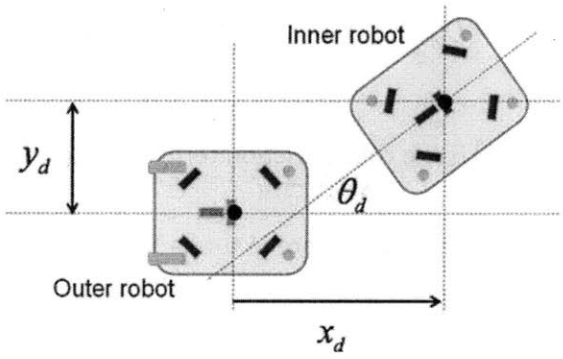


III - 2: Inner and outer robot as rigid bodies

We start by examining a free body diagram of the outer robot. The dynamic equations for the outer robot are very similar to those for the inner robot. The forces acting on the outer robot are (1) reaction forces from the skin, (2) magnet forces, (3) torque due to the motor, (4) dissipation terms due to friction and eddy current damping, (5) gravity and, (6) Lorentz forces due to coils.

The state of the robotic system is given by the following:

$$\mathrm{X}_i = \begin{bmatrix} x_i \\ \dot{x}_i \\ y_i \\ \dot{y}_i \\ \theta_i \\ \dot{\theta}_i \end{bmatrix}, \mathrm{X}_o = \begin{bmatrix} x_o \\ \dot{x}_o \\ y_o \\ \dot{y}_o \\ \theta_o \\ \dot{\theta}_o \end{bmatrix} \qquad \text{(III-1)}$$



III - 3: Definition of x_d, y_d and θ_d

X_i and X_o are the position of the inner and outer robots respectively, their derivative is velocity. X_d is the differential position between the robots. For use in dynamic equations, we define $\mathrm{X}_d = \mathrm{X}_i - \mathrm{X}_o$.

Dynamics of a rigid body in the plane are given by:

$$\Sigma F_x = ma_x$$
$$\Sigma F_y = ma_y \qquad \text{(III-2)}$$
$$\Sigma \tau_{COM} = I_{COM}\dot{\omega}$$

The challenge here is in finding the forces on the system. The forces on the outer robot are given as:

$$\begin{aligned}
\Sigma F_{ox} &= C_{ox}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right) - D_{ox}\left(\dot{\mathrm{X}}\right) - VR_x\left(\mathrm{X}_d\right) - \; i_x \cdot L_{xx}\left(\mathrm{X}_d\right) - i_y \cdot L_{yx}\left(\mathrm{X}_d\right) + F_{motor_x} \\
\Sigma F_{oy} &= C_{oy}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right) - D_{oy}\left(\dot{\mathrm{X}}\right) - VR_y\left(\mathrm{X}_d\right) - \; i_x \cdot L_{xy}\left(\mathrm{X}_d\right) - i_y \cdot L_{yy}\left(\mathrm{X}_d\right) + F_{motor_y} \qquad \text{(III-3)} \\
\Sigma \tau_o &= C_{o\theta}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right) - D_{o\theta}\left(\dot{\mathrm{X}}\right) - VR_\theta\left(\mathrm{X}_d\right) - \; i_x \cdot L_{x\theta}\left(\mathrm{X}_d\right) - i_y \cdot L_{y\theta}\left(\mathrm{X}_d\right) + \tau_{motor_\theta}
\end{aligned}$$

And the forces on the inner robot are given as:

$$\begin{aligned}
\Sigma F_{ix} &= C_{ix}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right) - D_{ix}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right) + VR_x\left(\mathrm{X}_d\right) + i_x \cdot L_{xx}\left(\mathrm{X}_d\right) + i_y \cdot L_{yx}\left(\mathrm{X}_d\right) \\
\Sigma F_{iy} &= C_{iy}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right) - D_{iy}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right) + VR_y\left(\mathrm{X}_d\right) + i_x \cdot L_{xy}\left(\mathrm{X}_d\right) + i_y \cdot L_{yy}\left(\mathrm{X}_d\right) \qquad \text{(III-4)} \\
\Sigma \tau_i &= C_{i\theta}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right) - D_{i\theta}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right) + VR_\theta\left(\mathrm{X}_d\right) + i_x \cdot L_{x\theta}\left(\mathrm{X}_d\right) + i_y \cdot L_{y\theta}\left(\mathrm{X}_d\right)
\end{aligned}$$

Explanation of terms. $C_{ox}\left(\mathrm{X}_d, \dot{\mathrm{X}}\right)$ is the rolling friction term. This term is dependent on R_o, the values of the reaction forces on the outer robot (which are in turn a function of X_d). The velocity of a (passive) wheel is given by

$$\begin{aligned}
&v_{wheel} = v_{COM} + \omega \times r_{COM_wheel} \\
&\text{where } v_{COM} = \left(\dot{x}_o, \dot{y}_o\right)
\end{aligned} \qquad \text{(III-5)}$$

And the rolling friction force from a wheel traveling at said velocity can be expressed as

$$-R \cdot C \cdot \frac{v_{wheel}}{\left\| v_{wheel} \right\|} \qquad \text{(III-6)}$$

Where, for example, the x component of force is found from

$$\left[-R \cdot C \cdot \frac{v_{wheel}}{\|v_{wheel}\|}\right] \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \text{(III-7)}$$

Summing these forces over all passive wheels (2 for the outer robot and 4 for the inner robot) gives C.

$D_{ox}(\dot{X})$ is the term dealing with eddy current damping. We assume that eddy damping is a linear function of velocity. Later we justify this assumption.

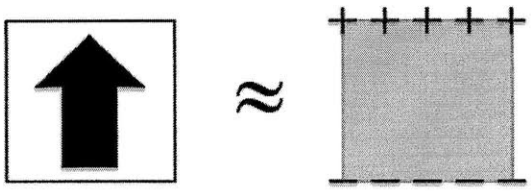
$VR_x(X)$ Is the term for variable reluctance force between paired magnets (the forces between a magnet on the inner robot and its corresponding magnet on the outer robot). This expression is highly nonlinear.

F_{motor_x} is the motor force acting on the chassis. Note that due to the low inertia of the wheels relative to the chassis, the motor is modeled as a pure force source acting on the chassis.

$L_{yx}(X_d)$ is the Lorentz force term due to the coils in the system. For our system, the Lorentz force is only used when the robots are close to one another; in this case we assume the expression $L_{yx}(X_d)$ is constant. Note the relationship between the expressions for force on the inner and outer robot: variable reluctance and Lorentz force terms are equal and opposite (as they act between the robots). Eddy damping and coulomb friction terms are of approximately the same form.

III.1.2. Magnetic forces

We start by modeling the permanent magnets using the magnetic charge model. This is a common approach to this type of problem [4].



III - 4: Magnetic charge model

This model approximates a permanent magnet by a pair of surfaces (located at the poles of the magnet) with some density of magnetic charge. For a magnet with magnetization M_i, the surface charge can be found by looking at the change in magnetization across a boundary.

$$\sigma_{sm} = -n \cdot \mu_o \left(M_i - M_o \right) \tag{III-8}$$

Free space has no magnetization, so

$$\sigma_{sm} = \mu_o M \tag{III-9}$$

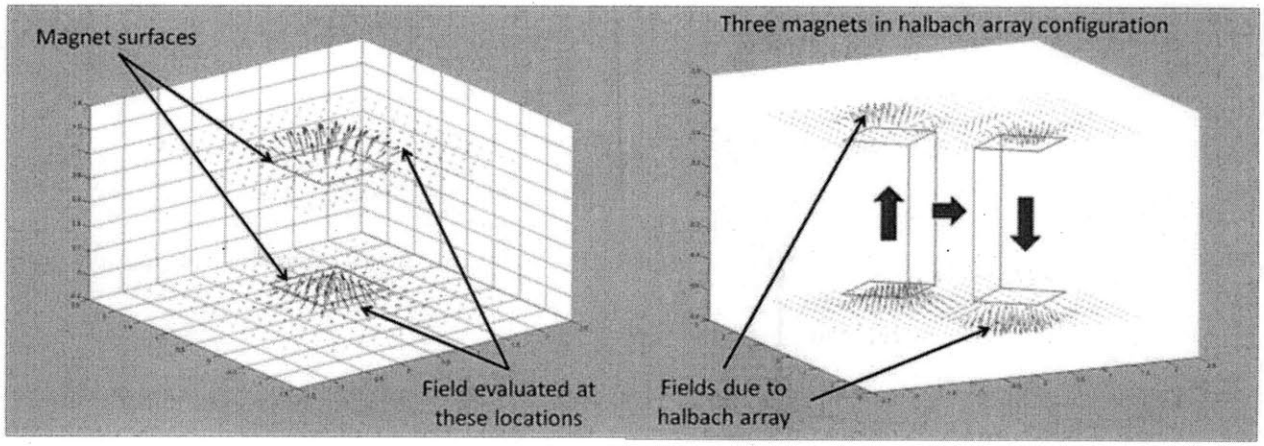
Manufacturers supply data on a magnet's $\left(BH\right)_{\max}$, a measure of the potential energy in a magnet. Magnetization at saturation can be related to $\left(BH\right)_{\max}$ through

$$-\left(BH\right)_{\max} = \mu_0 \left(\frac{M_{sat}}{2} \right)^2 \tag{III-10}$$

We assume the magnet is operating near saturation and use this value to find the magnetic surface charge. In addition, we assume the surface charge is uniform over the magnet pole surfaces.

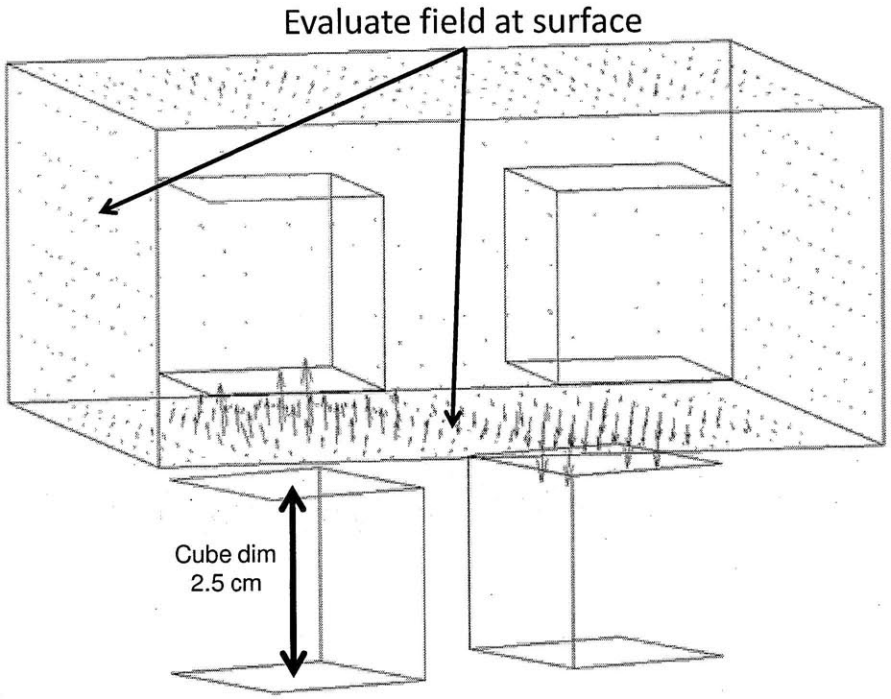
An analytical solution for the fields of a trapezoidal permanent magnet assumed to have uniform surface magnetic charge density is found in [4]. We assume linearity of the neodymium permanent magnets

and apply superposition to find the field due to a pair of halbach arrays. Fig. III-5 shows the fields due to a single magnet and those due to a Halbach array.



III - 5: Fields due to a single magnet (left) and Halbach array (right)

When the field as a function of position is known, it is possible to determine the force and torque on an arbitrary volume in space using Maxwell's stress tensor. To find the forces on the robots, we chose a volume surrounding one Halbach array as shown in Fig. III-6.



III - 6: Evaluating field at some surface surrounding one Halbach array

The surface surrounding this volume was discretized and the field was evaluated at each of these locations. Maxwell's stress tensor about any surface is given as:

$$\begin{bmatrix} \frac{\mu}{2}\left(H_1^2 - H_2^2 - H_3^2\right) & \mu H_1 H_2 & \mu H_1 H_3 \\ \mu H_1 H_2 & \frac{\mu}{2}\left(H_2^2 - H_1^2 - H_3^2\right) & \mu H_2 H_3 \\ \mu H_1 H_3 & \mu H_2 H_3 & \frac{\mu}{2}\left(H_3^2 - H_2^2 - H_1^2\right) \end{bmatrix} \quad \text{(III-11)}$$

Where the subscript on the field refers to the magnitude of field intensity in that direction. That is, H_1 is the magnitude of field strength in the x direction. The force on any surface is given by the surface integral:

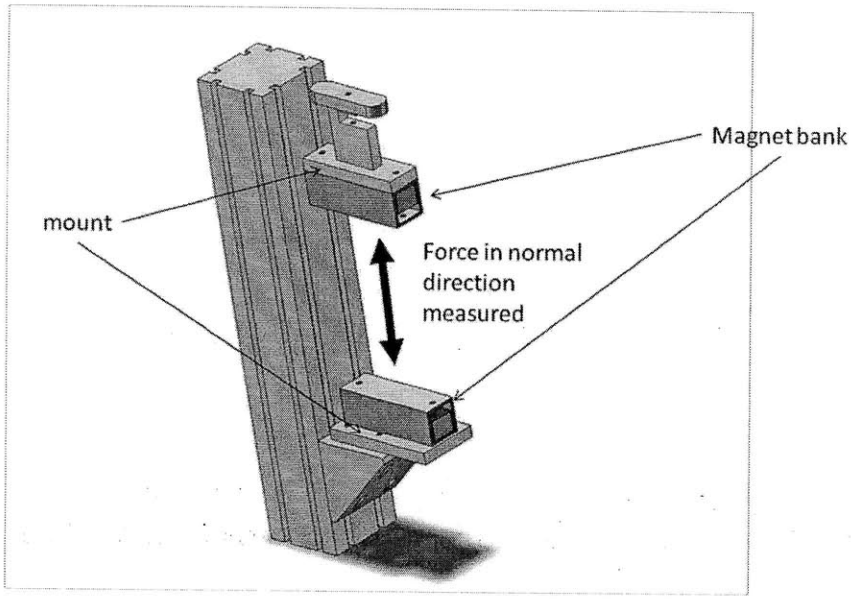
$$f_i = \int T_{ij} n_j dA \qquad \text{(III-12)}$$

Where T_{ij} is the ijth entry of Maxwell's stress tensor and n_j is ± 1, giving the direction of the normal from that surface outwards. For example, to find the force in the y direction on a differential surface element parallel to the yz plane where the interior of the volume is larger in x than the element, one would evaluate

$$-\mu H_1 H_2 dA \qquad \text{(III-13)}$$

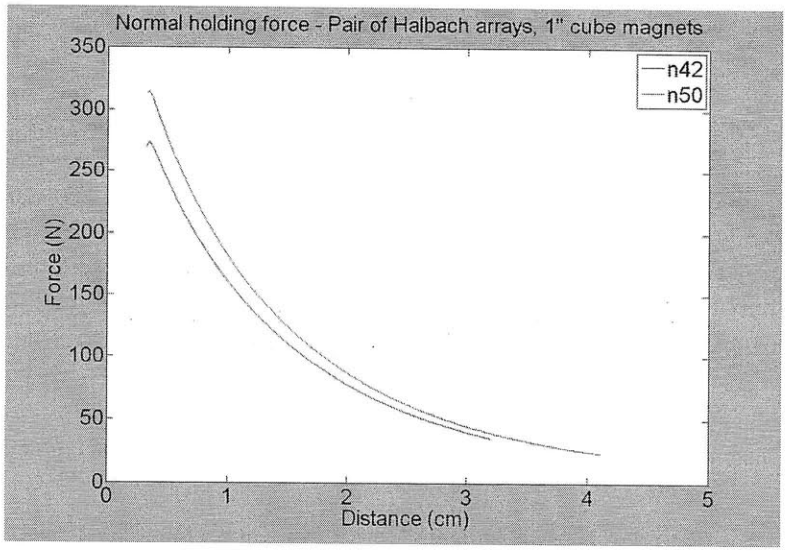
This method gives us forces and torques as a function of relative displacement and orientation of a pair of Halbach arrays of any strength (this is essentially a FEA software solution built from scratch). The resultant force and torque data was fit to a high order polynomial curve. Approximate magnet force data was taken from the curve fit in practice to speed computation in simulations (as expected, calculating a value from a polynomial fit turned out to be much faster than generating and discretizing a surface, evaluating fields, finding the stress tensor and computing a numerical surface integral). Forces obtained in this method did a reasonably good job of matching the magnitudes of those found experimentally. They did a very good job of scaling similarly to experimental results.

Experimental magnet force measurements were taken with an Admet machine able to vary displacement while recording data. A single axis load cell was used to get force information. The setup used is shown.



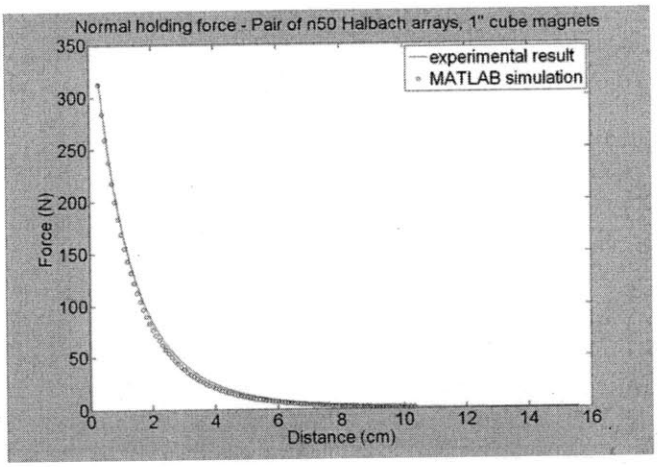
III - 7: Normal force measurement setup

Data was taken for magnet banks of different strengths, as shown. We were unable to take full six axis force measurements. We instead use these uni-axial force measurements as a way of verifying the analytical approach we chose.



III - 8: Normal force measurements for two different grades of magnets

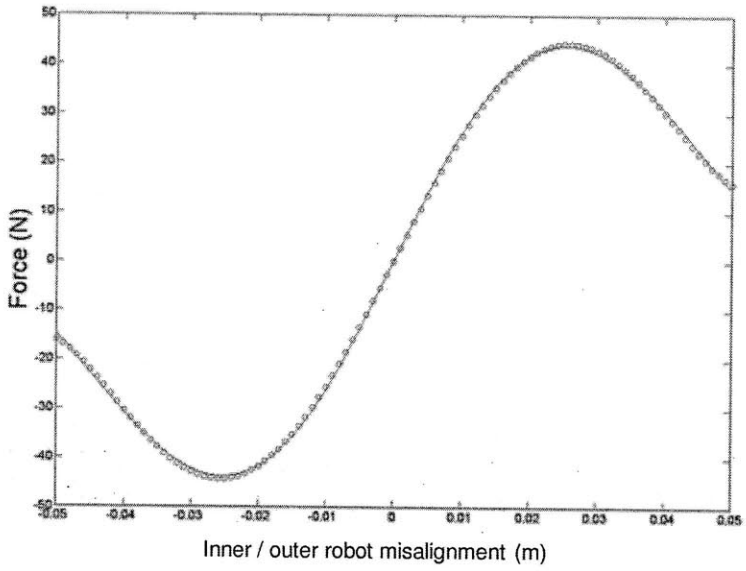
This data was matched to that obtained analytically as described in the previous section.



III - 9: Comparison of simulation to experimental results. Assumptions in magnetic charge density were scaled in simulation to better match experimental data

It should be noted that the data shown in Fig. III-9 was scaled to match experimental results. Although the magnets used were grade N50 (according the manufacturer), the data only matched well when we ran the analysis assuming that the magnets were grade N35. Fortunately, the data scaled very well as a function of position. In practice, we suggest taking a single force measurement from a pair of magnets, and using that information to select the 'assumed grade' of the magnet when performing this type of analysis. We believe that using 'assumed grade' in simulations will yield accurate force and torque data; this allows the engineer to explore the effects of a variety of magnet configurations and locations in simulation.

Code used to compute force and torque data can be found in APPENDIX A. Fig. III-10 shows an example of restoring force as a function of inner / outer robot misalignment generated by this code.



III - 10: Restoring force as a function of inner / outer robot misalignment

The simulation shows results as expected. For small relative displacement of the inner and outer robot, the restoring force is small. This restoring force increases monotonically as the displacement between the robot pair increases, until some point in which its magnitude begins to drop off. Additionally, for small misalignment, the restoring force can be linearized. This plot shows the resultant misalignment force for the case when y_d and θ_d are zero. For $y_d \neq 0$, we would expect a similarly shaped plot, but with reduced magnitude.

III.1.3. Eddy damping

Maxwell's equations state that a changing magnetic field through a surface induces a proportional voltage through a line around that surface.

$$\frac{\partial}{\partial t}\int_A B \cdot dA = -\int_S E \cdot dl \qquad \text{(III-14)}$$

When that surface is in a conductive medium, this voltage creates a current. This current interacts with magnetic fields in the system to oppose the original change in magnetic field. When the change in magnetic field is due to a moving magnet, this is manifested by a force that opposes the motion of the magnet. This results in a 'damping' force felt by a moving magnet.

Eddy current damping is a significant factor in this system due to the high magnetic fields generated by the neodymium magnets as well as the proximity to a large conductive surface (the aluminum skin). This force is often modeled as scaling linearly with respect to velocity. We assume that our system operates in a regime where this is the case, and tested our assumption with an experiment, described here.

We used the passive inner robot from the prototype and ran two sets of trials. In the first set of trials the inner robot was operating on a conductive aluminum skin. In the second set of trials it operated on a non-conductive acrylic sheet. In all trials the robot was accelerated to some velocity (not consistent between trials) and released. An accelerometer attached to the robot recorder acceleration data as the robot slowed; integrating this data gives velocity information.

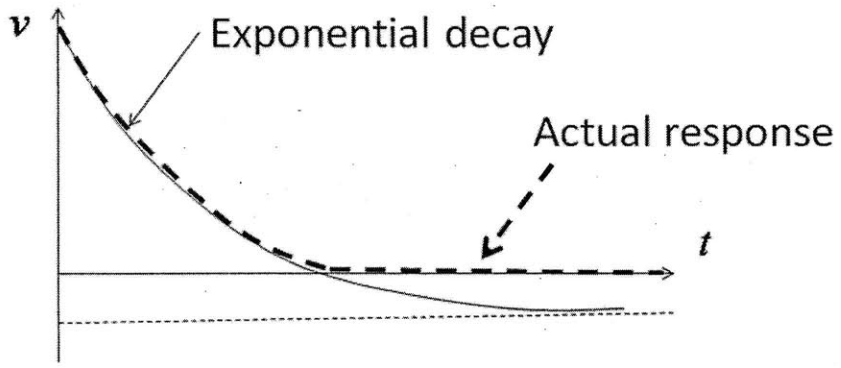
If eddy current damping did in fact act linearly with respect to velocity, the dynamic equations of motion of the system are given as:

$$\begin{aligned} m\ddot{x} &= -b\dot{x} - cN \\ m\dot{v} &= -bv - cN \end{aligned} \qquad \text{(III-15)}$$

Where b is the coefficient of eddy damping, c is the coefficient of rolling friction, and N is the normal force from the skin (equivalent to mg: the product of the mass of the robot and the acceleration due to gravity). Solving these equations is straightforward:

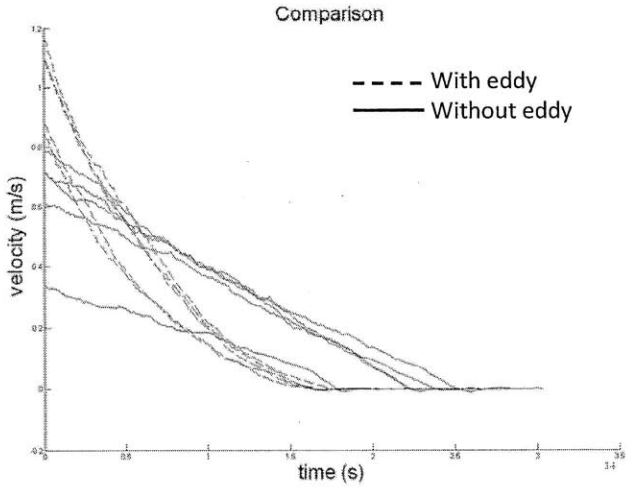
$$v = k\exp\left[-\frac{b}{m}t\right] - \frac{cN}{b} \qquad \text{(III-16)}$$

Note that k is a function of the initial conditions. This response is an exponential decay with an offset. Fig. III-11 shows this response graphically.



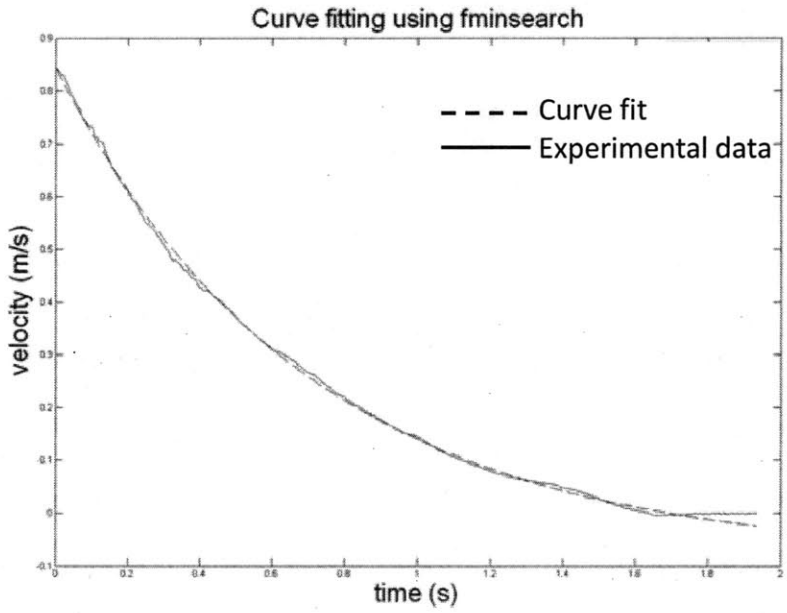
III - 11: Expected response for a system with linear coloumb and viscous damping

Five trials of each scenario (with or without eddy current) were run. Fig. III-12 shows the comparison between these cases. Clearly there is a significant difference when the robot is operating on a conductive surface. As expected, in the case without eddy damping, the velocity has a linear profile.



III - 12: Experimental results with and without eddy currents

A nonlinear minimization was used to fit an exponential curve to the leading (non-zero) portion of one of the trials. Fig. III-13 shows the results.

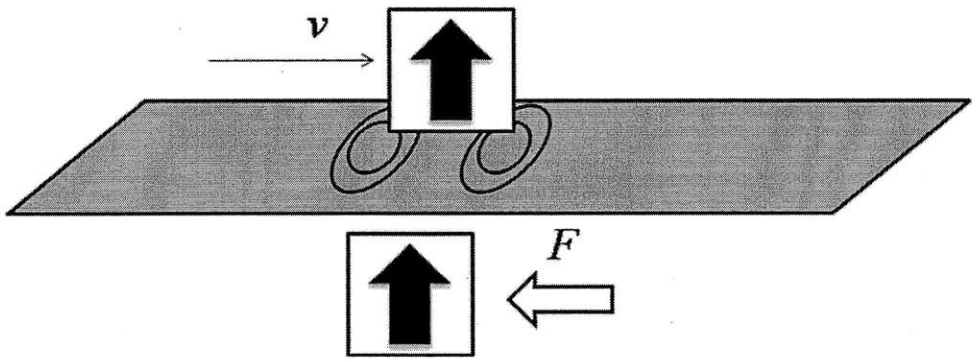


III - 13: Experimental data compared to exponential curve fit

The curve does an excellent job of matching an exponential decay. This suggests that our linear assumption for eddy currents is reasonable. From this set of data, we were also able to find values of b and c. Average values found across trials were:

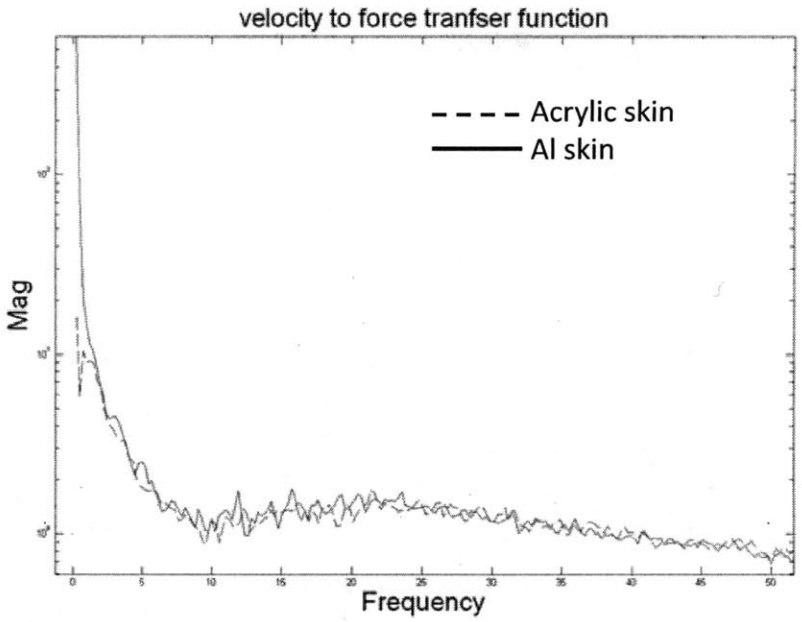
$b \approx 6.8$
$c \approx 0.025$

This system is unique in that it has a pair of magnets across a thin wall. This means that eddy currents produced by magnets from the inner robot could induce forces on magnets on the outer robot.



III - 14: Eddy currents caused by moving inner magnet induce forces on outer magnet

In order to test the magnitude of this force, another test was performed. A pair of Halbach arrays was mounted to opposite sides of a sheet. Again, two sets of trials were run. In the first set of trials the sheet was made of aluminum. In the second set, the sheet was non-conductive acrylic. Position sensors were mounted to the inner array while the outer array was fixed in place to a force sensor. By differentiating position, we were able to find the velocity of the inner Halbach array. We compared the power spectra of the force measurements to the power spectra of the velocity measurements we were able to numerically find a transfer function relating the two variables. Fig. III-15 shows this transfer function.

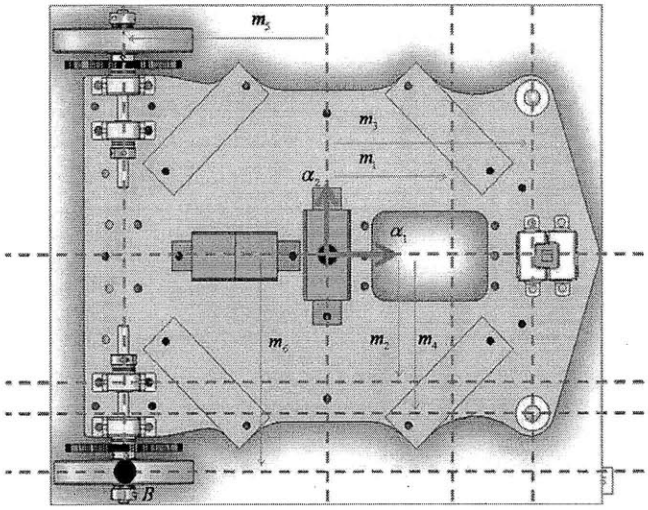


III - 15: Velocity to force transfer function for acrylic and aluminum skin

There appeared to be little discernable difference between the two cases. Based on this result, we chose to ignore the 'cross skin' eddy current forces in our model.

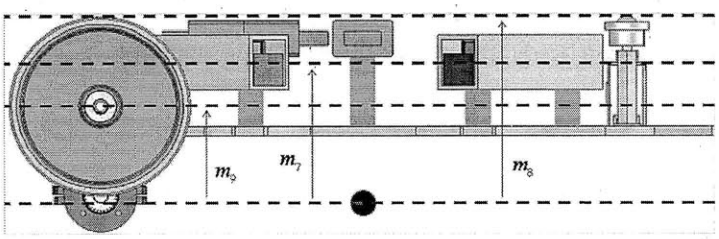
III.1.4. Reaction forces

Fig. III-16 shows the outer robot. We start here by defining some dimensions on the robot. Note the axis $\alpha_1, \alpha_2, \alpha_3$



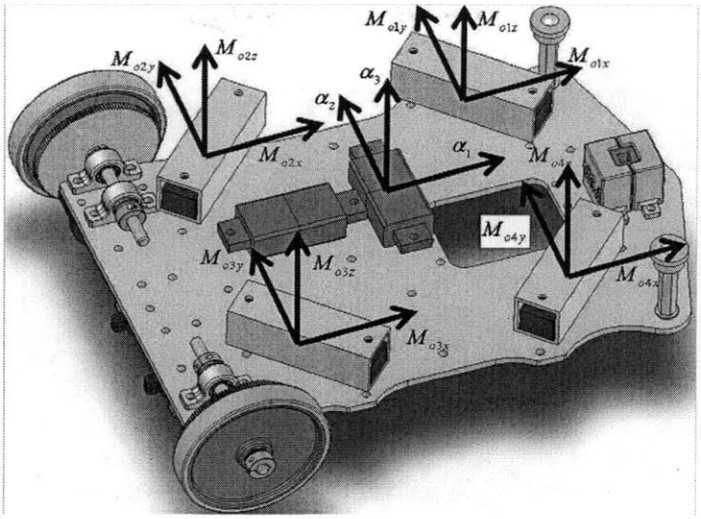
III - 16: Dimensions on outer robot

And we define some dimensions out of plane.



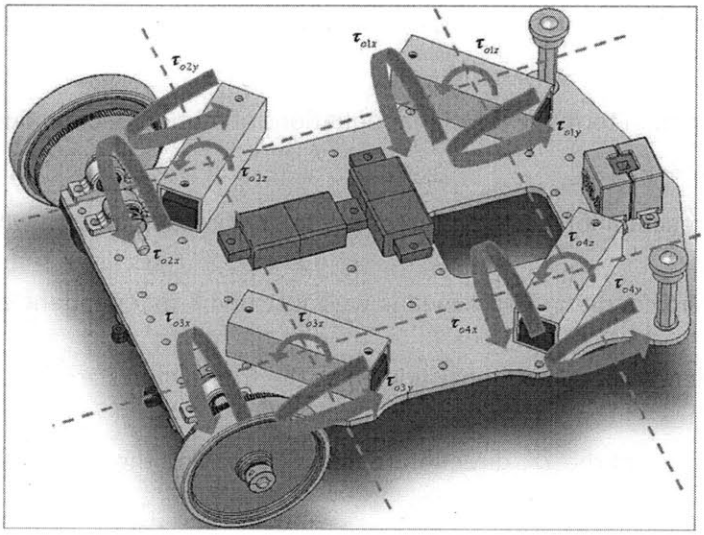
III - 17: Dimensions on outer robot

There is some force on the magnets that can be broken into components:



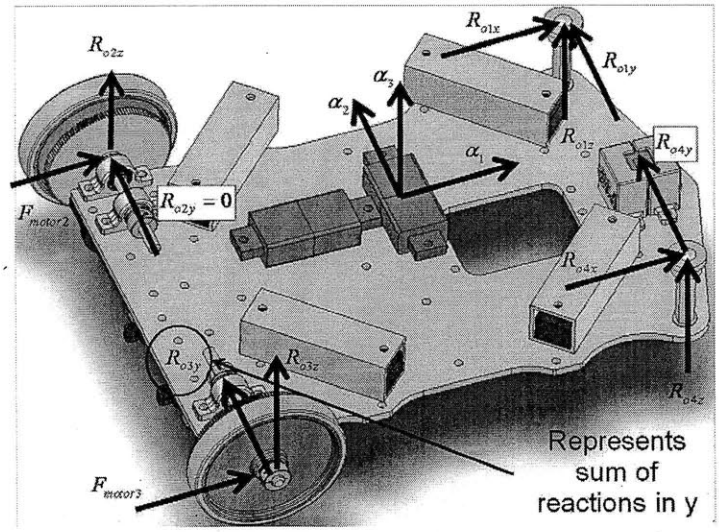
III - 18: Magnet force components

And some torque on the magnets that can also be broken into components:



III - 19: Magnet torques

Now consider the resultant reaction forces at the wheels.



III - 20: Reaction force on outer robot

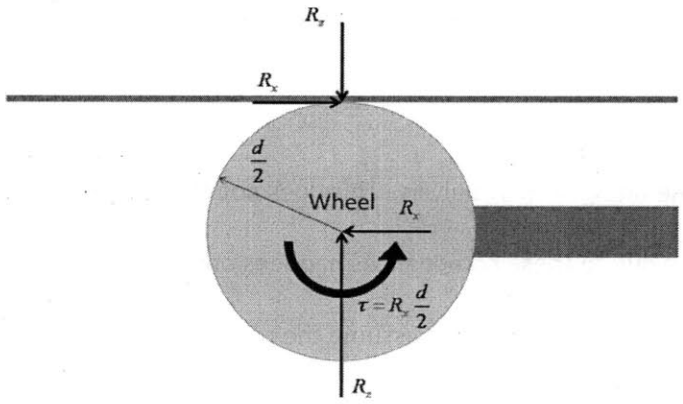
It is important to determine the reaction forces at the wheels for two reasons.

- When a reaction forces goes to zero, the robot is in danger of falling from the ceiling
- The coloumb friction terms in the planar equations of motion depend on reaction forces

There are 5 unknown reaction forces, the 4 forces in the α_3 direction at each wheel, as well as the force that enforces the no-slip condition. This means we must find 5 independent equations. The reaction forces in the α_1, α_2 plane (lateral and longitudinal) at the front wheels are due to rolling friction. They are deterministic functions of the α_3 direction reaction forces and the robot velocity and can be solved for directly.

The constraint reaction force R_{o3y} enforces the no slip condition. In reality this force is the sum of the y direction forces at both rear wheels. Mathematically, however, this representation is equivalent and is chosen for simplicity in analysis.

Note that the α_1 direction forces on the rear wheels (due to motor torque) act at the *axle* of the wheel. This is because the intertia of the wheel is small enough that we neglect it, assuming instead that the motor torque (and resultant force) acts directly on the robot chassis. Fig. III-21 shows this relationship for a wheel with zero inertia.



III - 21: Effect of massless rear wheel assumption

We assume planar motion of the outer robot. This implies the sum of torques about the α_1 and α_2 axes are equal to zero. This allows us to write two equations.

$$\begin{aligned} &m_2 M_{o1z} - m_7 M_{o1y} + m_2 M_{o2z} - m_7 M_{o2y} - m_2 M_{o3z} - m_7 M_{o3y} \\ &- m_2 M_{o4z} - m_7 M_{o4y} + \tau_{o1x} + \tau_{o2x} + \tau_{o3x} + \tau_{o4x} \\ &+ m_4 R_{o1z} - m_8 R_{o1y} + m_4 R_{o2z} - m_4 R_{o3z} - m_9 R_{o3y} - m_4 R_{o4z} - m_8 R_{o4y} = 0 \end{aligned} \qquad \text{(III-17)}$$

And

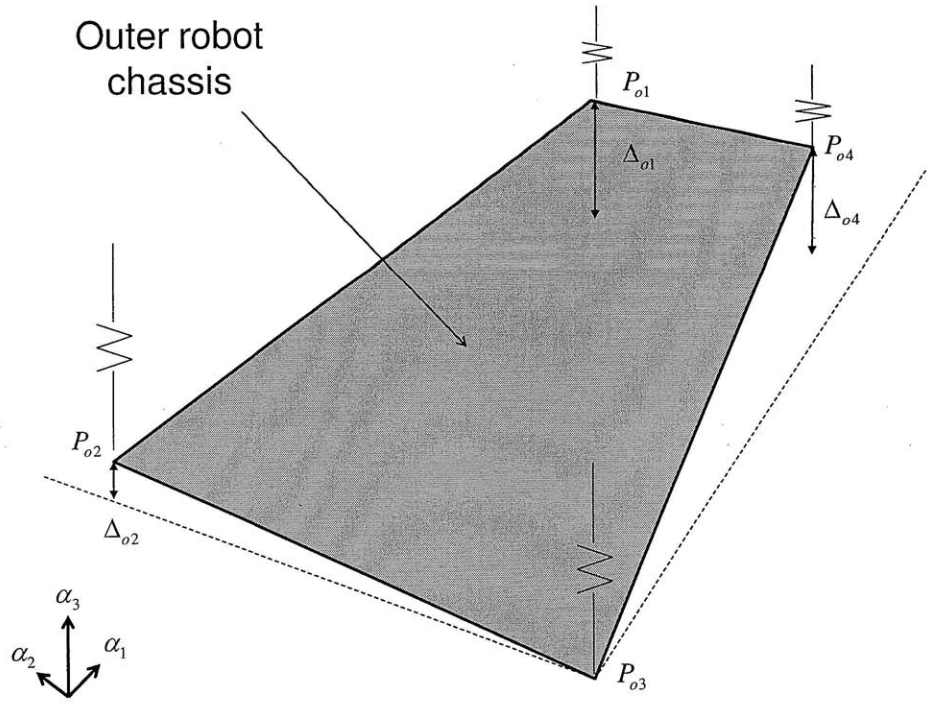
$$\begin{aligned} &m_7 M_{o1x} - m_1 M_{o1z} + m_7 M_{o2x} + m_1 M_{o2z} + m_7 M_{o3x} + \\ &m_1 M_{o3z} + m_7 M_{o4x} - m_1 M_{o4z} + \tau_{o1y} + \tau_{o2y} + \tau_{o3y} + \tau_{o4y} \\ &+ m_8 R_{o1x} - m_3 R_{o1z} + m_9 F_{motor2} + m_3 R_{o2z} + m_9 F_{motor3} + m_3 R_{o3z} + m_8 R_{o4x} - m_3 R_{o4z} = 0 \end{aligned} \qquad \text{(III-18)}$$

A third equation comes by setting forces along the z axis equal to 0 (no z axis motion is allowed).

$$M_{o1z} + M_{o2z} + M_{o3z} + M_{o4z} + R_{o1z} + R_{o2z} + R_{o3z} + R_{o4z} = m_o g \qquad \text{(III-19)}$$

Our next equation comes from the method of deformations. A common approach in statically indeterminate systems, the method of deformation assumes some finite stiffness for portions of the body and distributes forces accordingly. Here we assume the chassis remains infinitely stuff but that there is some compliance due to Hertz contact stresses in the ball wheel bearings as well as compliance in the rubber wheels.

We allow for out of plane tilting in this analysis. This does not contradict our previous assertions as we assume any deformation due to these compliances happens on a much faster time scale than the previously describe dynamics. From this, we assume the relationships found from the method of deformation are algebraic, not dynamic.

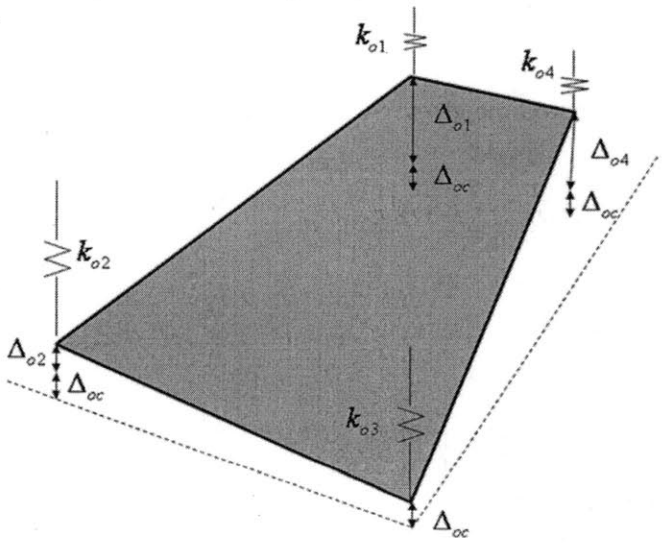


III - 22: Tilt of outer robot using method of deformations

Fig. III-22 shows the outer robot chassis. Δ_{oi} is the deflection of that corner of the chassis. Deflection is assumed to be strictly in the α_3 direction (based on a small angle assumption). Because the chassis is a rigid plane, if we are given the location of three points, we can find the fourth. If we assume P_{o3} is at the origin, we can say.

$$a_o\left(m_5+m_3\right)+\frac{\Delta_{o2}}{2m_6}\left(m_6-m_4\right)=\Delta_{o4} \qquad \text{(III-20)}$$

This gives us a relationship between the *differences* in vertex locations. This is essentially an expression for the normal to this plane. To find actual vertex locations, we need to offset by Δ_{oc} as shown in Fig. III-23.



III - 23: Total deflection of outer robot using method of deformations

From this figure we can relate total displacement, stiffness at that corner, and reaction force at that corner. For a compliant member, $F=k\Delta$. We plug in and get:

$$-\frac{m_4 R_{o2z}}{m_6 k_{o2}}+\frac{m_4 R_{o3z}}{m_6 k_{o3}}-\frac{R_{o4z}}{k_{o4}}+\frac{R_{o1z}}{k_{o1}}=0 \qquad \text{(III-21)}$$

This is our 4th equation. To find our last equation, we look at the constraint imposed by the constraint force. The no slip condition requires that velocity at the rear wheels is only in the α_1 direction. That is, for some point B located at the rear wheel,

$$v_B=\begin{bmatrix} v_{bx} \\ 0 \\ 0 \end{bmatrix} \qquad \text{(III-22)}$$

And if we know the velocity of the center of mass, v_{COM}, we can say

$$v_B = v_{COM} + w \times r_{COM\_B} \qquad \text{(III-23)}$$

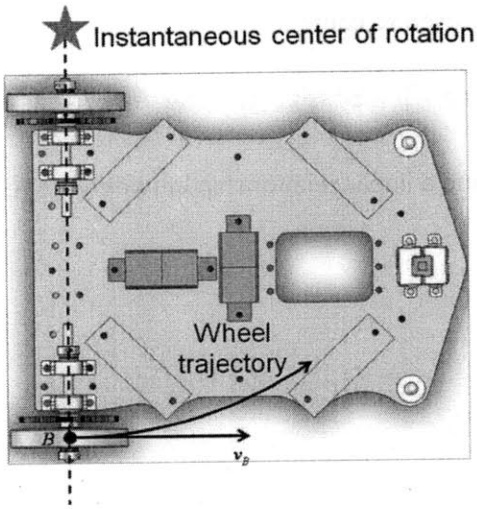
Differentiating this with respect to time gives

$$a_B = a_{COM} + \dot{w} \times r_{COM\_B} + w \times \dot{r}_{COM\_B} \qquad \text{(III-24)}$$

This can be written in vector form.

$$\begin{bmatrix} a_{Bx} \\ a_{By} \\ 0 \end{bmatrix} = \begin{bmatrix} a_{COMx} \\ a_{COMy} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix} \times \begin{bmatrix} -m_5 \\ -m_6 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \times \left(\begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \times \begin{bmatrix} -m_5 \\ -m_6 \\ 0 \end{bmatrix} \right) \qquad \text{(III-25)}$$

Consider the term a_{By}, the acceleration of the point B in the y (α_2) direction. Because of the no slip condition, the instantaneous center of rotation of the system is located on an axis as show in Fig. III-24.



III - 24: Instantaneous center of rotation showing acceleration of wheel

The point B is instantaneously traveling along a circular (with radius r) trajectory. This means the acceleration in the α_2 direction of the point B is $\frac{v_B^2}{r}$. Plugging this in to the acceleration equation above yields:

$$\frac{v_B^2}{r} = a_{COMy} - \ddot{\theta} m_5 - \dot{\theta}^2 m_6 \tag{III-26}$$

The acceleration of the center of mass in the y direction is a function of all the forces in the y direction. This expression (and our final equation) is:

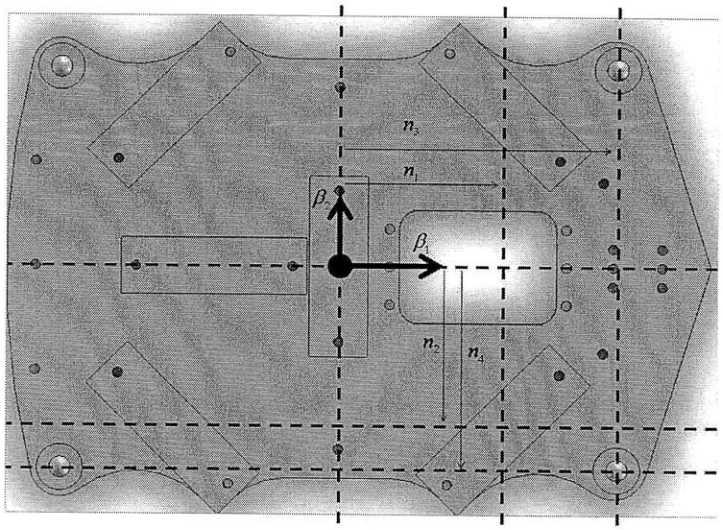
$$0 = \frac{v_{oCy} v_{oCx}}{m_5} + \frac{R_{o1y} + R_{o3y} + R_{o4y} + M_{o1y} + M_{o2y} + M_{o3y} + M_{o4y}}{m_o} - \left(\begin{array}{l} m_1 M_{o1y} - m_2 M_{o1x} - m_1 M_{o2y} - m_2 M_{o2x} - m_1 M_{o3y} + m_2 M_{o3x} + \\ m_1 M_{o4y} + m_2 M_{o4x} + \tau_{o1z} + \tau_{o2z} + \tau_{o3z} + \tau_{o4z} + m_3 R_{o1y} - m_4 R_{o1x} - \\ m_6 F_{motor2} - m_5 R_{o3y} + m_6 F_{motor3} + m_3 R_{o4y} + m_4 R_{o4x} \end{array} \right) \frac{m_5}{I_{oC}} - 2 \frac{v_{oCy}^2 m_6}{m_5^2} \tag{III-27}$$

An analytical solver in MATLAB was used to come up with these expressions. See code in APPENDIX for more details. As previously mentioned, it is possible to write R_{oix}, R_{oiy} as a deterministic function of R_{oiz} and velocity. It is possible to find a linear relationship between the remaining unknown reaction forces of the form

$$\boldsymbol{A}\begin{bmatrix} R_{o1z} \\ R_{o2z} \\ R_{o3z} \\ R_{o4z} \\ R_{o3y} \end{bmatrix} = \boldsymbol{b} \qquad \text{(III-28)}$$

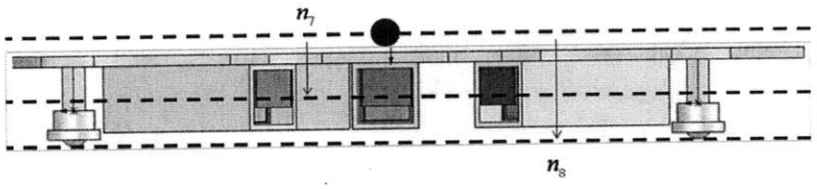
Where all the components of $\boldsymbol{A}$ and $\boldsymbol{b}$ are known. This can be solved by inverting the A matrix to find the reaction forces.

A similar method was used to find the reaction forces on the inner robot. The lack of constraint force makes the analysis simpler. We start again by defining robot dimensions, first in the plane



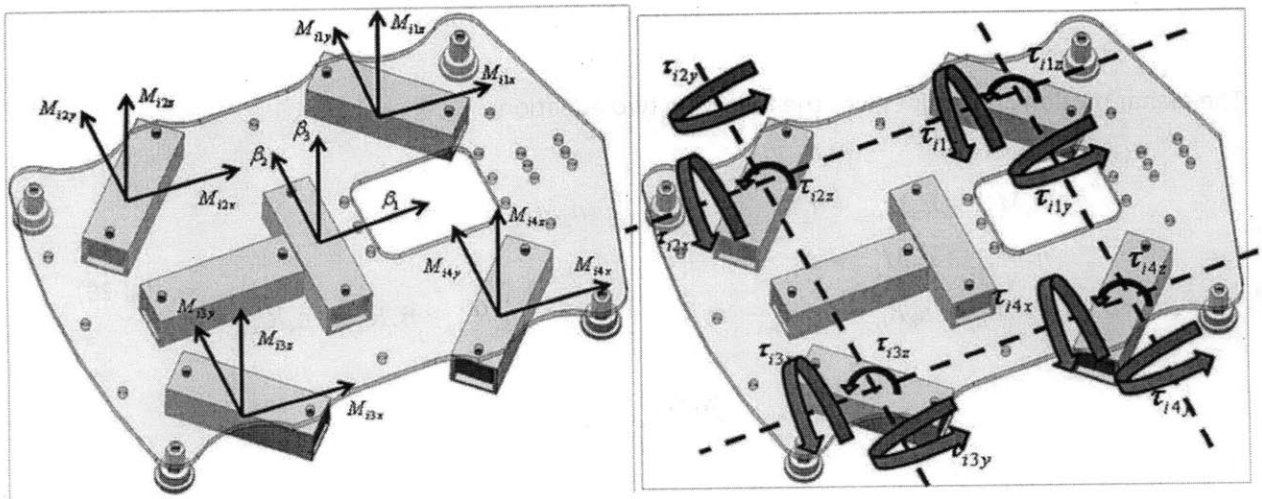
III - 25: Inner robot dimensions

And next, out of plane



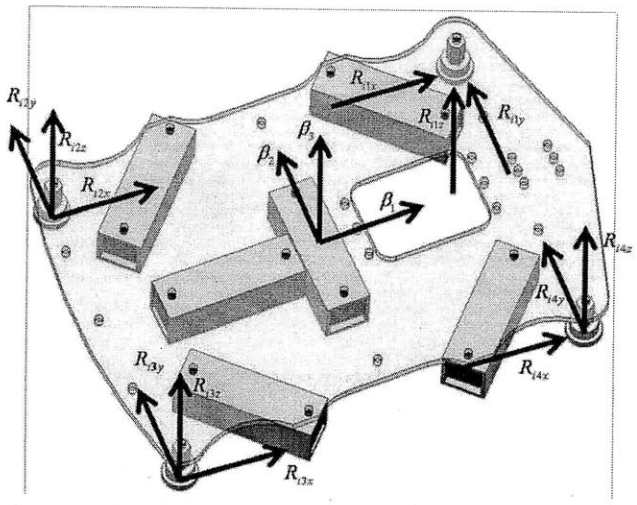
III - 26: Inner robot dimensions

Next we define magnet forces



III - 27: Magnet forces and torques on inner robot

And reaction forces



III - 28: Reaction forces inner robot

The planar motion constraint gives the following two equations

$$\begin{aligned} &n_2M_{i1z}+n_7M_{i1y}+n_2M_{i2z}+m_7M_{i2y}-n_2M_{i3z}+n_7M_{i3y}- \\ &n_2M_{i4z}+n_7M_{i4y}+\tau_{i1x}+\tau_{i2x}+\tau_{i3x}+\tau_{i4x} \\ &+n_4R_{i1z}+n_8R_{i1y}+n_4R_{i2z}+n_8R_{i2y}-n_4R_{i3z}+n_8R_{i3y}-n_4R_{i4z}+n_8R_{i4y}=0 \end{aligned} \quad \text{(III-29)}$$

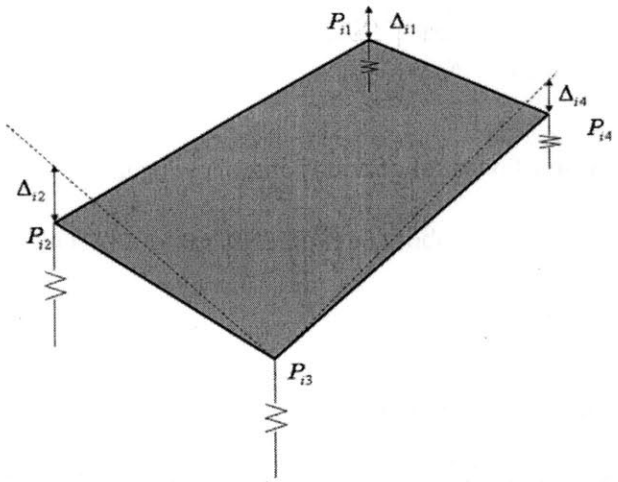
And

$$\begin{aligned} &-n_7M_{i1x}-n_1M_{i1z}-n_7M_{i2x}+n_1M_{i1z}-n_7M_{i3x}+n_1M_{i3z}- \\ &n_7M_{i4x}-n_1M_{i4z}+\tau_{i1y}+\tau_{i2y}+\tau_{i3y}+\tau_{i4y} \\ &-n_8R_{i1x}-n_3R_{i1z}-n_8R_{i2x}+n_3R_{i2z}-n_8R_{i3x}+n_3R_{i3z}-n_8R_{i4x}-n_3R_{i4z}=0 \end{aligned} \quad \text{(III-30)}$$

The third equation comes from summing forces in the β_3 direction to 0

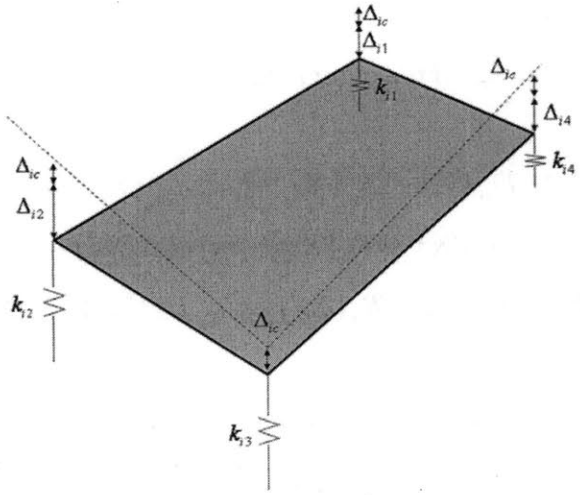
$$R_{i1z}+R_{i2z}+R_{i3z}+R_{i4z}+M_{i1z}+M_{i2z}+M_{i3z}+M_{i4z}-m_ig=0 \quad \text{(III-31)}$$

The fourth (and in this case final) equation comes again from the method of deformations. Fig. III-29 shows the chassis tilted to some angle due to deformation.



III - 29: Tilt of inner robot using method of deformations

As in the previous case, we generate an equation for the plane of the chassis. We then consider the offset of each corner as shown in Fig. III-30. Care should be taken to keep sign conventions correct.



III - 30: Total deflection of outer robot using method of deformations

This gives us our final equation:

$$-\frac{m_4 R_{o2z}}{m_6 k_{o2}}+\frac{m_4 R_{o3z}}{m_6 k_{o3}}-\frac{R_{o4z}}{k_{o4}}+\frac{R_{o1z}}{k_{o1}}=0 \qquad \text{(III-32)}$$

These four equations constitute a linear set of equations, allowing us to solve for reaction forces in the same manner as was used in the outer robot. See APPENDIX B for MATLAB code used to solve for these reaction forces.

Now that all the forces on the system are known, the dynamics of the system are straightforward (as mentioned previously). A simulation was written in MATLAB (see APPENDIX B). However, due to the complexity of this system, running simulations was computationally expensive. Several simpler models are described in the control chapter. These models capture the dynamics of the system relevant to the particular task, and are tractable enough so that it is possible to generate and test controllers.

III.2. Summary

In summary, the equations of motion of the outer robot are given as

$$\begin{aligned}
m_o\ddot{x}_o &= C_{ox}\left(\mathrm{X}_d,\dot{\mathrm{X}}\right)-D_{ox}\left(\dot{\mathrm{X}}\right)-VR_x\left(\mathrm{X}_d\right)- \quad i_x\cdot L_{xx}\left(\mathrm{X}_d\right)-i_y\cdot L_{yx}\left(\mathrm{X}_d\right)+F_{motor_x}\\
m_o\ddot{y}_o &= C_{oy}\left(\mathrm{X}_d,\dot{\mathrm{X}}\right)-D_{oy}\left(\dot{\mathrm{X}}\right)-VR_y\left(\mathrm{X}_d\right)- \quad i_x\cdot L_{xy}\left(\mathrm{X}_d\right)-i_y\cdot L_{yy}\left(\mathrm{X}_d\right)+F_{motor_y} \qquad \text{(III-33)}\\
I_o\ddot{\theta}_o &= C_{o\theta}\left(\mathrm{X}_d,\dot{\mathrm{X}}\right)-D_{o\theta}\left(\dot{\mathrm{X}}\right)-VR_\theta\left(\mathrm{X}_d\right)- \quad i_x\cdot L_{x\theta}\left(\mathrm{X}_d\right)-i_y\cdot L_{y\theta}\left(\mathrm{X}_d\right)+\tau_{motor_\theta}
\end{aligned}$$

Where $C_{ox}\left(\mathrm{X}_d,\dot{\mathrm{X}}\right)$ Is the coulomb friction term, acting as described in section III.1.1, and is dependent on reaction forces, which must be found as described in section III.1.4. $D_{ox}\left(\dot{\mathrm{X}}\right)$ Is an eddy current damping term, acting as described in section III.1.3. And $VR_x\left(\mathrm{X}_d\right)$ is a variable reluctance term, acting as described in III.1.2. The remaining terms are inputs to the system and are dealt with in section IV (system control).

In the vicinity of $X_d = 0$, this system model can be simplified to:

$$\begin{aligned} m_o \ddot{x}_o &= -c \cdot \text{sign}(\dot{x}_o) - B_x \dot{x}_o - k_x x_d - i_x \cdot L_{xx} + F_{motor\_x} \\ m_o \ddot{y}_o &= -c \cdot \text{sign}(\dot{y}_o) - B_y \dot{y}_o - k_y y_d - i_y \cdot L_{yy} + F_{motor\_y} \\ I_o \ddot{\theta}_o &= -c_\theta \cdot \text{sign}(\dot{\theta}_o) - B_\theta \dot{\theta}_o - k_\theta \theta_d + \tau_{motor\_\theta} \end{aligned} \qquad \text{(III-34)}$$

This is a nearly linearized (excepting the coulomb friction terms) version of the full system model. Similar equations have been obtained for the inner robot. The use of such equations of motion, linearized or otherwise, will be explored in section IV.

IV. Control

We start this section by splitting the functionality of the robotic system into two portions: gross motion and fine positioning. Gross motion refers to system traversal between fasteners. This is generally for distances > 1 cm. Our goal is rapid, safe traversal. Fine positioning refers to precise alignment of the robotic system with fasteners. Fine positioning stroke is generally limited to < 1 cm. Our goal in this situation is small positioning error achieved by a stable closed loop controller.

Because of the differences in functional requirements and system characteristics in these scenarios, distinct modeling and control approaches are used.

IV.1. Point to point optimal control

IV.1.1. Control strategy

In this section we present results for an optimal trajectory for a simplified system model. We start with a pair of aligned robots with some desired end location. First, we turn the robot pair to face the desired endpoint. Next, we move the robot pair in a straight line towards this endpoint, following a pre-computed optimal trajectory.

IV - 1: Point to point control: robot first aligns with desired endpoint, then drives in a straight line

IV.1.2. Key issues

The goal during this type of locomotion is to minimize time while maintaining 'safe' motion. Safe motion means the outer robot should not fall –we must monitor reaction forces on the outer robot

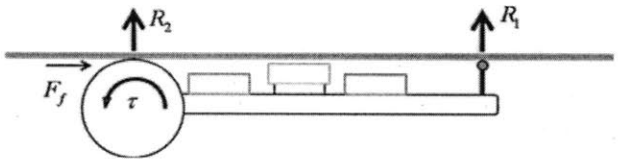
wheels. Additionally, we are interested in maintaining a no slip condition on the outer robot. Slipping renders encoder information for position and velocity useless and it makes our model invalid. Often, the no slip condition is a more conservative metric than the no-fall condition. This is because the no-slip condition requires reaction force greater than some constant while the no-fall condition requires reaction forces greater than zero.

In order to find an aggressive yet safe optimal trajectory, we are interested in exploring the full nonlinear effects of forces such as variable reluctance and rolling friction. To address this, we use a software program to find a numerically optimal trajectory.

IV.1.3. Simplified modeling and control for point to point optimal control

Due to our control strategy, we can use a greatly simplified version of the full system model. In this case, the system has only 2 DOF (inner and outer robot position along the axis of motion) and the dynamic equations of motion only use 4 state variables, making the state vector $x_d, \dot{x}_i, \dot{x}_o, x_o$

There are two constraints placed on the optimal control problem (1) no slip condition and (2) no fall condition.



IV - 2: No slip and no fall conditions are dependent on reaction forces at wheels

From Fig. IV-2, these can be expressed as (note sign convention):

$$0 > R_2, 0 > R_1 \qquad \text{(IV-1)}$$

And

$$\left|\frac{\tau}{r}\right| < \left|\mu R_2\right| \tag{IV-2}$$

Boundary conditions on the optimal control problem maintain that the position and velocity of the inner and outer robot start at zero. At the end of the trajectory, velocity of both robots must be zero and they must be located at the desired endpoint.

This optimization was performed to minimize time. The fastener installation procedure is highly repetitive and a bottleneck in the wing manufacturing process, making time savings of critical importance. Minimizing energy, for example, is not useful as potential energy savings would be dwarfed by tooling energy expenditure.

Formally, the solution to the optimal control problem is the trajectory $u^*(t)$, where

$$u^*(t) = \arg\min_{u(t)} \int_0^{t_f} 1 dt \tag{IV-3}$$

Subject to the dynamics (as described in eqs III-33)

$$\left(\dot{x}_d, \ddot{x}_i, \ddot{x}_o, \dot{x}_o\right) = f\left(x_d, \dot{x}_i, \dot{x}_o, x_o, u\right) \tag{IV-4}$$

With the boundary conditions:

$$\left(x_d\left(t=0\right),\dot{x}_i\left(t=0\right),\dot{x}_o\left(t=0\right),x_o\left(t=0\right)\right)$$
$$=\left(0,0,0,0\right)$$

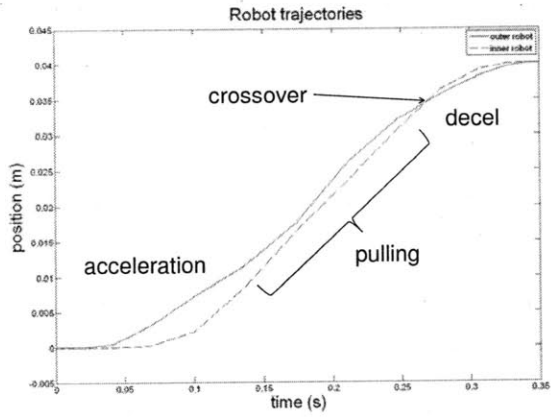
(IV-5)

$$\left(x_d\left(t=t_f\right),\dot{x}_i\left(t=t_f\right),\dot{x}_o\left(t=t_f\right),x_o\left(t=t_f\right)\right)$$
$$=\left(0,0,0,x_f\right)$$

And subject to the constraints (no slip and no fall) as described in eqs IV-1 and IV-2

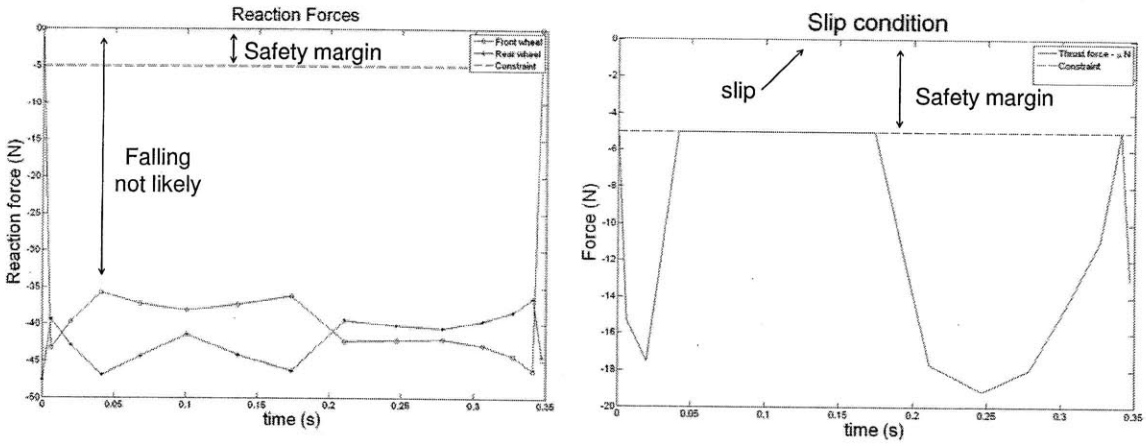
$$\underline{g}\left(x_d,\dot{x}_i,\dot{x}_o,x_o,u\right)>0 \qquad \text{(IV-6)}$$

Fig. IV-3 shows the results of an optimal trajectory. The software used in computing this trajectory is the academic version of DIDO.



IV - 3: Example optimal trajectory for point to point control

In the initial portion of the trajectory, the outer robot stays ahead of the inner robot and pulls it forward. Near the end of the trajectory, the outer robot drops behind the inner robot in order to slow it to a stop. This ensures that both robots get to the desired endpoint with zero velocity. This result is similar to that found in vibration suppression of cranes [36].



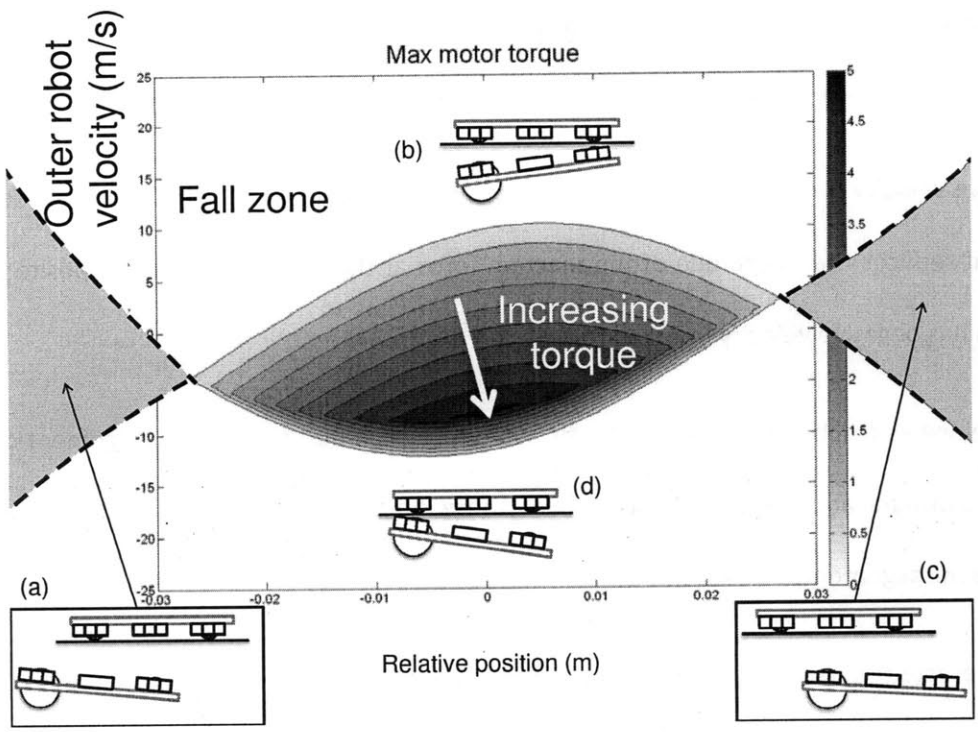
IV - 4: Time evolution of no fall and no slip condition

Fig. IV-4 shows the evolution of the no-slip condition and the reaction forces during this trajectory. Note that while the robot never comes close to falling, it does skirt the safety margin for the no slip condition.

IV.1.4. Further discussion

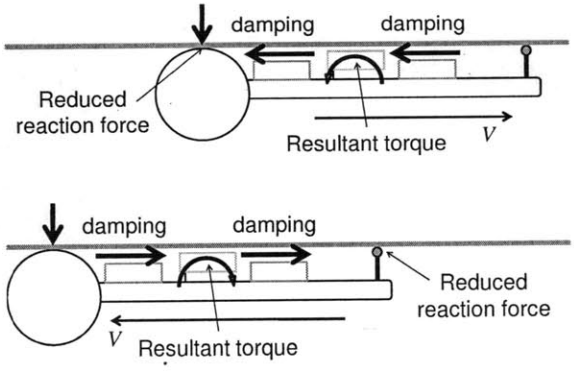
In order to better understand this system, we vary its parameters and observe their effect on performance. In particular we are interested in exploring parameters unique to such a system – principally in the variation of magnetic forces caused by robot misalignment and eddy current damping. Finding the numerical optimal trajectory is computationally intensive, so we focus on a simpler performance metric: ability of the outer robot to accelerate or decelerate.

We start by finding the maximum torque we can apply to the system before either slippage or falling of the outer robot. Fig. IV-5 shows how the maximum allowable torque varies as a function of relative displacement of the robots and velocity of the outer robot. Outside of this contour is the zone in which the outer robot falls.



IV - 5: Maximum torque applicable before slippage. Outside of the shaded region, the outer robot falls

When the system is grossly misaligned, the outer robot is likely to fall (see (a) and (b) in Fig. IV-5). For positive velocities the rear wheel tends to fall, while for negative velocities the front wheel tends to fall (see (c) and (d) in Fig. IV-5). This can be explained by eddy forces on the outer robot as shown in Fig. IV-6.



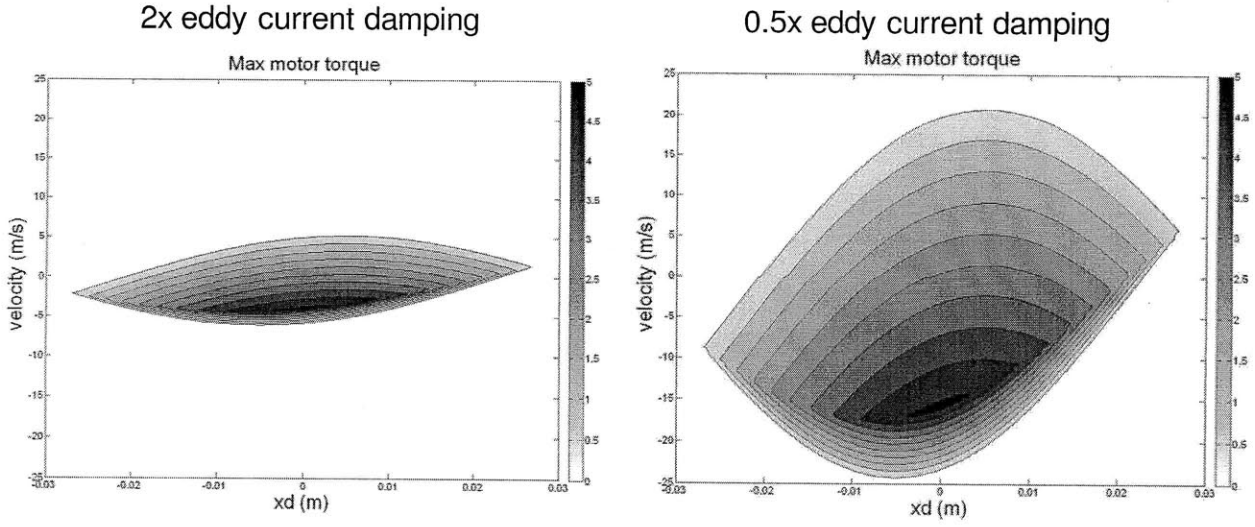
IV - 6: Changes in reaction forces due to velocity

These results can also be found by looking at a special (1 DOF case) of eqs IV-1 and IV-2. Note that to find the terms in these equations, we must solve for reaction forces as described in section III.1.4.

At a positive velocity (marked (b) on the figure), eddy damping on the magnets causes a torque on the system that reduces the magnitude of the reaction force at the rear wheel. The opposite is true when the system has negative velocity (marked (d)).

At small velocities (such as at (a) and (c)), the system can fail due to large misalignment of the robots (x_d). The contours are iso-torque lines where these opposing factors (falling due to eddy current damping vs misalignment) tradeoff.

We are interested in exploring how this behavior is affected by the system parameters, such as eddy current damping. Consider the case that the skin is thicker and the eddy damping more pronounced, as well as the case in which the skin in less conductive (reduced eddy damping). Max allowable torque for these cases is shown in Fig. IV-7.

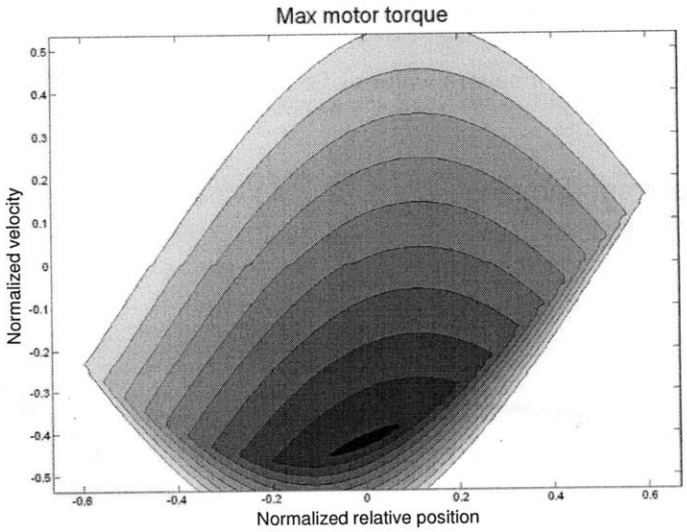


IV - 7: Allowable motor torques for variation in eddy current

For the most part, as eddy current damping increases, viable states of operation shrink.

We observe an interesting phenomenon if we nondimentionalize the velocity and position in the following manner: a characteristic position dimension is chosen as the displacement required before falling (assuming zero velocity). This is around 2.3 cm. A characteristic velocity dimension is chosen as the maximum steady velocity the inner robot can maintain without separation (at this velocity, the maximum possible variable reluctance realignment force is equal to eddy damping for the inner robot – it cannot be pulled any faster). This characteristic velocity does not deal with falling or slippage but is a function of eddy currents. For our system, it is approximately 20 m/s. This comes from solving for the reaction forces of the system – see section III.1.4. In this case, we are interested in a 1 DOF simplified case of the full system model, making solving for these reaction forces less computationally taxing.

Fig. IV-8 shows the results of this plot. Unlike the previous plots, this is *invariant with respect to eddy current coefficient.*



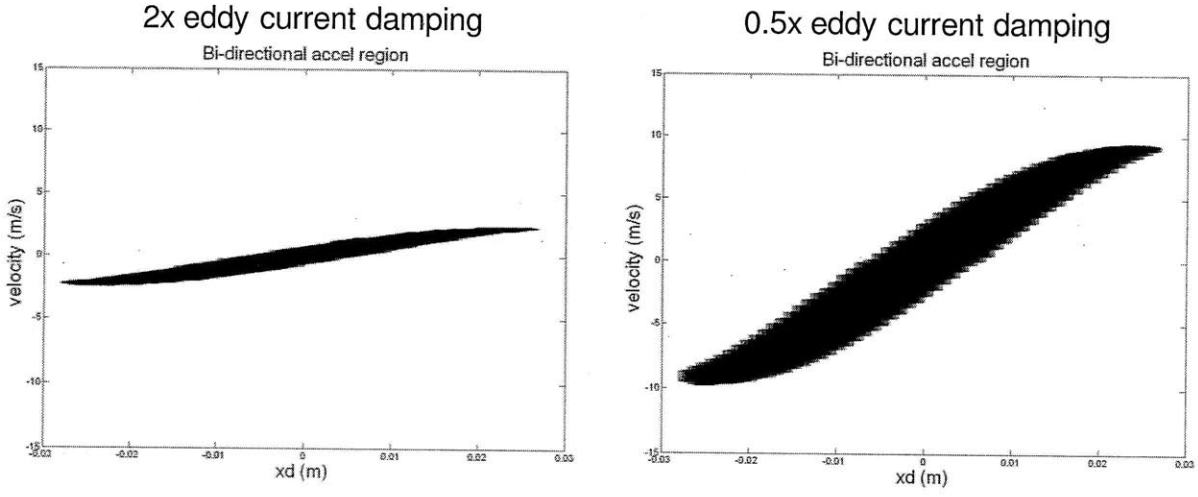
IV - 8: Normalized velocity and displacement

Note that the velocity never reaches 'maximum velocity' as described in the previous paragraph (reaches a maximum value around 0.6). This suggests that slippage / falling of the outer robot becomes

critical before factors such as separation of the inner and outer robot due to drag forces for our current design, regardless of eddy current properties. This graph is a useful tool when evaluating a candidate design for this system and eddy current properties are unknown.

The previous plots show the regions in which it is possible to apply a motor torque without slippage or falling of the outer robot. It should be noted, however, that even if it is possible to apply a motor torque, this torque may not be enough to specify the *direction* of acceleration. That is, in some cases, although we apply a motor torque, we can change on the *magnitude* of acceleration, not the *direction.* This is the case when other forces (such as magnet forces) are much larger than the forces we are able to apply. For example, sufficient misalignment may cause very large restoring forces from the permanent magnets. Although we may be able to provide a motor torque that opposes this realignment force, we cannot cancel it completely without wheel slippage.

We would prefer to always be able to control the direction of acceleration - Fig. IV-9 shows the conditions when this is possible for several values of eddy current damping coefficients.



IV - 9: Region in which we can control *direction* of acceleration for two eddy current values

Note that these are subsets of the contours shown in Fig. IV-7.

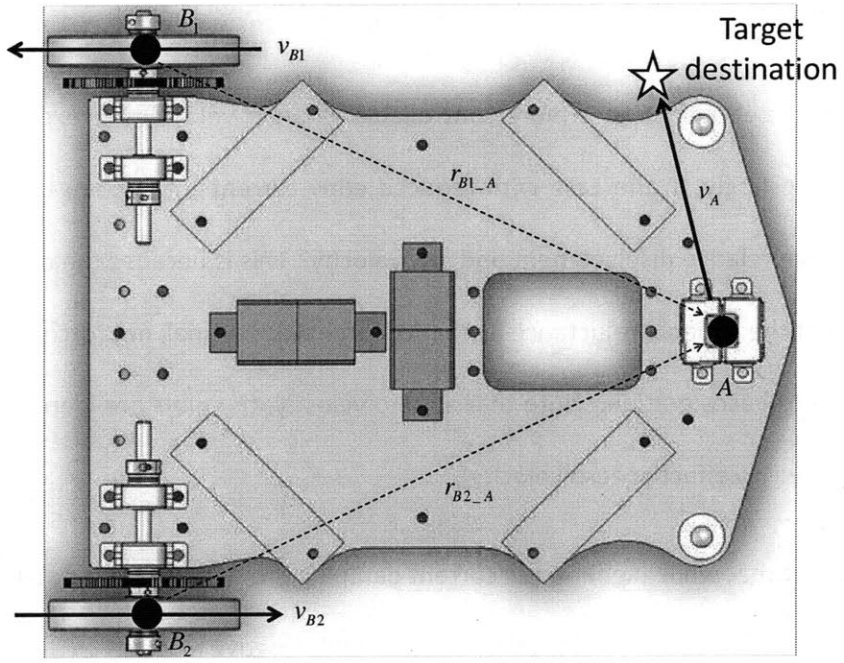
The case with reduced eddy friction is more controllable at larger velocities as expected and clearly has a larger area. Interestingly, in the case with reduced eddy currents, it becomes harder to operate at some regions of large relative displacement and low velocity. This is because the damping effects of the eddy currents offset the variable reluctance attractive force; with a small net force from the system, it is easier for us to control acceleration. Note that at zero velocity, the plots are identical (changes in eddy current damping have no effect at zero velocity).

These results suggest that while higher eddy current damping provides increased stability in a few cases, in general control of the system is improved by limiting its dissipative effects.

IV.2. Fine positioning

IV.2.1. Control strategy

Here we describe our strategy for achieving precise positioning. First, the outer robot uses its powered wheels to precisely align itself with the fastener. This is achieved using the straightforward method of solving the inverse kinematics of a wheeled mobile robot. If the tool is located at position A as shown in Fig. IV-10, and the desired location is at some other location, we know the direction of velocity we would like the tool to move. We apply to the system velocity in this direction with a unit magnitude. In order to apply this velocity to the tool, the wheels must move at some velocity.



IV - 10: Relative velocities for inverse kinematics

Kinematics tell us

$$v_A = v_{B1} + \omega \times r_{B1\_A}$$
$$v_A = v_{B2} + \omega \times r_{B2\_A} \quad \text{(IV-7)}$$

And we know the angular velocity

$$\omega = \left(v_{B2} - v_{B1}\right) 2m_6 \quad \text{(IV-8)}$$

From this we can find necessary wheel velocities to get our desired tool velocity.

$$v_A = v_{B1} + \omega \times r_{B1\_A}$$
$$v_A = v_{B2} + \omega \times r_{B2\_A} \quad \text{(IV-9)}$$

Once the outer robot has been accurately positioned, the wheels of the outer robot are locked. While the wheels are locked, current is applied to the coils on the outer robot. This generates a Lorentz force on the inner robot. We control the position of the inner robot through modulation of this force.

This Lorentz force strategy for positioning is important because though the outer robot is able to accurately align with the fastener, factors such as static friction, manufacturing tolerances and skin thickness aberrations may cause the inner robot to remain misaligned.

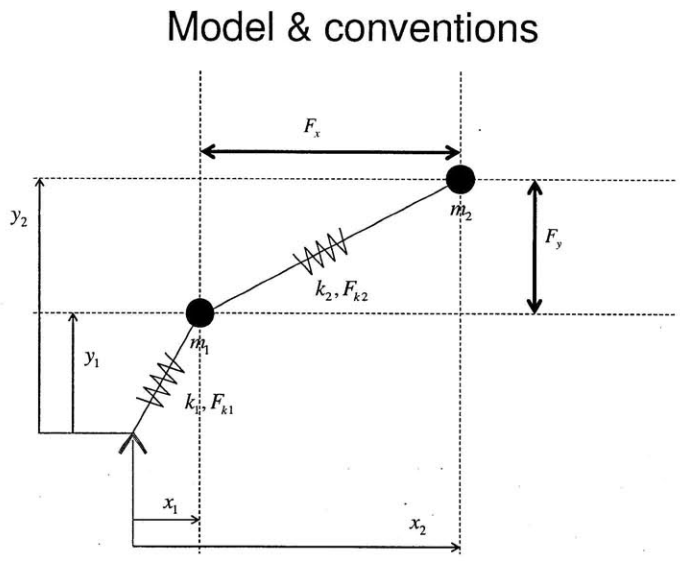
IV.2.2. Key issues

While fine positioning, the difference in position between the inner and outer robot remains small (< 5 mm). For this reason we assume normal magnetic forces are adequate to prevent falling or slipping of the outer robot. This small stroke also allows us to make other assumptions, such as linearizing the variable reluctance force between the robot pair, and assuming that Lorentz force is not state dependent. Our goal during this positioning is to eliminate steady state error, so our controller should contain an integrator.

IV.2.3. Simplified modeling and control for fine positioning

Once the outer robot has aligned to its desired location, the wheels on the outer robot are locked, and alignment of the inner robot is attempted. We attempt in this stage to hold the position of the outer robot stead. However, due to compliance in bearings and flexible elements such as rubber wheels, the outer robot may shift even while the wheels do not move. For this reason, the outer robot is modeled as a mass-spring system; the far end of the spring is ground fixed. The permanent magnets on the inner and outer robots create a restoring force between the pair. Linearizing this restoring force allows us to model it as a spring connecting the inner and outer robot. This linearization was introduced in eq. III-34. We simplify this linearization further by ignoring dynamics in the θ direction (the tool is rotationally symmetric – only xy positioning is relevant).

Fig. IV-11 shows the model of the robotic system while fine positioning is attempted.



IV - 11: Fine positioning model

Note that this model is of two uncoupled identical mass spring damper systems (in the x and the y direction). The coils in the outer robot produce forces F_x and F_y. Our goal is to step the inner robot to some location (x_2, y_2) while keeping the inner robot at the origin. Deriving equations of motion for this system is straightforward:

$$\begin{aligned} m_1\ddot{x}_1 &= -F_x + (-x_1)k_1 + (x_2 - x_1)k_2 - b_1\dot{x}_1 \\ m_2\ddot{x}_2 &= F_x + (x_1 - x_2)k_2 - b_2\dot{x}_2 \\ m_1\ddot{y}_1 &= -F_y + (-y_1)k_1 + (y_2 - y_1)k_2 - b_1\dot{y}_1 \\ m_2\ddot{y}_2 &= F_y + (y_1 - y_2)k_2 - b_2\dot{y}_2 \end{aligned} \qquad \text{(IV-10)}$$

Which can be written as:

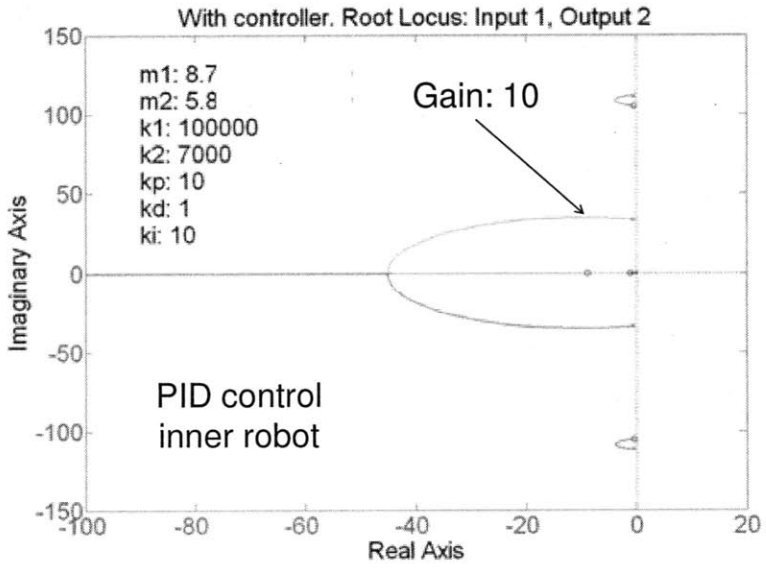
$$\dot{x} = Ax + Bu \qquad \text{(IV-11)}$$

From this, the controllability matrix is defined as:

$$H_c = \begin{bmatrix} B & AB & A^2B & A^3B & A^4B & A^5B & A^6B & A^7B \end{bmatrix} \quad \text{(IV-12)}$$

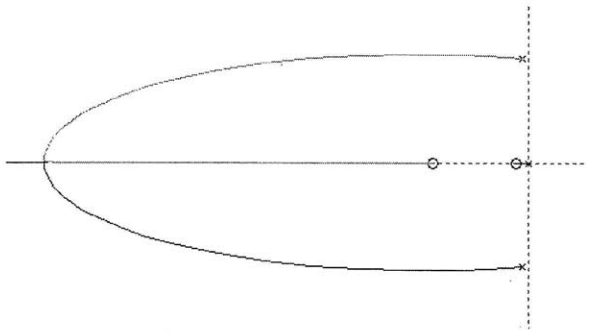
This matrix is full rank – the system is controllable.

We are interested in controlling the position of the inner robot, so we consider the transfer function from input F to output x_2. The following figure shows the root locus obtained when implementing PID control on the inner robot. Note that the system is stable for all gains. For our initial tests, we operated near the region marked on the chart. Eddy current damping values were taken from experimental results from section III.1.3.



IV - 12: Root locus for inner robot positioning

Fig. IV-13 shows a closer view near the center of the plot



IV - 13: Close view of root locus

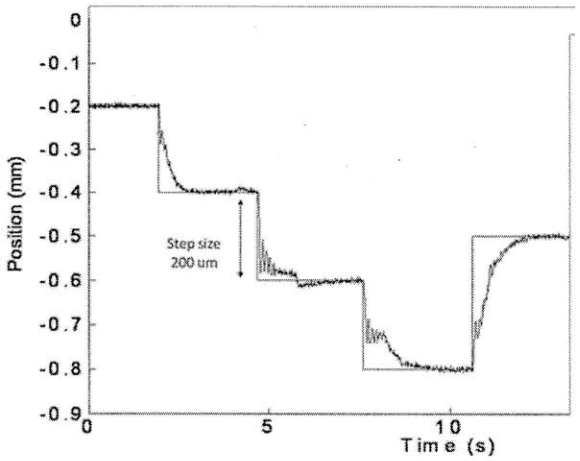
It should be noted that the location of the slow zero near the origin is dependent on the parameters of the system. In fact, this zero moves into the right half plane if the following expression holds:

$$\frac{m_1}{m_2} > \frac{k_1 + k_2}{k_2} \tag{IV-13}$$

Fortunately, plugging in typical fastener installation tooling values to this expression gives

$$10\frac{m_1}{m_2} \approx \frac{k_1 + k_2}{k_2} \tag{IV-14}$$

A PID controller was implemented on the actual system as shown below.



IV - 14: Servoing ability of inner robot

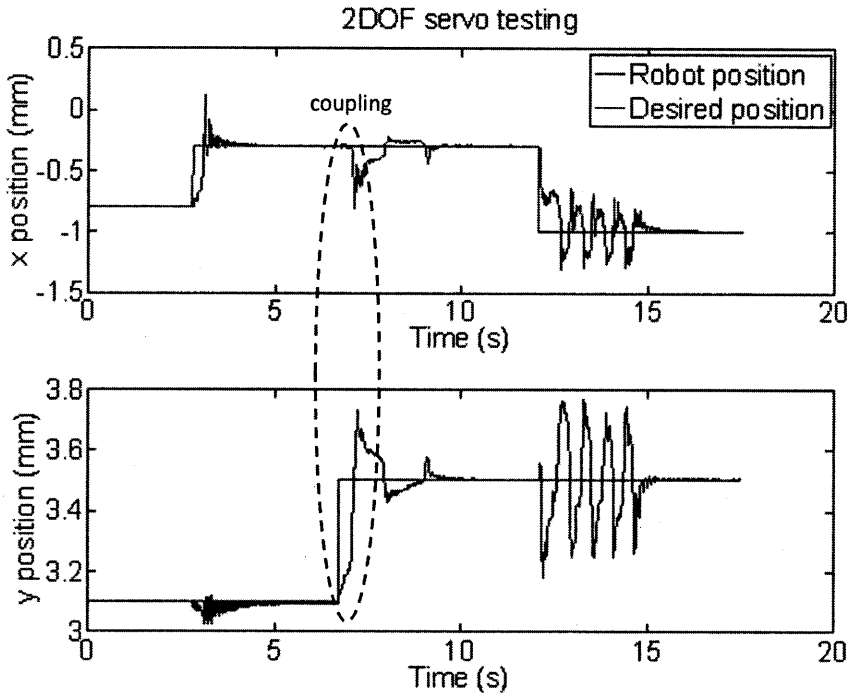
The plot shows a robot following a commanded ladder type reference.

For fastener installation, the tooling is required to have 100 μm positioning accuracy. This test demonstrates this goal is easily achievable. At times oscillations were present. These are believed to be due to unmodeled dynamics such as mechanical backlash in bearings. In practice, turning the controller off and then on again often stopped the oscillation. This suggests that a heuristic outer control loop could be added to the system to temporarily halt the PID loop in the case that unwanted oscillations are sensed.

IV.2.4. Further discussion

During some fine positioning attempts, motion of the tool was almost imperceptible. This suggests that the impressive accuracy present may be due to the fact that at least some portion of the positioning ability of the system is due to compliance in the mechanical elements themselves (stiffness in bearings, for example). Alignment in this fashion is acceptable for the task.

Although the system is modeled as two uncoupled systems, coupling was detected between the x and y axis, as shown in Fig. IV-15.



IV - 15: Coupling between axes, inner robot positioning

Despite this, 2 DOF servo positioning was successful as shown. This shows an instance of the aforementioned instability, which is fairly straightforward to deal with as described in section IV.2.3.

V. Conclusion

V.1. Summary of contributions

V.1.1. Applicability

This work was motivated by a real and immediate problem – automating fastener installation in aircraft assembly. This currently manually performed process is a significant bottleneck in aircraft production, raising the price of airplanes. Automation is a tool that has been used successfully by the automotive industry (among others) to dramatically lower prices. The work described here allows manufacturers to automate fastener installation, hopefully providing some of the same benefits to the aircraft industry that car makers have enjoyed.

Many aircraft makers are interested in changing their manufacturing techniques, building factories that are composed of mobile robotic systems rather than large heavy ground fixtures. The work described here fits within this anticipated factory framework.

V.1.2. Novel design

The authors are unaware of other robotic systems featuring the novel design described here – the use of a pair of robots operating across a thin wall, interacting with one another *through* the wall. Allowing for one side of the robot pair to remain passive allows for work in hazardous or hard to access areas. The support-less nature of the system reduces accessory equipment needed, as well as allows for effectively infinite stroke. Additionally, this self-supporting nature is possible even for heavy duty tooling. The system offers a simple way to achieve both fast, large scale positioning and precise alignment when required. It does all this while fitting into the 'future factory' paradigm many aircraft manufacturers predict.

V.1.3. Prototype

A functional prototype was built. Most components were found off-the-shelf. Minimal and standard machining was required, showing the cost effectiveness of utilizing such a robotic system. The prototype served as an excellent proof-of-concept for a dual robotic self supported partially passive system capable of performing gross locomotion and fine positioning.

V.1.4. General model

A general model for this system was developed. Parameters such as tooling weight, number of magnets, locations of wheels, etc, can be altered and the same analysis performed to predict the behavior of such a system for any application. This is enormously helpful for designers interested in using a similar architecture at a different scale. Key metrics for system failure were discussed. An FEA solution to find magnet forces for almost arbitrary placement of rectangular permanent magnets was presented.

V.1.5. Fine positioning

A control strategy was developed for fine positioning of the robotic system. Based on the features of this control strategy, a simplified system model capturing relevant dynamics was built. The control strategy in question was tested in simulation on the system model to verify its effectiveness. Finally, this controller was successfully implemented on the prototype.

V.1.6. Gross positioning

As in fine positioning, a control strategy was developed for gross positioning of the system. From this control strategy, a tractable system model containing relevant dynamics was created. Numerical

optimization was performed in the space of this control strategy to find time optimal trajectories that yielded physical insight into the system dynamics.

V.2. Future work

V.2.1. Hall Effect safety sensors

In practice, the system will run autonomously and should be able to monitor itself to prevent unusual failure. One of the worst types of failure is separation of the inner and outer robot caused by massive misalignment. This could happen if the outer robot moves to quickly or the inner robot gets stuck on an unexpected obstacle, for example. An array of Hall Effect sensors could be placed on the outer robot to detect the magnetic fields generated by the inner robot. In the case that the magnetic fields are drastically different than expected, the system could shut down before catastrophic misalignment. Development and calibration of such a sensor field would be enormously useful. Some work has been done in this direction and is presented in APPENDIX.

V.2.2. Obstacle avoidance

The interior of the wingbox is littered with obstacles. Designing an add-on to make the inner robot able to circumvent these obstacles would be useful in practice. Some work in this direction is currently underway.

A. MATLAB modeling

This section shows the MATLAB code used in modeling the system, first in using the full system model, and second in using the simplified model.

Full model

The following function gives state evolution of the full system model.

```
function[stateDot] = robotDynamics(t,state)
% Dynamics of the dual robot system
% Manas Menon
% 3/29/2010

% state = [x_oDot y_oDot x_iDot y_iDot theta_iDot xd yd thetad]';

x_oDot = state(1);
y_oDot = state(2);
x_iDot = state(3);
y_iDot = state(4);
theta_iDot = state(5);
xd = state(6);
yd = state(7);
thetad = state(8);

% load constants and assign stuff
load allInitializedData

% find static magnetic forces and torques, assign stuff
staticMagnetForces = getForcesRobotPair(state);
smf = staticMagnetForces;
tau_o1 = smf.tau_o1;
tau_o2 = smf.tau_o2;
tau_o3 = smf.tau_o3;
tau_o4 = smf.tau_o4;
tau_i1 = smf.tau_i1;
tau_i2 = smf.tau_i2;
tau_i3 = smf.tau_i3;
tau_i4 = smf.tau_i4;

% find magnetic forces due to eddy currents
eddyForcesOuter = getEddyForcesOuter(state,b);
eddyForcesInner = getEddyForcesInner(state,b);
efo = eddyForcesOuter;
efi = eddyForcesInner;

M_o1 = smf.M_o1 + efo.M_o1;
M_o2 = smf.M_o2 + efo.M_o2;
M_o3 = smf.M_o3 + efo.M_o3;
M_o4 = smf.M_o4 + efo.M_o4;

M_i1 = smf.M_i1 + efi.M_i1;
M_i2 = smf.M_i2 + efi.M_i2;
M_i3 = smf.M_i3 + efi.M_i3;
M_i4 = smf.M_i4 + efi.M_i4; % remeber these are in INNER ROBOT FRAME

% find reaction forces and assign stuff
% outer robot
% inputs

tau_motor2 = .01;
tau_motor3 = .1;
inputs = [tau_motor2 tau_motor3]; % here we have a control policy
reactionForcesOuter = getReactionForcesOuter([x_oDot y_oDot],inputs,...
    staticMagnetForces);
```

```
R_o1 = reactionForcesOuter.R_o1;
R_o2 = reactionForcesOuter.R_o2;
R_o3 = reactionForcesOuter.R_o3;
R_o4 = reactionForcesOuter.R_o4;
if abs(R_o3(2)) > abs((R_o3(3)+R_o2(3)))*c_r
    disp('slip')
stateDot = 'slip';
end

% inner robot
reactionForcesInner = getReactionForcesInner([x_iDot y_iDot theta_iDot],...
    staticMagnetForces);

R_i1 = reactionForcesInner.R_i1;
R_i2 = reactionForcesInner.R_i2;
R_i3 = reactionForcesInner.R_i3;
R_i4 = reactionForcesInner.R_i4;

% now sum the forces

magnetMomentsOuter = cross(r_om1,M_o1) + cross(r_om2,M_o2) + ...
    cross(r_om3,M_o3) + cross(r_om4,M_o4);
magnetMomentsInner = cross(r_im1,M_i1) + cross(r_im2,M_i2) + ...
    cross(r_im3,M_i3) + cross(r_im4,M_i4);

magnetTorquesOuter = subs(tau_o1 + tau_o2 + tau_o3 + tau_o4);
magnetTorquesInner = subs(tau_i1 + tau_i2 + tau_i3 + tau_i4);

reactionMomentsOuter = subs(cross(r_ow1,R_o1) + cross(r_ow2,R_o2) ...
    + cross(r_ow3,R_o3) + cross(r_ow4,R_o4)); % force due to motor included
reactionMomentsInner = subs(cross(r_iw1,R_i1) + cross(r_iw2,R_i2) ...
    + cross(r_iw3,R_i3) + cross(r_iw4,R_i4));

eddyMomentsOuter = [0 0 0];
eddyMomentsInner = [0 0 0];

totalMomentsOuter = magnetMomentsOuter + magnetTorquesOuter + ...
    reactionMomentsOuter + eddyMomentsOuter + subs(motorTorques);
totalMomentsInner = magnetMomentsInner + magnetTorquesInner + ...
    reactionMomentsInner + eddyMomentsInner;

% theta_oDotDot = (1/I_oC)*totalMomentsOuter(3);
theta_iDotDot = (1/I_iC)*totalMomentsInner(3);

magnetForcesOuterx = M_o1(1) + M_o2(1) + M_o3(1) + M_o4(1) + R_o1(1) + ...
    R_o2(1) + R_o3(1) + R_o4(1);
magnetForcesOutery = M_o1(2) + M_o2(2) + M_o3(2) + M_o4(2) + R_o1(2) + ...
    R_o2(2) + R_o3(2) + R_o4(2);
magnetForcesInnerx = M_i1(1) + M_i2(1) + M_i3(1) + M_i4(1) + R_i1(1) + ...
    R_i2(1) + R_i3(1) + R_i4(1);
magnetForcesInnery = M_i1(2) + M_i2(2) + M_i3(2) + M_i4(2) + R_i1(2) + ...
    R_i2(2) + R_i3(2) + R_i4(2);

% now do linear forces
x_oDotDot = magnetForcesOuterx/m_o;
y_oDotDot = magnetForcesOutery/m_o;

x_iDotDot = magnetForcesInnerx/m_i;
y_iDotDot = magnetForcesInnery/m_i;

xd_Dot = x_iDot*cos(thetad) - y_iDot*sin(thetad) - x_oDot;
yd_Dot = y_iDot*cos(thetad) + x_iDot*sin(thetad) - y_oDot;
% need to find theta_oDot;
theta_oDot = subs([0 0 y_oDot/-m_6]);
thetadDot = theta_iDot - theta_oDot;

% now construct the vector of stateDot
stateDot(1) = x_oDotDot;
stateDot(2) = y_oDotDot;
```

```
stateDot(3) = x_iDotDot;
stateDot(4) = y_iDotDot;
stateDot(5) = theta_iDotDot;
stateDot(6) = xd_Dot;
stateDot(7) = yd_Dot;
stateDot(8) = thetadDot(3);
stateDot = stateDot';
```

This program used the following functions:

- getForcesRobotPair.m
- getEddyForcesOuter.m
- getEddyForcesInner.m
- getReactionForcesOuter.m
- getReactionForcesInner.m

Code for these functions is shown below

```
function[forces] = getForcesRobotPair(state)
% this function gets the forces on a pair of robots due to the halbach
% array static forces.  it also gets torques

% manas menon, 3/29/2010

% state = [x_oDot y_oDot x_iDot y_iDot theta_iDot xd yd thetad]';

load allInitializedData

xd = state(6);
yd = state(7);
thetad = state(8);

zd = (m_8 - m_7 + skinThickness + n_8 - n_7); % z distance between magnets
% find position of inner magnets
R = [cos(thetad) -sin(thetad);sin(thetad) cos(thetad)];

c_o1 = [r_om1(1:2) 0]; % location of outer magnet 1
c_o2 = [r_om2(1:2) 0];
c_o3 = [r_om3(1:2) 0];
c_o4 = [r_om4(1:2) 0]; % location of outer magnet 4

c_i1 = [xd yd zd] + [R*r_im1(1:2)'; 0]'; % xd and yd come from 'state'
c_i2 = [xd yd zd] + [R*r_im2(1:2)'; 0]';
c_i3 = [xd yd zd] + [R*r_im3(1:2)'; 0]';
c_i4 = [xd yd zd] + [R*r_im4(1:2)'; 0]'; % location of inner magnet 4

theta_o1 = atan2(m_2,m_1) - pi/4;
theta_o2 = atan2(m_2,-m_1) - pi/4;
theta_o3 = atan2(-m_2,-m_1) + pi/4;
theta_o4 = atan2(-m_1,m_1) + pi/4;

theta_i1 = theta_o1 + thetad;
theta_i2 = theta_o2 + thetad;
theta_i3 = theta_o3 + thetad;
theta_i4 = theta_o4 + thetad;

% we calculate force on upper (inner) robot, so we take the negative of
```

```
% this force to find the force on the outer (lower) robot
% Magnet 1
[Fx Fy Fz Tx Ty Tz] = getApproximateForcesHalbachs(c_o1, 1, theta_o1,...
    c_i1, -1, theta_i1,J);
M_o1 = -[Fx Fy Fz];
tau_o1 = -[Tx Ty Tz];
% Magnet 2
[Fx Fy Fz Tx Ty Tz] = getApproximateForcesHalbachs(c_o2, 1, theta_o2,...
    c_i2, -1, theta_i2,J);
M_o2 = -[Fx Fy Fz];
tau_o2 = -[Tx Ty Tz];
% Magnet 3
[Fx Fy Fz Tx Ty Tz] = getApproximateForcesHalbachs(c_o3, 1, theta_o3,...
    c_i3, -1, theta_i3,J);
M_o3 = -[Fx Fy Fz];
tau_o3 = -[Tx Ty Tz];
% Magnet 4
[Fx Fy Fz Tx Ty Tz] = getApproximateForcesHalbachs(c_o4, 1, theta_o4,...
    c_i4, -1, theta_i4,J);
M_o4 = -[Fx Fy Fz];
tau_o4 = -[Tx Ty Tz];

R = [cos(thetad) sin(thetad) 0;
    -sin(thetad) cos(thetad) 0;
    0 0 1];

% now find inner robot forces - remember these are in the inner robot frame
M_i1 = -R*M_o1';
tau_i1 = -R*tau_o1';
M_i2 = -R*M_o2';
tau_i2 = -R*tau_o2';
M_i3 = -R*M_o3';
tau_i3 = -R*tau_o3';
M_i4 = -R*M_o4';
tau_i4 = -R*tau_o4';

% build output
forces.M_i1 = M_i1';
forces.M_i2 = M_i2';
forces.M_i3 = M_i3';
forces.M_i4 = M_i4';

forces.M_o1 = M_o1;
forces.M_o2 = M_o2;
forces.M_o3 = M_o3;
forces.M_o4 = M_o4;

forces.tau_i1 = tau_i1';
forces.tau_i2 = tau_i2';
forces.tau_i3 = tau_i3';
forces.tau_i4 = tau_i4';

forces.tau_o1 = tau_o1;
forces.tau_o2 = tau_o2;
forces.tau_o3 = tau_o3;
forces.tau_o4 = tau_o4;
```

This program used the following functions:

- getApproximateForcesHalbachs.m

Code for this is shown below

```
function[Fx Fy Fz Tx Ty Tz] = getApproximateForcesHalbachs(...
    centroid1,thetaZ1,d1,...
    centroid2,thetaZ2,d2,J)
% this function gets force information for a pair of halbach arrays from a
% lookup table

load allMagnetData

xDisplacements = allMagnetData.xDisplacements;
yDisplacements = allMagnetData.yDisplacements;
zDisplacements = allMagnetData.zDisplacements;
thetaDisplacements = allMagnetData.thetaDisplacements;

xd = centroid2(1) - centroid1(1);
yd = centroid2(2) - centroid1(2);
zd = centroid2(3) - centroid1(3);
thetad = thetaZ2 - thetaZ1;

[xValue xIndex] = min(abs(xd - xDisplacements));
[yValue yIndex] = min(abs(yd - yDisplacements));
[zValue zIndex] = min(abs(zd - zDisplacements));
[thetaValue thetaIndex] = min(abs(thetad - thetaDisplacements));

vec = allMagnetData.dataMatrix{xIndex(1), yIndex(1),...
    zIndex(1),thetaIndex(1)};
Fx = vec(1);
Fy = vec(2);
Fz = vec(3);
Tx = vec(4);
Ty = vec(5);
Tz = vec(6);
```

```
function[eddyForcesOuter] = getEddyForcesOuter(state,b);

% for now:
% state = [x_oDot y_oDot x_iDot y_iDot theta_iDot xd yd thetad]';

em = [-b*state(1) -b*state(2)];
eddyForcesOuter.M_o1 = [em 0];
eddyForcesOuter.M_o2 = [em 0];
eddyForcesOuter.M_o3 = [em 0];
eddyForcesOuter.M_o4 = [em 0];
eddyForcesOuter.tau_o1 = [0 0 0];
eddyForcesOuter.tau_o2 = [0 0 0];
eddyForcesOuter.tau_o3 = [0 0 0];
eddyForcesOuter.tau_o4 = [0 0 0];
```

```
function[eddyForcesInner] = getEddyForcesInner(state,b)

% for now:
% state = [x_oDot y_oDot x_iDot y_iDot theta_iDot xd yd thetad]';

em = [-b*state(3) -b*state(4)];
eddyForcesInner.M_i1 = [em 0];
eddyForcesInner.M_i2 = [em 0];
eddyForcesInner.M_i3 = [em 0];
eddyForcesInner.M_i4 = [em 0];
eddyForcesInner.tau_i1 = [0 0 0];
eddyForcesInner.tau_i2 = [0 0 0];
eddyForcesInner.tau_i3 = [0 0 0];
eddyForcesInner.tau_i4 = [0 0 0];
```

```
function[reactionForcesOuter] = getReactionForcesOuter(v_outerC,inputs,...
    staticMagnetForces)

% solves a system of equations to get reaction forces for the outer robot.
% I think I'll write a script to find the solution to this for the given
% inputs and then just write it here...

% Manas Menon
% started 3/19/2010, but it'll be some time before I finish

% It's 3/24/2010.  Finding the reaction forces has been reduced to solving
% a series of linear equations.

% inputs of the following form:
% v_oC = [v_oCx v_oCy];
% inputs = [tau_motor2 tau_motor3]

load allInitializedData

% ASSIGN INPUTS

% velocities
v_oCx = v_outerC(1);
v_oCy = v_outerC(2);

% inputs
tau_motor2 = inputs(1);
tau_motor3 = inputs(2);
F_motor2 = tau_motor2/wheelRadius;
F_motor3 = tau_motor3/wheelRadius;

% magnet forces
M_o1 = staticMagnetForces.M_o1;
M_o2 = staticMagnetForces.M_o2;
M_o3 = staticMagnetForces.M_o3;
M_o4 = staticMagnetForces.M_o4;
M_o1x = M_o1(1); %
M_o1y = M_o1(2);
M_o1z = M_o1(3);

M_o2x = M_o2(1); %
M_o2y = M_o2(2);
M_o2z = M_o2(3);

M_o3x = M_o3(1); %
M_o3y = M_o3(2);
M_o3z = M_o3(3);

M_o4x = M_o4(1); %
M_o4y = M_o4(2);
M_o4z = M_o4(3);

% magnet torques
tau_o1 = staticMagnetForces.tau_o1;
tau_o2 = staticMagnetForces.tau_o2;
tau_o3 = staticMagnetForces.tau_o3;
tau_o4 = staticMagnetForces.tau_o4;
tau_o1x = tau_o1(1); %
tau_o1y = tau_o1(2);
tau_o1z = tau_o1(3);

tau_o2x = tau_o2(1); %
tau_o2y = tau_o2(2);
tau_o2z = tau_o2(3);

tau_o3x = tau_o3(1); %
tau_o3y = tau_o3(2);
tau_o3z = tau_o3(3);
```

```
tau_o4x = tau_o4(1); %
tau_o4y = tau_o4(2);
tau_o4z = tau_o4(3);

% we have derived A from another program, plug it in here:
% need to check if the velocity of the passive wheels is 0, and make a few
% changes if it is:
if norm(subs(v_outerC)) <.001 % consider the case where the robot is ~still
   A_outer = A_outer(1:5,1:5);
   b_outer = b_outer(1:5);
   sA_outer = subs(A_outer);
   sb_outer = subs(b_outer);
   rvec = inv(sA_outer)*sb_outer;
   reactionForcesOuter.R_o1x = 0;
   reactionForcesOuter.R_o1y = 0;
   reactionForcesOuter.R_o4x = 0;
   reactionForcesOuter.R_o4y = 0;
else % if the robot is moving (more common case)
   sA_outer = subs(A_outer);
   sb_outer = subs(b_outer);
   rvec = inv(sA_outer)*sb_outer;
   reactionForcesOuter.R_o1x = rvec(6);
   reactionForcesOuter.R_o1y = rvec(7);
   reactionForcesOuter.R_o4x = rvec(8);
   reactionForcesOuter.R_o4y = rvec(9);
end

reactionForcesOuter.R_o1z = rvec(1);
reactionForcesOuter.R_o2z = rvec(2);
reactionForcesOuter.R_o3z = rvec(3);
reactionForcesOuter.R_o4z = rvec(4);
reactionForcesOuter.R_o3y = rvec(5);

reactionForcesOuter.R_o1 = [reactionForcesOuter.R_o1x ...
    reactionForcesOuter.R_o1y reactionForcesOuter.R_o1z];
reactionForcesOuter.R_o2 = [F_motor2 ...
    0 reactionForcesOuter.R_o2z];
reactionForcesOuter.R_o3 = [F_motor3 ...
    reactionForcesOuter.R_o3y reactionForcesOuter.R_o3z];
reactionForcesOuter.R_o4 = [reactionForcesOuter.R_o4x ...
    reactionForcesOuter.R_o4y reactionForcesOuter.R_o4z];

-----------------------------------------------------------------------------------------------

function[reactionForcesInner] = getReactionForcesInner(v_innerC,...
    staticMagnetForces)

% Manas Menon
% 3/29/2010

% this function solves some linear equations to find the reaction forces on
% the inner robot, in the frame of the inner robot

% v_innerC = (v_iCx v_iCy v_iCtheta) % NOTE V_THETA IS INCLUDED

load allInitializedData

% velocities
v_iCx = v_innerC(1);
v_iCy = v_innerC(2);
v_iCtheta = v_innerC(3);

% magnet forces
M_i1 = staticMagnetForces.M_i1;
M_i2 = staticMagnetForces.M_i2;
M_i3 = staticMagnetForces.M_i3;
M_i4 = staticMagnetForces.M_i4;
M_i1x = M_i1(1); %
```

```
M_i1y = M_i1(2);
M_i1z = M_i1(3);
M_i2x = M_i2(1); %
M_i2y = M_i2(2);
M_i2z = M_i2(3);
M_i3x = M_i3(1); %
M_i3y = M_i3(2);
M_i3z = M_i3(3);
M_i4x = M_i4(1); %
M_i4y = M_i4(2);
M_i4z = M_i4(3);

% magnet torques
tau_i1 = staticMagnetForces.tau_i1;
tau_i2 = staticMagnetForces.tau_i2;
tau_i3 = staticMagnetForces.tau_i3;
tau_i4 = staticMagnetForces.tau_i4;
tau_i1x = tau_i1(1); %
tau_i1y = tau_i1(2);
tau_i1z = tau_i1(3);
tau_i2x = tau_i2(1); %
tau_i2y = tau_i2(2);
tau_i2z = tau_i2(3);
tau_i3x = tau_i3(1); %
tau_i3y = tau_i3(2);
tau_i3z = tau_i3(3);
tau_i4x = tau_i4(1); %
tau_i4y = tau_i4(2);
tau_i4z = tau_i4(3);

if norm(subs(v_innerC))<.001;
    reactionForcesInner.R_i1x = 0;
    reactionForcesInner.R_i1y = 0;
    reactionForcesInner.R_i2x = 0;
    reactionForcesInner.R_i2y = 0;
    reactionForcesInner.R_i3x = 0;
    reactionForcesInner.R_i3y = 0;
    reactionForcesInner.R_i4x = 0;
    reactionForcesInner.R_i4y = 0;
    sA_inner = subs(A_inner(1:4,1:4));
    sb_inner = subs(b_inner(1:4));
    rvec = inv(sA_inner)*sb_inner;
elseif norm(subs(v_iw1)) <.001
    reactionForcesInner.R_i1x = 0;
    reactionForcesInner.R_i1y = 0;
    sA_inner = subs(A_inner([1:4,7:12],[1:4,7:12]));
    sb_inner = subs(b_inner([1:4,7:12]));
    rvec = inv(sA_inner)*sb_inner;
    reactionForcesInner.R_i2x = rvec(5);
    reactionForcesInner.R_i2y = rvec(6);
    reactionForcesInner.R_i3x = rvec(7);
    reactionForcesInner.R_i3y = rvec(8);
    reactionForcesInner.R_i4x = rvec(9);
    reactionForcesInner.R_i4y = rvec(10);
elseif norm(subs(v_iw2)) <.001;
    reactionForcesInner.R_i2x = 0;
    reactionForcesInner.R_i2y = 0;
    sA_inner = subs(A_inner([1:6,9:12],[1:6,9:12]));
    sb_inner = subs(b_inner([1:6,9:12]));
    rvec = inv(sA_inner)*sb_inner;
    reactionForcesInner.R_i1x = rvec(5);
    reactionForcesInner.R_i1y = rvec(6);
    reactionForcesInner.R_i3x = rvec(7);
    reactionForcesInner.R_i3y = rvec(8);
    reactionForcesInner.R_i4x = rvec(9);
    reactionForcesInner.R_i4y = rvec(10);
elseif norm(subs(v_iw3)) < .001
    reactionForcesInner.R_i3x = 0;
    reactionForcesInner.R_i3y = 0;
```

```
    sA_inner = subs(A_inner([1:8,11:12],[1:8,11:12]));
    sb_inner = subs(b_inner([1:8,11:12]));
    rvec = inv(sA_inner)*sb_inner;
    reactionForcesInner.R_i1x = rvec(5);
    reactionForcesInner.R_i1y = rvec(6);
    reactionForcesInner.R_i2x = rvec(7);
    reactionForcesInner.R_i2y = rvec(8);
    reactionForcesInner.R_i4x = rvec(9);
    reactionForcesInner.R_i4y = rvec(10);
elseif norm(subs(v_iw4)) < .001
    reactionForcesInner.R_i4x = 0;
    reactionForcesInner.R_i4y = 0;
    sA_inner = subs(A_inner(1:10,1:10));
    sb_inner = subs(b_inner(1:10));
    rvec = inv(sA_inner)*sb_inner;
    reactionForcesInner.R_i1x = rvec(5);
    reactionForcesInner.R_i1y = rvec(6);
    reactionForcesInner.R_i2x = rvec(7);
    reactionForcesInner.R_i2y = rvec(8);
    reactionForcesInner.R_i3x = rvec(9);
    reactionForcesInner.R_i3y = rvec(10);
else
    sA_inner = subs(A_inner);
    sb_inner = subs(b_inner);
    rvec = inv(sA_inner)*sb_inner;
    reactionForcesInner.R_i1x = rvec(5);
    reactionForcesInner.R_i1y = rvec(6);
    reactionForcesInner.R_i2x = rvec(7);
    reactionForcesInner.R_i2y = rvec(8);
    reactionForcesInner.R_i3x = rvec(9);
    reactionForcesInner.R_i3y = rvec(10);
    reactionForcesInner.R_i4x = rvec(11);
    reactionForcesInner.R_i4y = rvec(12);
end

reactionForcesInner.R_i1z = rvec(1);
reactionForcesInner.R_i2z = rvec(2);
reactionForcesInner.R_i3z = rvec(3);
reactionForcesInner.R_i4z = rvec(4);

reactionForcesInner.R_i1 = [reactionForcesInner.R_i1x ...
    reactionForcesInner.R_i1y reactionForcesInner.R_i1z];
reactionForcesInner.R_i2 = [reactionForcesInner.R_i2x ...
    reactionForcesInner.R_i2y reactionForcesInner.R_i2z];
reactionForcesInner.R_i3 = [reactionForcesInner.R_i3x ...
    reactionForcesInner.R_i3y reactionForcesInner.R_i3z];
reactionForcesInner.R_i4 = [reactionForcesInner.R_i4x ...
    reactionForcesInner.R_i4y reactionForcesInner.R_i4z];
```

Simplified model

Next we show code used in the point to point model of the system. This code assumes the robot moves along a straight line, has fewer states and is much less complicated. It is also worth noting that the structure of this code, as well as many file names, have been altered for simplicity in running DIDO (the numerical trajectory optimization program).

```
function[XDOT] = dynamics_fun(primal)
% Dynamics of the dual robot system
% Manas Menon
% 4/13/2010

% this version of the program written for numerical optimization by DIDO

% state = [x_oDot x_iDot xd x_o]';
```

```
% load constants and assign stuff
load allInitializedData

% preallocate
XDOT = zeros(size(primal.states));
for a = 1:size(XDOT,2)
    x_oDot = primal.states(1,a);
x_iDot = primal.states(2,a);
xd = primal.states(3,a);
x_o = primal.states(4,a);

state = [x_oDot x_iDot xd x_o];

    % look at the state vector at each time (node)
% find static magnetic forces and torques, assign stuff
% should be in a form that matches 'primal'
% for each 'node' in primal.nodes (which corresponds to a time), there
% should be a value for staticMagnetForcesArray.  So the first entry is
staticMagnetForces = getApproximateForcesRobotPair(state,...
    p_x,p_z,p_tau);
smf = staticMagnetForces;

% find magnetic forces due to eddy currents
eddyForcesOuter =  getEddyForcesOuter(state,b);
eddyForcesInner =  getEddyForcesInner(state,b);
efo = eddyForcesOuter;
efi = eddyForcesInner;

M_o1 = smf.M_o1 + efo.M_o1;
M_o2 = smf.M_o2 + efo.M_o2;
M_o3 = smf.M_o3 + efo.M_o3;
M_o4 = smf.M_o4 + efo.M_o4;
totalMagnetForcesOuter.M_o1 = M_o1;
totalMagnetForcesOuter.M_o2 = M_o2;
totalMagnetForcesOuter.M_o3 = M_o3;
totalMagnetForcesOuter.M_o4 = M_o4;
totalMagnetForcesOuter.tau_o1 = smf.tau_o1;
totalMagnetForcesOuter.tau_o2 = smf.tau_o2;
totalMagnetForcesOuter.tau_o3 = smf.tau_o3;
totalMagnetForcesOuter.tau_o4 = smf.tau_o4;

M_i1 = smf.M_i1 + efi.M_i1;
M_i2 = smf.M_i2 + efi.M_i2;
M_i3 = smf.M_i3 + efi.M_i3;
M_i4 = smf.M_i4 + efi.M_i4; % remember these are in INNER ROBOT FRAME
totalMagnetForcesInner.M_i1 = M_i1;
totalMagnetForcesInner.M_i2 = M_i2;
totalMagnetForcesInner.M_i3 = M_i3;
totalMagnetForcesInner.M_i4 = M_i4;
totalMagnetForcesInner.tau_i1 = smf.tau_i1;
totalMagnetForcesInner.tau_i2 = smf.tau_i2;
totalMagnetForcesInner.tau_i3 = smf.tau_i3;
totalMagnetForcesInner.tau_i4 = smf.tau_i4;


% find reaction forces and assign stuff
% outer robot
% inputs
tau_motor2 = primal.controls(a)/2;
tau_motor3 = primal.controls(a)/2;
inputs = [tau_motor2 tau_motor3];
reactionForcesOuter = getReactionForcesOuter([x_oDot 0],inputs,...
    totalMagnetForcesOuter);
R_o1 = reactionForcesOuter.R_o1;
R_o2 = reactionForcesOuter.R_o2;
R_o3 = reactionForcesOuter.R_o3;
R_o4 = reactionForcesOuter.R_o4;
%
% % inner robot
```

```
reactionForcesInner = getReactionForcesInner([x_iDot 0 0],...
    totalMagnetForcesInner);
R_i1 = reactionForcesInner.R_i1;
R_i2 = reactionForcesInner.R_i2;
R_i3 = reactionForcesInner.R_i3;
R_i4 = reactionForcesInner.R_i4;

% now sum the forces
magnetForcesOuterx = M_o1(1) + M_o2(1) + M_o3(1) + M_o4(1);
magnetForcesInnerx = M_i1(1) + M_i2(1) + M_i3(1) + M_i4(1);
reactionForcesOuterx = R_o1(1) + R_o2(1) + R_o3(1) + R_o4(1);
reactionForcesInnerx = R_i1(1) + R_i2(1) + R_i3(1) + R_i4(1);
totalForcesOuterx = magnetForcesOuterx + reactionForcesOuterx;

totalForcesInnerx = magnetForcesInnerx + reactionForcesInnerx;

% now do linear forces
x_oDotDot = totalForcesOuterx/m_o;
x_iDotDot = totalForcesInnerx/m_i;
xd_Dot = x_iDot - x_oDot;

% reminder:
% state = [x_oDot x_iDot xd x_o]';
XDOT(1,a) = x_oDotDot;
XDOT(2,a) = x_iDotDot;
XDOT(3,a) = xd_Dot;
XDOT(4,a) = x_oDot;
end
```

This function calls the following functions:

- getEddyForcesOuter.m
- getEddyForcesInner.m

This code is nearly identical to that shown in the previous section and is not shown here. Additionally, the function called:

- getApproximateForcesRobotPair.m
- getReactionForcesOuter.m
- getReactionForcesInner.m

This code is shown below.

```
function approximateForcesRobotPair = getApproximateForcesRobotPair(...
    state,p_x,p_z,p_tau)
% Manas Menon
% 4/7/2010
% This uses the 1 DOF robot model, tractability is sweet

% state = [x_oDot x_iDot xd x_o]';

xd = state(3);

xForce = polyval(p_x,xd);
zForce = polyval(p_z,xd);
yMoment = polyval(p_tau,xd); % these are forces on EACH OUTER ROBOT MAGNET
```

```
afrp.M_o1 = [xForce 0 zForce];
afrp.M_o2 = [xForce 0 zForce];
afrp.M_o3 = [xForce 0 zForce];
afrp.M_o4 = [xForce 0 zForce];
afrp.M_i1 = [-xForce 0 -zForce];
afrp.M_i2 = [-xForce 0 -zForce];
afrp.M_i3 = [-xForce 0 -zForce];
afrp.M_i4 = [-xForce 0 -zForce];

afrp.tau_o1 = [0 yMoment 0];
afrp.tau_o2 = [0 yMoment 0];
afrp.tau_o3 = [0 yMoment 0];
afrp.tau_o4 = [0 yMoment 0];
afrp.tau_i1 = -[0 yMoment 0];
afrp.tau_i2 = -[0 yMoment 0];
afrp.tau_i3 = -[0 yMoment 0];
afrp.tau_i4 = -[0 yMoment 0];

approximateForcesRobotPair = afrp;
```

```
function[reactionForcesOuter] = getReactionForcesOuter(v_outerC,inputs,...
    totalMagnetForcesOuter)

% solves a system of equations to get reaction forces for the outer robot.
% I think I'll write a script to find the solution to this for the given
% inputs and then just write it here...

% Manas Menon
% started 3/19/2010, but it'll be some time before I finish

% It's 3/24/2010.  Finding the reaction forces has been reduced to solving
% a series of linear equations.

% inputs of the following form:
% v_oC = [v_oCx v_oCy];
% inputs = [tau_motor2 tau_motor3]


% ASSIGN INPUTS

% velocities
v_oCx = v_outerC(1);

% inputs
tau_motor2 = inputs(1);
tau_motor3 = inputs(2);

% magnet forces
M_o1 = totalMagnetForcesOuter.M_o1;
M_o2 = totalMagnetForcesOuter.M_o2;
M_o3 = totalMagnetForcesOuter.M_o3;
M_o4 = totalMagnetForcesOuter.M_o4;
M_o1x = M_o1(1); %
M_o1z = M_o1(3);
M_o2x = M_o2(1); %
M_o2z = M_o2(3);
M_o3x = M_o3(1); %
M_o3z = M_o3(3);
M_o4x = M_o4(1); %
M_o4z = M_o4(3);

% magnet torques
tau_o1y = totalMagnetForcesOuter.tau_o1(2);
tau_o2y = totalMagnetForcesOuter.tau_o2(2);
tau_o3y = totalMagnetForcesOuter.tau_o3(2);
tau_o4y = totalMagnetForcesOuter.tau_o4(2);
```

```
% we have derived A from another program, plug it in here:
% need to check if the velocity of the passive wheels is 0, and make a few
% changes if it is:

% NEED TO CHECK AND MAKE SURE THIS IS ACCURATE ON MY FINAL RUN
wheelRadius = 0.061976;
c = .025;
F_motor2 = tau_motor2/wheelRadius;
F_motor3 = tau_motor3/wheelRadius;
A_outer = [          -18059/50000,             1811/5000,          15247/62500;
2,2,0;
 1/40*v_oCx/abs(v_oCx),                    0,                   -1];
b_outer = ...
    ([ -2281/25000*M_o1x+107/1000*M_o1z-2281/25000*M_o2x-107/1000*...
    M_o2z-2281/25000*M_o3x-107/1000*M_o3z-2281/25000*M_o4x+107/1000*...
    M_o4z-tau_o1y-tau_o2y-tau_o3y-tau_o4y-3/50*F_motor2-3/50*F_motor3-...
    tau_motor2-tau_motor3;
    -M_o1z-M_o2z-M_o3z-M_o4z+4263/50;
    0]);
%%%% CHECK ABOVE THIS LINE TO MAKE SURE A, B AND WHEEL RADIUS AND C...
% % % ARE CORRECT TO FIND CORRECT VALUES, RUN initializeValues.m


smallV = .001;
if norm(v_outerC) <smallV % consider the case where the robot is ~still
   sA_outer = (A_outer(1:2,1:2));
   sb_outer = (b_outer(1:2));
   rvec = inv(sA_outer)*sb_outer;
   reactionForcesOuter.R_o1x = -abs(c*rvec(1))/smallV*norm(v_outerC);
else % if the robot is moving (more common case)
   sA_outer = (A_outer);
   sb_outer = (b_outer);
   rvec = inv(sA_outer)*sb_outer;
   reactionForcesOuter.R_o1x = rvec(3);
end

reactionForcesOuter.R_o1z = rvec(1);
reactionForcesOuter.R_o2z = rvec(2);

reactionForcesOuter.R_o1 = [reactionForcesOuter.R_o1x ...
    0 reactionForcesOuter.R_o1z];
reactionForcesOuter.R_o2 = [F_motor2 ...
    0 reactionForcesOuter.R_o2z];
reactionForcesOuter.R_o3 = reactionForcesOuter.R_o2;
reactionForcesOuter.R_o4 = reactionForcesOuter.R_o1;

-------------------------------------------------------------------------------------------------

function[reactionForcesInner] = getReactionForcesInner(v_innerC,...
    totalMagnetForcesInner)

% Manas Menon
% 3/29/2010

% this function solves some linear equations to find the reaction forces on
% the inner robot, in the frame of the inner robot

% v_innerC = (v_iCx v_iCy v_iCtheta) % NOTE V_THETA IS INCLUDED

% velocities
v_iCx = v_innerC(1);
v_iCy = 0;

% magnet forces
M_i1 = totalMagnetForcesInner.M_i1;
M_i2 = totalMagnetForcesInner.M_i2;
M_i3 = totalMagnetForcesInner.M_i3;
M_i4 = totalMagnetForcesInner.M_i4;
```

```
M_i1x = M_i1(1); %

M_i1z = M_i1(3);
M_i2x = M_i2(1); %

M_i2z = M_i2(3);
M_i3x = M_i3(1); %

M_i3z = M_i3(3);
M_i4x = M_i4(1); %

M_i4z = M_i4(3);

% magnet torques
tau_i1 = totalMagnetForcesInner.tau_i1;
tau_i2 = totalMagnetForcesInner.tau_i2;
tau_i3 = totalMagnetForcesInner.tau_i3;
tau_i4 = totalMagnetForcesInner.tau_i4;

tau_i1y = tau_i1(2);
tau_i2y = tau_i2(2);
tau_i3y = tau_i3(2);
tau_i4y = tau_i4(2);


% CHECK THIS BEFORE RUNNING FINAL VERSION
c = .025;
A_inner =...
[      -18059/50000,       18059/50000,-459/5000,       -459/5000;
          2,              2,                0,              0;
 -1/40*v_iCx/abs(v_iCx),          0,          -1,          0;
                        0, -1/40*v_iCx/abs(v_iCx),          0,     -1];
b_inner = ...
[1/50*M_i1x+107/1000*M_i1z+1/50*M_i2x-107/1000*M_i2z+1/50*...
M_i3x-107/1000*M_i3z+1/50*M_i4x+107/1000*M_i4z-tau_i1y-tau_i2y-...
tau_i3y-tau_i4y;
                -M_i1z-M_i2z-M_i3z-M_i4z+1421/25;
                                          0;
                                             0];
% CHECK ABOUT THIS BEFORE RUNNING FINAL VERSION
% NEED TO CHECK A_INNER, B_INNER, c
% unknownsVector = [R_i1z; R_i2z; R_i1x; R_i2x];


smallV = .001;
if norm(v_innerC)<smallV;
    sA_inner = (A_inner(1:2,1:2));
    sb_inner = (b_inner(1:2));
    rvec = inv(sA_inner)*sb_inner;
    reactionForcesInner.R_i1x = -abs(c*rvec(1))/.001*v_innerC;
    reactionForcesInner.R_i2x = -abs(c*rvec(2))/.001*v_innerC;
else
    sA_inner = (A_inner);
    sb_inner = (b_inner);
    rvec = inv(sA_inner)*sb_inner;
    reactionForcesInner.R_i1x = rvec(3);
    reactionForcesInner.R_i2x = rvec(4);
end

reactionForcesInner.R_i1z = rvec(1);
reactionForcesInner.R_i2z = rvec(2);

reactionForcesInner.R_i1 = [reactionForcesInner.R_i1x ...
    0 reactionForcesInner.R_i1z];
reactionForcesInner.R_i2 = [reactionForcesInner.R_i2x ...
    0 reactionForcesInner.R_i2z];
reactionForcesInner.R_i3 = reactionForcesInner.R_i2;
reactionForcesInner.R_i4 = reactionForcesInner.R_i1;
```

B. MATLAB initialization

Magnet Data

This section shows the MATLAB code used to generate magnet force data. This data is fit to a high order polynomial curve so that it does not have to be re-generated every time the dynamics simulation needs force information.

```
% generateDualRobotMagnetData.m

% This function generates data for a bunch of different robot misalignment
% scenarios.  This data will later be fit to a curve for quick access

% state = [x_oDot x_iDot xd]';
clear all
close all
clc

initializeValues

xDisplacements = -.05 :.001: .05;

cubeDim = 0.0254;
step = cubeDim/10;

dataMatrix = cell(length(xDisplacements));

for xIndex = 1:length(xDisplacements)
    xd = xDisplacements(xIndex);
    state = [0 0 xd];
    [M_o1 tau_o1] = getForcesRobotPair(state,cubeDim,step);
    magnetDataMatrix(xIndex,:) = [xd M_o1(1) M_o1(3) tau_o1(2)]
end

mdm = magnetDataMatrix;

p_x = polyfit(mdm(:,1),mdm(:,2),6);
p_z = polyfit(mdm(:,1),mdm(:,3),6);
p_tau = polyfit(mdm(:,1),mdm(:,4),6);

save('mdm','mdm')

plot(mdm(:,1),mdm(:,2),'o')
hold on
plot(mdm(:,1),polyval(p_x,mdm(:,1)))
title('Accuracy of polynomial approximation to magnet forces',...
    'Fontsize',28)
xlabel('displacement (m)','Fontsize',28)
ylabel('Force (N)','Fontsize',28)
legend('Actual magnet force','Polynomial approximation')
```

This function calls

- getForcesRobotPair

which is shown below

```
function[M_o1 tau_o1] = getForcesRobotPair(state,cubeDim,step)
% this function gets the forces on a pair of robots due to the halbach
```

```
% array static forces.  it also gets torques.  this function now finds
% ACTUAL FORCES, not estimates, so use this with:
% generateDualRobotMagnetData.m

% also due to symmetry, I've been able to cut this way down

% manas menon, 3/29/2010

% state = [x_oDot x_iDot xd]';

load allInitializedData

xd = state(3);

zd = (m_8 - m_7 + skinThickness + n_8 - n_7); % z distance between magnets
% find position of inner magnets

c_o1 = [r_om1(1:2) 0]; % location of outer magnet 1

c_i1 = [xd 0 zd] + [r_im1(1:2)'; 0]'; % xd and yd come from 'state'

theta_o1 = atan2(m_2,m_1) - pi/2;

theta_i1 = theta_o1;


% we calculate force on upper (inner) robot, so we take the negative of
% this force to find the force on the outer (lower) robot

% Magnet 1
[Fx Fy Fz Tx Ty Tz] = getForcesHalbachs(c_o1, theta_o1,1, ...
    c_i1, theta_i1, -1,J,cubeDim,step);
M_o1 = -[Fx 0 Fz];
tau_o1 = -[0 Ty 0];
```

This function calls

- getForcesHalbachs.m

Which is shown below:

```
function[Fx Fy Fz Tx Ty Tz] = getForcesHalbachs(centroid1,thetaZ1,d1,...
    centroid2,thetaZ2,d2,J,cubeDim,step)

% this function finds the forces / (and torques?) between a pair of halbach
% arrays.

% Manas Menon
% 3/10/2010

if centroid1(3) > centroid2(3)
    error('FIRST halbach array must be lower array')
end

mu = 1.26e-6;

% First create a surface over which we will calculate maxwell's stress
% tensor.
[x1 x2 y1 y2 z1 z2] = getTensorSurface(centroid1,thetaZ1,centroid2...
    ,thetaZ2,cubeDim);

% discretize the tensor surface - this should return a set of locations as
% well as vectors normal to the surface corresponding to each of these
% locations
[locations normals] = getEvaluationLocations(x1,x2,y1,y2,z1,z2,step);
temp = ones(size(locations));
```

```
% create matrix full of vectors from magnet COM to evaluation surface
temp(:,1) = temp(:,1)*centroid2(1);
temp(:,2) = temp(:,2)*centroid2(2);
temp(:,3) = temp(:,3)*centroid2(3);
momentArms = locations - temp;
clear temp

% find field at each of these locations due to BOTH halbach arrays
fields = nan(size(locations));
for k = 1:length(fields)
    location = locations(k,:);
    [Bx1 By1 Bz1] = findFieldHalbach(centroid1,cubeDim,J,d1,thetaZ1,...
        location);
    [Bx2 By2 Bz2] = findFieldHalbach(centroid2,cubeDim,J,d2,thetaZ2,...
        location);
    if sum(isnan([Bx1 By1 Bz1 Bx2 By2 Bz2]))
        disp([Bx1 By1 Bz1 Bx2 By2 Bz2])
        disp(location)
        error('nan')
    end
    fields(k,:) = [Bx1 + Bx2,By1 + By2,Bz1 + Bz2];
end

% SOME PLOTTING CODE HERE TO HELP IN DEBUGGING ***************************
drawHalbach(centroid1,cubeDim,J,d1,thetaZ1)
drawHalbach(centroid2,cubeDim,J,d2,thetaZ2)
drawTensorSurface(x1,x2,y1,y2,z1,z2)
quiver3(locations(:,1),locations(:,2),locations(:,3),fields(:,1)...
    ,fields(:,2),fields(:,3))
% quiver3(locations(:,1),locations(:,2),locations(:,3),normals(:,1)...
%     ,normals(:,2),normals(:,3))
% ************************************************************************

% calculate maxwell's stress tensor
forces = zeros(size(locations));
for i = 1:3
    for t = 1:length(fields)
        H = fields(t,:)/mu;
        n = normals(t,:);
        j = find(n ~= 0);
        if i == j;
            Hk = H(1)^2 + H(2)^2 + H(3)^2;
            forces(t,i) = (H(i)^2 - (1/2)*Hk)*sum(n);
        else
            forces(t,i) = H(i)*H(j)*sum(n);
        end
    end
end
forces = forces.*step^2*mu;

% now find torques
torques = nan(size(forces));
for t = 1:length(forces)
    torques(t,:) = cross(momentArms(t,:),forces(t,:));
end

% assign outputs
Fx = sum(forces(:,1));
Fy = sum(forces(:,2));
Fz = sum(forces(:,3));
Tx = sum(torques(:,1));
Ty = sum(torques(:,2));
Tz = sum(torques(:,3));
```

This function calls:

- getTensorSurface.m

- getEvaluationLocations.m
- findFieldHalbach.m

These are shown below:

```
---------------------------------------------------------------------------------------------
function[x1 x2 y1 y2 z1 z2] = getTensorSurface(centroid1,thetaZ1,...
    centroid2,thetaZ2,cubeDim)

% Given 2 halbach arrays, this function generates a rectangular surface
% used to calculate the maxwell stress tensor.  Outputs are the locations
% of the surfaces.

% Arrays are assumed to be the same size

% Manas Menon
% 3/10/2010

% first put some vectors into a nice usable form

c1(1) = centroid1(1);
c1(2) = centroid1(2);
c1(3) = centroid1(3);
c2(1) = centroid2(1);
c2(2) = centroid2(2);
c2(3) = centroid2(3);

c1 = [c1(1);c1(2)];
c2 = [c2(1);c2(2)];

R1 = [cos(thetaZ1) -sin(thetaZ1);
    sin(thetaZ1) cos(thetaZ1) ];

R2 = [cos(thetaZ2) -sin(thetaZ2) ;
    sin(thetaZ2) cos(thetaZ2) ];

cd = cubeDim;
cornersx = [-1.5*cd,-1.5*cd,1.5*cd,1.5*cd];
cornersy = [-0.5*cd,0.5*cd,0.5*cd,-0.5*cd];

rotatedCorners = [R1*[cornersx;cornersy] R2*[cornersx;cornersy]];
finalCorners = rotatedCorners + [c1 c1 c1 c1 c2 c2 c2 c2];

maxX = max(finalCorners(1,:));
minX = min(finalCorners(1,:));
maxY = max(finalCorners(2,:));
minY = min(finalCorners(2,:));
% max / min X and Y give the locations of the corners

bottomOfTopArray = max(centroid1(3),centroid2(3)) - cd/2;
topOfBottomArray = min(centroid1(3),centroid2(3)) + cd/2;

if bottomOfTopArray < topOfBottomArray
    error('Arrays are intersecting')
end

z1 = mean([bottomOfTopArray topOfBottomArray]);

clearance = cd / 2;

z2 = bottomOfTopArray + cd + clearance;

x1 = minX - clearance;
x2 = maxX + clearance;
y1 = minY - clearance;
y2 = maxY + clearance;
```

```
function[locations,normalVectors] = getEvaluationLocations(x1,x2,y1,y2,...
    z1,z2,step)

% this function gets evaluation locations as well as normal vectors for a
% given tensor surface


% create discretization points
xVec = [x1 + step/2 : step : x2 - step/2];
yVec = [y1 + step/2 : step : y2 - step/2];
zVec = [z1 + step/2 : step : z2 - step/2];

% initialize outputs
numberOfPoints = length(xVec)*length(yVec)*2 + length(xVec)*length(zVec)*2 ...
    + length(yVec)*length(zVec)*2;

locations = ones(numberOfPoints,3)*NaN;
normalVectors = locations;
% these just initialize the values, need to fix this

% Bottom surface
k = 1;
for x = xVec
    for y = yVec
        locations(k,:) = [x,y,z1];
        normalVectors(k,:) = [0 0 -1];
        k = k + 1;
    end
end

% Top surface
for x = xVec
    for y = yVec
        locations(k,:) = [x,y,z2];
        normalVectors(k,:) = [0 0 1];
        k = k + 1;
    end
end

% Front surface
for x = xVec
    for z = zVec
        locations(k,:) = [x,y1,z];
        normalVectors(k,:) = [0 -1 0];
        k = k + 1;
    end
end

% Back surface
for x = xVec
    for z = zVec
        locations(k,:) = [x,y2,z];
        normalVectors(k,:) = [0 1 0];
        k = k + 1;
    end
end

% 'Right' surface
for z = zVec
    for y = yVec
        locations(k,:) = [x2,y,z];
        normalVectors(k,:) = [1 0 0];
        k = k + 1;
    end
end

% 'Left' surface
```

```
for z = zVec
    for y = yVec
        locations(k,:) = [x1,y,z];
        normalVectors(k,:) = [-1 0 0];
        k = k + 1;
    end
end

if sum(isnan(locations) + isnan(normalVectors))
    error('something didn''t get assigned')
end
```

```
function[Bx By Bz]=findFieldHalbach(centroid,cubeDim,J,direction,thetaZ,location)

% findFieldHalbach.m
% Manas Menon
% 3/9/2010

% This program generates the fields due to a halbach array

% The halbach array is composed of cuboidal magnets.  The function takes as
% inputs the CENTROID of the magnet ARRAY, the dimension of a CUBE side,
% the magnetic charge density J, the direction (1 or -1 - ie is it facing
% up or down) and the angle theta z about the z axis.  No rotation is
% allowed about x or y.  Field is found at LOCATION

% Two halbach arrays with the same thetaZ and opposite directions are
% 'mated' with one another.

% choose different locations that correspond to each magnet
% remember to change Bx By Bz as needed

% generate rotation matrix:
R = [cos(thetaZ) -sin(thetaZ) 0;
    sin(thetaZ) cos(thetaZ) 0;
    0 0 1];

a = cubeDim/2;

% first try one of the magnets
magnet1location = R*[-3*a;-a;-a];
r = location - centroid - magnet1location';
% need to put r in the correct frame:
r = inv(R)*r';

% assume for now that this is correct,
dimensions = [cubeDim,cubeDim,cubeDim];
[X1 Y1 Z1] = findFieldSingle(dimensions,r,J);
vtemp = R*[X1;Y1;Z1];
X1 = vtemp(1);
Y1 = vtemp(2);
Z1 = vtemp(3);

% now add other 'vertical' magnet
magnet3location = R*[a;-a;-a];
r = location - centroid - magnet3location';
% need to put r in the correct frame:
r = inv(R)*r';

[X3 Y3 Z3] = findFieldSingle(dimensions,r,-J); % note negative J
vtemp = R*[X3; Y3 ;Z3];
X3 = vtemp(1);
Y3 = vtemp(2);
Z3 = vtemp(3);

% now find field due to middle magnet - this is tricky
magnet2location = R*[-a;-a;a];
```

```
r = location - centroid - magnet2location';
% need to put r in the correct frame:
RR = inv([0 0 -1;0 1 0;1 0 0]);
r = inv(RR)*inv(R)*r';

[X2 Y2 Z2] = findFieldSingle(dimensions,r,-direction*J);
vtemp = R*RR*[X2;Y2;Z2];
X2 = vtemp(1);
Y2 = vtemp(2);
Z2 = vtemp(3);

Bx = X1 + X3 + X2;
By = Y1 + Y3 + Y2;
Bz = Z1 + Z3 + Z2; % BAM IT WORKS
```

This function calls

- findFieldSingle.m

This is shown below:

```
function [Bx By Bz] = findFieldSingle(dimensions,location,J)

% findFieldSingle.m
% Manas Menon
% started 3/3/2010

% This script finds the field due to a single rectangular magnet.  It uses
% the formulation found in the paper, 'Three-Dimensional Analytical
% Optimization of Permanent Magnets Alternated Structure' by F. Bancel and
% G. Lemarquand.  We utilize a magnetic charge model of the system where we
% assume the magnetic surface charge density is known.  B is then found
% from a fairly straightforward but difficult integral, so thanks to
% Bancel and Lemarquand for solving it

% magnet is a box with sides length a (along x) b (along y) and c (along
% z).  The corner of the magnet is located at the origin, and the following
% expressions give fields at the location (x,y,z) in each direction Bx, By,
% and Bz.  J is the magnetic pole density at the surface of the magnet


a = dimensions(1);
b = dimensions(2);
c = dimensions(3);
x = location(1);
y = location(2);
z = location(3);

Bx = NaN;
By = NaN;
Bz = NaN;



while isnan(Bx)||isinf(Bx)||imag(Bx)
Bx = (J/(4*pi))*log(((b-y+sqrt(x^2-2*x*a+y^2+a^2+b^2-2*y*b+z^2-2*z*c+c^2))*...
    (-y+sqrt(x^2-2*z*c+z^2+y^2+c^2))*(b-y+sqrt(x^2+b^2-2*y*b+z^2+y^2))*...
    (-y+sqrt(x^2-2*x*a+y^2+a^2+z^2)))/((b-y+sqrt(x^2+b^2-2*y*b+z^2+y^2-2*z*c+c^2))*...
    (-y+sqrt(x^2-2*x*a+y^2+a^2+c^2+z^2-2*z*c))*(b-y+sqrt(x^2-2*x*a+y^2+a^2+b^2-2*y*b+z^2))*...
    (-y+sqrt(x^2+z^2+y^2))));
x = x + randn*eps^.5;
y = y + randn*eps^.5;
z = z + randn*eps^.5;
```

```
% disp('wiggle')
end
d1 = x + y + z - sum(location);


while isnan(By)||isinf(By)||imag(By)
% % used to be: By =(J/(4*pi))*log(((a-x+sqrt(x^2-2*x*a+y*y+a*a+x*x+b*b+c*c-2*y*b-2*z*c))*..
By = (J/(4*pi))*log(((a-x+sqrt(x^2-2*x*a+y*y+a*a+z*z+b*b+c*c-2*y*b-2*z*c))*...
    (-x+sqrt(x*x+z*z-2*z*c+y*y+c*c))*(a-x+sqrt(x*x-2*x*a+y*y+a*a+z*z))*...
    (-x+sqrt(x*x+z*z+b*b+y*y-2*y*b)))/((a-x+sqrt(x*x-2*x*a+y*y+a*a+z*z-2*z*c+c*c))*...
    (-x+sqrt(x*x+z*z+b*b+y*y-2*y*b+c*c-2*z*c))*(a-x+sqrt(x*x-2*x*a+y*y+a*a+z*z+b*b-2*y*b))*...
    (-x+sqrt(x*x+z*z+y*y))));
x = x + randn*eps^.5;
y = y + randn*eps^.5;
z = z + randn*eps^.5;
% disp('wiggle')
end
d2 = x + y + z - sum(location);

while isnan(Bz)||isinf(Bz)||imag(Bz)
% used to be Bz =
% (-J/(4*pi))*(atan(x*y/((c-z)*sqrt(x^2+y^2+x^2-2*z*c+c^2)))+atan(((a-x)*y)/((c-z)*sqrt(z^2+
2*a*x+x^2+y^2+z^2-2*z*c+c^2)))+..
Bz = (-J/(4*pi))*(atan(x*y/((c-z)*sqrt(x^2+y^2+z^2-2*z*c+c^2)))+atan(((a-x)*y)/((c-z)*sqrt(a^2-
2*a*x+x^2+y^2+z^2-2*z*c+c^2)))+...
    atan((x*(b-y))/((c-z)*sqrt(b^2-2*b*y+x^2+y^2+z^2-2*z*c+c^2)))+...
    atan((((-a+x)*(-b+y))/((c-z)*sqrt(a^2+b^2-2*a*x+x^2-2*b*y+y^2+z^2-2*z*c+c^2))))+...
    atan(x*y/(z*sqrt(x^2+y^2+z^2)))+atan(((a-x)*y)/(z*sqrt(a^2-2*a*x+x^2+y^2+z^2)))+...
    atan((x*(b-y))/(z*sqrt(b^2+x^2-2*b*y+y^2+z^2)))+atan(((a-x)*(b-y))/(z*sqrt(a^2+b^2-2*a*x+x^2-
2*b*y+y^2+z^2))));
% used to be atan((x*(b-y))/(z*sqrt(b^2+x^2-2*b*y+y^2+z^2)))+atan(((a-x)*(b-y))/(z*sqrt(z^2+b^2-
2*a*x+x^2-2*b*y+y^2+z^2))));
x = x + randn*eps^.5;
y = y + randn*eps^.5;
z = z + randn*eps^.5;
% disp('wiggle')
end
d3 = x + y + z - sum(location);

if max([ d1,d2,d3])>1e6
    disp(max([ d1,d2,d3]))
end
```

Deriving reaction forces

In this section we show the derivation of reaction forces for the inner and outer robot. For the sake of brevity, we only show the derivation for the full system case – the simplified case is a subset of this.

```
% deriveReactionForcesInner.m

% Manas Menon
% started 3/21/2010

% This script is very similar to (but simpler than)
% deriveReactionForcesOuter.m - here we don't have to deal with stuff like
% constrained motion and such.  I'll copy paste a lot of code over and trim
% it up as needed.

% here we are only trying to find 4 reaction forces, so we only need 4
% equations.  We get these equations by taking sum of moments (2) linear
% force in z direction (1) and method of deformations (1)

% 3/23/2010 - update - I realized that in this formulation, like the last
% one, we are going to need to find a few extra reaction forces.  Here we
% need to find 8 extra, so we need 8 more equations for a total of 12
```

```
% 3/25/2010 - another update - those extra equations can be immediately
% substituted back into the original 4, so we're back to only 4

clear all
close all
clc

% magnet forces
syms M_i1 M_i2 M_i3 M_i4 M_i1x M_i1y M_i1z M_i2x M_i2y M_i2z M_i3x ...
    M_i3y M_i3z M_i4x M_i4y M_i4z

% magnet torques
syms tau_i1 tau_i2 tau_i3 tau_i4 tau_i1x tau_i2x tau_i3x tau_i4x ...
    tau_i1y tau_i2y tau_i3y tau_i4y tau_i1z tau_i2z tau_i3z tau_i4z

% reaction forces
syms R_i1 R_i2 R_i3 R_i4 R_i1x R_i2x R_i3x R_i4x R_i1y R_i2y R_i3y ...
    R_i4y R_i1z R_i2z R_i3z R_i4z

% location vectors
syms r_im1 r_im2 r_im3 r_im4 r_iw1 r_iw2 r_iw3 r_iw4

% dimensions
syms n_1 n_2 n_3 n_4 n_7 n_8

% deflections
syms delta_i1 delta_i2 delta_i3 delta_i4 delta_ic k_i1 k_i2 k_i3 k_i4

% misc
syms m_i g x y z a_i b_i I_iC I_iB thetadotdot_i c

% velocities
syms v_iCx v_iCy v_iCtheta v_iw1 v_iw2 v_iw3 v_iw4


% build some vectors here:
M_i1 = [M_i1x M_i1y M_i1z];
M_i2 = [M_i2x M_i2y M_i2z];
M_i3 = [M_i3x M_i3y M_i3z];
M_i4 = [M_i4x M_i4y M_i4z];

R_i1 = [R_i1x R_i1y R_i1z];
R_i2 = [R_i2x R_i2y R_i2z];
R_i3 = [R_i3x R_i3y R_i3z];
R_i4 = [R_i4x R_i4y R_i4z];

tau_i1 = [tau_i1x tau_i1y tau_i1z];
tau_i2 = [tau_i2x tau_i2y tau_i2z];
tau_i3 = [tau_i3x tau_i3y tau_i3z];
tau_i4 = [tau_i4x tau_i4y tau_i4z];

% INNER ROBOT SPECIFIC
% magnet locations
r_im1 = [n_1,n_2,-n_7];
r_im2 = [-n_1,n_2,-n_7];
r_im3 = [-n_1,-n_2,-n_7];
r_im4 = [n_1,-n_2,-n_7];

% wheel locations
r_iw1 = [n_3 n_4 -n_8];
r_iw2 = [-n_3 n_4 -n_8];
r_iw3 = [-n_3 -n_4 -n_8];
r_iw4 = [n_3 -n_4 -n_8];

% moments
magnetForceTorques = cross(r_im1,M_i1) + cross(r_im2,M_i2) + ...
    cross(r_im3,M_i3) + cross(r_im4,M_i4);
```

```
magnetTorques = tau_i1 + tau_i2 + tau_i3 + tau_i4;

reactionTorques = cross(r_iw1,R_i1) + cross(r_iw2,R_i2) + ...
    cross(r_iw3,R_i3) + cross(r_iw4,R_i4);

allTorques = magnetForceTorques + magnetTorques + reactionTorques;
eq1 = allTorques(1);
eq2 = allTorques(2);

% force in z direction
zForces = R_i1z + R_i2z + R_i3z + R_i4z + M_i1z + M_i2z + M_i3z + M_i4z ...
    - m_i*g;
eq3 = zForces;

% method of deformations INNER ROBOT SPECIFIC
% first find the equation for the plane
plane = 'a_i*x + b_i*y = z';
deformationEq1 = subs(plane,{x,y,z},{0,2*n_4,-delta_i2});
deformationEq2 = subs(plane,{x,y,z},{2*n_3,0,-delta_i4});

[a_i b_i] = solve(deformationEq1,deformationEq2,a_i,b_i);

% plug reaction forces into deltas - be careful with sign!
% now pick either point 1 or 4.  I choose 1!
delta_ic = R_i3z/k_i3;
delta_i1 = R_i1z/k_i1 - delta_ic;
delta_i2 = R_i2z/k_i2 - delta_ic;
delta_i4 = R_i4z/k_i4 - delta_ic;

a_i = subs(a_i,{'delta_i1','delta_i2','delta_i4'},...
    {delta_i1,delta_i2,delta_i4});
b_i = subs(b_i,{'delta_i1','delta_i2','delta_i4'},...
    {delta_i1,delta_i2,delta_i4});
eq4 = a_i*(2*n_3) + b_i*(2*n_4) + delta_i1;

% finding lateral and longitudinal forces.  First of all we need to find
% velocities of the wheels.
v_ic = [v_iCx v_iCy 0];
w_i = [0 0 v_iCtheta];

v_iw1 = v_ic + cross(w_i,r_iw1);
v_iw2 = v_ic + cross(w_i,r_iw2);
v_iw3 = v_ic + cross(w_i,r_iw3);
v_iw4 = v_ic + cross(w_i,r_iw4);

% we know the following relationships exist:
% R_i1x = -c*R_i1z*v_iw1(1)/(v_iw1(1)^2 + v_iw1(2)^2)^(1/2);
% R_i1y = -c*R_i1z*v_iw1(2)/(v_iw1(1)^2 + v_iw1(2)^2)^(1/2);

% R_i2x = -c*R_i2z*v_iw2(1)/(v_iw2(1)^2 + v_iw2(2)^2)^(1/2);
% R_i2y = -c*R_i2z*v_iw2(2)/(v_iw2(1)^2 + v_iw2(2)^2)^(1/2);

% R_i3x = -c*R_i3z*v_iw3(1)/(v_iw3(1)^2 + v_iw3(2)^2)^(1/2);
% R_i3y = -c*R_i3z*v_iw3(2)/(v_iw3(1)^2 + v_iw3(2)^2)^(1/2);

% R_i4x = -c*R_i4z*v_iw4(1)/(v_iw4(1)^2 + v_iw4(2)^2)^(1/2);
% R_i4y = -c*R_i4z*v_iw4(2)/(v_iw4(1)^2 + v_iw4(2)^2)^(1/2);
% we can write these as our equations:
eq5 = -c*R_i1z*v_iw1(1)/(v_iw1(1)^2 + v_iw1(2)^2)^(1/2)-R_i1x;
eq6 = -c*R_i1z*v_iw1(2)/(v_iw1(1)^2 + v_iw1(2)^2)^(1/2)-R_i1y;

eq7 = -c*R_i2z*v_iw2(1)/(v_iw2(1)^2 + v_iw2(2)^2)^(1/2)-R_i2x;
eq8 = -c*R_i2z*v_iw2(2)/(v_iw2(1)^2 + v_iw2(2)^2)^(1/2)-R_i2y;

eq9 = -c*R_i3z*v_iw3(1)/(v_iw3(1)^2 + v_iw3(2)^2)^(1/2)-R_i3x;
eq10 = -c*R_i3z*v_iw3(2)/(v_iw3(1)^2 + v_iw3(2)^2)^(1/2)-R_i3y;

eq11 = -c*R_i4z*v_iw4(1)/(v_iw4(1)^2 + v_iw4(2)^2)^(1/2)-R_i4x;
eq12 = -c*R_i4z*v_iw4(2)/(v_iw4(1)^2 + v_iw4(2)^2)^(1/2)-R_i4y;
```

```
% now build the A matrix and b vector to solve Ax = b
% x = [R_i1z R_i2z R_i3z R_i4z]
equationsVector = [eq1 eq2 eq3 eq4 eq5 eq6 eq7 eq8 eq9 eq10 eq11 eq12];
unknownsVector = [R_i1z; R_i2z; R_i3z; R_i4z; R_i1x; R_i1y; R_i2x; ...
    R_i2y; R_i3x; R_i3y; R_i4x; R_i4y];

for i = 1:length(equationsVector)
    equationChoice = equationsVector(i);
    for j = 1:length(unknownsVector)
        unknownChoice = unknownsVector(j);
        A(i,j) = diff(equationChoice,unknownChoice);
    end
    b(i,1) = A(i,:)*unknownsVector - equationChoice;
end
A_inner = A;
b_inner = b;
clear b
```

```
% deriveReactionForcesOuter.m

% a script to find an expression to get the reaction forces of the outer
% robot.  The forces we want are: z axis reaction forces at wheels (4) and
% y axis reaction force that constrains the motion (1) which means we need
% 5 equations total

% 3/23/2010 - I've realized we also need lateral and longitudinal reaction
% forces at wheels - this requires 4 more equations for 9 equations total

% Manas Menon
% started 3/20/2010

clear all
close all
clc

% magnet forces
syms M_o1 M_o2 M_o3 M_o4 M_o1x M_o1y M_o1z M_o2x M_o2y M_o2z M_o3x ...
    M_o3y M_o3z M_o4x M_o4y M_o4z

% magnet torques
syms tau_o1 tau_o2 tau_o3 tau_o4 tau_o1x tau_o2x tau_o3x tau_o4x ...
    tau_o1y tau_o2y tau_o3y tau_o4y tau_o1z tau_o2z tau_o3z tau_o4z

% reaction forces
syms R_o1 R_o2 R_o3 R_o4 R_o1x R_o2x R_o3x R_o4x R_o1y R_o2y R_o3y ...
    R_o4y R_o1z R_o2z R_o3z R_o4z

% inputs
syms F_motor2 F_motor3 tau_motor2 tau_motor3

% location vectors
syms r_om1 r_om2 r_om3 r_om4 r_ow1 r_ow2 r_ow3 r_ow4 r_bm1 r_bm2 r_bm3 ...
    r_bm4 r_bw1 r_bw2 r_bw3 r_bw4

% dimensions
syms m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8 m_9

% deflections
syms delta_o1 delta_o2 delta_o3 delta_o4 delta_oc k_o1 k_o2 k_o3 k_o4

% misc
syms m_o g x y z a_o b_o I_oC I_oB thetadotdot_o v_Bx v_By v_oCx v_oCy c

% velocities
syms x_owdot y_owdot theta_owdot v_ow1 v_ow1 v_ow2 v_ow4
```

```
% build some vectors here:
M_o1 = [M_o1x M_o1y M_o1z];
M_o2 = [M_o2x M_o2y M_o2z];
M_o3 = [M_o3x M_o3y M_o3z];
M_o4 = [M_o4x M_o4y M_o4z];

R_o1 = [R_o1x R_o1y R_o1z];
R_o2 = [F_motor2 0 R_o2z];
R_o3 = [F_motor3 R_o3y R_o3z];
R_o4 = [R_o4x R_o4y R_o4z];

tau_o1 = [tau_o1x tau_o1y tau_o1z];
tau_o2 = [tau_o2x tau_o2y tau_o2z];
tau_o3 = [tau_o3x tau_o3y tau_o3z];
tau_o4 = [tau_o4x tau_o4y tau_o4z];

% OUTER ROBOT SPECIFIC
% magnet locations
r_om1 = [m_1,m_2,m_7];
r_om2 = [-m_1,m_2,m_7];
r_om3 = [-m_1,-m_2,m_7];
r_om4 = [m_1,-m_2,m_7];

% wheel locations
r_ow1 = [m_3 m_4 m_8];
r_ow2 = [-m_5 m_6 m_9];
r_ow3 = [-m_5 -m_6 m_9];
r_ow4 = [m_3 -m_4 m_8];

% from the rear axle
RAvec = [m_5,m_6,-m_9];

r_bm1 = r_om1 + RAvec;
r_bm2 = r_om2 + RAvec;
r_bm3 = r_om3 + RAvec;
r_bm4 = r_om4 + RAvec;

r_bw1 = r_ow1 + RAvec;
r_bw2 = r_ow2 + RAvec;
r_bw3 = r_ow3 + RAvec;
r_bw4 = r_ow4 + RAvec;

% moments about COM
magnetForceTorques = cross(r_om1,M_o1) + cross(r_om2,M_o2) + ...
    cross(r_om3,M_o3) + cross(r_om4,M_o4);

magnetTorques = tau_o1 + tau_o2 + tau_o3 + tau_o4;

reactionTorques = cross(r_ow1,R_o1) + cross(r_ow2,R_o2) + ...
    cross(r_ow3,R_o3) + cross(r_ow4,R_o4);

% chose the convention that positive motor torque (which is the input)
% leads to positive forces at the rear axle of the robot.
motorTorques = [0 tau_motor2+tau_motor3 0];

allTorques = magnetForceTorques + magnetTorques + reactionTorques + ...
    motorTorques;

eq1 = allTorques(1);
eq2 = allTorques(2);

% force in z direction
zForces = R_o1z + R_o2z + R_o3z + R_o4z + M_o1z + M_o2z + M_o3z + M_o4z ...
    - m_o*g;
eq3 = zForces;

% method of deformations OUTER ROBOT SPECIFIC
% first find the equation for the plane
plane = 'a_o*x + b_o*y = z';
```

```
deformationEq1 = subs(plane,{x,y,z},{0,2*m_6,delta_o2});
deformationEq2 = subs(plane,{x,y,z},{m_3 + m_5,m_6 - m_4,delta_o4});
[a_o b_o] = solve(deformationEq1,deformationEq2,a_o,b_o);

% plug reaction forces into deltas - be careful with sign!
% now pick either point 1 or 4.  I choose 1!
delta_oc = -R_o3z/k_o3;
delta_o1 = -R_o1z/k_o1 - delta_oc;
delta_o2 = -R_o2z/k_o2 - delta_oc;
delta_o4 = -R_o4z/k_o4 - delta_oc;
a_o = subs(a_o,{'delta_o1','delta_o2','delta_o4'},{delta_o1,delta_o2,...
    delta_o4});
b_o = subs(b_o,{'delta_o1','delta_o2','delta_o4'},{delta_o1,delta_o2,...
    delta_o4});
eq4 = a_o*(m_3+m_5) + b_o*(m_6 + m_4) - delta_o1;

% need a few more equations - holonomic wheel constraints - find
% thetadotdot_o from moments around z axis, taken at center of rear axle
% magnetForceTorquesB = cross(r_bm1,M_o1) + cross(r_bm2,M_o2) + ...
%     cross(r_bm3,M_o3) + cross(r_bm4,M_o4);
%
% magnetTorquesB = tau_o1 + tau_o2 + tau_o3 + tau_o4;
%
% reactionTorquesB = cross(r_bw1,R_o1) + cross(r_bw2,R_o2) + ...
%     cross(r_bw3,R_o3) + cross(r_bw4,R_o4);
%
% motorTorquesB = motorTorques;
%
% allTorquesB = magnetForceTorquesB + magnetTorquesB + reactionTorquesB + ...
%     motorTorquesB;

% now, given velocity of COM, we find the angular velocity
v_oC = [v_oCx v_oCy 0];
P = m_o*v_oC;
temp = v_oC + cross([0 0 theta_owdot],-RAvec); % only care about 2nd term
theta_owdot = solve(temp(2),theta_owdot);

% find acceleration of center of mass
% ax = (R_o1x + F_motor2 + F_motor3 + R_o4x + M_o1x + M_o2x + M_o3x + ...
%     M_o4x)/m_o;
ay = (R_o1y + R_o3y + R_o4y + M_o1y + M_o2y + M_o3y + M_o4y)/m_o;
% a = [ax ay 0];

% use thetadot to find vb
vb = v_oC + cross([0 0 theta_owdot],-RAvec);
% cp = cross(vb,P);
% thetadotdot_o = (allTorquesB(3) - cp(3))/I_oB;
% now that we have thetadotdot_o, can take torques about COM and get
% reaction.  Note that we already computed all the torques about the COM
% earlier in the script.  At this point, we are only interested in the 3rd
% component.

thetadotdot_o  = allTorques(3)/I_oC;

eq5 = ay + thetadotdot_o*(-m_5) - vb(1)*theta_owdot - ...
    theta_owdot^2*m_6;

% % temporary section to check some stuff
% temp = (allTorquesB(3) - v_oCy*m_o*(v_oCx - theta_owdot*-m_6))/I_oB


% next, we need to deal with the lateral and longitudinal reaction forces.
% next find wheel velocities
w = [0 0 theta_owdot];
v_ow1 = v_oC + cross(w,r_ow1);
v_ow2 = v_oC + cross(w,r_ow2);
v_ow3 = v_oC + cross(w,r_ow3);
v_ow4 = v_oC + cross(w,r_ow4);
```

```
% for each wheel velocity, we have the following relationships
% R_o4x = c*R_o4z*v_ow4(1)/(v_ow4(1)^2 + v_ow4(2)^2)^(1/2);
% R_o4y = c*R_o4z*v_ow4(2)/(v_ow4(1)^2 + v_ow4(2)^2)^(1/2);
%
% R_o1x = c*R_o1z*v_ow1(1)/(v_ow1(1)^2 + v_ow1(2)^2)^(1/2);
% R_o1y = c*R_o1z*v_ow1(2)/(v_ow1(1)^2 + v_ow1(2)^2)^(1/2);
% which we can write as equations:

eq6 = c*R_o4z*v_ow4(1)/(v_ow4(1)^2 + v_ow4(2)^2)^(1/2)- R_o4x;
eq7 = c*R_o4z*v_ow4(2)/(v_ow4(1)^2 + v_ow4(2)^2)^(1/2)- R_o4y;

eq8 = c*R_o1z*v_ow1(1)/(v_ow1(1)^2 + v_ow1(2)^2)^(1/2)- R_o1x;
eq9 = c*R_o1z*v_ow1(2)/(v_ow1(1)^2 + v_ow1(2)^2)^(1/2)- R_o1y;

% we now have a set of LINEAR EQUATIONS -

% now build the A matrix and b vector for Ax - b = 0.
equationsVector = [eq1 eq2 eq3 eq4 eq5 eq6 eq7 eq8 eq9];
unknownsVector = [R_o1z; R_o2z; R_o3z; R_o4z; R_o3y; R_o1x; R_o1y; ...
    R_o4x; R_o4y];

for i = 1:length(equationsVector)
    equationChoice = equationsVector(i);
    for j = 1:length(unknownsVector)
        unknownChoice = unknownsVector(j);
        A(i,j) = diff(equationChoice,unknownChoice);
    end
    b(i,1) = A(i,:)*unknownsVector - equationChoice;
end
A_outer = subs(simple(A));
b_outer = subs(simple(b));
clear b;
```

Initialized data

This script describes the parameters of the current system and initialized a lot of values for the simulations.

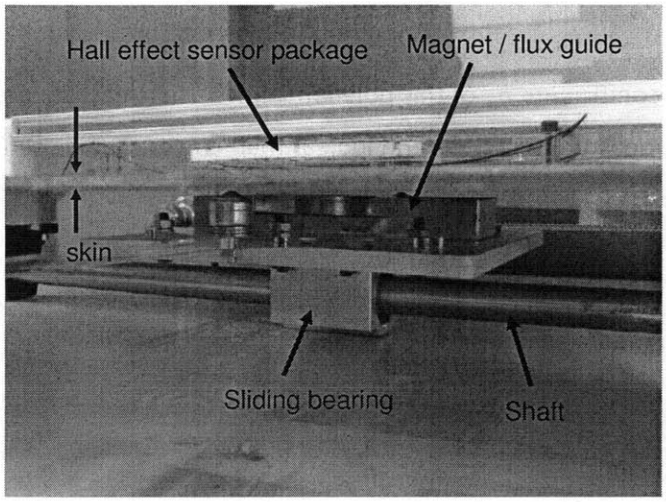
```
% initializeValues.m
% Manas Menon
% 3/26/2010

% This program initializes all the values for the dual robot simulation

clear all
close all
clc

% start by getting matrices required to find the reaction forces
deriveReactionForcesOuter
save outerExpressions
deriveReactionForcesInner
load outerExpressions
% ROBOT CONSTANTS ************************************************************
% dimensions
% outer
m_1 = 0.107;
m_2 = 0.107;
m_3 =  0.18059;
m_4 = 0.133;
m_5 = 0.1811;
m_6 = 0.18288;
m_9 = .06;
m_7 = m_9 + 0.03124;
```

```
wheelRadius = 0.061976;
m_8 = m_9 + wheelRadius;


% inner
n_1 = m_1;
n_2 = m_2;
n_3 = m_3;
n_4 = m_4;
n_7 = .02;
n_8 = n_7 + 0.0259;

% stifnesses
% outer
k_o1 = 1e6;
k_o2 = 1e5;
k_o3 = 1e5;
k_o4 = 1e6;

% inner
k_i1 = 1e6;
k_i2 = 1e6;
k_i3 = 1e6;
k_i4 = 1e6;

% intertia / masses / friction / gravity
BHmax = -48; % MGOe
mu = 1.26e-6;
m_o = 8.7;
m_i = 5.8; % kg
c = .025;
c_r = .5; % coefficient of static friction at rubber / Al surface
b = 6.8; % damping - eddy currents
g = 9.8;
skinThickness = 1/8*2.54/100;
J = (-BHmax*100/(4*pi*10^-3)/mu*4)^(1/2)*mu;

% location vectors
% outer
r_om1 = [m_1 m_2 m_7];
r_om2 = [-m_1 m_2 m_7];
r_om3 = [-m_1 -m_2 m_7];
r_om4 = [m_1 -m_2 m_7];
% inner
r_im1 = [n_1 n_2 -n_7];
r_im2 = [-n_1 n_2 -n_7];
r_im3 = [-n_1 -n_2 -n_7];
r_im4 = [n_1 -n_2 -n_7];

% ************************************************************************

% magnet stuff
load mdm
p_x = polyfit(mdm(:,1),mdm(:,2),4);
p_z = polyfit(mdm(:,1),mdm(:,3),4);
p_tau = polyfit(mdm(:,1),mdm(:,4),4);

% do a substitution here - should help speed stuff up

A_outer = subs(A_outer);
b_outer = subs(b_outer);
A_inner = subs(A_inner);
b_inner = subs(b_inner);

save allInitializedData
```

C. Hall Effect sensors

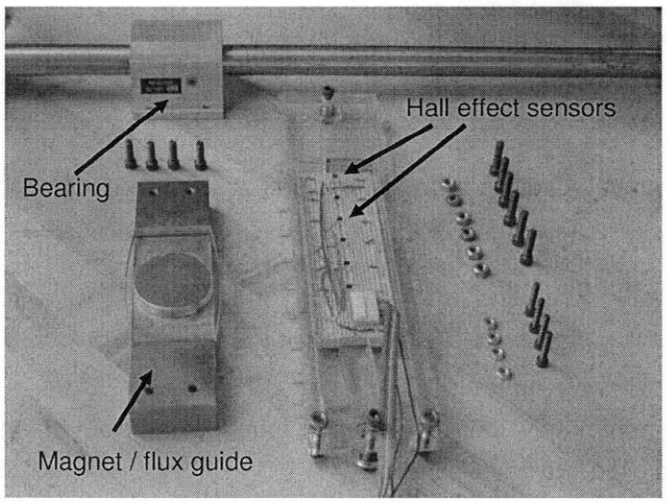
This section describes a proof of concept experiment run to see the effectiveness of using Hall Effect sensors for position estimation across the skin. Sensors distributed over the outer robot would use the magnetic fields to estimate the position of the inner robot. This test was on a one degree of freedom test bed.

Fig. C-1 shows the setup used to test the position sensing ability of this system.



C - 1: Test setup

The magnet / flux guide assembly shown is able to slide back and forth along the steel shaft. Across the aluminum skin, an array of Hall Effect sensors detects the magnetic field produced by the tooling and estimates its position. Fig. C-2 shows the critical components of the assembly.



C - 2: Critical components

Let us call the position of the tooling x, and the output of each sensor f(x). We choose to model the response of each of these sensors as a different n^{th} degree polynomial in x. For each sensor, we can write:

$$y_i = f_i(x) = \begin{bmatrix} x^n & x^{n-1} & \dots & 1 \end{bmatrix} \begin{bmatrix} b_{i1} \\ b_{i2} \\ \vdots \\ b_{in} \end{bmatrix} = \varphi^T \theta_i$$

In order to estimate the coefficients b_{ij} of this polynomial, we use a least squares approximation. We take a series of position measurements, φ, and look at their corresponding sensor outputs. The parameters of the θ matrix can be estimated in a least squares sense for each sensor using:

$$\hat{\theta} = PB$$

Where, if we have taken N data samples, we can find the P and B matrices from the following expressions.

$$P = \left[\sum_{t=1}^{N} \left(\varphi(t) \varphi^T(t) \right) \right]^{-1}$$

$$B = \sum_{t=1}^{N} y(t) \varphi(t)$$

This results in a polynomial fit to the sampled data. We can use this polynomial fit to estimate the position of the tooling, based on the data from the sensors. If we estimate that the tooling is located at some position $\hat{x}$, then we would expect sensor i to have the output

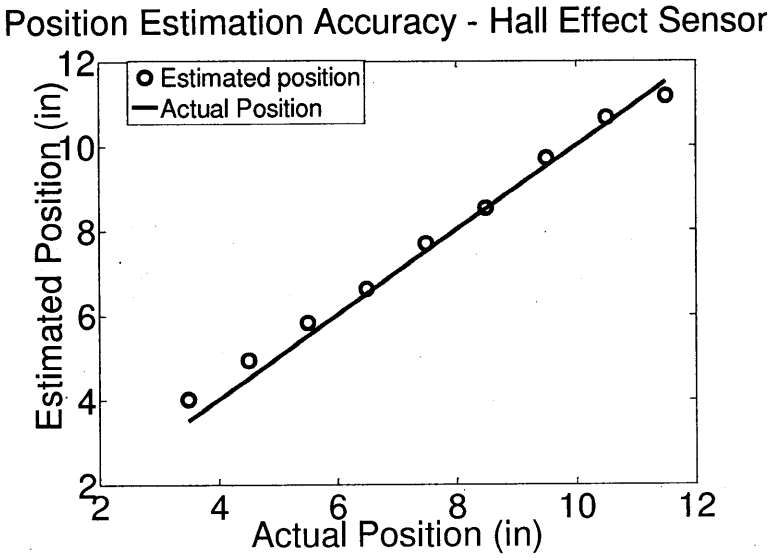
$$\hat{y}_i = b_{i1} \hat{x}^n + b_{i2} \hat{x}^{n-1} + \dots + 1$$

We compare this to the actual readings from the sensors, $y_1 \dots y_l$, and look at the squared error. This error function is:

$$\sum_{i=1}^{l} \left(\hat{y}_i - y_i \right)^2$$

We numerically find the minimum values for this error function and from that estimate the location $\hat{x}$ of the tooling. In the setup used, we had 6 sensors.

The benefit of this methodology is that we can achieve almost arbitrarily accurate positioning, by sampling more data and fitting a larger order polynomial to the data. By fitting 10 points at approximately 1'' intervals we were able to achieve accuracy better than 0.25'' for most of the 10'' of stroke of the tooling, as shown in **Error! Reference source not found.**. The data points used for the fit were chosen on the inch marks, while the data points used for testing were at the ½ inch marks.



C - 3: Actual position vs Estimated position

Noise from the sensors manifested itself in position estimation noise with magnitude around 0.01.'' In addition, position was estimated visually, and is prone to errors due to play in the bearings.

References

[1] Abbott, J.J.; Ergeneman, O.; Kummer, M.P.; Hirt, A.M.; Nelson, B.J., *Modeling magnetic torque and force for controlled manipulation of soft-magnetic bodies,* Advanced intelligent mechatronics, 2007 ieee/asme international conference on , vol., no., pp.1-6, 4-7 Sept. 2007

[2] Abdel-Rahman, E. Nayfeh, A. Masoud, Z. *Dynamics and Control of Cranes: A Review,* Journal of Vibration and Control 2003; 9; 863

[3] Armstrong-Helouvry, B. *Control of Machines with Friction,* Kluwer Academic Publishers, 1991.

[4] Bancel, F. Lemarquand, G. *Three-Dimensional Analytical Optimization of Permanent Magnets Alternated Structure.* IEEE transactions on magnetics mag. 1998 vol 34 no. 1 no. 2 pages 242-247.

[5] Baruh, H. *Analytical Dynamics.* WCB McGraw-Hill, 1999

[6] Campbell, P. *Permanent Magnet Materials and their Application,* Cambridge University Press 1994.

[7] Canudas de Wit, C., Olsson, H. Astrom, K.J. Lischinsky, P. *A New Model for Control of Systems with Friction.* IEEE Transactions on Automatic Control. Vol 40, no. 3 March 1995.

[8] Chatzandroulis, S. Tsoukalas, D. Neukomm, P.A. *A miniature pressure system with a capacitive sensor and a passive telemetry link for use in implantable applications,* Journal of Microelectromechanical Systems, Vol. 9, pp. 18-23, March 2000

[9] Wright, C. Johnson, A. Peck, A. McCord, Z. Naaktgeboren, A. Gianfortoni, P. Gonzalex-Rivero, M. Hatton, R. Choset, H. *Design of a Modular Snake Robot.* Proc of 2007 IEEE/RSJ Int Conf on Intelligent Robots and Systems. San Diego, CA. 2007

[10] Chuah, Y.T. Chan, P.K. Siek, L. *A wireless telemetry system for strain measurement,* Canadian Conf. on Electrical and Computer Engineering, Vol.2 , pp. 1018-1021, 2000

[11] Dukkipati, R.V. *Vehicle Dynamics,* Boca Raton, CRC Press, ISBN 0-8493-0976-X, 2000

[12] Finkenzeller, K. *RFID Handbook Radio-Frequency Identification: Fundametnals and Applications*, John Wiley & Son, New York, 1999.

[13] Franklin, G.F. Powell, J. D. and Emami-Naeini, *A. Feedback Control of Dynamic Systems.* Addison-Wesley, second edition, 1991.

[14] Griffiths, D. J. (1989). *Introduction to electrodynamics*. Englewood Cliffs, NJ: Prentice-Hall.

[15] Hadfield, D. *Permanent magnets and magnetism: theory, materials, design, manufacture and applications.* [Editorial panel: F.A. Benson, D. Harrison [and] W. Sucksmith. Contributors: E.N. da C. Andrade [and others]. London, Iliffe Books; New York, J. Wiley [1962]

[16] Halbach, K. *Design of permanent multipole magnets with oriented rare earth cobalt material.* Nuclear Instruments and Methods, vol. 169. Issue 1, pp 1-10

[17] Hammond, P. Sykulski, J. *Engineering Electromagnetism : Physical Processes and Computation.* Oxford University Press. Oxford, England, 1994.

[18] Harpster, T. J. Stark, B. and Najafi, K. *A Passive Wireless Integrated Humidity Sensor,* The 14th IEEE International Conference on MEMS, pp. 553-557, 2001

[19] Jang, S.M. Lee, S.H. Jeong, S.S. *Characteristic Analysis of Eddy-Current Brank System Using the Linear Halbach Array.* IEEE Transactions on Magnetics, 2005

[20] Kim S.; Spenko, M.; Trujillo, S.; Heyneman, B.; Santos, D.; Cutkosky, M.R., *Smooth Vertical Surface Climbing With Directional Adhesion*, Robotics, IEEE Transactions on , vol.24, no.1, pp.65-74, Feb. 2008

[21] Kronemer, A. Henneberger, J.E. *Productivity in Aircraft Manufacturing.* Monthly Labor Review 1993

[22] Luk, B.L.; Collie, A.A.; Billingsley, J., *Robug II: An intelligent wall climbing robot,* Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on , vol., no., pp.2342-2347 vol.3, 9-11 Apr 1991

[23] Martel, S. Madden, P. Sosnowski, L. Hunter, I. Lafontaine, S. *Nanowalker: A fully Autonomous Highly Integrated Miniature Robot for Nanoscale Measurements.* Proc of SPIE Vol 3825, 111. 1999

[24] Massachusetts Institute of Technology Global Airline Industry Program. *Airline Industry Overview.* <http://web.mit.edu/airlines/analysis/analysis_airline_industry.html>

[25] McCaig, M. *Permanent Magnets in Theory and Practice*, John Wiley and Sons, Inc. New York, 1977

[26] Menon, M. ; Asada, H. H, *Design of a Semi-Passive Heavy-Duty Mobile Robotic System for Automated Assembly Inside an Aircraft Body,* Intelligent Robots and Systems, 2008 (IROS 2008).

[27] Menon, M. Asada, H. H, *Design of an Instrumented Multifunctional Foot for Application to a Heavy Duty Mobile Robot Manufacturing System.* M.S. Thesis

[28] Meriam, J. L. (1975). *Dynamics: Second Edition - SI Version.* Toronto: John Wiley & Sons, Inc.

[29] Pack, R.T.; Christopher, J.L., Jr.; Kawamura, K., *A Rubbertuator-based structure-climbing inspection robot*, Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on , vol.3, no., pp.1869-1874 vol.3, 20-25 Apr 1997Asbeck, A. Kim, S. Cutkosky, M.R. Provancher, W. Lanzetta, M. *Scaling Hard Vertical Surfaces with Compliant Microspine Arrays*, The International Journal of Robotics Research, Dec 2006; 25: 1165 - 1179.

[30] Parker, R. Studders, R. *Permanent Magnets and Their Application.* John Wiley and Sons Inc, New York, USA, 1962

[31] Rajamani, R. *Vehicle Dynamics and Control.* Springer, 2005.

[32] Ravaud, R. Lemarquand, G. Lemarquand, V. *Magnetic Field Created by Tile Permanent Magnets.* IEEE transactions on Magnetics 45, 7 (2009) 2920-2926.

[33] Roters, H., *Electromagnetic Devices.* John Wiley and Sons, Inc. USA, 1941.

[34] Roy, B.; Asada, H.H., *Design of a Reconfigurable Robot Arm for Assembly Operations inside an Aircraft Wing-Box*, Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on pp. 590-595, 18-22 April 2005

[35] Roy, B.; Asada, H.H., *An Under-actuated Robot with a Hyper-articulated Deployable Arm Working Inside an Aircraft Wing-box,* Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on pp. 4046-4050, 2-6 Aug. 2005

[36] Singhose, W. Porter, L. Seering, W. *Input Shaped Control of a Planar Gantry Crane with Hoisting.* Proceedings of the American Control Conference. Albuquerque, NM June 1997.

[37] Slocum, A. *Precision Machine Design*, Prentice-Hall, 1992.
[38] Vollmers, K. Frutiger, D. R. Kratochvil, B. E. Nelson, B. J. *Wireless Resonant Magnetic Microactuator for Untethered Mobile Microrobots*, Applied Physics Letters, Vol. 92, No. 14, 2008.
[39] Williams J.H. Jr (1996) *Fundamentals of applied dynamics.* Wiley, NewYork
[40] Woodson, H, and Melcher, J, *Electromagnetic Dynamics, Part II: Fields Forces and Motion,* The Rober E.Kriger Publishing Company. New York, 1985.
[41] Yamanishi, Y. Sakuma, S. and Arai, F. *On-chip Cell Manipulation by Magnetically Modified Soft Microactuators*, Proceedings of the 2008 IEEE International Conference on Robotics and Automation, pp.899-904, 2008.
[42] Young, H. Freedman, R. *University Physics,* Addison-Wesley Publishing Company 1996.