ABSTRACT

Title of thesis: A STUDY OF SELECTED ASPECTS OF ELECTROMAGNETIC FORMATION FLIGHT

Peter Nathaniel Gardner, Master of Science, 2008

Thesis directed by: Professor Raymond Sedwick
Department of Aerospace Engineering

Electromagnetic Formation Flight (EMFF) is a technique for electromagnetically controlling the relative position and velocity of satellites in close proximity, without using propellant.

An optimal design for an EMFF system for clusters of small satellites was calculated. Trends in parameters were identified, taking into account thermal issues.

A power transfer system, using strongly coupled magnetic resonance, was simulated, using the same coils as the EMFF system. The efficiencies were calculated for the same parameters.

A scheme for EMFF control was tested, in which two satellites at a time were active, with their dipoles aligned with each other on-axis. This system was shown to keep clusters of four satellites within specified boundaries.

A STUDY OF SELECTED ASPECTS OF ELECTROMAGNETIC FORMATION FLIGHT

by

Peter Nathaniel Gardner

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2008

Acknowledgments

Thank you to everyone who helped.

My family, for everything.

Dr. Raymond Sedwick, for all the advice.

Dr. David Akin, for the rest of the advice.

Dr. Derek Paley, for being on my committee, despite never having met me before.

DARPA, for paying a contractor to pay a subcontractor to pay another subcontractor to pay Dr. Sedwick to pay the department to pay the university to pay me to write this thesis.

Holy Apostles Orthodox Church, for providing non-thesis-related work to do now and then.

Typeset with LaTeX 2_ε; figures and calculations done with MATLAB R2007a.

Contents

List of Figures

List of Symbols

a	Acceleration (m/s^2)
A	Cross-sectional area (m^2)
a_{cons}	Acceleration due to constant external force (m/s^2)
a_{min}	Minimum acceptable acceleration (m/s^2)
a_{sat}	Orbit semi-major axis of satellite (km)
AWG	Wire gauge
α	Thermal diffusivity (m^2/s)
α_{Al}	Thermal diffusivity of aluminum (m^2/s)
α_{ins}	Thermal diffusivity of insulation (m^2/s)
$\vec{B}_2$	Magnetic field from coil 2 (T)
c	Speed of light ($299, 792, 458m/s$)
C	Capacitance (F)
c_p	Specific heat ($J/kg \cdot K$)
d_{in}	Distance within which the system switches on (too close) (m)
d_{out}	Distance outside of which the systems switches on (too far) (m)
d_{sep}	Separation distance between two satellites (m)
ϵ	Emissivity
ϵ_o	Permittivity of free space ($8.85419 \cdot 10^{-12} F/m$)
η_{max}	Maximum achievable power transfer efficiency
f	Frequency (Hz)
$\vec{F}$	Force vector (N)
Γ_D	Device (target) decay rate
Γ_S	Source decay rate
Γ_W	Load decay rate
h	Coil height (m)
I	Current (A)
$\mathbf{I}$	Moment of inertia tensor ($kg \cdot m^2$)
$\vec{J}$	Current density (A/m^3)
J_2	J_2 perturbation constant ($1.083 \cdot 10^{-3}$)
κ	Coupling constant
l	Total length of wire (m)
L	Inductance (H)
m	Mass (kg)
M	Effective mutual inductance ($H \cdot s$)

m_{bat}	Mass of battery (kg)
m_{bus}	Mass which is not part of the EMFF system (kg)
m_{ins}	Insulation mass (kg)
m_{prop}	Mass of propulsion system (kg)
m_{sol}	Solar panel mass (kg)
m_{total}	Total satellite mass (kg)
m_{wire}	Wire mass (kg)
$\vec{\mu}_i$	Dipole moment of coil i ($A \cdot m^2$)
μ_o	Magnetic constant ($4\pi \cdot 10^{-7} N/A^2$)
$\mu_\oplus$	Standard gravitational parameter of the Earth (398600 km^3/s^2)
N	Number of turns of wire in loop
q_o	Electric charge (C)
$\dot{q}$	Total heat flow (W)
$\dot{q}_{cond}$	Conductive heat flow (W)
$\dot{q}_{rad}$	Radiative heat flow (W)
$\dot{q}_{res}$	Heat input from resistive heating (W)
$r(x)$	Distance to the center of the wire from point x (m)
$\vec{r}$	Position vector (m)
$\hat{r}$	Direction from one satellite to another
R	Resistance (Ω)
r_{loop}	Loop radius (m)
R_o	Ohmic resistance (Ω)
R_r	Radiative resistance (Ω)
r_{wire}	Wire radius (m)
$R_\oplus$	Radius of the Earth (6378 km)
ρ	Mass density (kg/m^3)
ρ_{Al}	Mass density of Aluminum (kg/m^3)
ρ_{bat}	Energy density of battery (J/kg)
ρ_e	Electrical resistivity (Ωm)
ρ_{ins}	Mass density of insulation (kg/m^3)
ρ_q	Charge density (C/m^3)
ρ_{sol}	Solar panel mass density (kg/m^3)
ρ_{th}	Thermal resistivity (mK/W)
σ	Boltzmann Constant ($1.38065 \cdot 10^{-23} J/K$)
t_{on}	Maximum time magnetic coils can be on (s)
t_{off}	Minimum time coils must be off after t_{on} (s)
$t_{on,dark}$	Maximum time shadowed per orbit with the coils on (s)
T	Temperature (K)
T_c	Background temperature (K)
Δt	Time step (s)
ΔT	Temperature difference (K)
th_{ins}	Insulation thickness (m)
τ	Time constant (s)
$\vec{\tau}$	Torque vector (Nm)
$\vec{v}$	Velocity vector (m/s)

$\vec{v}_t$	Velocity vector at time t (m/s)
${}^1\vec{v}_2$	Velocity of satellite 2 with respect to satellite 1 (m/s)
V	Voltage (V)
v_{rel}	Relative velocity between two satellites (m/s)
$\Delta\vec{v}_{cons}$	Change in velocity due to constant external force (m/s)
$\Delta\vec{v}_{J2}$	Change in velocity due to J_2 perturbation (m/s)
ω	Angular frequency (rad/s)
$\vec{\omega}$	Angular velocity (rad/s)
$\vec{x}$	Location vector (m)
$\vec{x}_t$	Location vector at time t (m)
${}^1\vec{x}_2$	Position of satellite 2 with respect to satellite 1 (m)
$\bar{x}$	Location of the center of mass of the system (m)
Δx	Distance step (m)

Chapter 1

Introduction

1.1 Electromagnetic Formation Flight

There are many situations in space where it is useful to keep a small group of satellites in a relatively close formation. A good example of this is a satellite cluster behaving as a single modular satellite, but physically separated. This could be done in order to minimize inert connecting structure, as in a space-based interferometer, or for ease of switching out non-functioning components. For example, a weather satellite cluster whose infrared camera satellite had broken could simply replace the infrared camera satellite without having to replace the perfectly functioning visible light camera satellite, or the high-gain radio transmitter satellite. There are many such applications.

The problem with these clusters is that it takes a lot of fuel for station-keeping. Orbital mechanics dictates that satellites at different distances from the Earth, even very slight differences, will almost always, over time, drift apart. On top of this, the J_2 perturbation tends to scatter satellites which are together. If propellant is used to counteract this drift, the propellant will run out eventually, rendering the satellite, along with the considerable investment inherent in its construction and launch, completely useless.

What satellite clusters need is some way of keeping the satellites together without expending propellant. In principle, tethers could be used; in tension, they could keep the satellites from flying apart. Unfortunately, tethers are useless in compression or shear, and have problems of their own in implementation. Their use is not straightforward, also, if the formation needs to change. However, magnetism could, in principle, be used to maintain the formation.

Electromagnets are capable of pulling things (such as satellites) toward each other, and in addition, can push things apart (for collision avoidance) by means of inverting one dipole. Also, they can interact with either the Earth's magnetic field or other satellites to generate torques, and in concert with reaction wheels, they can generate usable transverse forces.

The Earth's magnetic field adds additional complexity. While there is little translational effect,[1] the torque effects from the Earth's magnetic field must be taken into account; thus torque rods are currently in use on some satellites for attitude control.[2]

If a satellite has loops of wire, these loops can be used to generate controllable magnetic dipoles. If two such satellites are lined up axially, they can attract or repel each other. If they are lined up off-axis, there is a torque. If the torque is canceled out by reaction wheels, an off-axis translational force remains. Thus, electromagnetism can be used to move the satellites relative to each other in any way.

A cluster of satellites equipped with controllable magnetic coils can be operated in such a way as to remain in formation using only magnetic forces. This is Electromagnetic Formation Flight, or EMFF.

[1]Translational force from dipoles scales according to the inverse fourth power; torque scales according to the inverse cube. (Sedwick *et al.*, 2005, 4)

[2]Sedwick *et al.*, 2005, 3

1.2 Previous Research

Much of the work on EMFF has been done at MIT. A significant summation of the state of Electromagnetic Formation Flight research is in a report by Sedwick, Miller, et. al.[3] In it, the practicality and utility of EMFF, for a wide variety of applications, is demonstrated.

Aya Sakaguchi, in her 2007 thesis[4], studied micro-EMFF systems (EMFF systems for satellites smaller than about 100 kg), finding that they are practical for keeping small groups of satellites in close proximity. She implemented an algorithm for finding the optimum mass configuration for a μ-EMFF system on a small satellite, which served as a starting point for chapter two of this thesis.

Kwon[5], in his master's thesis, applied EMFF to a deep-space interferometer. He has also done work on the utility of superconducting wire, including thermal analysis.

Work has also been done on EMFF dynamics and control. Schweighart[6] and Elias[7], in particular, did work on dynamics and control of spacecraft using EMFF. The analysis is in greater depth than that which is in chapter four of this thesis, but the control methodology is quite different, as Elias and Schweighart do not limit satellites to on-axis interaction.

Chapter 3 of this thesis builds on work done at MIT on power transfer through coupled magnetic resonance. A paper published in *Science*[8] contains the key parts of their work, on which chapter three depends.

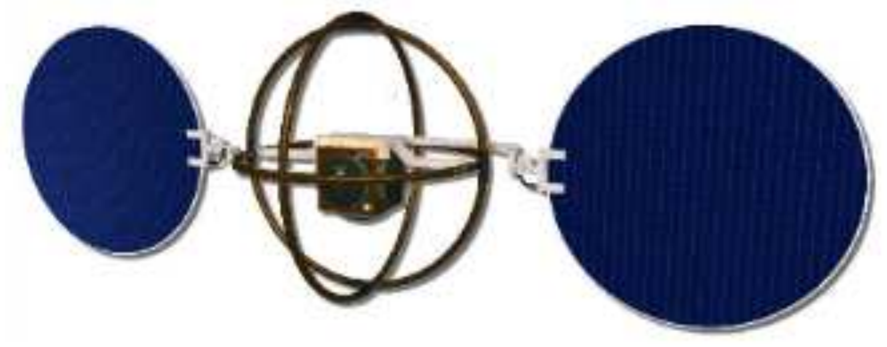
[3]Sedwick *et al.*, 2005
[4]Sakaguchi, 2007
[5]Kwon, 2004
[6]Schweighart, 2005
[7]Elias, 2004
[8]Kurs *et al.*, 2007

1.3 Research Goals

This thesis examines three aspects of Electromagnetic Formation Flight. First, in chapter 2, a minimum-mass design was found for a micro-EMFF system. The variables involved in this, both geometric and electrical, were studied, and their trends discussed.

The second portion of this research, chapter 3, deals with power transfer by inductive coupling. In both the power transfer and in EMFF, helical antennas are used. Due to similarity of antenna shapes, it has been suspected that the same coils used to accelerate satellites could be used to transfer power between them. Using mathematical models, the feasibility of such actions was evaluated. The power transfer efficiencies can then be factored in to the previous mass optimization as an additional output variable.

The final problem, chapter 4, is an evaluation of the paired satellites technique for using EMFF systems. In this technique, rather than dealing with complex interactions between multiple off-axis dipoles, only two satellites at a time activate their dipoles, and activate them parallel to each other. Two-body problems are always more straightforward than three- or four-body problems, and the dipoles being on-axis eliminates torque from consideration. However, it is important to ensure that it will suffice to maintain formation against the J_2 perturbation, caused by irregularities in the mass distribution of the Earth, and against one satellite having an external thrust, such as a rocket motor.



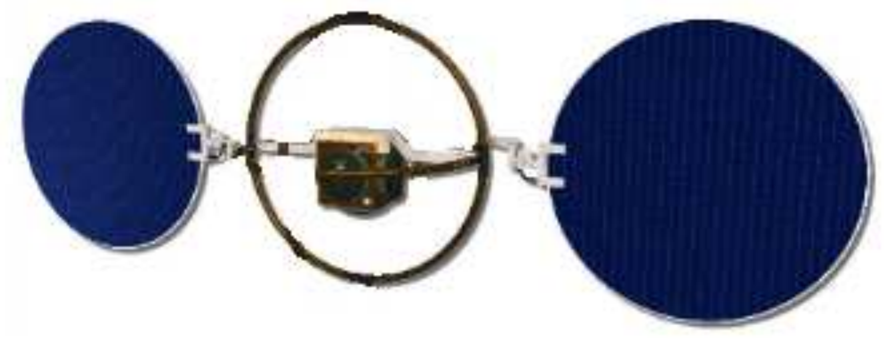


Figure 1.1: Artist's impression of EMFF satellites (Sedwick *et al.*, 2005). Top: satellite with three orthogonal coils; able to create dipole in any direction (assumed in chapter 4). Bottom: satellite with one coil; must physically rotate to change dipole orientation (assumed in chapters 2 and 3).

Chapter 2

Optimization

2.1 Goals

One important part of the space field which could benefit from Electromagnetic Formation Flight is small satellites, on the order of 100 kg. ("Micro-EMFF" being EMFF on satellites of this scale.) Their size precludes significant fuel reserves. Due to their cheaper launch and replacement costs, they are at an ideal scale for satellite clusters. However, without the fuel reserves for station-keeping, any close clusters will drift apart eventually. Electromagnetic Formation Flight has the potential to solve this problem.

Small satellites present some rather stringent limitations on the EMFF system mass. The mass of the coils, the insulation, the solar panels (if that is to be the power system used), and any batteries that are required, needs to be low in order to not drive up the total mass unnecessarily, but large enough to effectively maneuver the satellites.

The coils have to be sized so as to provide sufficient ΔV capability, while minimizing power consumption and coil mass. Since the micro-EMFF system is mass-limited, low-temperature superconductors will not be available to reduce power consumption, as the cryogenic systems would also draw power and add mass, in addition to taking up volume.

2.2 Methodology

2.2.1 Design Space

The design of a micro-EMFF system involves quite a few variables. The most straightforward are the geometric parameters — the radius of the wire loop, the radius of the wire itself, the thickness of the wire insulation, and the number of turns of wire in the coil. For this thesis, it was assumed that the loop radius is set by the design of the satellite. A one-meter radius was selected, that being a round number at about the right order of magnitude. The insulation thickness is a function of the wire radius (see equations 2.3 and 2.4 below), but the wire radius itself and the number of turns can vary freely.

Connected with the geometric variables are the electric variables: resistance, current, and voltage. Resistance can be easily found from the geometry — loop radius (r_{loop}), wire cross-section (A), and electrical resistivity (ρ_e) — as in equation 2.1,

$$R = \frac{r_{loop}\,\rho_e}{A} \tag{2.1}$$

and resistance with current, of course, determines the voltage required according to Ohm's Law, $V = IR$. In similar manner, the power requirements can be found. Driving all this is the current variable, which can vary, like the wire radius and number of turns.

The easiest and most reliable power system for long-term missions in Earth orbit is solar power; photovoltaic panels are a mature and cost-effective technology, and have been proved reliable on many satellites throughout the last few decades.

They provide a quantity of power that is appropriate for most space missions of this scale. However, eclipses complicate the matter of the panels: for a satellite orbiting Earth (especially in low orbit) there will be significant periods of eclipse. Because of this, it is necessary to have power available that is not directly from solar panels; thus this micro-EMFF system will include both solar panels and small batteries to power the system when the satellite is in eclipse. It is necessary to keep these systems as small as possible, to minimize satellite mass.

In addition, there are thermal issues which must be taken into consideration. As current flows through the wire bundle, the wire heats up. If the current is on for too long, the temperature may reach the melting point of the wire insulation. For sustainable operation of the system, it is necessary to first calculate the maximum length of time that the coils may be activated, and to calculate how long the wires take to cool off afterward. These calculations will be discussed in greater detail in section 2.2.3.

The variables which relate to how the EMFF system is used are separation distance and mass. The first is the average length between satellites during the operation of the EMFF system. Separation distance could easily change over the course of even one pulse, but a typical value is used to ensure that sufficient accelerations can be produced. Since force falls off quite rapidly with increasing distance, this typical value is approximately an upper bound to separation distance.

The geometry and power requirements yield the mass of the propulsion system. Meanwhile, once the system is switched on, a force will be generated; using the geometry and power, this force will be calculated, but in order to find out if the acceleration is sufficient, the total mass is needed, including propulsion mass, structural mass, any reaction wheels, or other such non-EMFF maneuvering devices, and the mass of the payload and supporting equipment.

Since the total mass of the spacecraft is the sum of the the EMFF system mass and spacecraft bus mass (consisting of everything which is not directly EMFF-related),

$$m_{total} = m_{prop} + m_{bus} \tag{2.2}$$

and m_{prop} is known, then if either m_{total} or m_{bus} is specified, the other can be found. For the purposes of this optimization, it was assumed that the spacecraft bus mass would be known already, the final mass needing to be calculated. The alternative would be to have the total launch mass known, with the mass constraints for the spacecraft bus to be calculated. In either case, the necessary work consists of a simple addition or subtraction; the choice depends on which input datum is available. For this thesis, the decision was arbitrary, since no specific design constraints were given.

2.2.2 Procedure

The optimization begins with the creation of an three-dimensional array of potential design points. Six different current levels, eight numbers of turns, and eight wire radii are combined into a 6 X 8 X 8 array of 384 possible combinations. These combinations are summarized in table 2.1.

Wire radius varies from the approximate equivalents of AWG 0 to 18. This range covers wires large enough to carry fairly large currents, but not so large as to be unwieldy. The number of turns of wire varies from 20 to 160, as larger numbers of turns, especially with the larger wire gauges, make for overly-heavy bundles of wire. Current varies from 25 to 50 amps, which is large enough to make thorough use of the smaller gauge wires, but not so large as to render them useless.

The insulation thickness is calculated for each point from the wire radius, by way of the wire gauge, as in equations 2.3 and 2.4.[1]

$$\text{AWG} = 36 - 39 \cdot \frac{\log\left(\frac{inches}{.005}\right)}{\log(92)} \tag{2.3}$$

[1]Sullivan, 1999

Table 2.1: Design Space

Variable	Number studied	min	max
Wire radius	8 radii studied	0.5 mm	4 mm
Loop radius	Fixed by design constraints	1 m (assumed)	
Number of turns	8 numbers studied	20	160
Current	Six currents studied	25 A	50 A
Voltage	Varies with current, wire and loop radii, and turns.		
Maximum on-time	Time until temperature reaches 360 K		
Required off-time	Time to cool back to 295 K afterward		

$$th_{ins} = 10^{0.518-\frac{\text{AWG}}{44.8}} \cdot 2.54 \cdot 10^{-5} \tag{2.4}$$

This being done, the maximum cycle time is calculated. To do this, a thermal model is run at each point, first heating up from room temperature (295 K) to the maximum temperature (set to 360 K, which is a typical value for the melting point of rubbers[2]), and then cooling back down to the starting point. This cycle time factors in to the calculations of the battery and solar panel masses.

Assuming a ninety-minute orbit, of which approximately one third is in the Earth's shadow, the satellite will be in this shadow for approximately 2000 seconds per orbit. From that time, the amount of time within one orbit in which the system is likely to be both on and in the shadow of the Earth can be calculated. The number of dark cycles is calculated by dividing the shadow time by the total cycle time, rounding up, and that is in turn multiplied by the amount of on-time per cycle.

$$t_{on,dark} = t_{on}\text{ceil}\left(\frac{2000\text{s}}{t_{on} + t_{off}}\right) \tag{2.5}$$

From the cycle times and wire geometry, the masses of the wire (eq. 2.6), insulation (eq. 2.7), battery (eq. 2.8), and solar panel (eq. 2.9) can be calculated for each potential design point:

$$m_{wire} = N \cdot (2\pi r_{loop}) \cdot (\pi r_{wire}^2) \cdot \rho_{Al} \tag{2.6}$$

[2]Mark, 1996

$$m_{ins} = (2\pi r_{wire}) \cdot \pi((r_{loop} + th_{ins})^2 - r_{loop}^2) \cdot \rho_{ins} \tag{2.7}$$

$$m_{bat} = \left(\frac{V^2}{R}\right) t_{on,dark} \frac{1}{\rho_{bat}} \tag{2.8}$$

$$m_{sol} = \frac{t_{on} - t_{on,dark}}{t_{off} + t_{on}} \left(\frac{V^2}{R}\right) \rho_{sol} \tag{2.9}$$

The total mass of the EMFF propulsion system is the sum of these, equation 2.10.

$$m_{prop} = m_{wire} + m_{ins} + m_{bat} + m_{sol} \tag{2.10}$$

2.2.3 Thermal Analysis

The maximum length of time the system can be on is dependent on how quickly the wire bundle reaches the melting point of the insulation. After this, some amount of time is required for the system to return to its original temperature. From the thermal characteristics of the wire bundle, the maximum heating time and required cooling time can be found using a numerical simulation.

Since axial symmetry simplifies calculations considerably, the wire bundle is modeled as a series of concentric rings, with metal at the center, surrounded by alternating layers of insulation and metal, with insulation always being the outermost layer. In addition, the coil is dealt with in cross-section, allowing the simulation to be one-dimensional. The number of layers of metal is calculated from the number of turns of wire, patterned after centered hexagonal numbers. This is illustrated in fig. 2.1. The resulting thermal profiles are illustrated in figs. 2.2 and 2.3.

Centered hexagonal numbers are "figurate numbers," which can be represented by dots making up a regular pattern. In the case of centered hexagonal numbers, they can be visualized as a filled hexagon on a triangular lattice (see fig. 2.1)[3]. One is the first such number, followed by seven (one surrounded by a hexagon of six, two

[3]Weisstein, 2008

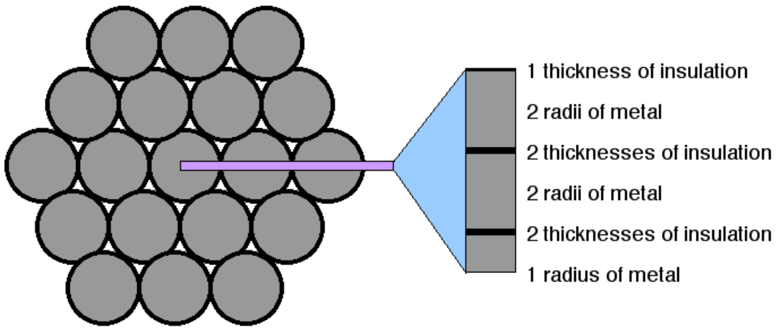

1 thickness of insulation
2 radii of metal
2 thicknesses of insulation
2 radii of metal
2 thicknesses of insulation
1 radius of metal

Figure 2.1: 1-d model for thermal analysis

on a side), followed by nineteen (seven surrounded by a hexagon of twelve, three on a side), and so on, according to the pattern $1 + 3n(n - 1)$. The number of layers is found by rounding the number of turns of wire to the nearest centered hexagonal number.

For the numerical simulation, the wire bundle must be split up into a finite number of distance steps, and time must progress in discrete time steps. If the distance steps are larger than the layers, vital detail is lost; if they are a small fraction of the layers, little detail is added, but the simulation time increases. It would seem, then, that the insulation thickness, being much smaller than the wire size, would be the distance step. However, the steps need not all be of identical size, so each layer, metal or insulation, is treated as one distance step for the numerical simulation.

Jaluria and Torrance[4] showed that in a 1-D finite difference model, as is used here, the maximum time step that can be used with a given distance step is $\Delta x^2/2\alpha$, where Δx is the distance step, and α is the thermal diffusivity of the material. The distance step over the insulation is th_{ins}, so the maximum time step found from the insulation is th_{ins}^2/α_{ins}. Meanwhile, in the wire itself, the minimum distance step is

[4]Jaluria & Torrance, 2003, p. 75

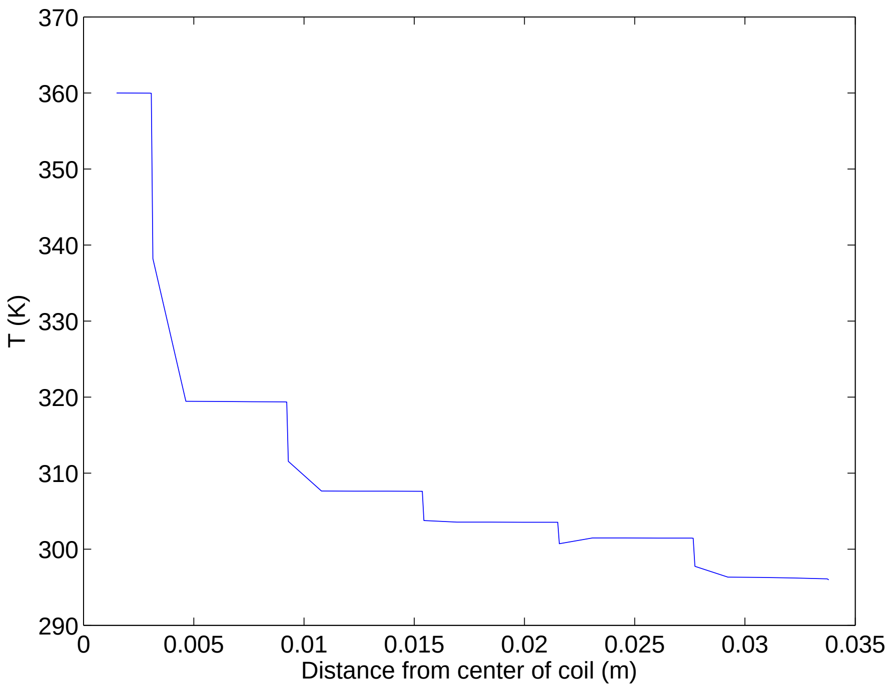


Figure 2.2: Thermal profile after one heating cycle; $N = 100$, $r_w = .003m$, $I = 40A$. Plateaus are the conductive regions, large slopes correlate with insulation. Apparent change in slope across insulation is an artifact of distance step size.

at the center, where the step is $r_{wire}/2$. The maximum time step found from the wire is $\left(\frac{r_{wire}}{2}\right)^2/\alpha_{Al}$. The lesser of these two maxima is used as the time step for the simulation. Whether this lesser maximum is from the metal or from the insulation depends on the wire radius used.

Radiation only applies to the outermost distance step. It is calculated according to equation 2.11.

$$\dot{q}_{rad} = \epsilon\sigma A(T_c^4 - T^4) \tag{2.11}$$

where T_c is the ambient temperature, ϵ is emissivity and σ is the Boltzmann constant.

Conduction, on the other hand, is addressed at every distance step. As implemented in equation 2.12, the temperature difference between adjacent points is divided by the thermal resistivity. At the innermost and outermost points, $T(i \pm 1)$

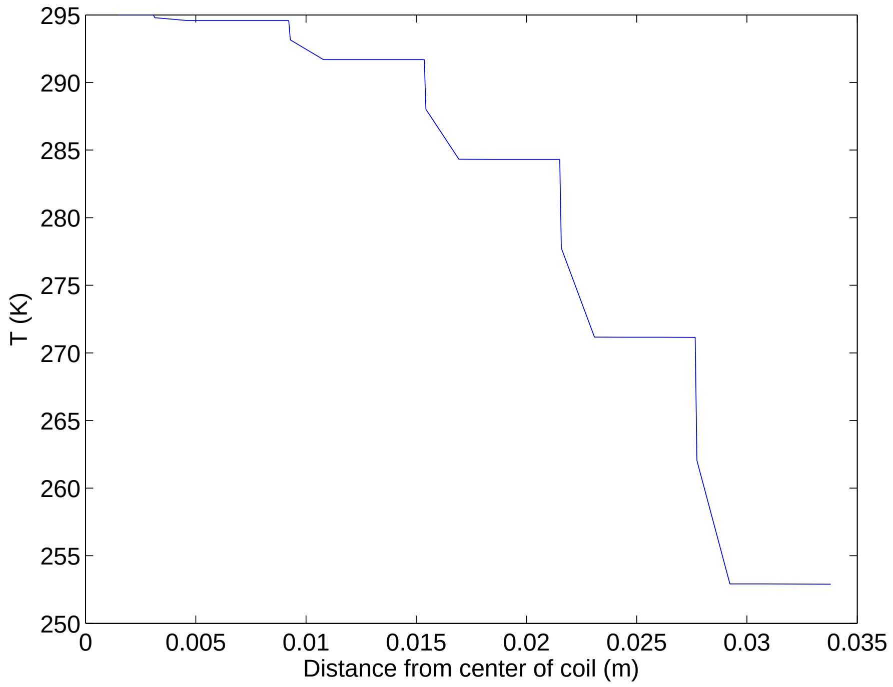


Figure 2.3: Thermal profile after fig. 2.2 followed by one cooling cycle.

is set equal to $T(i)$.

$$\dot{q}_{cond} = \frac{T_{i-1} - T_i}{\rho_{th,i-1}} + \frac{T_{i+1} - T_i}{\rho_{th,i}} \tag{2.12}$$

In the heating code, Joule heating is taken into account, and equal to the power loss over the wire due to resistance, as in equation 2.13, where $r(x)$ is the distance from a given layer to the center of the wire.

$$\dot{q}_{res} = I^2 R = I^2 \frac{r_{loop}\, \rho_e}{r(x)\Delta x} \tag{2.13}$$

Finally, the various heat flows are summed together, as in equation 2.14, and heat flow rates are converted into temperature changes, as in equation 2.15.

$$\dot{q} = \dot{q}_{rad} + \dot{q}_{cnd} + \dot{q}_{res} \tag{2.14}$$

$$\Delta T = \frac{\dot{q}\Delta t}{\rho A c_p \Delta x} \tag{2.15}$$

2.2.4 Optimization Calculations

With the propulsion mass (and cycle time) for each potential design point calculated, the total spacecraft mass is $m_{total} = m_{inert} + m_{propulsion}$.

The goal of the propulsion system being to counteract the J_2 perturbation, the minimum acceleration requirement can be found as in equation 2.16[5].

$$a_{min} = 3J_2 \left(\frac{R_\oplus}{a_{sat}} \right)^2 \frac{\mu_\oplus}{a_{sat}^3} d_{sep} \tag{2.16}$$

If the orbit is 500 km above the Earth's surface ($a_{sat} = R_\oplus + 500$ km), this comes out to

$$a_{min} = 3.4213 \cdot 10^{-9} d_{sep} \tag{2.17}$$

The system is on for $\frac{t_{on}}{t_{off}+t_{on}}$ of the cycle, so that time fraction is multiplied by the maximum acceleration to get the mean. The force from these coils[6] is equal to $\frac{\mu_o N^2 I^2 r_{loop}^2}{d_{sep}^2}$. That force, times the fraction of the time on, and divided by the mass, yields the acceleration.

$$a = \frac{t_{on}}{t_{off} + t_{on}} \frac{3\mu_o \pi N^2 I^2 r_{loop}^4}{2d_{sep}^4 (m_{bus} + m_{prop})} \tag{2.18}$$

For a given spacecraft bus mass and average separation distance, of all the potential design points that allow sufficient acceleration, the minimum mass solution — the lowest mass where $a > a_{min}$ — is selected.

[5]Sakaguchi, 2007, p. 27
[6]Sakaguchi, 2007, p. 91

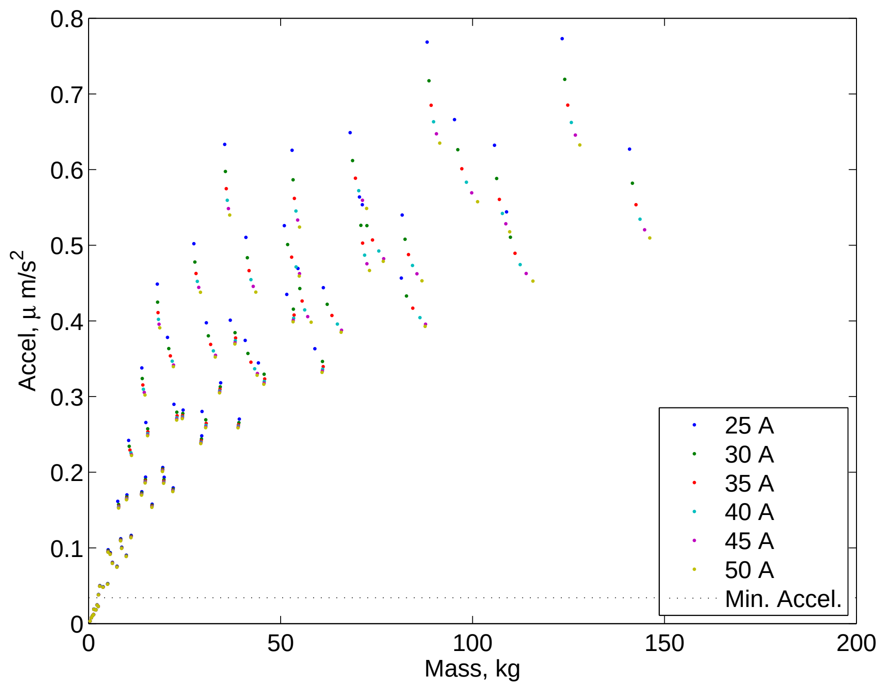


Figure 2.4: Acceleration capability by EMFF system mass. Vertical clusters have common numbers of turns and wire radii (see fig. 2.6). Compare to fig. 2.7.

2.3 Results

2.3.1 Trends in Spacecraft Geometry

Mass and Cross-sectional Area

The goal of the Electromagnetic Formation Flight system is to effectively propel the satellites relative to each other. Thus, acceleration capacity is desired. Since mass is expensive to launch, it is important to keep the mass low and the acceleration high. The first graphs generated examine this tradeoff.

Mass and acceleration are not related linearly. Figure 2.4 illustrates the increase of acceleration capacity with mass. At the cost of raising the mass somewhat, more acceleration can be squeezed out of the EMFF system. However, the larger the mass, the less additional acceleration you can get. Figure 2.5 clarifies this by showing the acceleration per kilogram of EMFF system mass; this ratio is the key

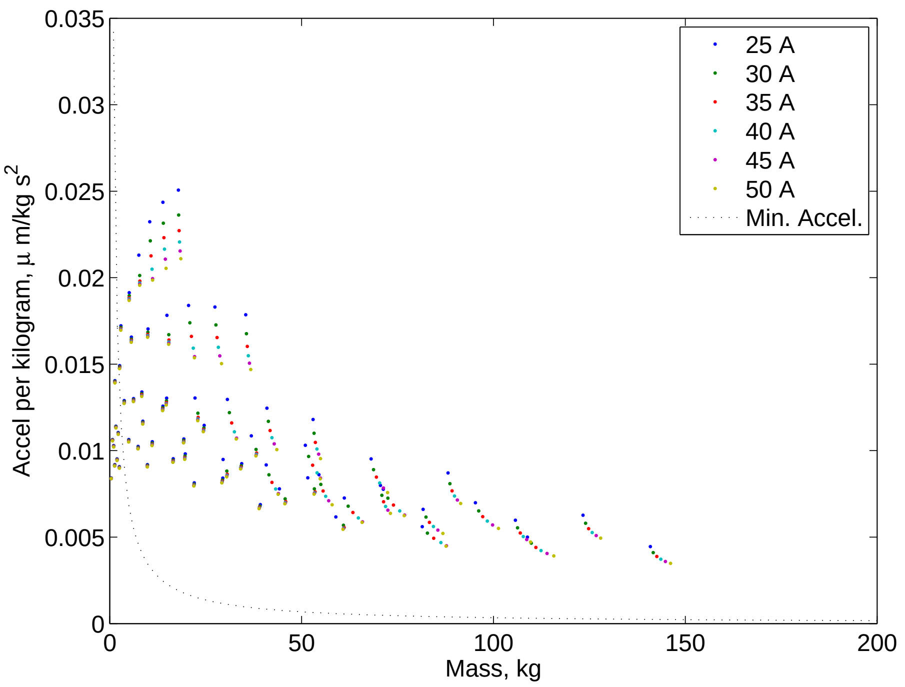


Figure 2.5: Acceleration capability per kilogram. Clusters have common numbers of turns and wire radii (see fig. 2.6).

tradeoff in EMFF design. The highest acceleration per kilogram comes with very small systems. In figures 2.4 and 2.5, and also in fig. 2.7, a dotted line indicates the minimum allowable acceleration for the baseline case, with a spacecraft bus mass of 50 kg and an average separation distance of ten meters between spacecraft.

As shown in fig. 2.6, the wire mass dominates the total mass enough that the cross-section of the coil is related to mass in an almost constant ratio, with only a little variation from power use, due to the various current levels given here. Because of this, fig. 2.4 can be easily modified into fig. 2.7, showing more clearly how the properties go in 'clumps' based on wire radius and the number of turns of wire in the coil.

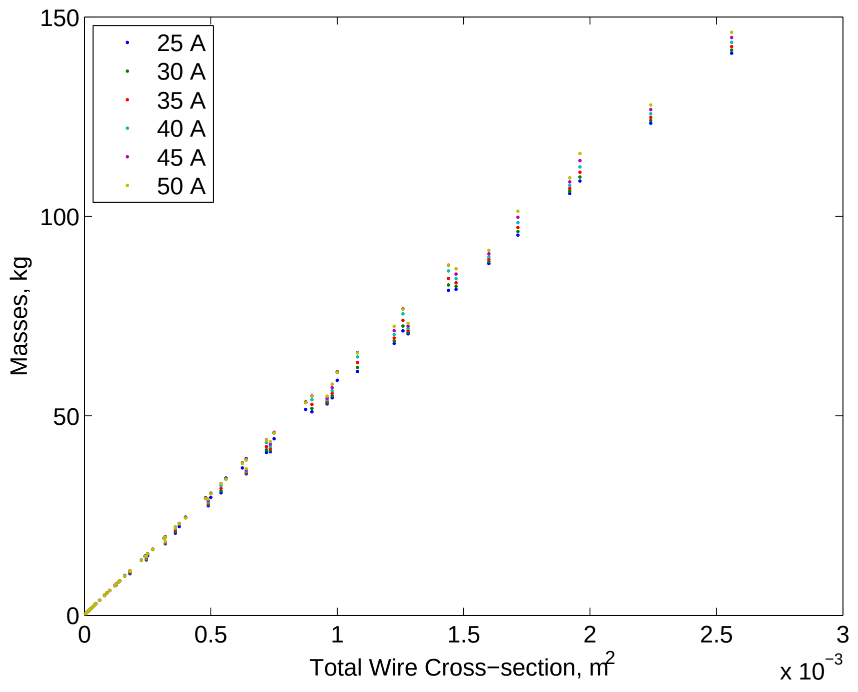


Figure 2.6: EMFF system mass by cross-sectional area of coil. Area is equal to $\pi r_{wire}^2 N$, and is directly proportional to wire mass.

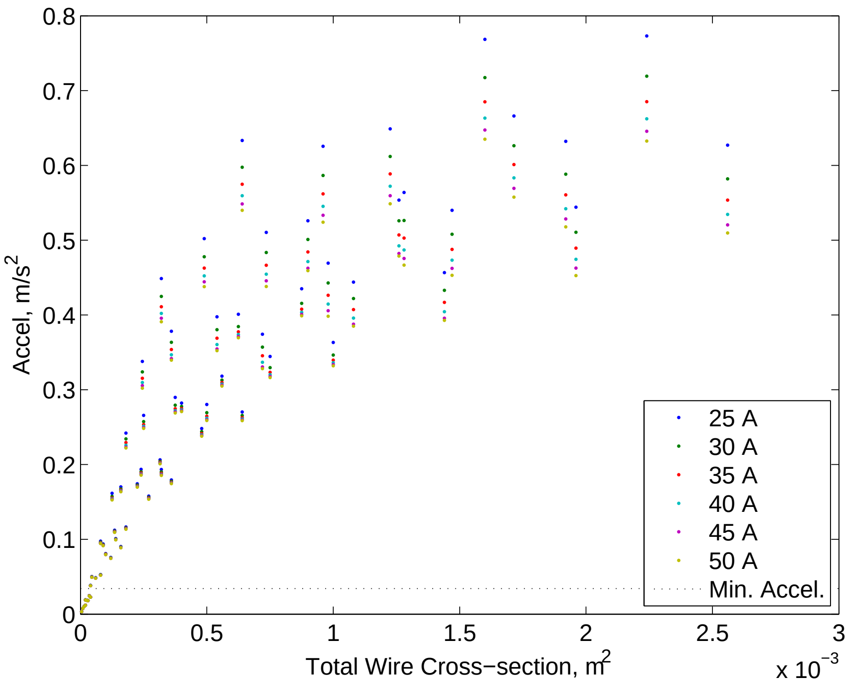


Figure 2.7: Acceleration capability by coil cross-sectional area. Compare to fig. 2.4.

Current

As a representative sample of trends from changing current levels, a data set was selected with the maximum wire radius and number of turns. This extreme end of the available data has the same trends which are visible elsewhere in the data, but more pronounced.

The thermal properties of the system vary based on the geometry, but vary even more with changing current levels. The most readily apparent consequence of increasing the amount of current flowing through the coil is the decrease — quadratically — of the amount of time that the system can remain on without the risk of the insulation melting (see fig. 2.8). As resistive heating increases by the square of the current, as in equation 2.13, this trend is to be expected.

Faster heating for the higher current levels tends to concentrate the heat toward the center of the coil. This concentration then takes longer to make it to the surface of the coil, and be radiated away. Thus the cooling time (fig. 2.9) follows the same pattern as the heating time, as does the fraction of the cycle the system may be on (figure 2.10), which is approximately equal to $\frac{t_{on}}{t_{off}}$. Since the quadratic shape of t_{off} is mitigated somewhat by the heating taking place throughout the coil, not just in the center, the quadratic quality of t_{on} comes through.

Somewhat curiously, there is a slight tendency for the lower current levels to produce higher masses, as seen in fig. 2.11. This is explained by the battery requirements. Though decreased power requirements lower the required battery mass, from the calculation used for the size of the battery, a longer t_{on} and a larger on-time fraction will increase the battery mass slightly more than the lower power requirements will decrease it.

Lower currents also allow a slightly larger average acceleration (see fig. 2.12). This is due to the longer periods of time the system can remain active; fig. 2.12 (acceleration) parallels fig. 2.10 (fraction of time on in a cycle) quite closely.

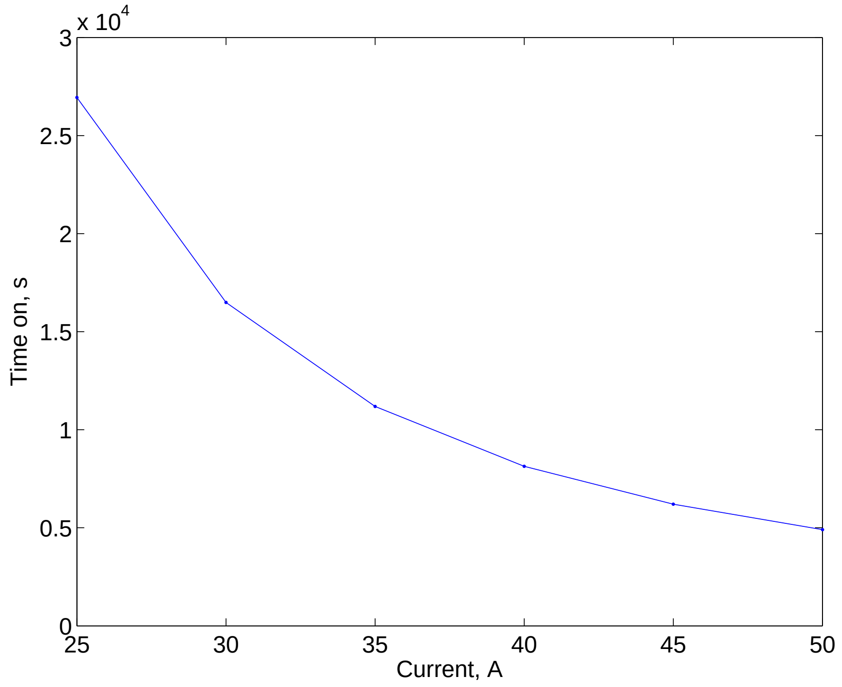


Figure 2.8: On time, 4-mm radius, 160 turns

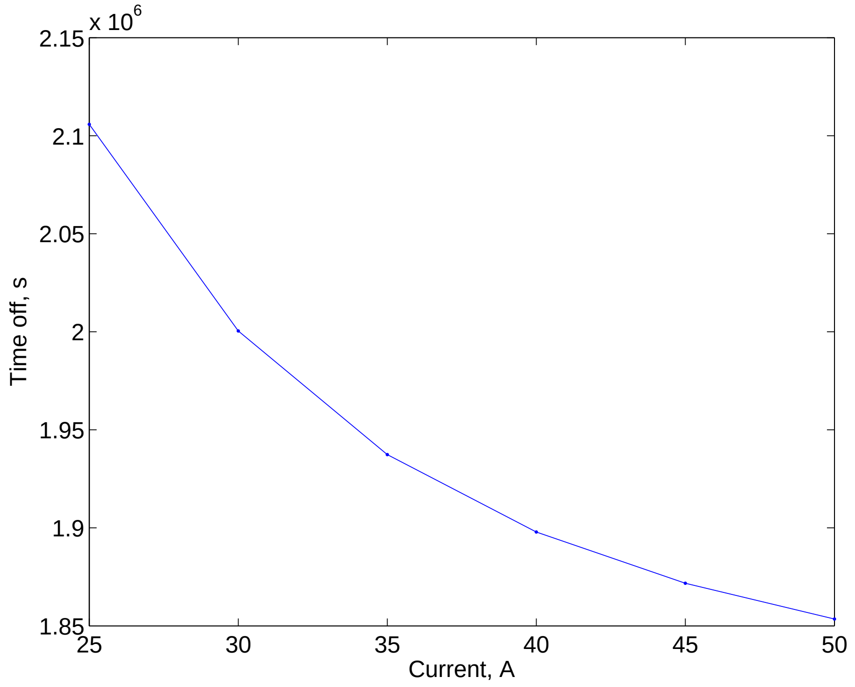


Figure 2.9: Off time, 4-mm radius, 160 turns

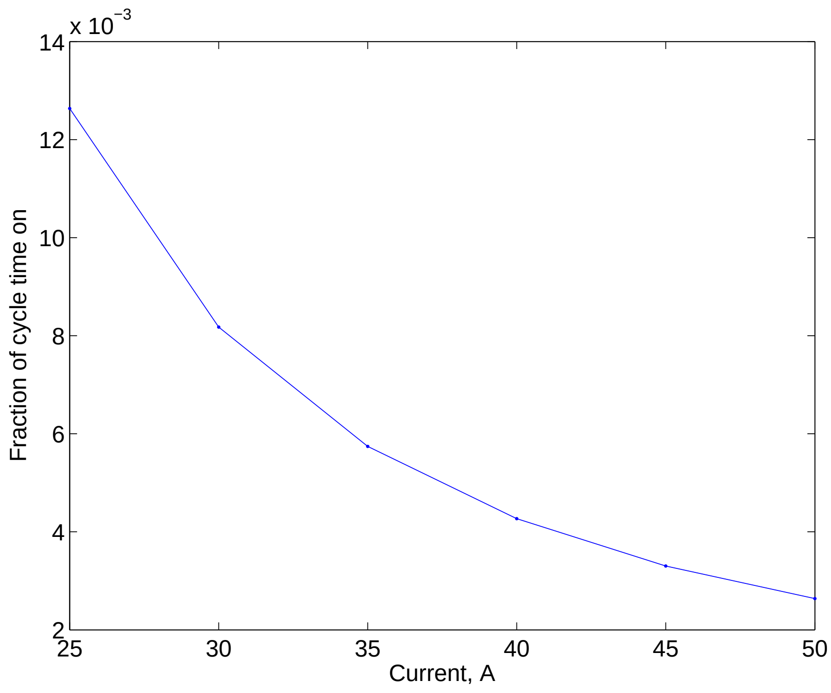


Figure 2.10: Fraction of cycle time on, 4-mm radius, 160 turns

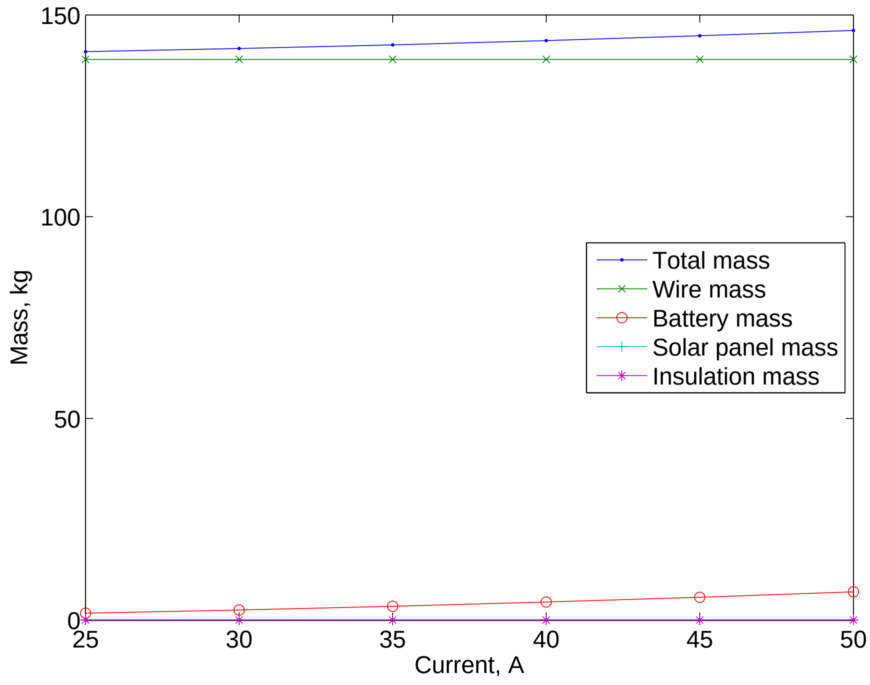


Figure 2.11: Mass, 4-mm radius, 160 turns. Solar panel and insulation masses are very small.

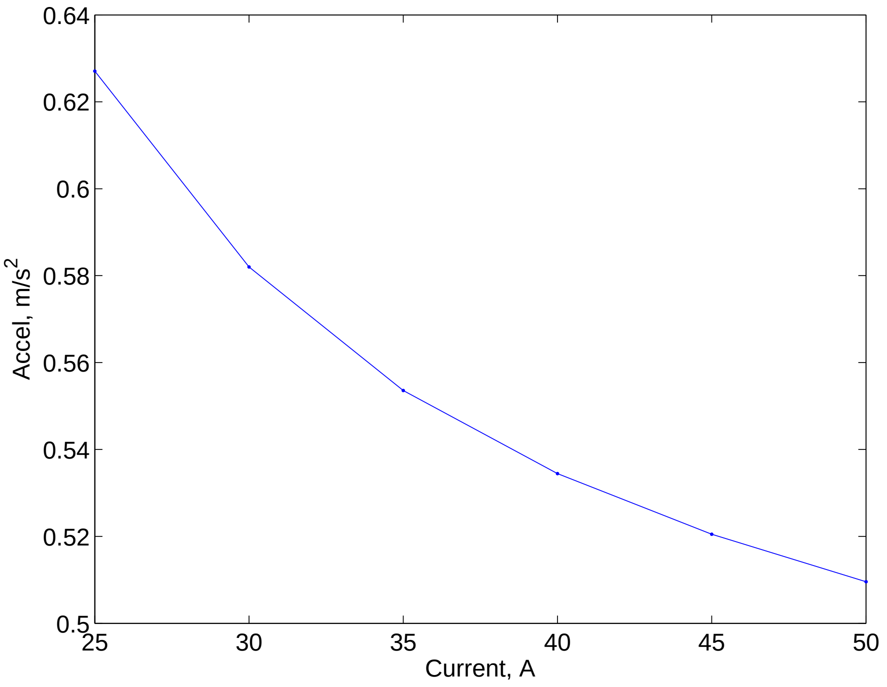


Figure 2.12: Acceleration capability, 4-mm radius, 160 turns

Number of turns

A section of the data was also examined at the high end of current and wire radius, with the number of turns of wire varying. The most interesting resulting variances were due to the geometric constraints of large bundles of wire — if they are put into an approximately circular bundle, increasingly large layers will be formed, as seen in fig. 2.1.

The more layers in the coil of wire there are, the longer it takes for the heat to move out of the coil through the layers. Thus, cooling takes longer as the number of coils increases, as in fig. 2.13. There are terrace-like regions of the graph; these are due to the wire being layered, as in the centered hexagonal numbers, above (see discussion on page 8).

With the time to heat, on the other hand, there is only a slight effect from the number of turns of wire (fig. 2.13). If there are few wires in the bundle, they can radiate a slightly larger percentage of the power away during the heating. The

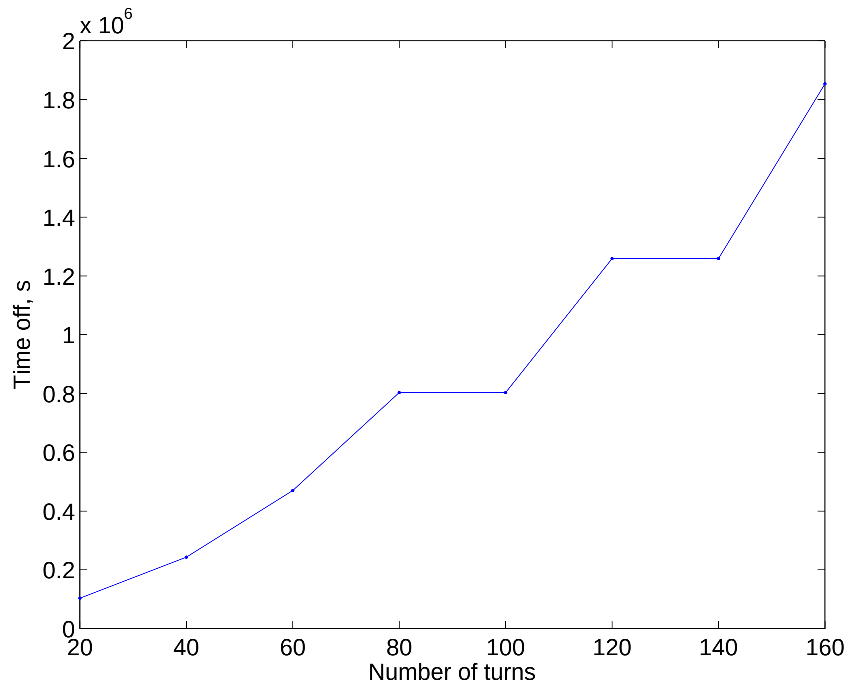


Figure 2.13: Off time, 4-mm radius, 50A

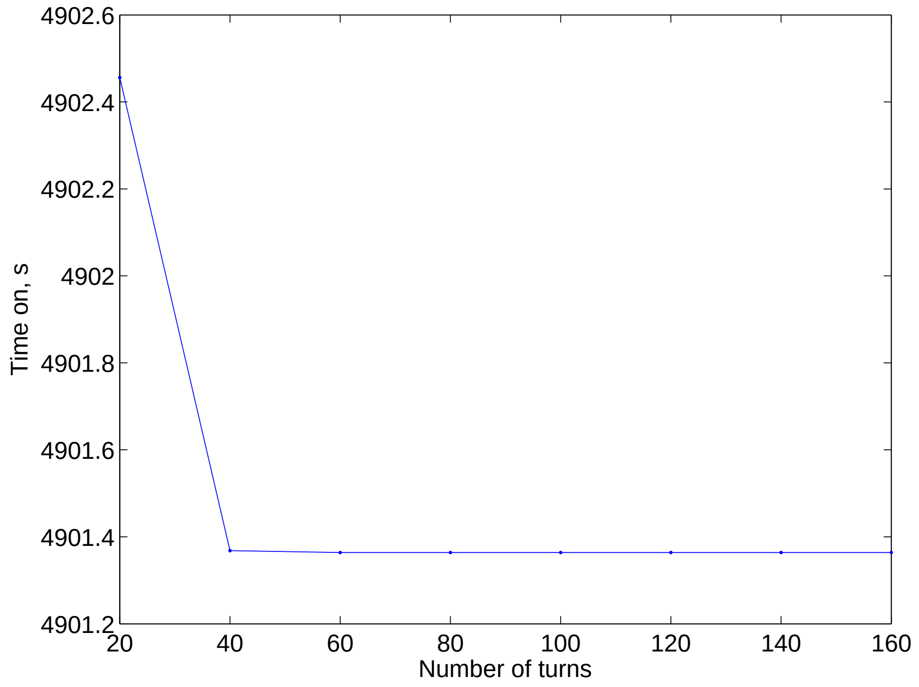


Figure 2.14: On time, 4-mm radius, 50A

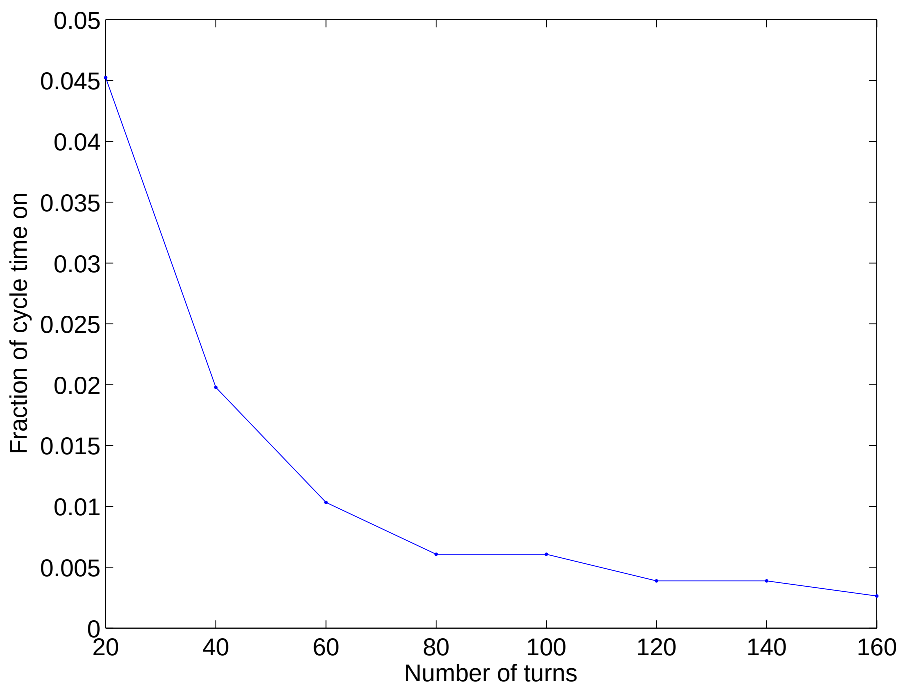


Figure 2.15: Fraction of cycle time on, 4-mm radius, 50A

difference, however, is almost negligible — a thousandth of a second out of several minutes total cycle time. Effectively, t_{on} is constant with respect to the number of turns.

Since t_{on} is nearly constant, and the fraction of the cycle in which the system is on is approximately equal to $\frac{t_{on}}{t_{off}}$, fig. 2.15, which shows this fraction, approximates the inverse of fig. 2.13.

The number of turns of wire is linearly related to mass, since the longer the wire, the heavier it is. This is linear, since only one dimension of the wire is being changed — the length. Likewise, the longer the wire, the more electrical resistance it has, so there is again a linear relationship with the power requirements. These can be clearly seen in fig. 2.16.

The relationship between the number of turns and the acceleration capability is quite complex, as seen in fig. 2.17. Comparing to fig. 2.13, which shows the 'terraces' where the same number of layers of wire are involved, it can be seen that

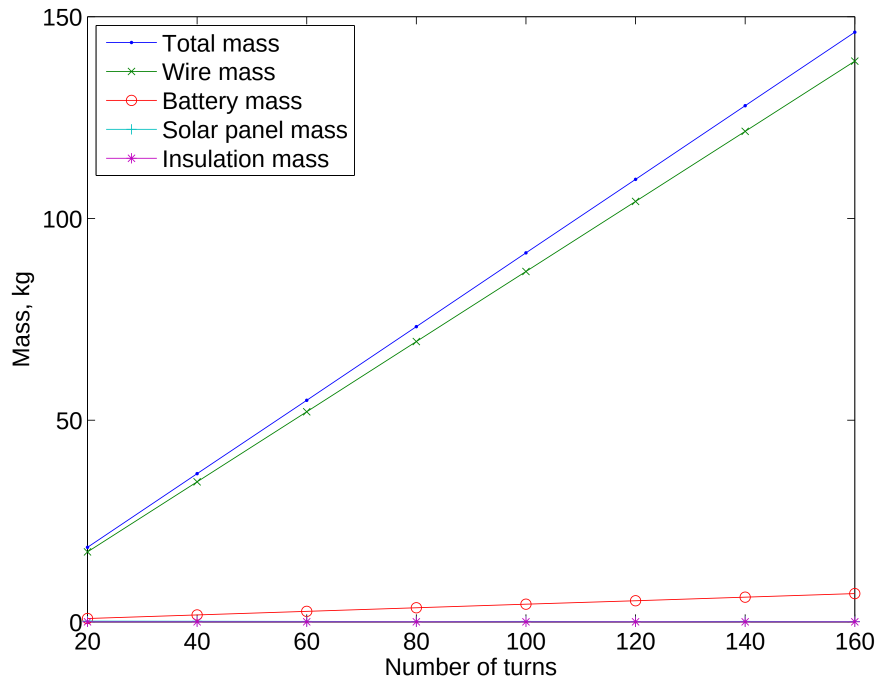


Figure 2.16: Mass, 4-mm radius, 50A. Solar panel and insulation masses are very small.

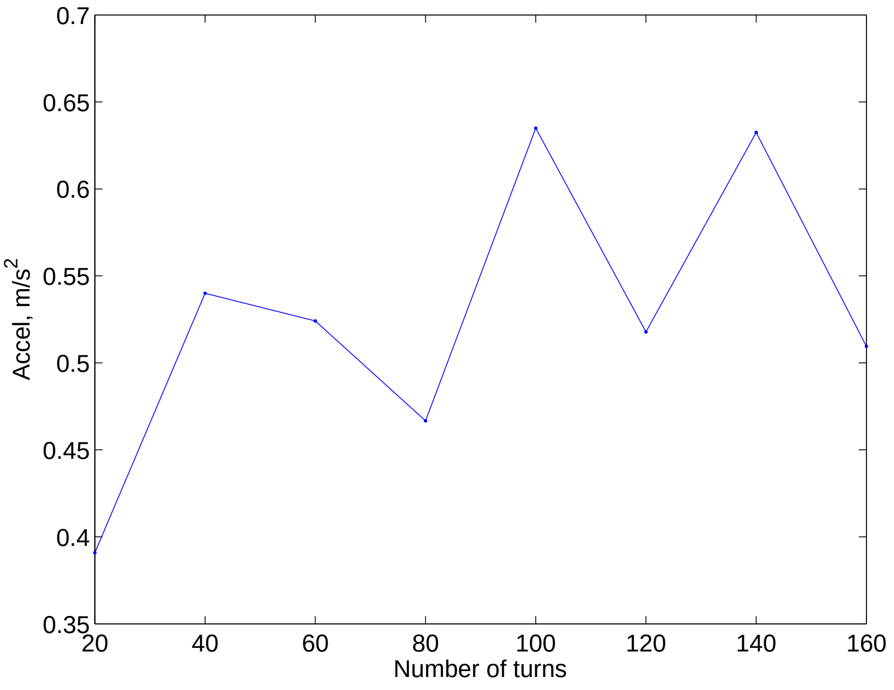


Figure 2.17: Acceleration capability, 4-mm radius, 50A

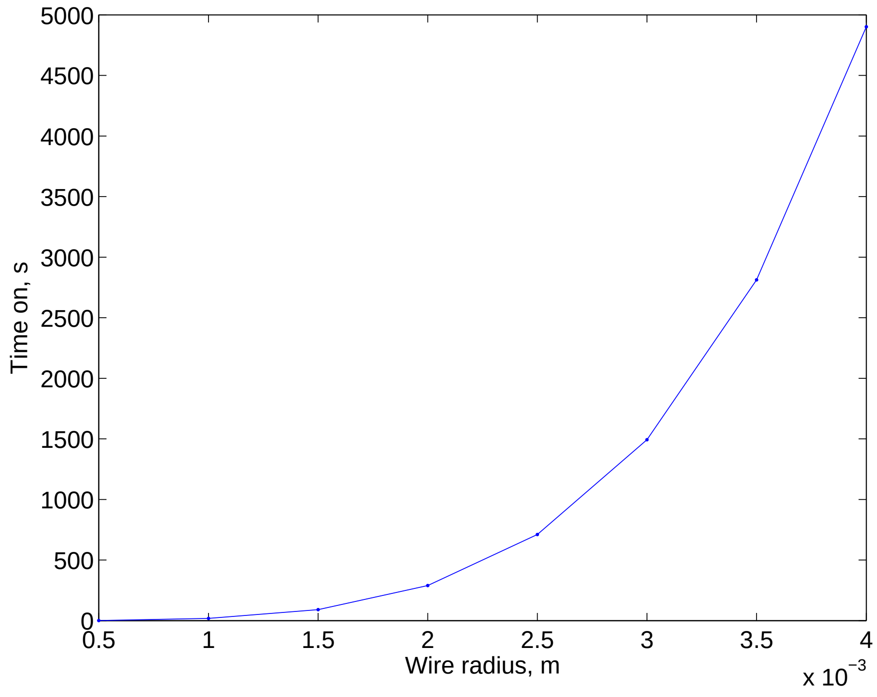


Figure 2.18: On time, 160 turns, 50A

the acceleration capacity decreases as a layer 'fills up', and increases with each new layer. The end result is not strongly varying — acceleration varies from only about 0.35 to 0.55 $\frac{m}{s^2}$ between 20 and 160 turns of wire.

Wire radius

Likewise, a sample of the data was taken with the maximum current and number of turns, varying the radius of the wire. The wire radius has a dramatic and regular effect on thermal properties, especially.

As the cross-sectional area of an individual wire increases, the time before it reaches the maximum temperature increases quadratically. In terms of the variables studied directly, t_{on} increases with the fourth power of r_{wire} (fig. 2.18). On the other hand, for the time to cool, the remnants of the heat buildup pattern are quadratic, since the cooling is dependent on the square of the distance to the outside of the wire, this distance being r_{wire} (fig. 2.19).

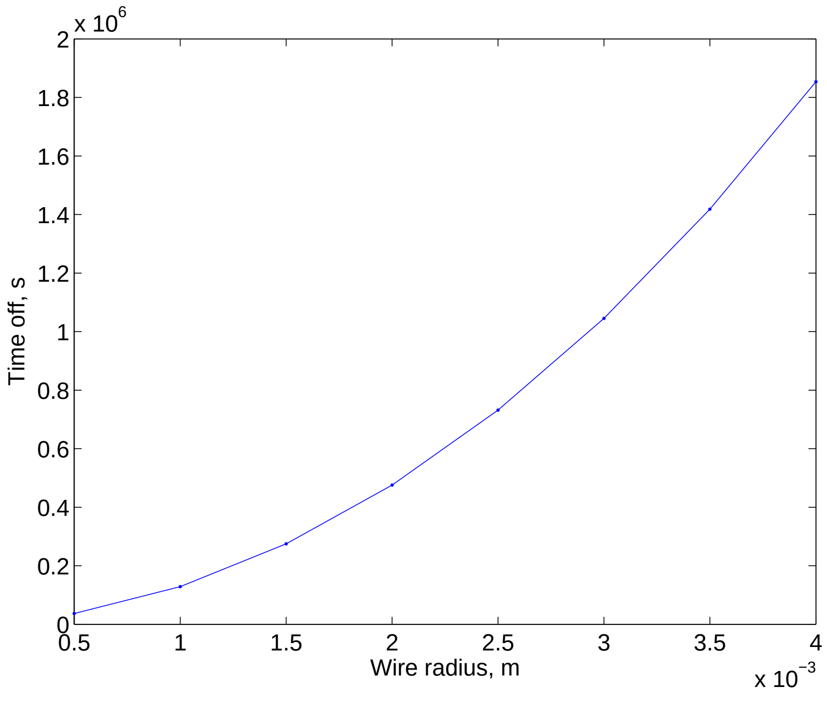


Figure 2.19: Off time, 160 turns, 50A

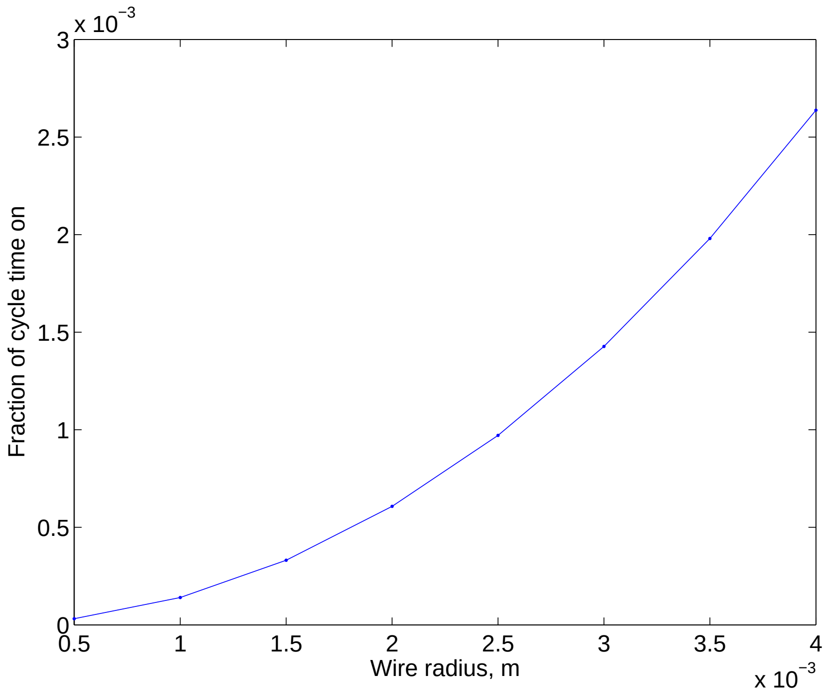


Figure 2.20: Fraction of cycle time on, 160 turns, 50A

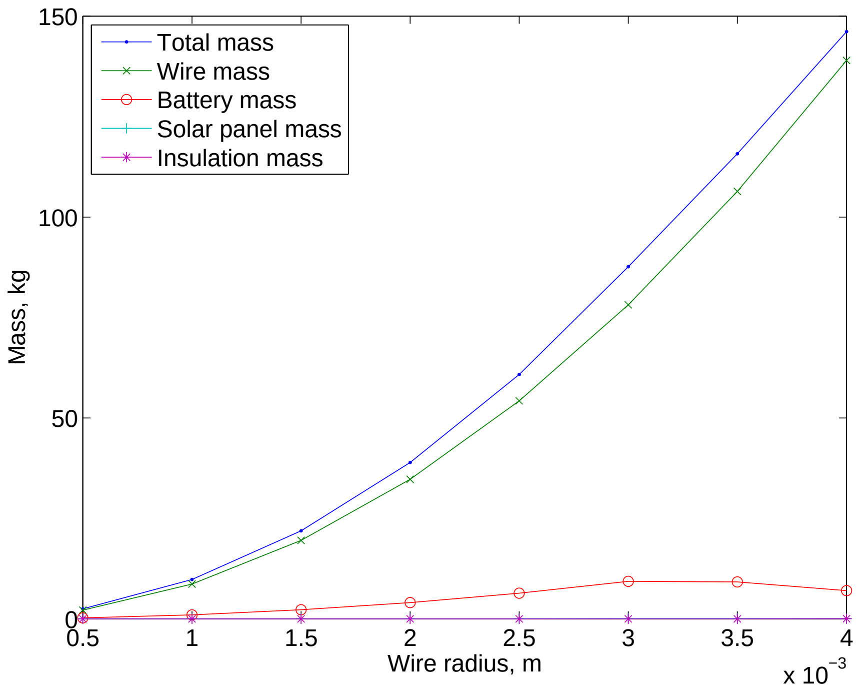


Figure 2.21: Mass, 160 turns, 50A. Solar panel and insulation masses are very small.

Since t_{on} is quartal, and t_{off} is quadratic, with respect to r_{wire}, the fraction of the cycle in which the system is on, $\frac{t_{on}}{t_{off}+t_{on}} \approx \frac{t_{on}}{t_{off}}$ (fig. 2.20) is quadratic. The curve is quite similar in shape to that of t_{off} (fig. 2.19).

The mass of the wires is proportional to the cross-sectional area, which is in turn proportional to the square of the radius. This is seen clearly in the quadratic shape of the green line in fig. 2.21. The red line, meanwhile, shows the increasing battery mass, as the power requirements also increase by the square of the radius.

As radius increases, the acceleration capability also increases, approximately linearly, as in fig. 2.22. It is not exactly linear, since the relationship between radius and acceleration is rather complicated. Equation 2.18 includes t_{on} in the numerator (proportional to the fourth power of r_{wire}), and t_{off}, and m_{prop} in the denominator. Both of the variables in the denominator are proportional to r_{wire} squared. Thus, despite some irregularity, acceleration varies linearly with wire radius.

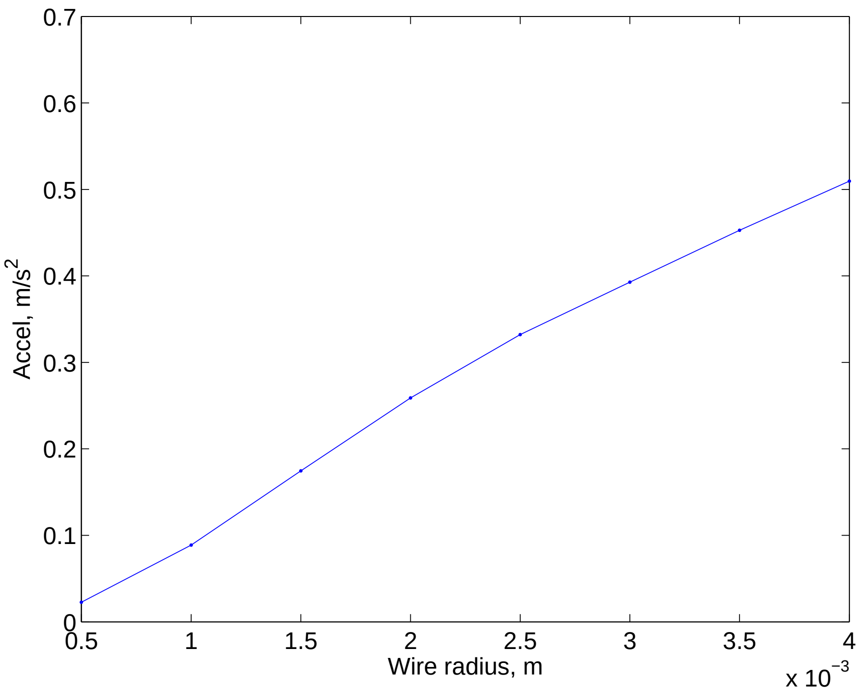


Figure 2.22: Acceleration capability, 160 turns, 50A

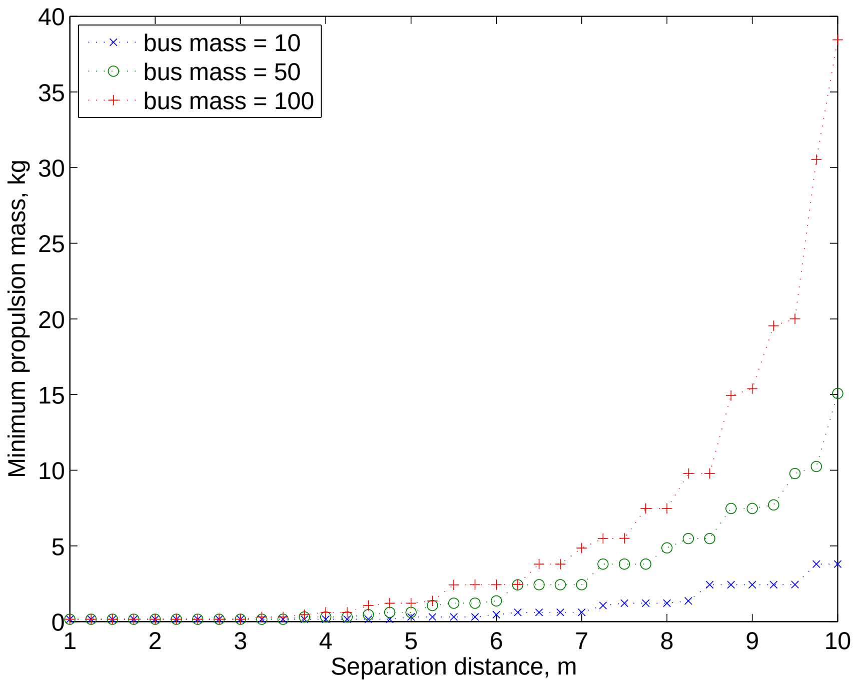


Figure 2.23: Minimum EMFF system mass for various bus masses and separation distances. See fig. 2.24 for the same data on a logarithmic scale.

2.3.2 Trends in Operational Parameters

In `findmultibest.m` (appendix A.4), the optimal design, of the 384 possibilities discussed above, is selected for spacecraft bus masses of 10, 50, and 100 kilograms, and average separation distances between 1 and 10 meters. For each mass and separation distance, the acceleration capacity is found for each of the design points, and the lowest-mass solution which is sufficient to counteract the J_2 perturbation is selected.

As expected, as the separation distance increases, so too does the necessary mass of the propulsion system, proportional to the fourth power of the distance (fig. 2.23; fourth-power effects more clearly seen in fig. 2.24). Likewise, increasing bus mass increases EMFF mass, approximately linearly. The relation between bus mass and EMFF mass is clearer in fig. 2.25, which relates distance to mass fraction. The mass fraction is relatively constant across the three bus masses, with the exception

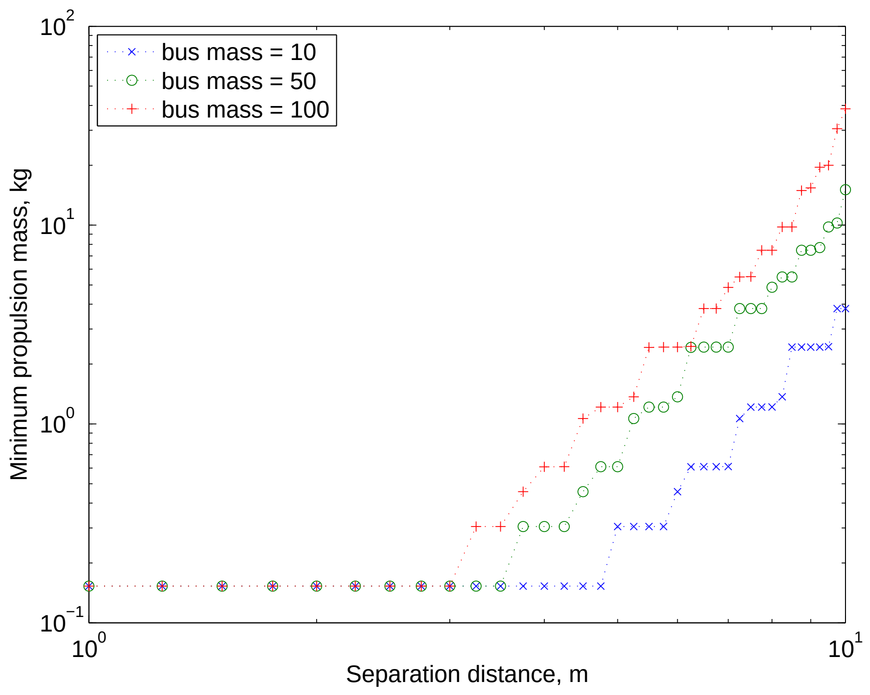


Figure 2.24: Minimum EMFF system masses, logarithmic scale. Fig. 2.23 has the same data on a linear scale.

of the 10-kg case. With the small mass, even a minimally-sized EMFF system is going to be a rather sizable fraction of the total. The graphs show mass in a stair step pattern; because there are only 384 design points studied, there are only 384 possible masses. Since the best design points can come up more than once, there tend not to be a large variety in masses.

In figure 2.26, each parameter is looked at individually. As discussed in section 2.3.1, acceleration capacity is dependent mainly on wire radius, while mass is based on wire radius and the number of turns. Current and number of turns do not vary much across the different separation distances and bus masses; mainly the wire radius changes.

On the whole, the EMFF mass and acceleration clusters around each discrete radius, as seen in fig. 2.4 with the acceleration cutoff coming between clusters. However, occasionally, it will cut off the list in the middle of a radius cluster, and thus a different number of turns or current level will be chosen.

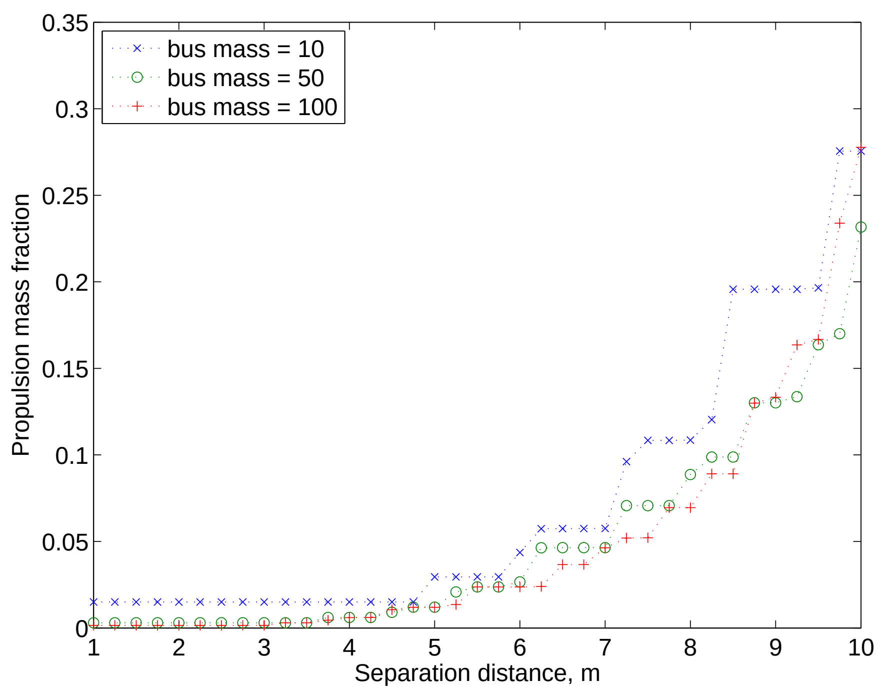


Figure 2.25: Minimum EMFF mass fraction for various bus masses and separation distances

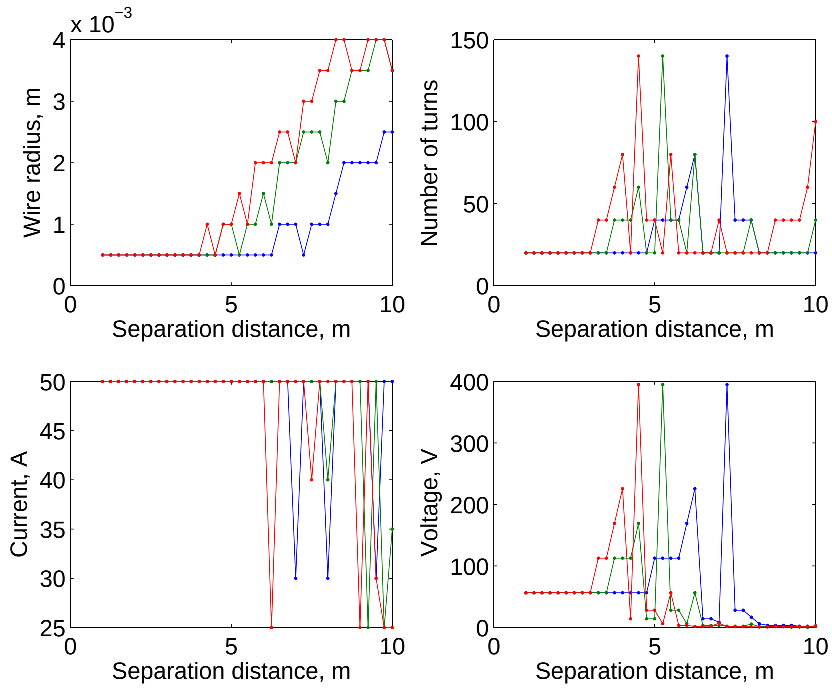


Figure 2.26: EMFF system parameters for various bus masses and separation distances

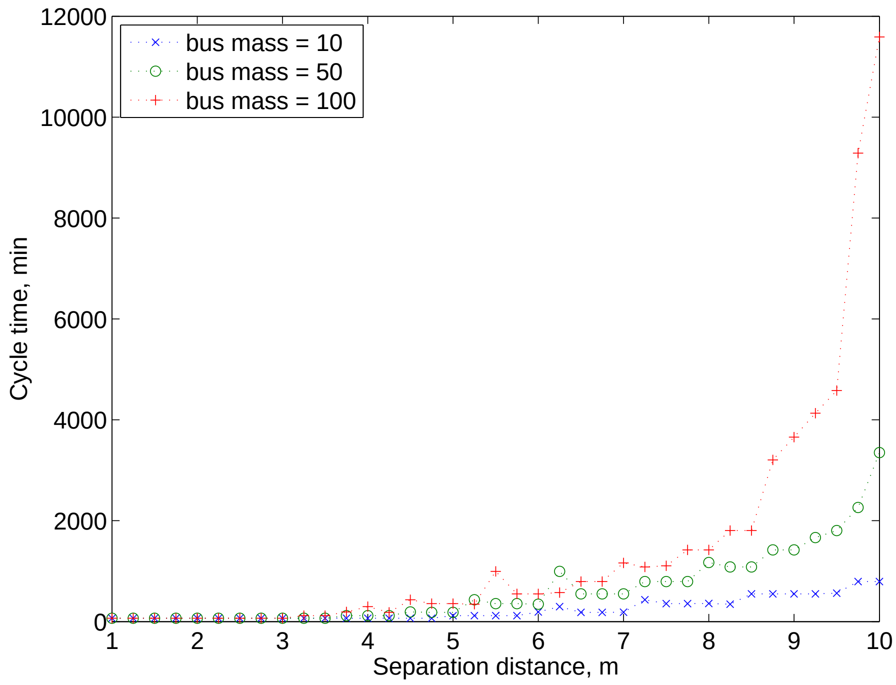


Figure 2.27: Maximum cycle times for various bus masses and separation distances

Voltage tends to follow the number of turns very closely. Since $V = IR$, voltage is proportional resistance, which in turn is proportional to the length of wire, and thus to the number of turns. The voltage curve parallels the curve of the number of turns, but dips where the current level dips. As resistance goes also by the inverse square of radius, so also voltage drops accordingly with increasing radius.

The cycle time is strongly affected by separation distance, as seen in fig. 2.27. Cycle time increases with increasing radius, and increasing separation distance is, as said above, correlated with increasing wire radius. Thus, the farther apart the satellites get, the longer the system can remain on without overheating, and the longer it must stay off afterward to cool down.

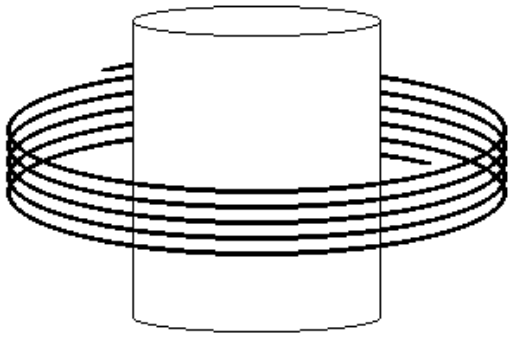


Figure 2.28: Helical geometry

2.4 Alternative Geometry

2.4.1 Helical Geometry

The hexagonally-bundled configuration above is not the only way of arranging the wires. Another potential arrangement is to have the wires arranged in a helix, as in fig. 2.28. In this arrangement, the coil of wire describes a cylinder surrounding the satellite. This allows each part of the coil surface area to radiate out heat, which dramatically reduces the cooling time of the system.

2.4.2 Results

In the helical geometry, mass and acceleration capacity are much more strongly correlated, as seen in fig. 2.29. Their relationship is almost linear. Without the increased cooling times from the bundled wires, the system can remain on for a more consistent period of time, thus allowing the thermal factors to have less of an impact on the mass-acceleration curve.

In this geometry, cooling the wire takes approximately the same amount of time for any given wire radius. Heating, on the other hand, is dependent on the current. Because of this, increasing the current tends to quadratically decrease the

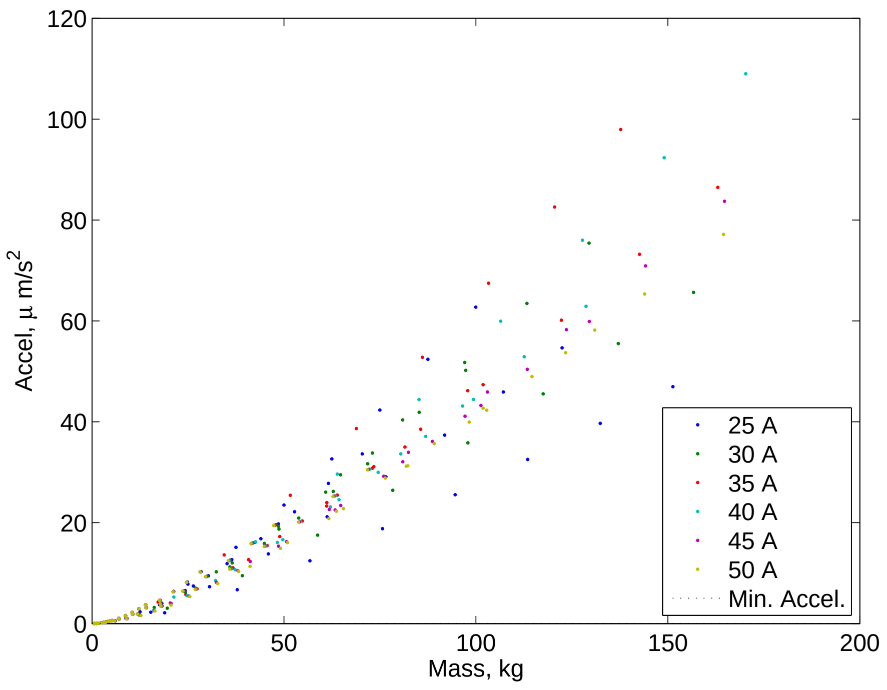


Figure 2.29: Acceleration capability by mass, helical geometry

fraction of the cycle time over which the system is active (fig. 2.32). The quadratic decrease of active cycle time cancels out the performance increase from increased current, leading to a fairly constant acceleration capability (fig. 2.31). The non-flat shape of the graph is due to current and cycle time fraction not canceling out perfectly. This imprecise cancellation leads to a small increase in battery mass at 35A (fig. 2.30).

The number of turns, in the helical geometry, has no effect on any thermal issues (see in particular fig. 2.35). Thus, there are no thermal nonlinearities introduced, and the mass (fig. 2.33) and acceleration (fig. 2.34) are linear with respect to the number of turns.

Increasing the radius increases the wire mass quadratically, as expected (fig. 2.36); the solar panel mass also increases, but the battery mass has an odd peak around a radius of 3mm. This peak is due to the interaction between the cubic increase of the fraction of the cycle time on (fig. 2.38) and the almost-quadratic

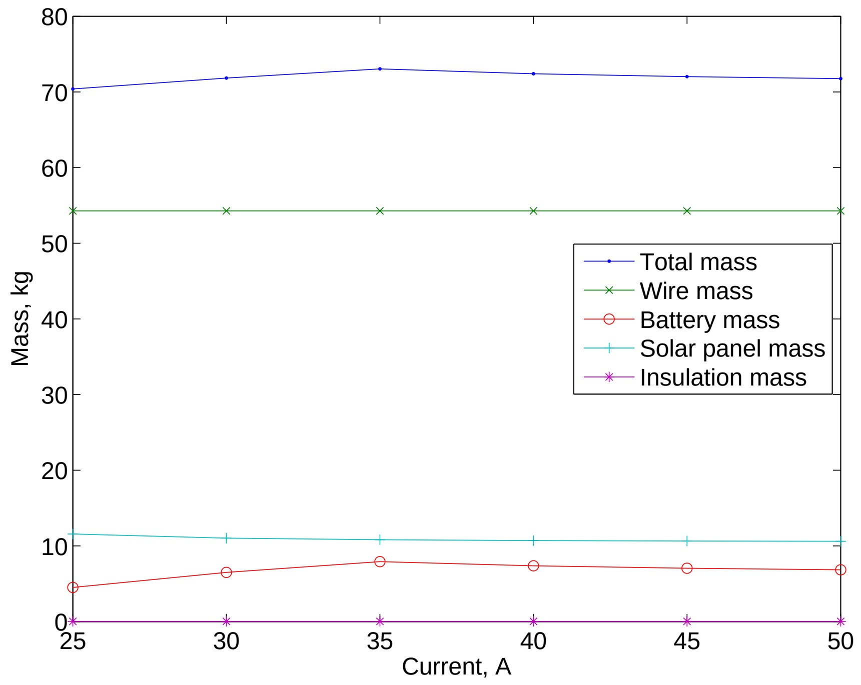


Figure 2.30: Mass, 2.5-mm radius, 160 turns, helical geometry

increase of acceleration capacity (fig. 2.37).

When the mass and separation distances are varied, a similar pattern emerges with the helical geometry as with the default geometry (see fig. 2.39; compare to fig. 2.23), the most significant difference being that the helical geometry optima are significantly lighter in mass than the default geometry optima. A comparison of figures 2.40 and 2.26 also shows that the patterns taken by the various parameters are much more regular over the given data set in the helical geometry. While the wire radius is the most powerful parameter, and increasing it gives the most increase in acceleration capacity, the lighter masses of the helical geometry make the less sensitive variable of the number of turns a better way to fine-tune the acceleration capacity of the satellite. The wire radius is changed only once, for the 100-kg satellite, at a 9-meter separation distance. It is probable that a different range of parameters, with smaller wire radii, would have provided data more similar to that in section 2.3.2.

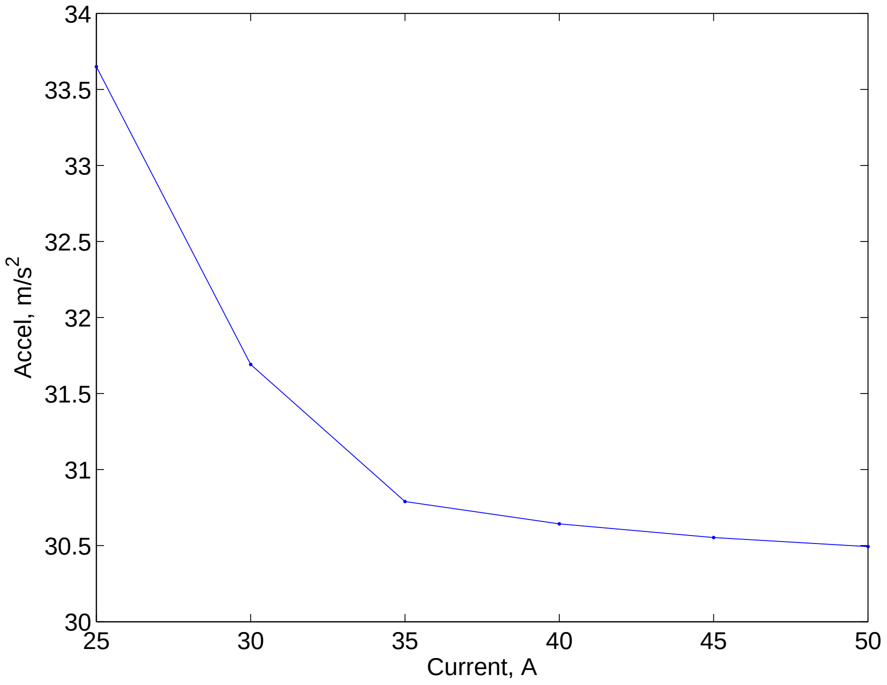


Figure 2.31: Acceleration capability, 2.5-mm radius, 160 turns, helical geometry

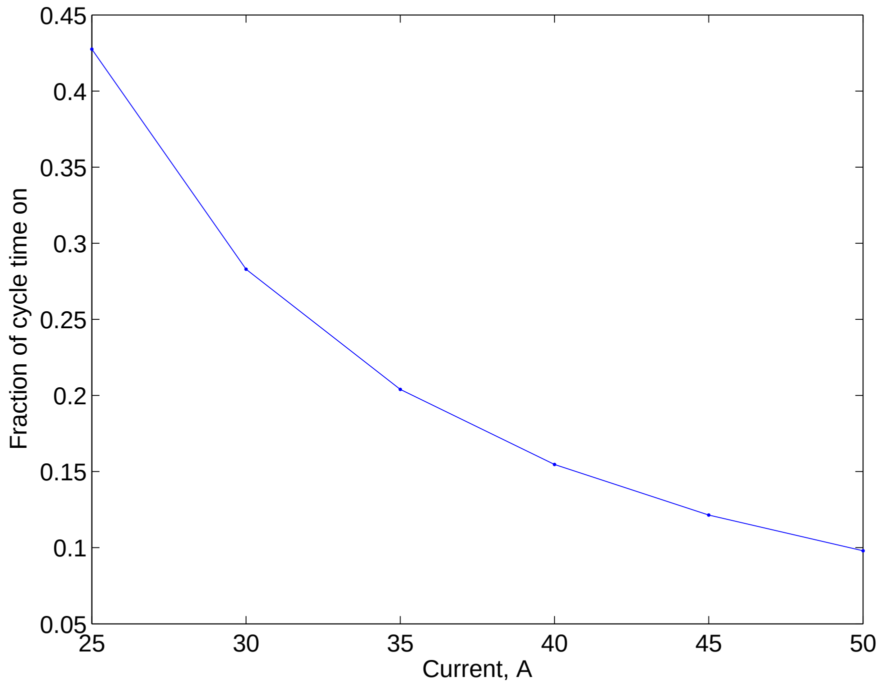


Figure 2.32: Fraction of cycle time on, 2.5-mm radius, 160 turns, helical geometry

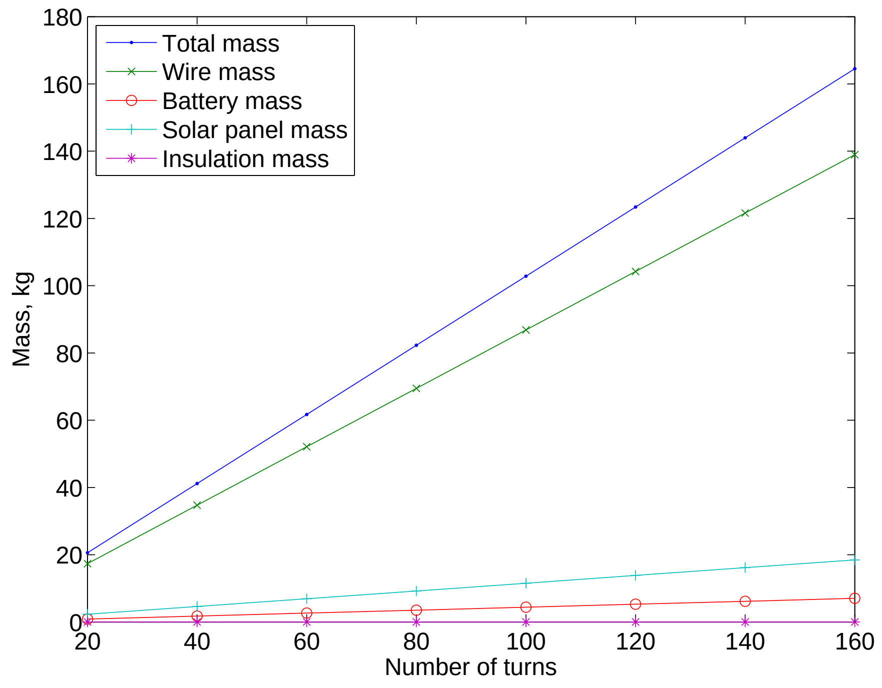


Figure 2.33: Mass, 4-mm radius, 50A, helical geometry

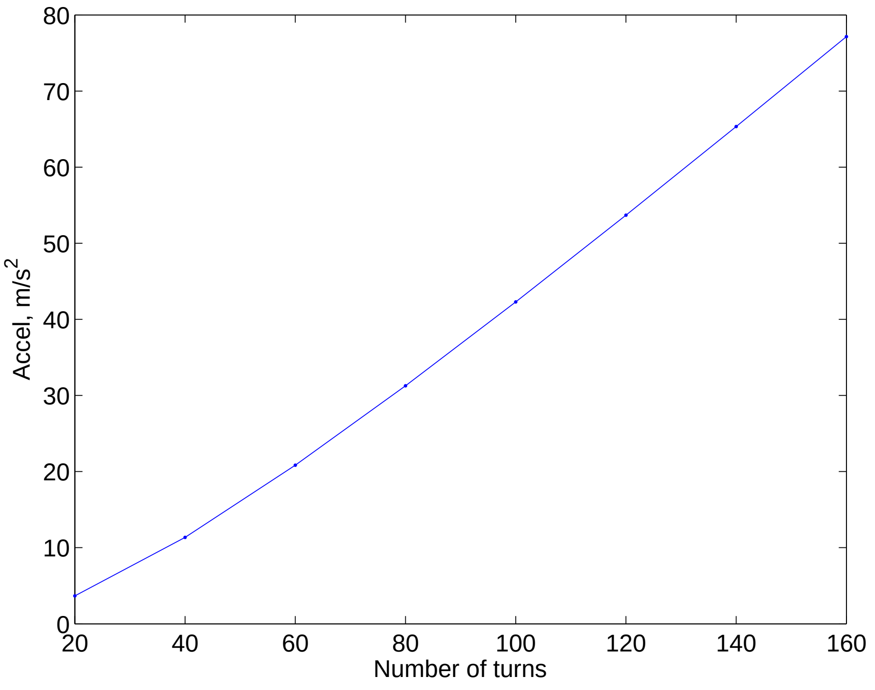


Figure 2.34: Acceleration capability, 4-mm radius, 50A, helical geometry

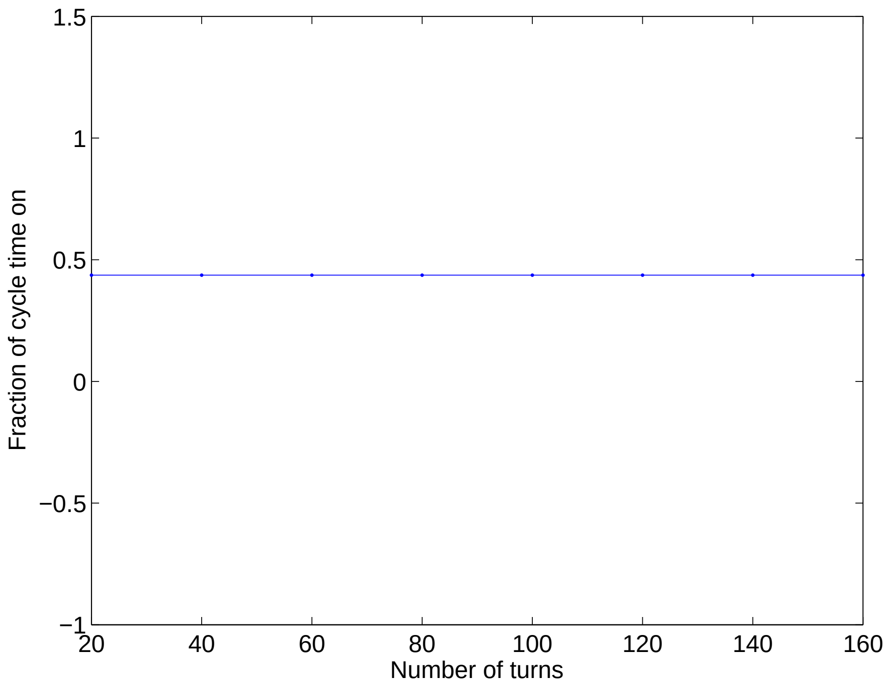


Figure 2.35: Fraction of cycle time on, 4-mm radius, 50A, helical geometry

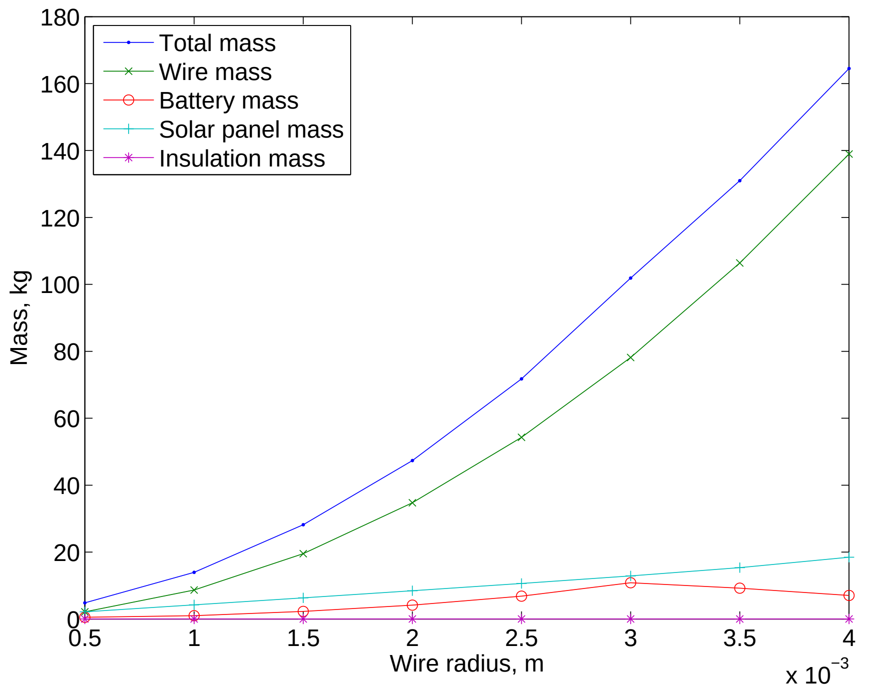


Figure 2.36: Mass, 160 turns, 50A, helical geometry

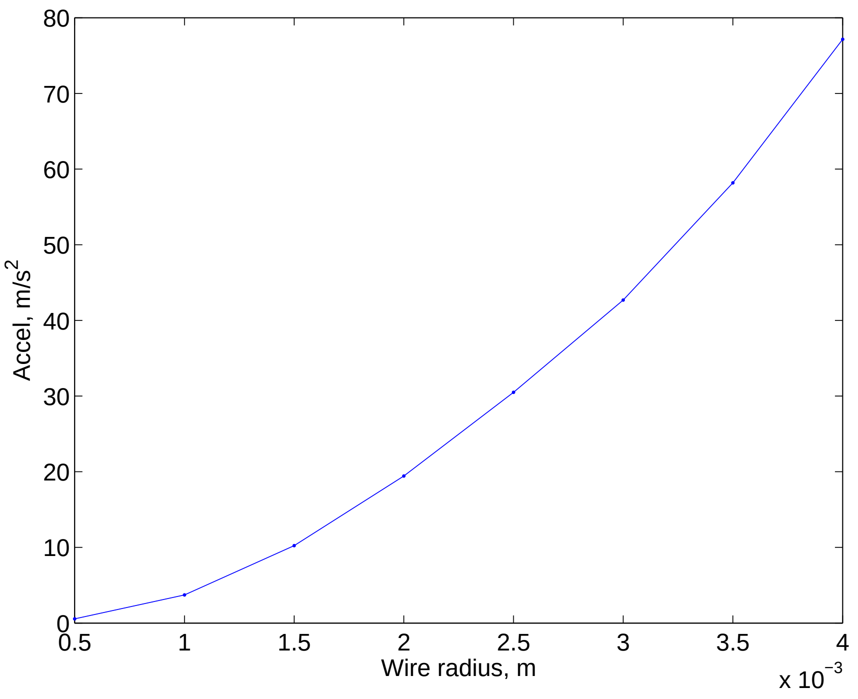


Figure 2.37: Acceleration capability, 160 turns, 50A, helical geometry

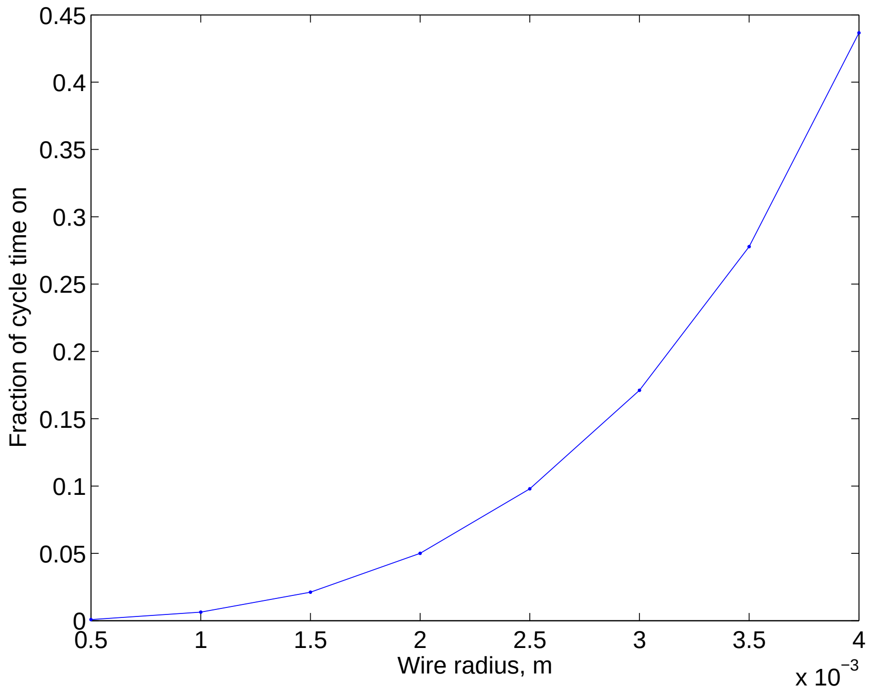


Figure 2.38: Fraction of cycle time on, 160 turns, 50A, helical geometry

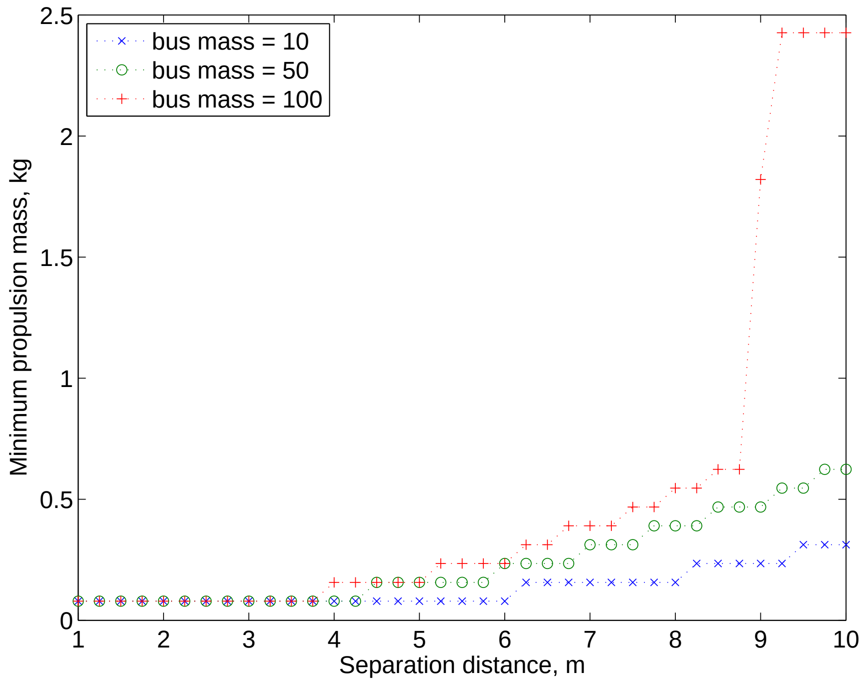


Figure 2.39: Minimum EMFF system ass for various bus masses and separation distances, helical geometry

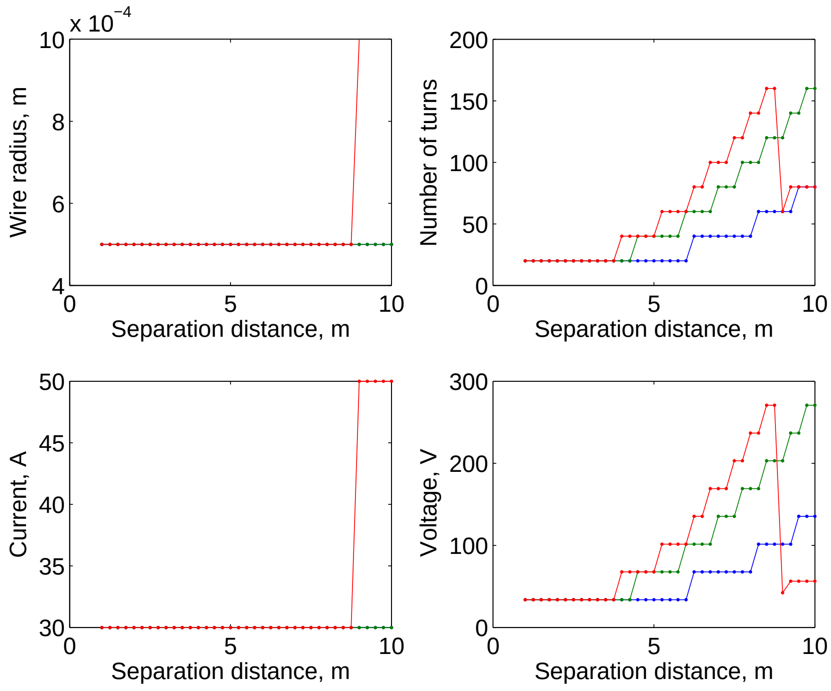


Figure 2.40: EMFF system parameters for various bus masses and separation distances, helical geometry

2.4.3 Significant Differences

The helical geometry has a lighter mass relative to acceleration than the bundled geometry, due to thermal effects from the bundled packing. However, one problem that might arise also comes from the geometry: if the wire radius is large and there are many turns, the helix could get quite high, and could obstruct the satellite's line of sight. In the scenarios studied here, this is not a problem; the wire radius never gets over 1 mm, and there are never more than 160 turns.

The near-linear relationship of mass to acceleration capacity would seem to make the helical geometry much better for larger satellites. However, with larger masses and separation distances, the see-over problem could manifest itself and partially neutralize the benefits.

2.5 Conclusion

The acceleration capacity can be increased by increasing the wire radius, at the cost of adding mass. Increasing the number of turns also increases acceleration capacity, but more slowly; however, when the geometry of the coil requires the wire to be packed closely together, the resulting thermal constraints reduce the advantage of having many turns of wire.

Using Electromagnetic Formation Flight on small satellites requires a very low system mass. But in this chapter, it has been demonstrated that the EMFF system can be effectively implemented in 100-kilogram-scale satellites.

Chapter 3

Power Transfer

3.1 Coupled Resonance Power Transfer

In the situation where several satellite components are distributed into several different satellites, maneuvered relative to each other using EMFF or other means, it may prove useful to have the power systems on only one or two satellites. If so, it would be very advantageous to have a method of wireless power transmission between the satellites. A system of inductive power transfer via strongly coupled magnetic resonances may be available for these purposes.

While radiative power transfer tends to be rather low-efficiency, a system for power transfer by means of strongly coupled magnetic resonances has been proposed.[1]. Inductive power transfer is relatively efficient at close range, but the efficiency falls off quickly as the separation distance increases. By using resonant coils, power transfer can remain efficient up through several meters of separation distance.[2]

Considering that both the power transfer system and the EMFF system use coils of wire, it would be convenient for both to use the same coils of wire, thus

[1]Kurs *et al.*, 2007
[2]Kurs *et al.*, 2007

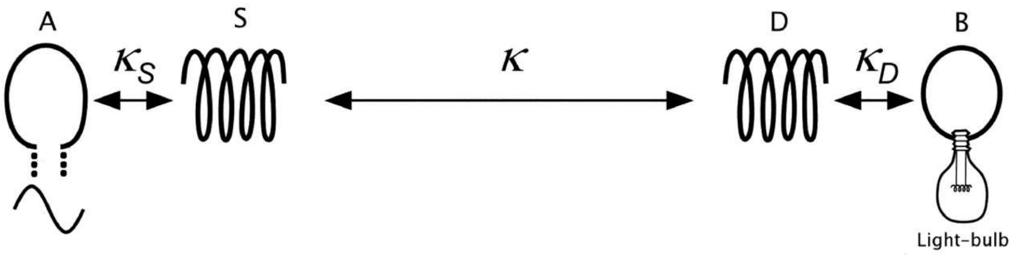


Figure 3.1: Power transfer setup from Kurs *et al.*, 2007. A is the power supply, connected inductively or physically to S, the source coil. D, the device coil, connects physically or inductively to B, the load.

saving mass. It is therefore necessary to determine how efficiently power can be transmitted over the expected typical separation distance using the coils of wire for EMFF.

3.2 Methodology

The power transfer efficiency is a function of κ, the coupling constant, and Γ_S, Γ_D, and Γ_W, the decay rate for the source coil, device coil, and load coil, respectively.

To find these variables, it is first necessary to calculate the inductance and capacitance of the coils of wire being used. The equations[3] for these are eq. 3.1 and 3.2:

$$L = \frac{\mu_o}{4\pi|I|^2} \iint d\vec{r}d\vec{r}' \, \frac{\vec{J}(\vec{r}) \bullet \vec{J}(\vec{r}')}{|\vec{r} - \vec{r}'|} \tag{3.1}$$

$$\frac{1}{C} = \frac{1}{4\pi\epsilon_o|q_o|^2} \iint d\vec{r}d\vec{r}' \, \frac{\rho_q(\vec{r})\rho_q(\vec{r}')}{|\vec{r} - \vec{r}'|} \tag{3.2}$$

In order to calculate these, first a computer model is constructed for each coil: a helix, with height $h = r_{wire}\sqrt{N}$, radius r_{loop}, N turns, wire radius r_{wire}, and wire length $l = 2\pi r_{loop}N$, as seen in appendix B.1, `inductionarray.m`.

[3]Kurs *et al.*, 2007

Kurs, *et. al.* calculate an effective mutual inductance, as in eq. 3.3:

$$M = -\frac{1}{4\pi I_S I_D \omega} \iint d\vec{r} d\vec{r}' \left[\mu_o \frac{\vec{J}_S(\vec{r}')}{\vec{r}' - \vec{r}} + \frac{\rho_{q,S}(\vec{r}')}{\epsilon_o} \frac{\vec{r}' - \vec{r}}{|\vec{r}' - \vec{r}|^3} \right] \bullet \vec{J}_D(\vec{r}') \tag{3.3}$$

in which the S and D subscripts, following the conventions used by Kurs, *et. al.*, refer to the "source" and "device" coils, respectively. Equation 3.3 includes the resonant frequency of the source coil, found with equations 3.4 and 3.5:

$$f = \frac{1}{2\pi\sqrt{LC}} \tag{3.4}$$

$$\omega = 2\pi f \tag{3.5}$$

Once M is known, the coupling coefficient κ can be found using equation 3.6:

$$\kappa = \frac{\omega M}{2\sqrt{L_S L_D}} \tag{3.6}$$

To find Γ_S and Γ_D, the coupled-mode theory decay constants, equation 3.7 is used:

$$\Gamma = (Ro + Rr)/2/L; \tag{3.7}$$

where R_o and R_r are ohmic and radiation resistance, respectively. Kurs, et. al. [4] give the formulas for these as eq. 3.8 and 3.9.

$$R_o = \sqrt{\frac{\mu_o \omega}{2\sigma}} \frac{l}{4\pi r_{wire}} \tag{3.8}$$

$$R_r = \sqrt{\frac{\mu_o}{\epsilon_o}} \left(\frac{\pi N^2}{12} \left(\frac{\omega r_{loop}}{c} \right)^4 + \frac{2}{3\pi^3} \left(\frac{\omega h}{c} \right)^2 \right) \tag{3.9}$$

Since κ and each Γ are now known, the maximum efficiency can be calculated,

[4] Kurs *et al.*, 2007

through equation 3.10:

$$\eta_{max}(d_{sep}) = \frac{\frac{\Gamma_W}{\Gamma_D}\frac{\kappa^2(d_{sep})}{\Gamma_S\Gamma_D}}{\left(\left(1+\frac{\Gamma_W}{\Gamma_D}\right)\frac{\kappa^2(d_{sep})}{\Gamma_S\Gamma_D}+\left(1+\frac{\Gamma_W}{\Gamma_D}\right)^2\right)} \tag{3.10}$$

where Γ_W is the decay rate of the load on the target satellite.[5] For maximum efficiency, Γ_W needs to be

$$\frac{\Gamma_W}{\Gamma_D} = \sqrt{1+\frac{\kappa^2}{\Gamma_S\Gamma_D}} \tag{3.11}$$

Since the EMFF power transfer system involves identical coils on the sending and receiving ends, $\Gamma_S = \Gamma_D = \Gamma$. Equation 3.11 reduces to eq. 3.12:

$$\frac{\Gamma_W}{\Gamma} = \frac{\sqrt{\Gamma^2+\kappa^2(d_{sep})}}{\Gamma} \tag{3.12}$$

3.3 Design Space

In order to mesh well with chapter 2, the geometric parameters were selected to match those in table 2.1. The electrical and chronological variables (voltage, current, time on, time off) do not at all affect power transfer, since power will be transferred only when the EMFF system is inactive. Since loop radius is fixed, the only remaining variables are the number of turns and the wire radius. Wire radius does not affect inductance as much as the number of turns, and was thus neglected, so the only geometric variable that was analyzed was the number of turns, which again, varies from 20 to 160.

The remaining variables for power transfer are the separation distance and the angle between the two coils. For this thesis, it is assumed that all interactions are on-axis, so the angle is fixed at 0°. The separation distance is varied from 1 to 10 meters, just as in `findmultibest.m` (Appendix A.4).

[5] Kurs *et al.*, 2007

3.4 Results

Using `inductionarray.m` (Appendix B.1), coupling coefficients and maximum power transfer efficiencies were found for separation distances from 1 through 10 meters in steps of 1 meter, and for coils of 20 to 160 turns in steps of 20 turns.

Increasing the number of turns increases the value of the coupling coefficient κ, as seen in figs. 3.2 and 3.3. This increase is mostly linear, with a few minor irregularities. With large separation distances, the linear nature is significantly less obvious; however, κ, in close proximity, is on the order of 10^6; the coupling coefficients at a ten-meter distance are in the hundreds, so deviation from the pattern is far less significant there.

Figures 3.4 and 3.5 show power transfer efficiencies across various numbers of turns. Increasing the number of turns increases the efficiency. As the turns increase, additional turns have less effect. When the coils are in close proximity, this effect is less significant; the efficiencies are very high no matter how many turns there are.

As seen in figs. 3.6 and 3.7, κ falls off quite precipitously with increasing separation distance. Coupling coefficients are on the order of 10^6 with one-meter separations; however, with ten-meter separations, they are on the order of 100.

Power transfer efficiency drops off as distance increases, as seen in figs. 3.8 and 3.9. Larger numbers of turns postpone the dropoff to a more distant point, but make the fall more precipitous.

Figure 3.10 shows the power transfer efficiencies for the designs found in section 2.3.2, based on the number of turns for each bus mass found at various distances (see fig. 2.26 for numbers of turns). Since most of the optimal designs have few turns of wire, the trend tends to follow that of fig. 3.8. There are several notable outliers here. While most of the mass-optimal designs have low numbers of turns of wire, a few have higher turn counts, and these have higher power transfer efficiencies than would otherwise be expected.

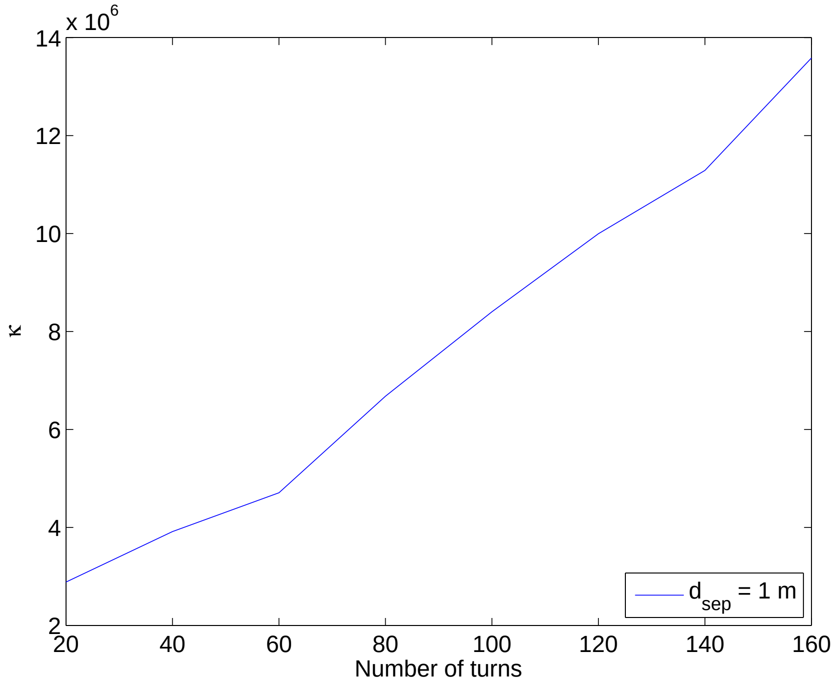


Figure 3.2: κ with 20 to 160 turns, 1 m separation

3.5 Conclusion

At close range, an inactive electromagnetic formation flight system can be used for coupled magnetic resonance power transfer. Larger numbers of turns of wire increase the range over which the power transfer is efficient. For the mass-optimal solutions found in section 2.3.2, power transfer is quite efficient within a range of five meters.

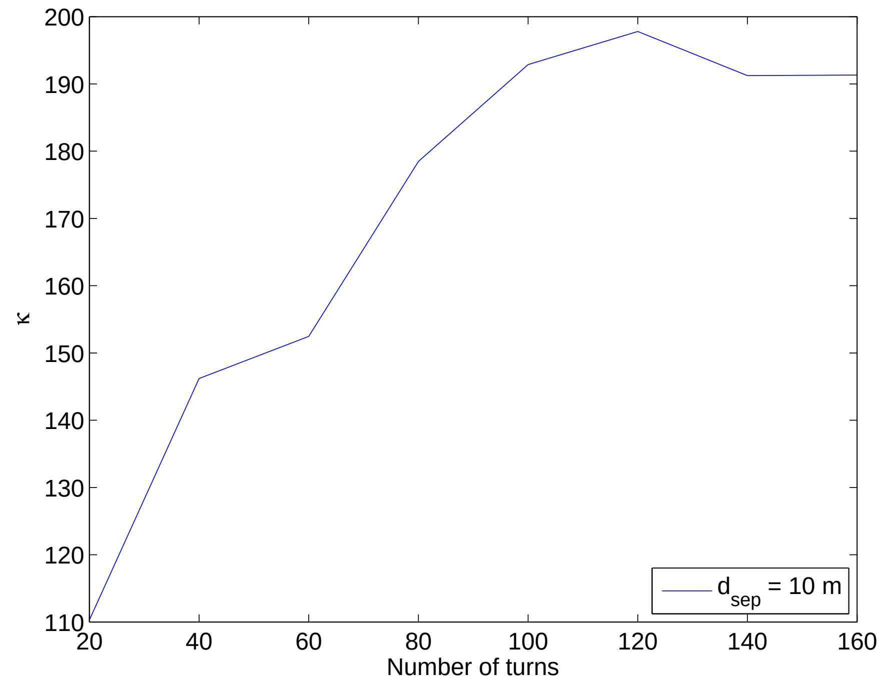


Figure 3.3: κ with 20 to 160 turns, 10 m separation

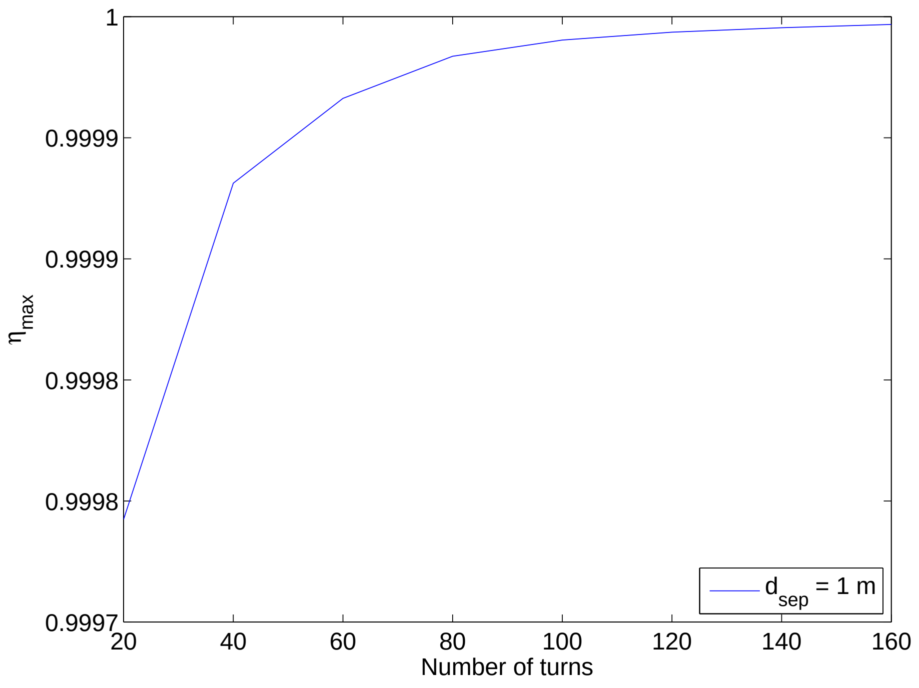


Figure 3.4: η_{max} with 20 to 160 turns, 1 m separation

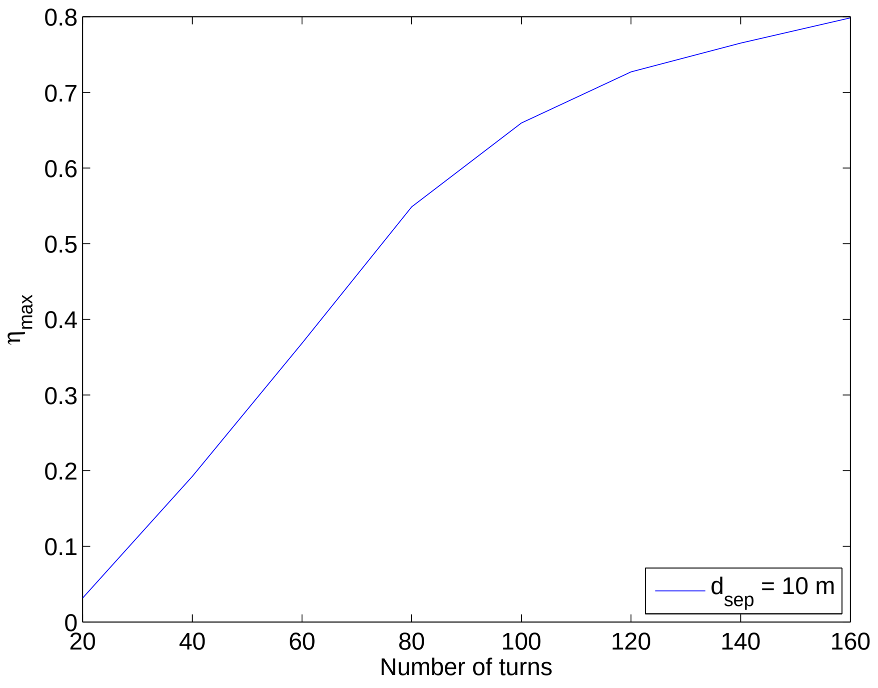


Figure 3.5: η_{max} with 20 to 160 turns, 10 m separation

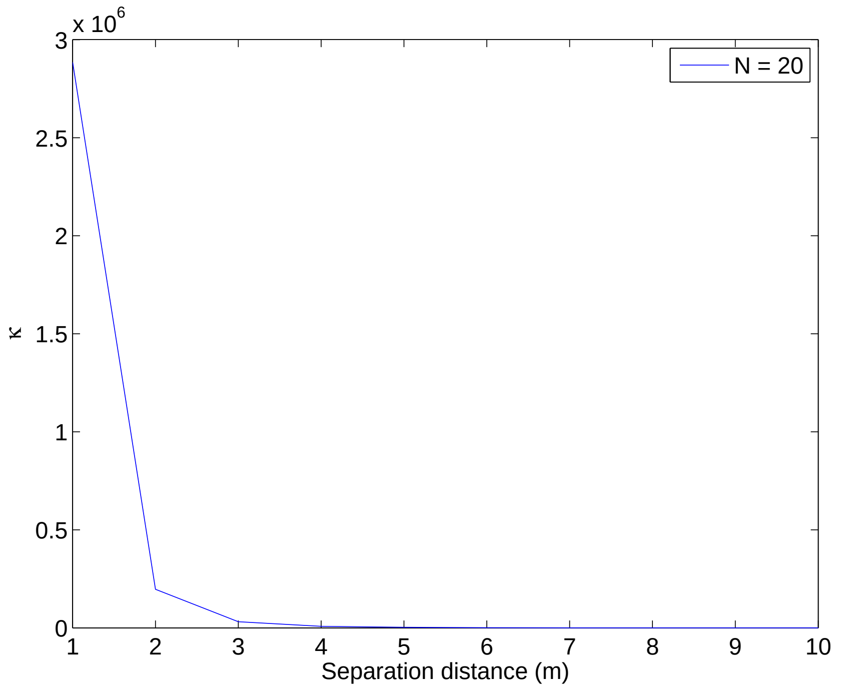


Figure 3.6: κ at 1 to 10 meters apart, 20 turns

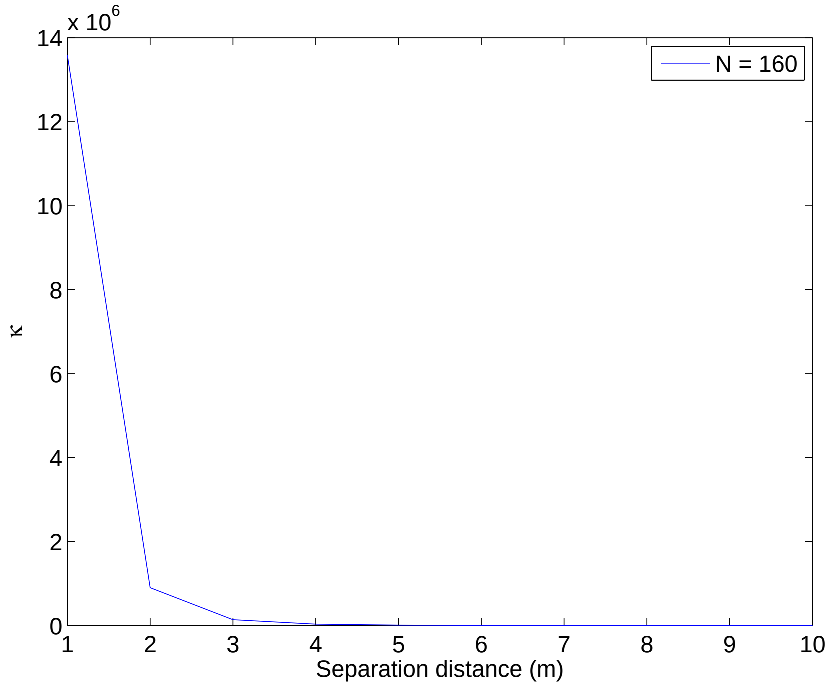


Figure 3.7: κ at 1 to 10 meters apart, 160 turns

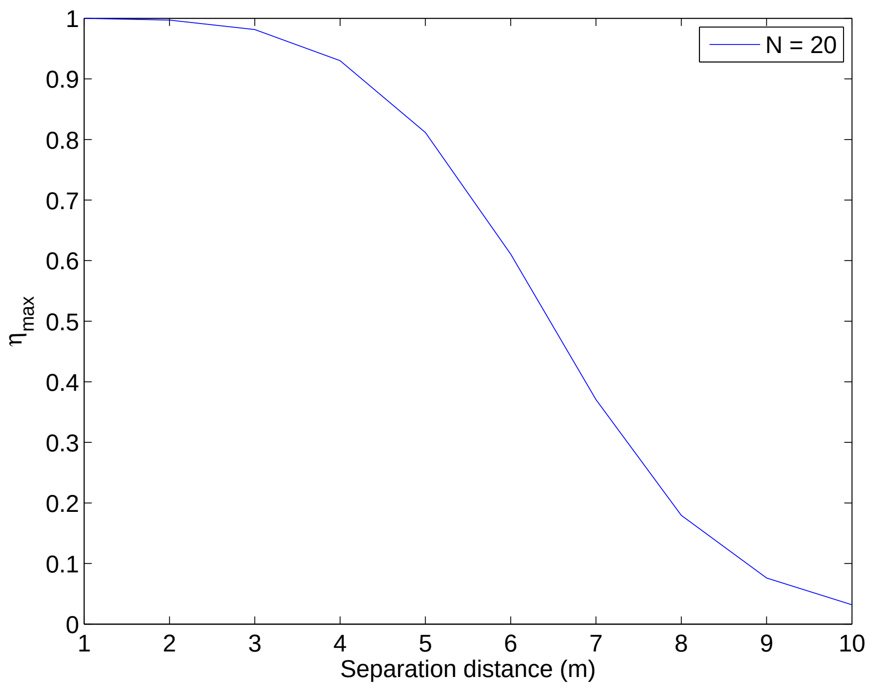


Figure 3.8: η_{max} at 1 to 10 meters apart, 20 turns

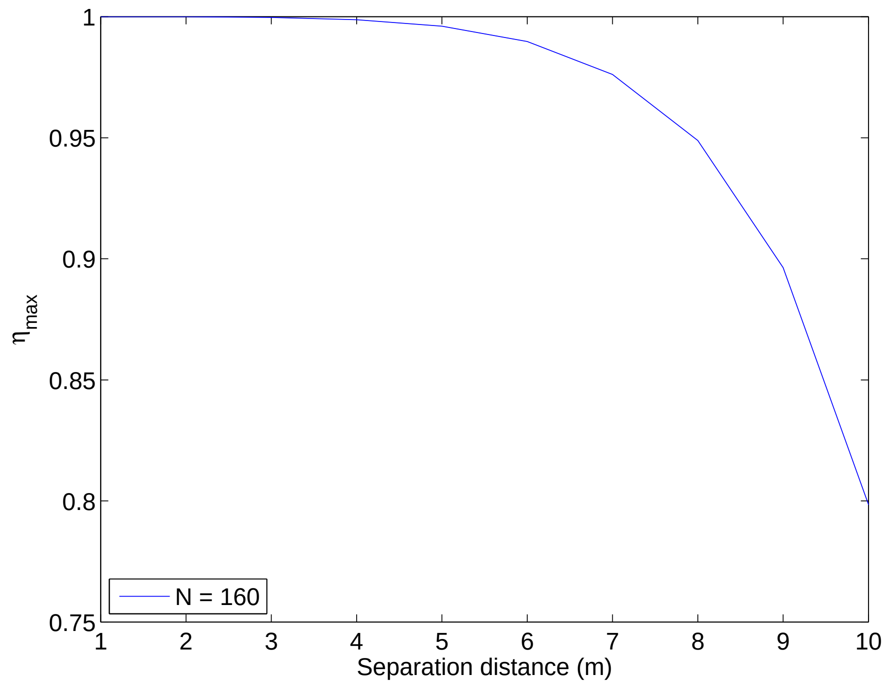


Figure 3.9: η_{max} at 1 to 10 meters apart, 160 turns

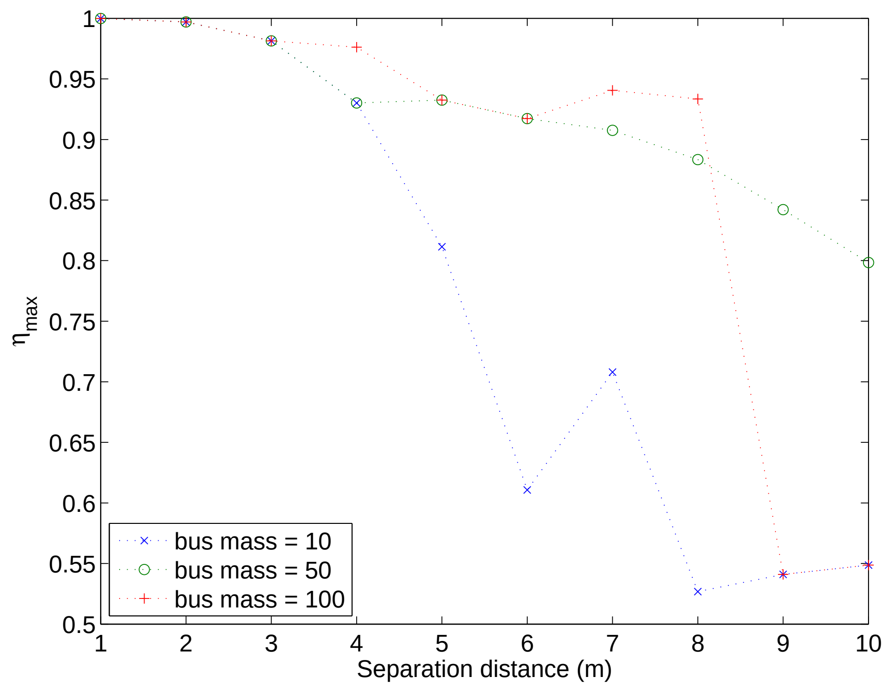


Figure 3.10: η_{max} for mass-optimal designs at 1 to 10 meters apart

Chapter 4

Paired Satellites

4.1 Introduction

4.1.1 Problem

Once we have satellites which are capable of using Electromagnetic Formation Flight technology, it is necessary to develop control techniques for their operation. One potential technique is to maneuver satellites axially in pairs. Satellite dipoles are only activated such that they are coaxial. This arrangement minimizes torque, since magnetic torque comes from off-axis interaction. The use of one pair at a time also simplifies the control calculations quite a bit.

For this research, formal control theory was not used. The simulations were set up in such a way as to give general results for feasibility, rather than any specific control law.

4.1.2 Approach

Four satellites are modeled, among which six pairs of satellites can be made. One pair of satellites at a time is selected for maneuvering; only their magnetic dipoles are turned on; the other two satellites remain magnetically inert. This activated pair is selected by identifying which of the six pairs is either farthest away (and need to get closer) or closest together (and need to move apart). Due to the importance of collision avoidance, priority is given to satellite pairs which are too close over pairs which are too far apart. If the closest pair is inside the minimum distance, or the farthest pair is outside the maximum, a dipole is activated on each in order to impart an acceleration. It is assumed that a dipole can be established arbitrarily quickly in any orientation.

The paired satellite simulation was implemented in MATLAB. Three scripts were used for each scenario: one common display script for output (Appendix C.2), one common calculation script for updating the state at every time step (Appendix C.1), and one setup and control script, specific for each scenario (Appendices C.3 through C.6).

4.2 Calculations

4.2.1 State

In the paired satellites control scheme, two satellites at a time need to simultaneously activate, and then later deactivate, their dipoles. Because of this, the state is updated at discrete times, in order to keep everything synchronized. The new positions and velocities at each time step are found using equations 4.1 and 4.2, derived from Newton's second law:

$$\vec{x}_j(t+1) = \vec{x}_j(t) + \vec{v}_j(t)\Delta t + \frac{1}{2}\frac{\vec{F}_j}{m_j}\Delta t^2 \tag{4.1}$$

$$\vec{v}_j(t+1) = \vec{v}_j(t) + \frac{\vec{F}_j}{m_j}\Delta t \tag{4.2}$$

Angular velocity is updated using equation 4.3, where $\mathbf{I}^{(j)}$ is the moment of inertia tensor[1]. Though attitude is not simulated in this simulation, angular velocity is included to ensure that angular velocities do not get beyond the reasonable capacity of reaction wheels, or other attitude control devices.

$$\vec{\omega}_j(t+1) = \mathbf{I}^{(j)-1}(\vec{\tau}_j - \vec{\omega}_j(t) \times (\mathbf{I}^{(j)}\, \vec{\omega}_j(t)))\Delta t + \vec{\omega}_j(t) \tag{4.3}$$

The force on each satellite ($\vec{F}_j$) is modeled (as in eq. 4.4) as the sum of the force from interaction with the other satellites ($\sum_{k=1}^{4} \vec{F}_{j,k}$), the force of the J_2 perturbation ($\vec{F}_j^{(J_2)}$), and a scalar force ($\vec{F}_{o,j}$).

$$\vec{F}_j = \sum_{k=1}^{4}(\vec{F}_{j,k}) + \vec{F}_j^{(J_2)} + \vec{F}_{o,j} \tag{4.4}$$

The forces and torques are calculated in equations 4.5 and 4.6:[2]

$$\vec{F}_{j,k} = \vec{\mu}_j \cdot \nabla \vec{B}_k|_j \tag{4.5}$$

$$\vec{\tau}_{j,k} = \vec{\mu}_j \times \vec{B}_k|_j \tag{4.6}$$

The magnetic field strength $\vec{B}_k|_j$ from satellite k at point j is found in eq. 4.7:

$$\vec{B}_k|_j = \frac{\mu_o}{4\pi d_{j,k}}(-\vec{\mu}_k + 3(\vec{\mu}_k \cdot \hat{r}_{k,j})\hat{r}_{k,j}) \tag{4.7}$$

where $d_{j,k} = \|\vec{x}_j - \vec{x}_k\|$ and $\hat{r}_{j,k} = \frac{\vec{x}_j - \vec{x}_k}{\|\vec{x}_j - \vec{x}_k\|}$. Thus, we can find equations 4.8 and 4.9.

$$\vec{F}_{j,k} = \frac{3\mu_o}{4\pi d_{j,k}^4}\left((\vec{\mu}_j \cdot \vec{\mu}_k)\,\hat{r}_{k,j} + (\vec{\mu}_j \cdot \hat{r}_{k,j})\,\vec{\mu}_k + (\vec{\mu}_k \cdot \hat{r}_{k,j})\,\vec{\mu}_j - 5\,(\vec{\mu}_j \cdot \hat{r}_{k,j})\,(\vec{\mu}_k \cdot \hat{r}_{k,j})\,\hat{r}_{k,j}\right) \tag{4.8}$$

$$\vec{\tau}_{j,k} = \vec{m}_j \times \frac{\mu_o}{4\pi d_{j,k}^3}\left(3\,(\vec{\mu}_2 \cdot \hat{r}_{k,j})\,\hat{r}_{k,j} - \vec{\mu}_k\right) \tag{4.9}$$

[1]Pines, 2007

[2]Elias, 2004, p. 95

The first of the two remaining forces from eq. 4.4 is the force from the J_2 perturbation. This is calculated as in equation 4.10.[3]

$$\vec{F}_j^{(J_2)} = m_j(\vec{x} - \bar{x}\,)\,3.25 \cdot 10^{-9} \tag{4.10}$$

The other force is a constant acceleration, which in the cases being studied, is only applied to one satellite, as if there were a constant thrust on it.

$$\vec{F}_{o,j} = m\vec{a}_{cons} \tag{4.11}$$

4.2.2 Control

To control the satellites, it is necessary to determine which, if any, dipoles need to be activated. A list of the distances between each of the six potential satellite pairs is generated. First, the minimum distance is checked, to see if it's too close — within d_{in}. If not, the maximum distance is checked, to see if it's too far apart — more than d_{out}. If one of those cases is met, the satellites are checked to see if they are moving toward the nominal bounds yet. If ${}^2\vec{r}_1 \cdot {}^2\vec{v}_1 > 0$ or ${}^1\vec{r}_2 \cdot {}^1\vec{v}_2 > 0$, then the satellites are considered to be moving together; if those conditions are not met, they are considered to be moving apart.

If a maneuver is needed, the dipole orientation must be found. Where $\vec{r} = \vec{x}_1 - \vec{x}_2$, $\frac{\vec{r}}{\|\vec{r}\|}$ yields the dipole direction, unless the satellites are too close, in which case one of the two satellites orients its dipole $-\frac{\vec{r}}{\|\vec{r}\|}$, for collision avoidance. The dipole strength is scaled to produce sufficient force to zero out the satellite's outward velocity. Putting eq. 4.8 into scalar form yields

$$F = \frac{3\mu_o\|\vec{\mu}\|^2}{2\pi d_{sep}^4} \tag{4.12}$$

[3]Sakaguchi, 2007, p. 28

and therefore

$$\|\vec{\mu}\| = \sqrt{\frac{2\pi d_{sep}^4 F}{3 \cdot 4\pi \cdot 10^{-7}}} \tag{4.13}$$

To cancel out the relative velocity of the satellites in τ seconds, F needs to be mv_{rel}/τ, therefore

$$\|\vec{\mu}\| = \sqrt{\frac{mv_{rel} d_{sep}^4}{\tau 6 \cdot 10^{-7}}} \tag{4.14}$$

The time constant, giving the time to come to a halt, is arbitrary; a τ of one second was selected, in order to simplify the calculation.

If d_{out} is used for d_{sep} — since $d_{sep} = d_{out}$ when the system switches on — all but one of the terms in eq. 4.14 are constant for a given scenario. A "dipole strength constant" $\sqrt{\frac{md_{out}^4}{6 \cdot 10^{-7}}}$ can be found, needing only to be multiplied by $\sqrt{v_{rel}}$ to produce the dipole magnitude.

However, due to other pairs possibly taking precedence, it is possible that a given satellite pair may be well over d_{out} apart before their dipoles have a chance to be activated. To account for the increased distance, d_{out} can be replaced by d_{sep} by means of multiplying the "dipole strength constant" by $\left(\frac{d_{sep}}{d_{out}}\right)^2$.

Unfortunately, when $d_{sep} > d_{out}$, even after the adverse velocity has been eliminated, the satellite pair tends to drift ever farther apart before the next available round of magnetic attraction. To counteract this, larger $\frac{d_{sep}}{d_{out}}$ ratios need somewhat higher proportional dipole moments. Empirically, multiplying the dipole strength by an additional $\frac{d_{sep}}{d_{out}}$ is insufficient to counteract the drift, but $\left(\frac{d_{sep}}{d_{out}}\right)^2$ works quite nicely, leading to the equation used for dipole moments for pairs which are too far apart,

$$\|\vec{\mu}\| = \sqrt{\frac{md_{out}^4}{6 \cdot 10^{-7}}} \left(\frac{d_{sep}}{d_{out}}\right)^4 \sqrt{v_{rel}} \tag{4.15}$$

If the dipoles need to be activated because the satellites are too close, the dipole strength must be much less. First, the "dipole strength constant" must be multiplied by $\left(\frac{d_{in}}{d_{out}}\right)^2$ to cancel out the d_{out} in the constant's formula. Then, as with the distant case, $\left(\frac{d_{sep}}{d_{in}}\right)^2$ can translate d_{in} into d_{sep}. Finally, one of the two dipoles is multiplied by -1, so that they repel each other, instead of attracting each other.

Due to magnetic force falling off as the inverse square of distance, at close range, quite impressively large forces can be had. To further complicate matters, due to the small distances involved, things happen very quickly, so problems tend to compound themselves within one or two time steps. If velocities are a little bit too high, the simulated satellites might pass through each other, for example, across one time step, or if they are particularly close, interact so as to gain several thousand meters per second velocity. Because of this, the strength has to be very carefully adjusted, lest on one hand, satellites fly apart at excessive velocity, or on the other, they collide. Empirically, it was determined that multiplying by an additional $\left(\frac{d_{in}}{d_{out}}\right)^2$ and then dividing by $\left(\frac{d_{sep}}{d_{in}}\right)^5$ achieves a workable balance. In effect, the gain has been empirically calibrated. From this, we get equation 4.16:

$$\|\vec{\mu}\| = \sqrt{\frac{md_{out}^4}{6\cdot 10^{-7}}}\left(\frac{d_{in}}{d_{out}}\right)^4\left(\frac{d_{in}}{d_{sep}}\right)^3\sqrt{v_{rel}} \tag{4.16}$$

Plugging equations 4.15 and 4.16 back into equation 4.12, we get, for the resulting force for lost-in-space, eq. 4.19

$$F = \frac{3\mu_o(\sqrt{\frac{md_{out}^4}{6\cdot 10^{-7}}}\left(\frac{d_{sep}}{d_{out}}\right)^4\sqrt{v_{rel}})^2}{2\pi\tau d_{sep}^4} \tag{4.17}$$

$$F = \frac{md_{out}^4 v_{rel}}{\tau d_{sep}^4}\left(\frac{d_{sep}}{d_{out}}\right)^8 \tag{4.18}$$

$$F = \frac{mv_{rel}}{\tau}\left(\frac{d_{sep}}{d_{out}}\right)^4 \tag{4.19}$$

and likewise, for collision avoidance, eq. 4.22.

$$F = \frac{3\mu_o(\sqrt{\frac{md_{out}^4}{6\cdot 10^{-7}}}\left(\frac{d_{in}}{d_{out}}\right)^4\left(\frac{d_{in}}{d_{sep}}\right)^3\sqrt{v_{rel}})^2}{2\pi\tau d_{sep}^4} \tag{4.20}$$

$$F = \frac{md_{out}^4 v_{rel}}{\tau d_{sep}^4}\left(\frac{d_{in}}{d_{out}}\right)^8\left(\frac{d_{in}}{d_{sep}}\right)^6 \tag{4.21}$$

$$F = \frac{mv_{rel}}{\tau}\left(\frac{d_{in}}{d_{out}}\right)^4\left(\frac{d_{in}}{d_{sep}}\right)^{10} \tag{4.22}$$

Once the dipoles have been set to the appropriate values, the new state is calculated using equations 4.8 through 4.3.

4.2.3 Program Architecture

The setup and control portions of the scenarios are in the MATLAB m-files `pairedsats.m` (see appendix C.5), `linearsats.m` (see appendix C.3), `squaresats.m` (see appendix C.4), and `tetrasats.m` (see appendix C.6). These scripts consist of a brief setup section, where initial values are given, followed by the main control loop, which will be discussed in more detail in section 4.2.2.

The state is updated at each time step using `tstepSat.m` (see appendix C.1), which contains the magnetic force equations and the kinematic equations needed in any scenario. Any constant accelerations or additional forces are added in the scenario files.

Output is done using `plotSat.m` (see appendix C.2), which has a selection of the various diagnostic plots used in the creation and debugging of the code, in addition to the outputs necessary for understanding the results. `linearsats.m` and `squaresats.m` have additional specialized plots as well, and an additional script, `slice.m` (see appendix C.5.1), produces plots of the four satellites at a given point in time, to supplement the other plots for `pairedsats.m`.

Each satellite was given a mass of 50 kg, and moments of inertia $I_{xx} = I_{yy} = I_{zz} = 100$ kg·m^2. Mass is adjusted for in the force calculations, so changing the mass does not change the behavior of the satellites. The ratio of mass to moment of inertia does change the rate of angular momentum buildup somewhat, though. These masses and moments were chosen as round numbers within the range of typical satellites in μ-EMFF setups, as in chapter 2.

To simplify calculation the satellite volume was neglected. Since the code has no collision detection, the main use of volume would be in setting the minimum allowable separation distance between satellites. In these scenarios, aside from `tetrasats.m`, this minimum distance ranges from 0.6 m to 0.2 m, which would imply a rather small satellite.

In `linearsats.m`, the satellites are arranged in a line, one meter apart, as illustrated in fig. 4.1. Satellite A has a forward acceleration imposed, and can be positioned at any point along this line. All motion in this scenario is along this line. Since the satellites start out only one meter apart, the minimum separation distance is quite small in this case, so that the satellites have room to move before the collision avoidance maneuvers begin.

Going from one dimension to two, `squaresats.m` has the four satellites arranged in a square, and moving in that plane, as in fig. 4.2. The initial setup is a square, $\sqrt{2}$ meters to a side, with the vertices aligned so as to point along the y and z axes.

`pairedsats.m` implements a three-dimensional scenario, wherein four satellites have random starting positions, within $\pm 4m$ of the origin, and random velocities, within $\pm 0.04\frac{m}{s}$. Satellite A is given a position and velocity such that the center of gravity is at the origin, and the average velocity is zero.

`tetrasats.m` adds to the previous scenario a specific formation — a tetrahedron — which it is to maintain. This is done by increasing the minimum distance

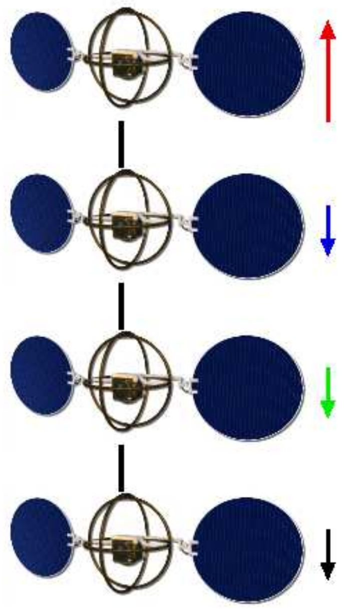


Figure 4.1: Linear configuration. See also fig. 4.5

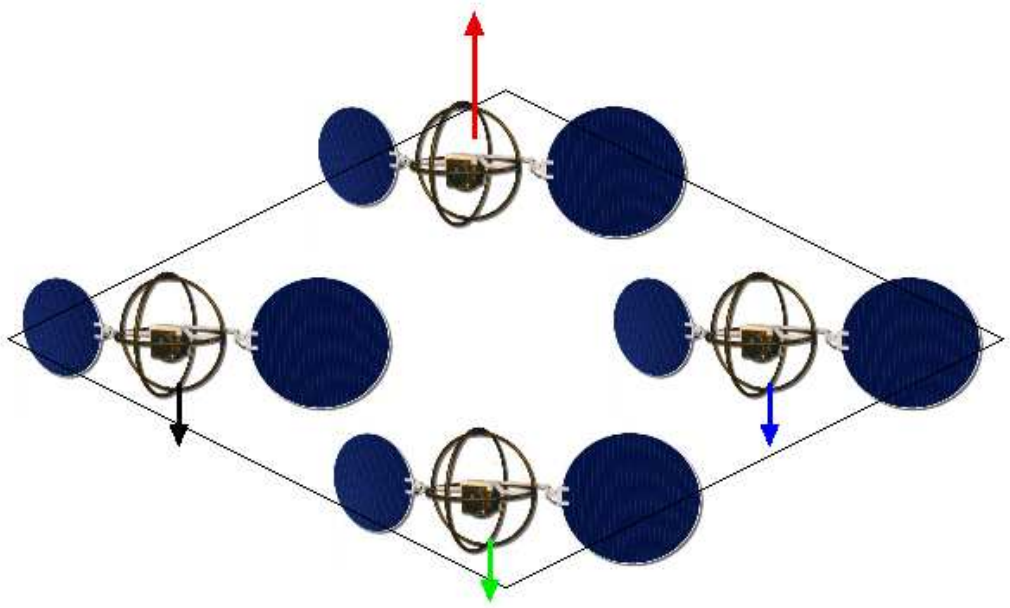


Figure 4.2: Square configuration. See also fig. 4.11

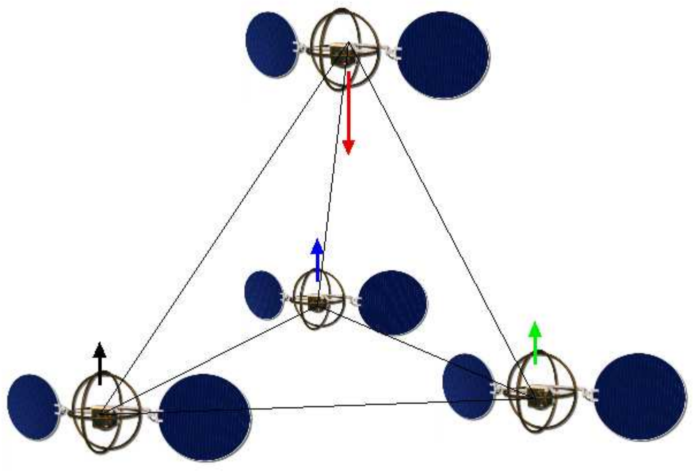


Figure 4.3: Tetrahedral configuration. See also fig. 4.20

to just under the maximum distance, thus requiring the separation between the satellites to be nearly constant. With four satellites, this necessitates a tetrahedron. This setup is illustrated in fig. 4.3

If one satellite is given a constant acceleration — as from a rocket motor — the center of gravity of the whole system will begin to move; the resulting plots of motion would be somewhat hard to read. This situation could come up if a cluster of satellites leaves each major function, such as propulsion, to one satellite. If the system is working, all satellites will be accelerating together. In order to clarify the plots, the center of gravity is kept to the origin of the coordinate system, by giving the accelerated satellite a constant acceleration of $0.75 \cdot a_{cons}$, and each of the other satellites an acceleration of $-0.25 \cdot a_{cons}$, as illustrated in fig. 4.4. This keeps the output in the center of mass frame.

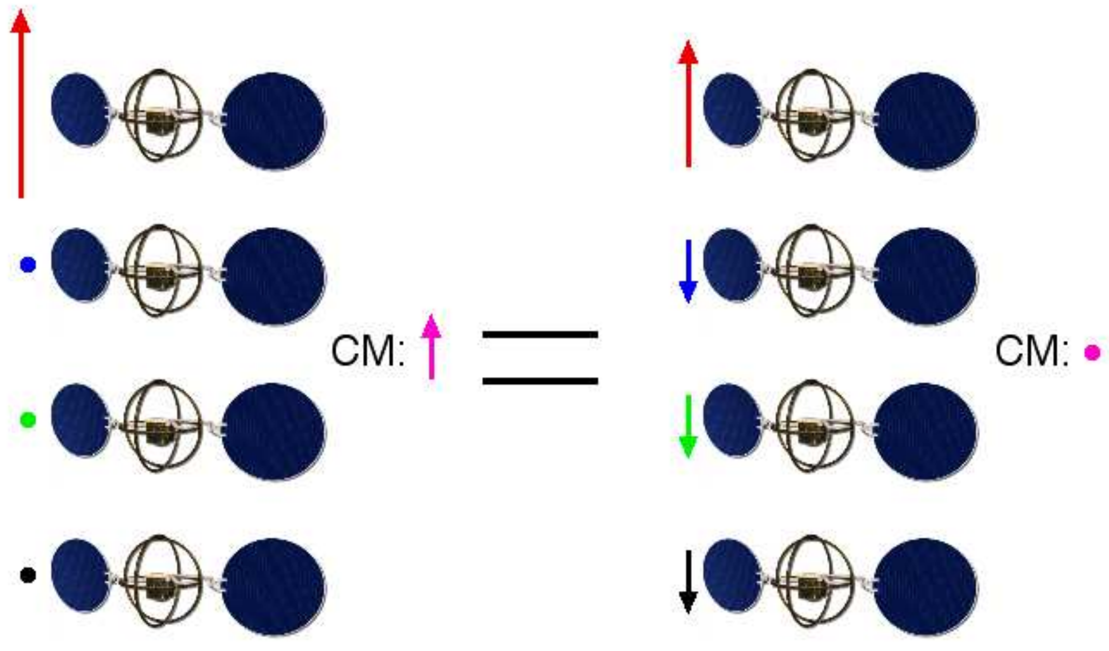


Figure 4.4: Center of mass frame compared with inertial frame

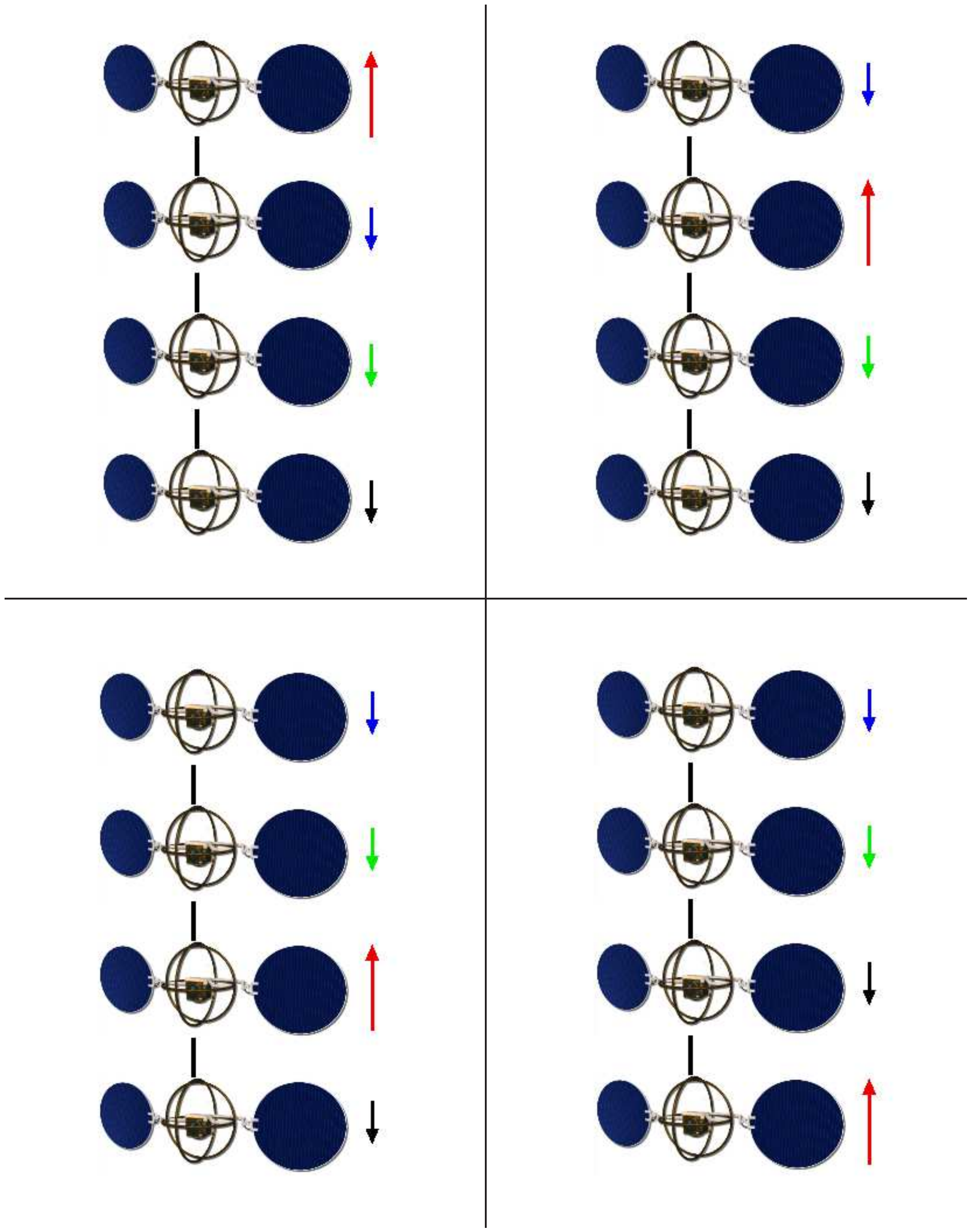


Figure 4.5: Linear configurations:
Top left: Driven satellite (A) frontmost (see fig. 4.9).
Top right: Satellite A second (see fig. 4.8).
Bottom left: Third satellite is A (see fig. 4.7).
Bottom right: Sat A in rearmost position (see fig. 4.6).

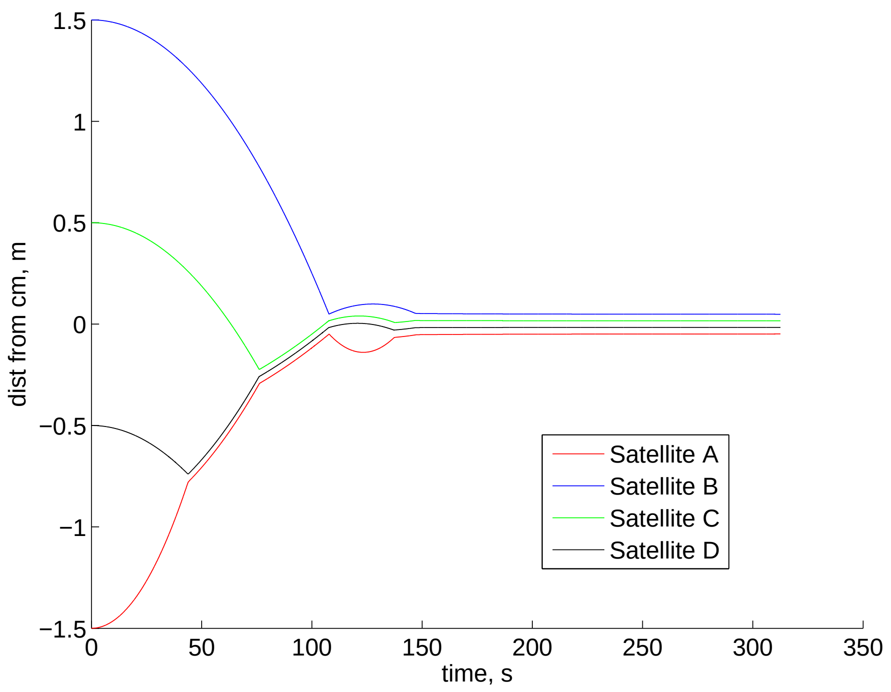


Figure 4.6: Motion of satellites in linear formation, thrust at back. The red satellite moves forward relative to the other three, and ultimately pushes them all forward together.

4.3 Results

4.3.1 Linear

The simplest scenario studied using the paired satellites technique consisted of a line of four evenly-spaced satellites, constrained to a one-dimensional line. (Control script `linearsats.m` in Appendix C.3.) In it, the four satellites are distributed along a line, and move only along that line. One of these four satellites has a constant acceleration. The behavior of the four-satellite system varies considerably, depending on which position the accelerated satellite (satellite A) has.

If satellite A is the rearmost satellite (the satellites being arranged, front to back, B C D A; in the MATLAB code, this is `linearsats(4)`), the thrusting satellite A moves forward until it comes within the minimum distance to the next satellite in line, D. They rebound, initiating a series of successively smaller bounces with C and

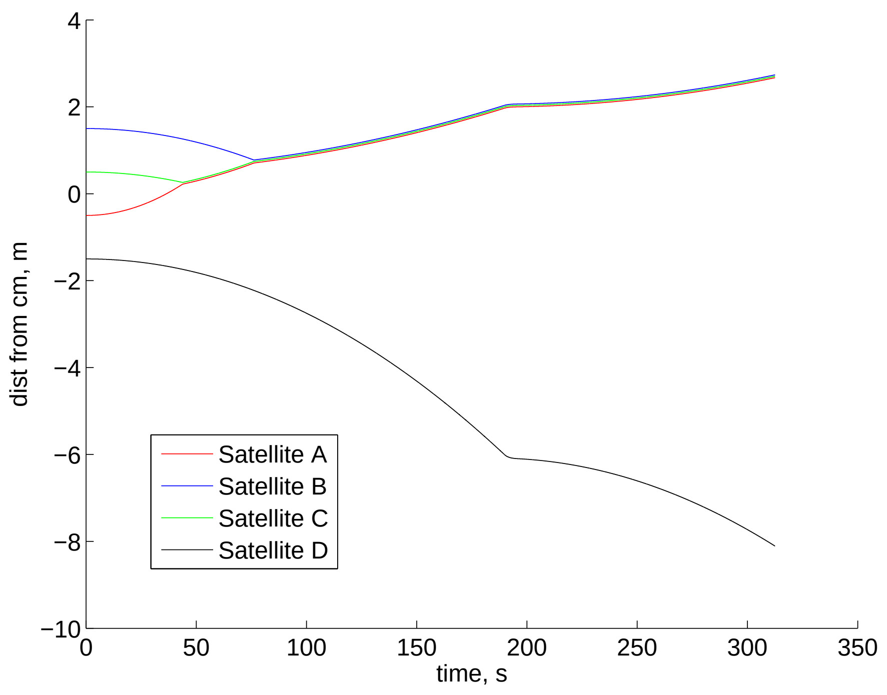


Figure 4.7: Motion of satellites in linear formation, thrust second to back. The one satellite behind red gets left behind, with only one insufficient attempt at recovery.

B, until finally, all settle down to a separation distance right around the minimum allowable distance, as in fig. 4.6. At steady state, there is a series of tiny, momentary pulses (see fig. 4.10, bottom right) pushing the satellites apart, and a steady force from behind — the satellite with the constant force — pushing them together.

As the position of the accelerating satellite moves forward, the behavior changes significantly. Fig. 4.7 shows the situation with satellite A being the second from the back (`linearsats(3)`). The satellites ahead of the thrust — B and C — behave as in the rearmost case, coming to a stable arrangement, bouncing along in front, while the rear satellite — D — never can catch up. The small bounces sufficiently occupy the system with collision avoidance that only one attempt can be made to bring the back satellite forward, and it is insufficient. Possibly, were the dipole strengths increased for the too-far case, the one pulse would be sufficient to bring them together, and the system could be made stable.

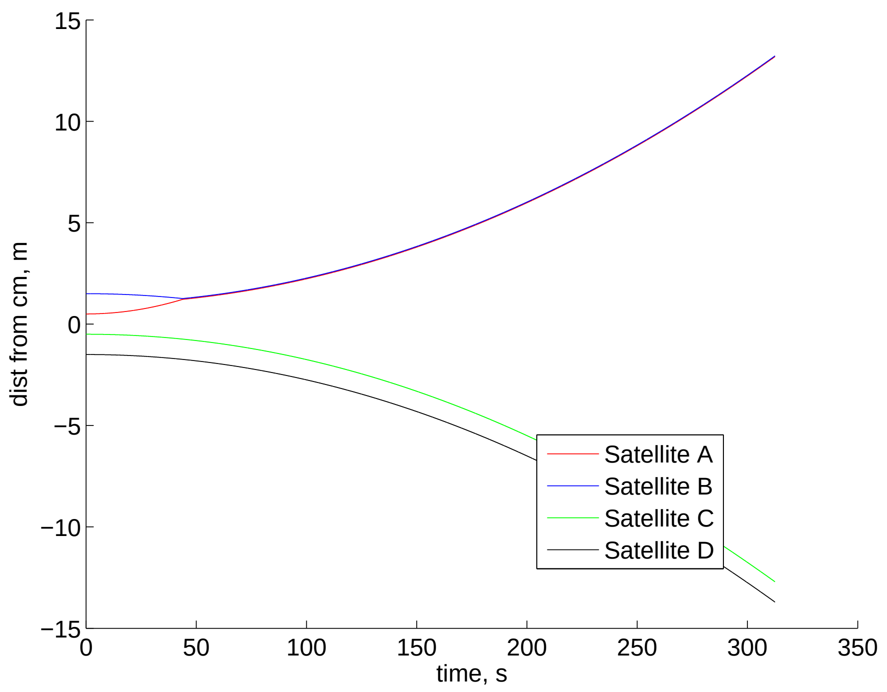


Figure 4.8: Motion of satellites in linear formation, thrust second to front. The two front satellites interact so much that neither of the other satellites gets a chance to catch up.

Fig. 4.8 shows the system with A being the second satellite from the front (`linearsats(2)`). The frontmost satellite, B, is continually being pushed on by the driving satellite, A. Because collision avoidance must take priority in this control scheme, the dipoles are never given the opportunity to close the rearward gap.

When the thrust is at the front, as in fig. 4.9 (`linearsats(1)`), the first maneuver that is done is between the frontmost and rearmost satellites. This slows the frontmost satellite quite a bit, and accelerates the rearmost satellite, which then collides with its forward neighbors. The three rear satellites then begin their own series of tiny bounces, much as the satellites in the rearmost-drive case. These bounces are punctuated by frequent tugs from the frontmost satellite, which serve to keep all four together.

The key finding here is that the system remains stable as long as one pair or triad of satellites does not get locked into a collision-avoidance cycle.

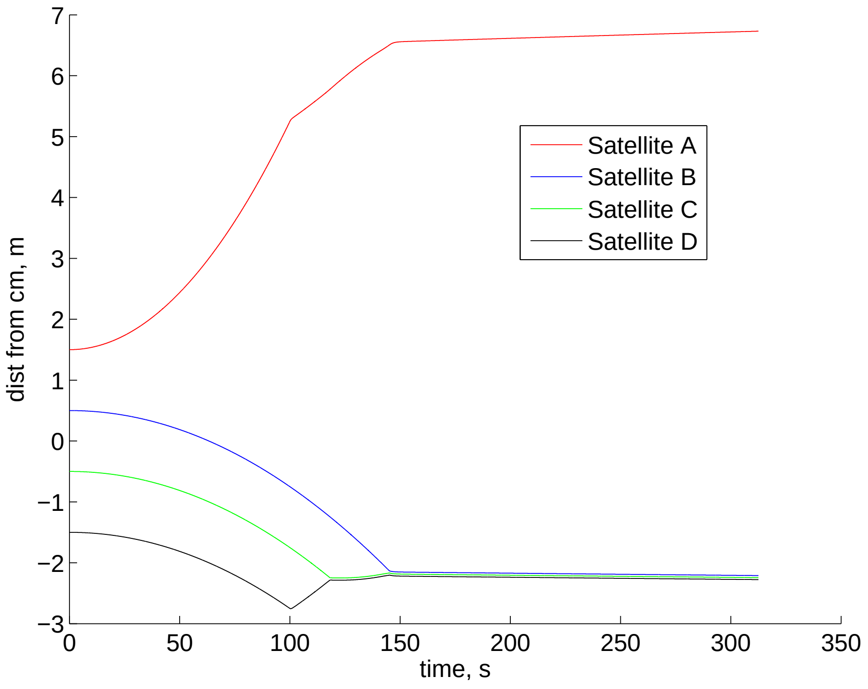


Figure 4.9: Motion of satellites in linear formation, thrust at front. The front satellite pulls the rear satellites forward, leading to a stable configuration.

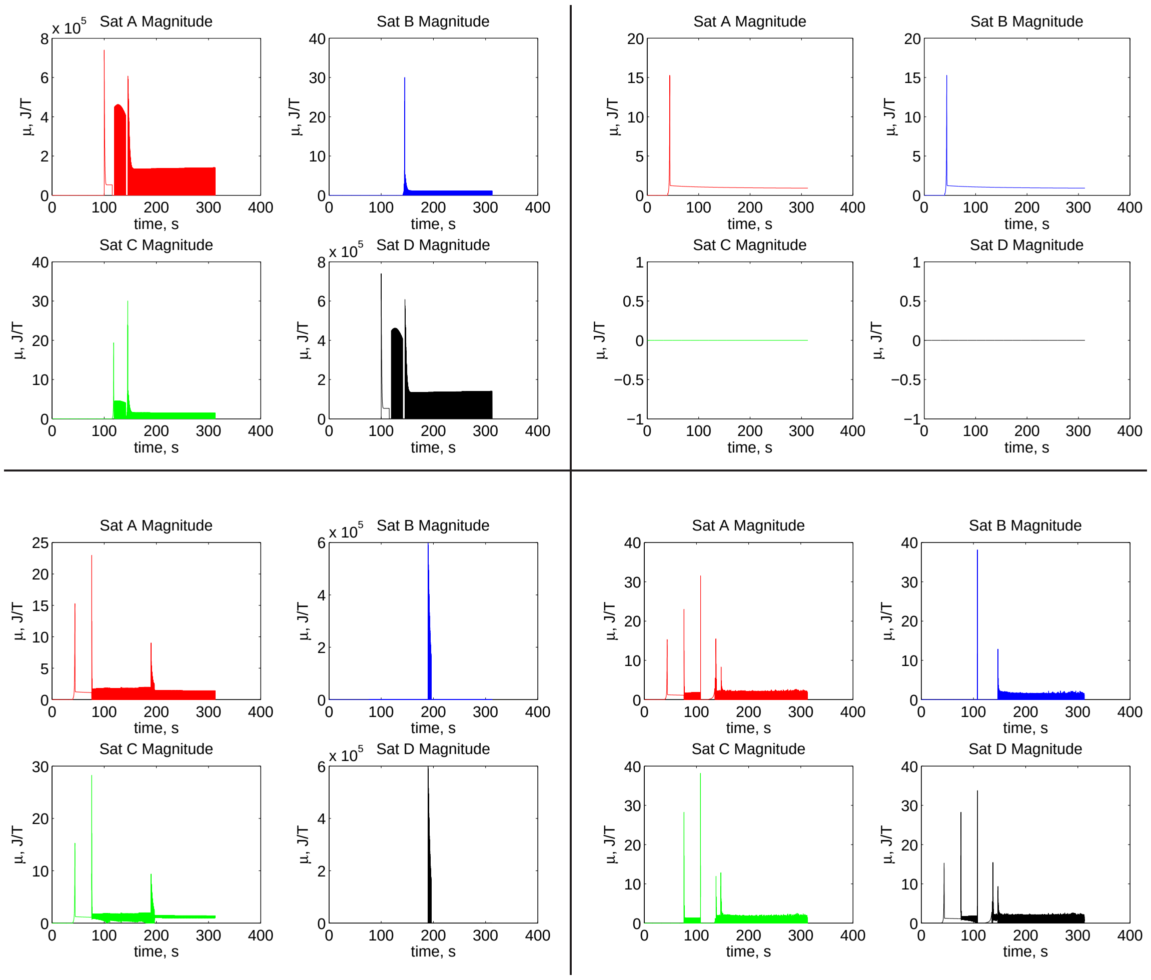


Figure 4.10: Dipole moment strengths, linear formation.
Top left: Driven satellite (A) frontmost (see fig. 4.9); primary maneuvering done by front (A) and rear (D) satellites, corresponding to their stronger dipole moments.
Top right: Satellite A second (see fig. 4.8); note A and B's constant interaction, precluding any other maneuvering.
Bottom left: Third satellite is A (see fig. 4.7); spike in B and D satellites corresponds to the one attempt at bringing D into formation.
Bottom right: Sat A in rearmost position (see fig. 4.6); initial spikes correspond to bounces, followed by low-level oscillations.

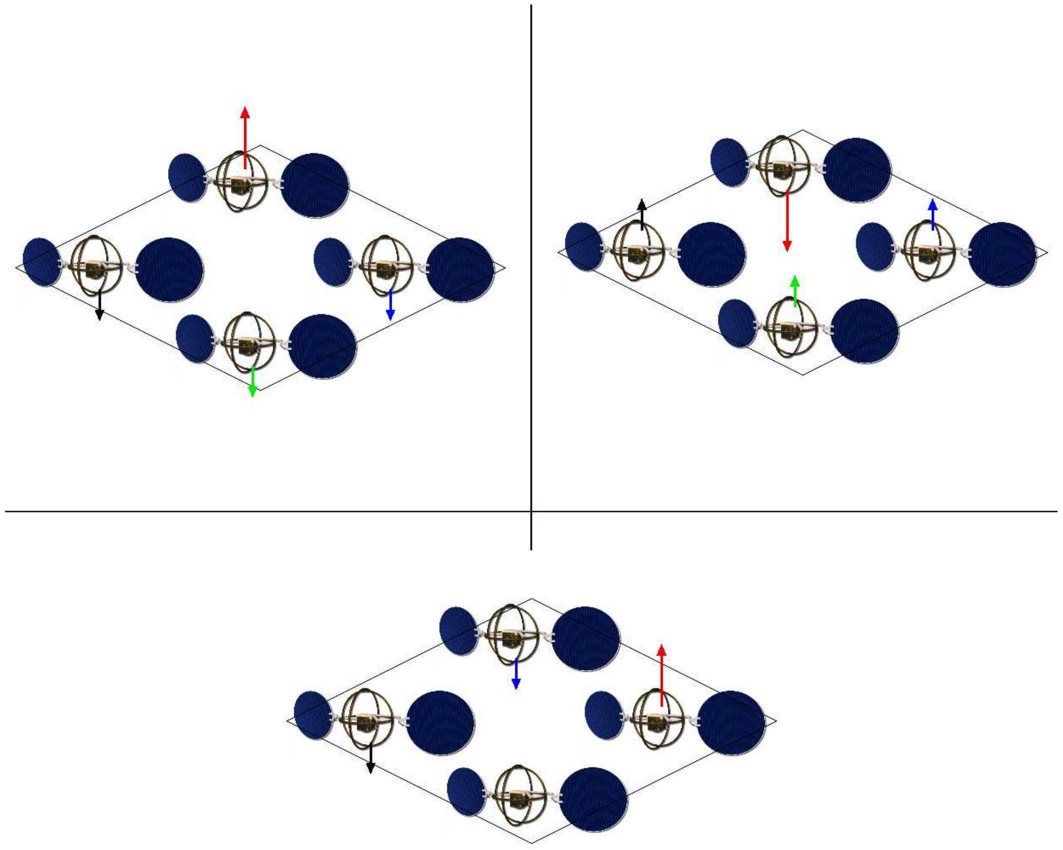


Figure 4.11: Square configurations:
Top left: Driven satellite thrusting forward. Results in figs. 4.12 and 4.13.
Top right: Driven satellite thrusting backward. Results in fig. 4.14.
Bottom: Driven satellite thrusting sideways. Results in fig. 4.15.

4.3.2 Square

The next scenario has four satellites, arranged in a plane, in a square formation. Satellite A, again, is accelerating, in the direction away from the center of the square. Satellite A is at front, with C opposite A, at the back. Satellites B and D are on either side.

As can be seen in figure 4.12, satellite C gets pulled forward first. Once it comes closer to the front, the two side satellites start getting pulled toward the center. The side satellites B and D, though, do not remain close together, but repel each other, and bounce outward before being pulled back in again. This cycle repeats several times. Meanwhile, the front satellite maintains itself near the maximum separation distance with the other three by dint of a series of continual small pulses. Eventually, when the side satellites return to the center, being pulled toward the leading satellite, the interaction of the three following satellites, now in very close proximity, have sufficient collision-avoidance pulses to let the leading satellite slip forward.

When the three following satellites finally start to move away from each other, satellite A is far enough away to require a rather large dipole moment to recapture. This large moment, on one pair of satellites at a time, breaks the symmetry of the formation. The motion quickly winds up being a stable, but somewhat chaotic, oscillation by each satellite, with satellite A out in front in the direction of thrust.

The important thing to note here is that the system can keep the four satellites together as long as close multi-satellite interactions are not taking place. When satellites B, C, and D come very close to each other, satellite A begins to slip away.

If the thrust direction is reversed (as in fig. 4.14), with the accelerated satellite pointed directly inward, the driven satellite hits the opposite satellite head-on, and the ensuing constant collision-avoidance force distracts the system from keeping the satellites together. The two side satellites are completely ignored, and the two

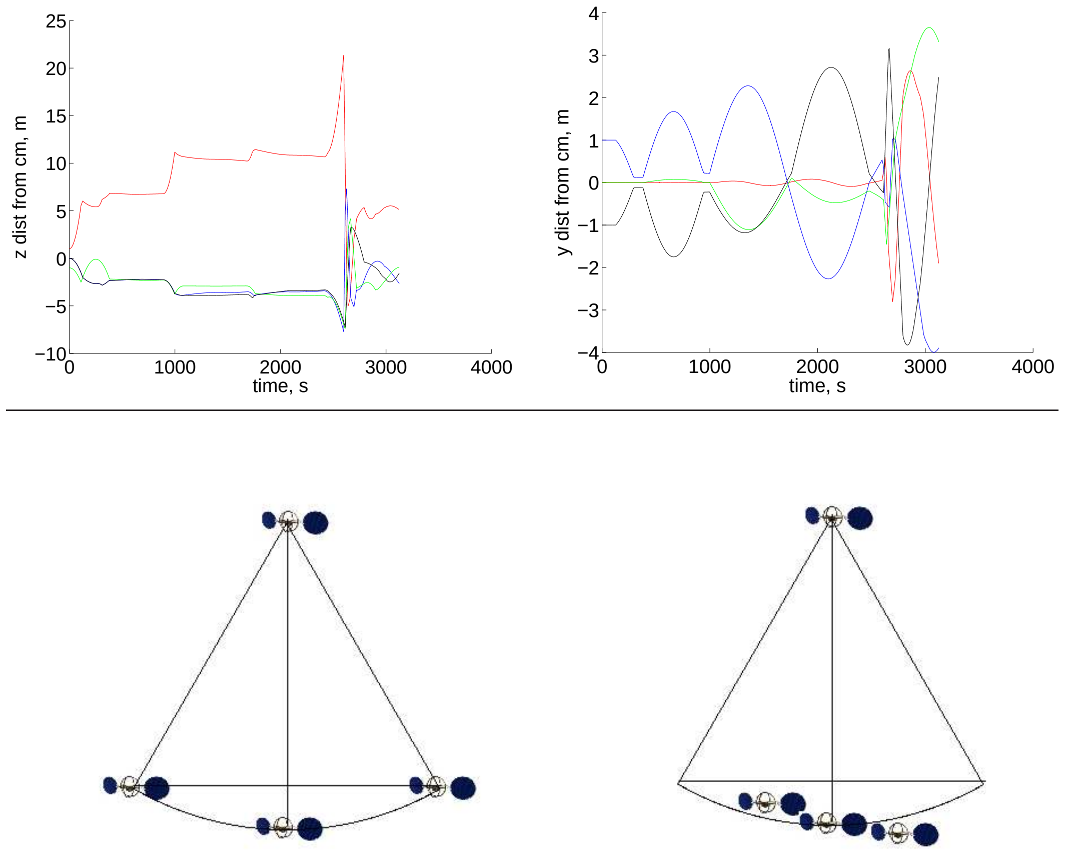


Figure 4.12: Motion of satellites in square formation. Upper plots show satellite position in time by axis; lower plot shows satellite position tracks: during the symmetrical phase, the two side satellites move in an out in tandem along the curve shown, equidistant from the leading satellite. Eventually, symmetry is broken, as one begins to lag behind the other.

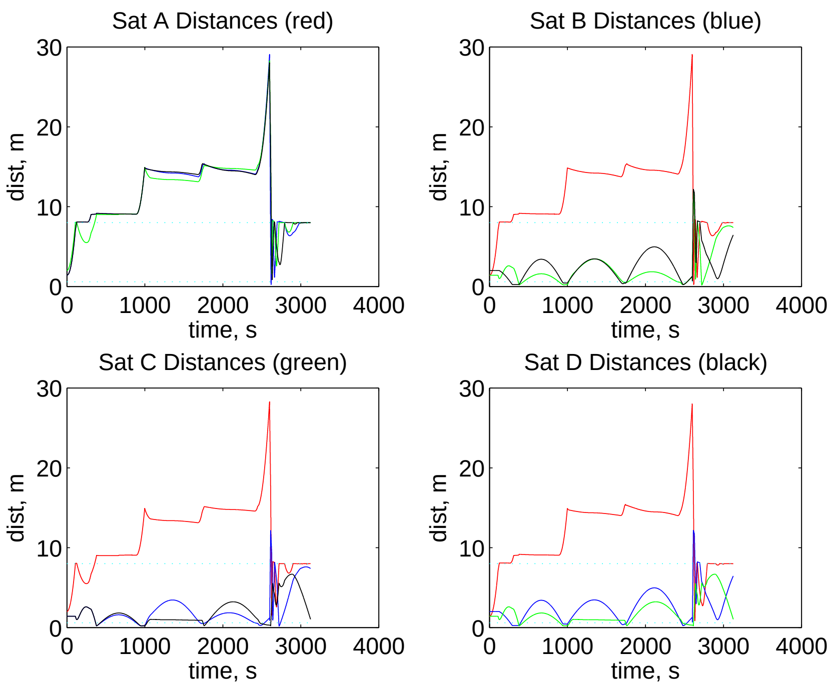


Figure 4.13: Separation distances of satellites in square formation. Minimum and maximum distances marked in cyan dotted lines.

colliding satellites fly off together.

If the thrust is off to the side, as in figure 4.15, the motion quickly winds up being a stable, but somewhat chaotic, oscillation by each satellite, as in the first case.

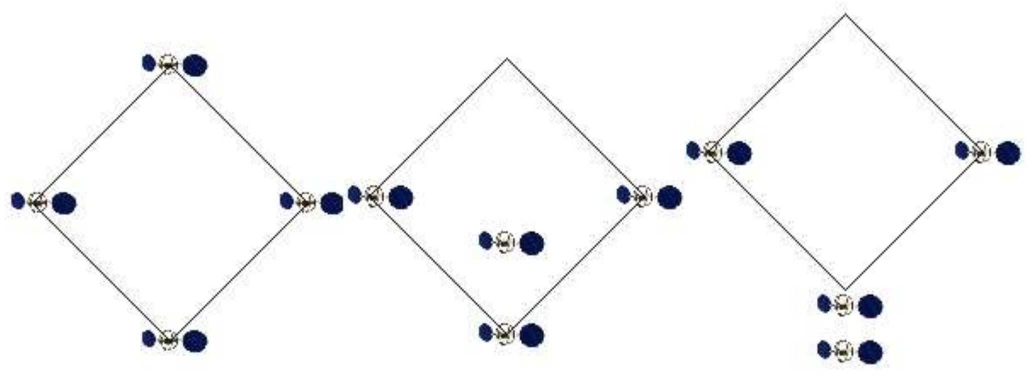
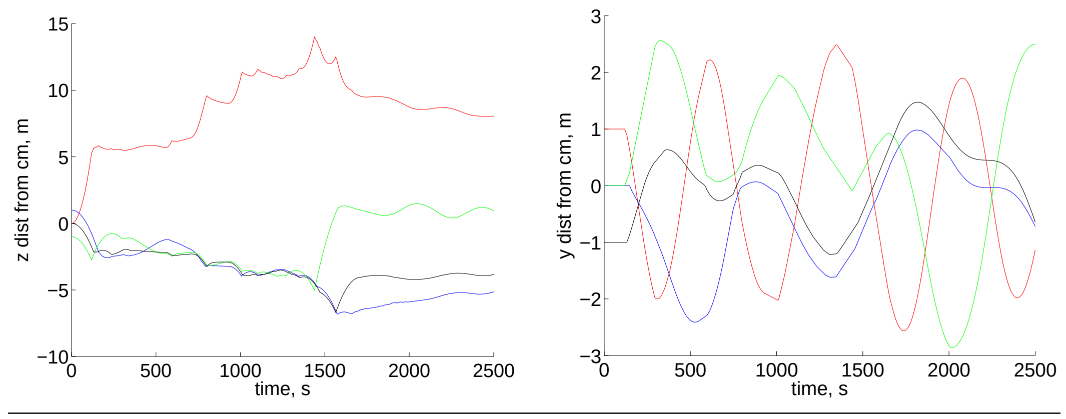


Figure 4.14: Motion of satellites in square formation, reverse thrust. A pushes C backwards; B and D remain stationary, which is the same as moving forward relative to the center of gravity.



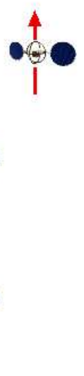


Figure 4.15: Position of satellites in square formation, side thrust. Oscillatory motion is chaotic, but stable. Symmetry is broken immediately.

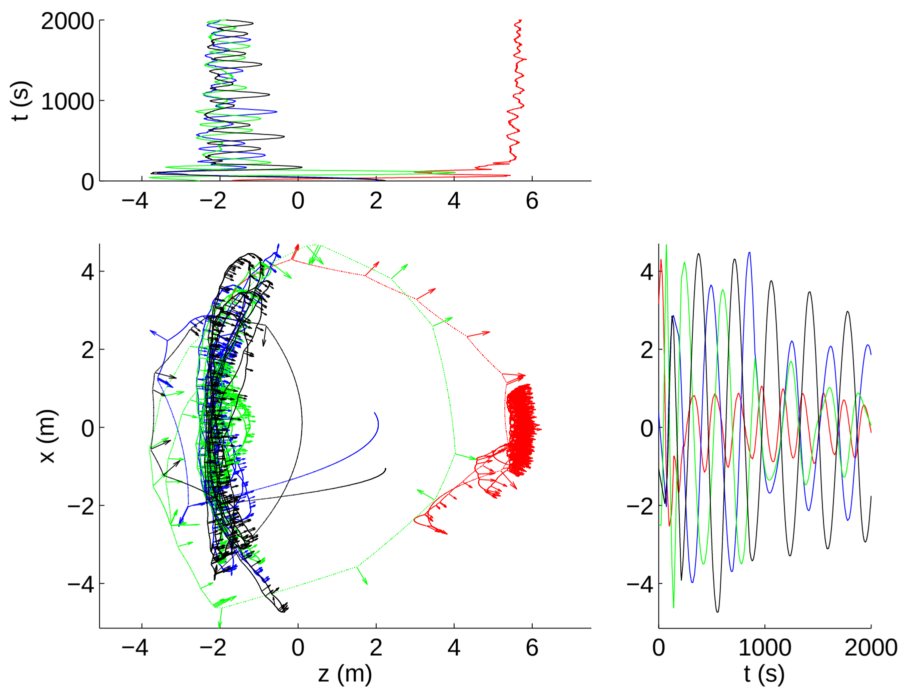


Figure 4.16: X-Z plane, general case; Y-Z plane has a similar appearance.

4.3.3 Paired Satellites — General Case

The stable, but chaotic oscillations are exemplified in the general case. In `pairedsats.m`, each of the four satellites is given a random starting location, and a small, random starting velocity and direction. The results are fairly consistent — they stay within the bounds set out for them, making looping motions. The thrusting satellite stays out ahead of the other three, as shown in fig. 4.16.

In the general case, the random initial formation changes into a rotating tetrahedron, somewhat squished on one face. The accelerated satellite takes a position at one vertex, staying at about the maximum distance from each of the other three. Meanwhile, the other three make up the opposite face; a triangle, albeit one with continually changing side lengths. (The changing distances are shown in fig. 4.19) The three remaining faces are isosceles triangles, with the rear edge shorter than the two that meet the accelerating satellite. The whole tapered tetrahedron then rotates about an axis which passes near to the accelerated satellite. This is especially visible

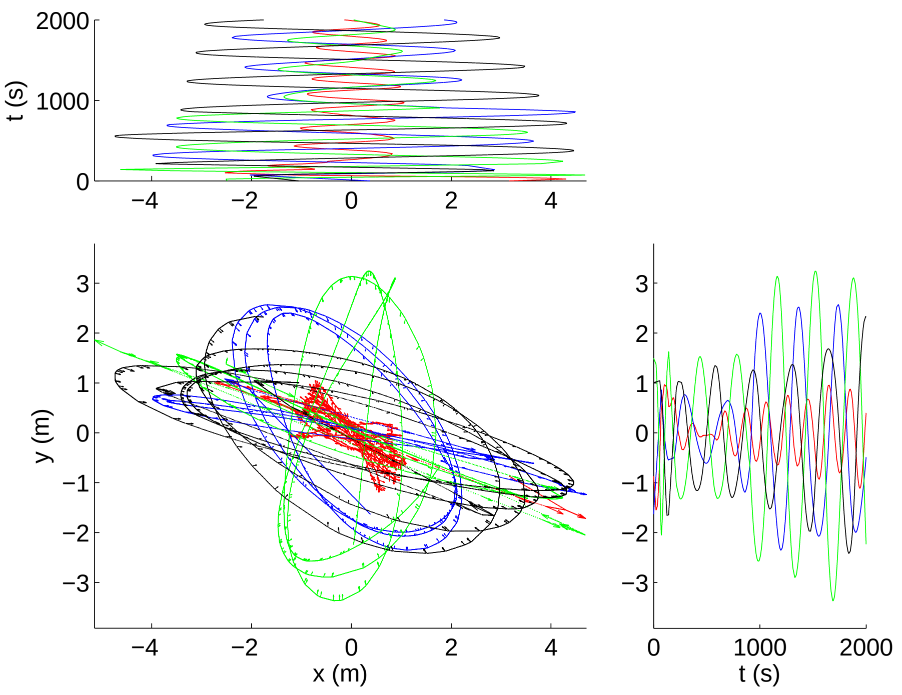


Figure 4.17: X-Y plane, general case, perpendicular to the axis of rotation of the tetrahedron. See also fig. 4.18.

in fig. 4.17, which shows the plane perpendicular to the thrust direction. Figure 4.18 shows one half of loop in the same plane, for clarity.

Interestingly, satellite A oscillates with about twice the frequency of the other three. It may be that while the four satellites revolve about the central axis of their tetrahedron, the central axis itself nutates about the direction of thrust, at approximately the same rate. For the three following satellites, this would not be noticeable, but for the leading satellite, which is very close to the central axis and to the direction of thrust, the amplitudes would be on the same order of magnitude, and thus noticeable.

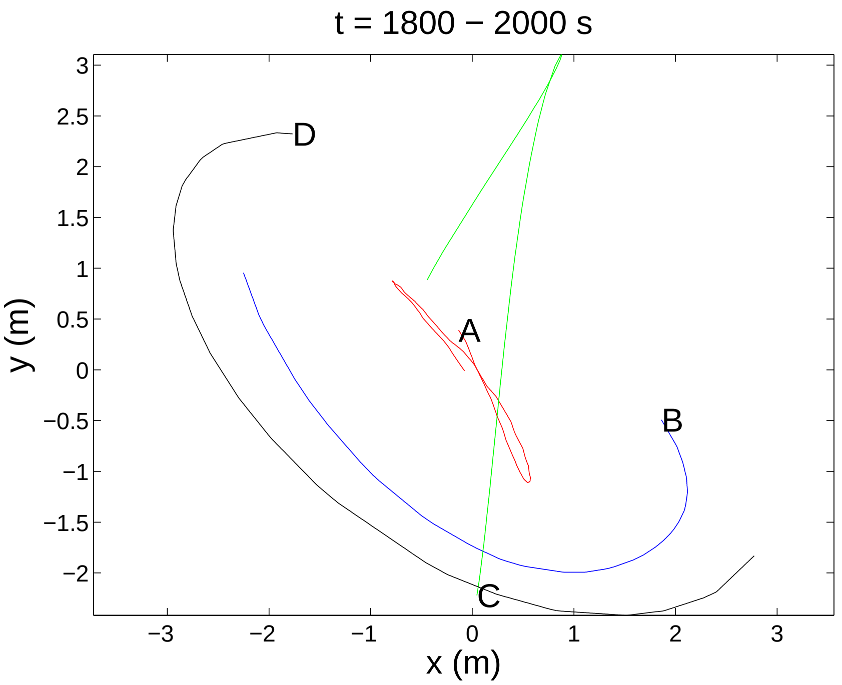


Figure 4.18: One half-loop of the satellites, X-Y plane. See also fig. 4.17.

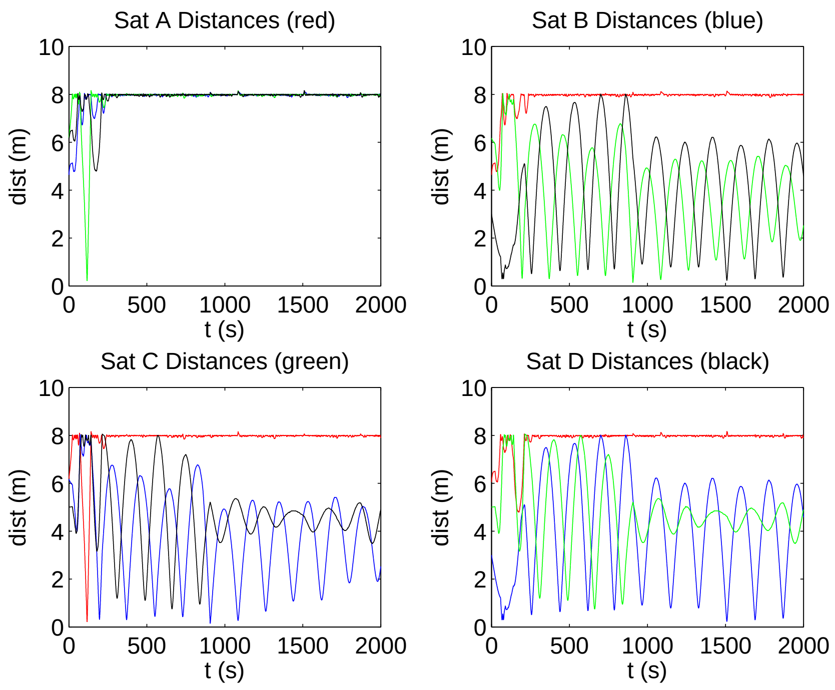


Figure 4.19: Satellite distances, general case. The driven satellite (A) stays at a fairly constant distance ahead of the other three, which loop around each other, following behind A.

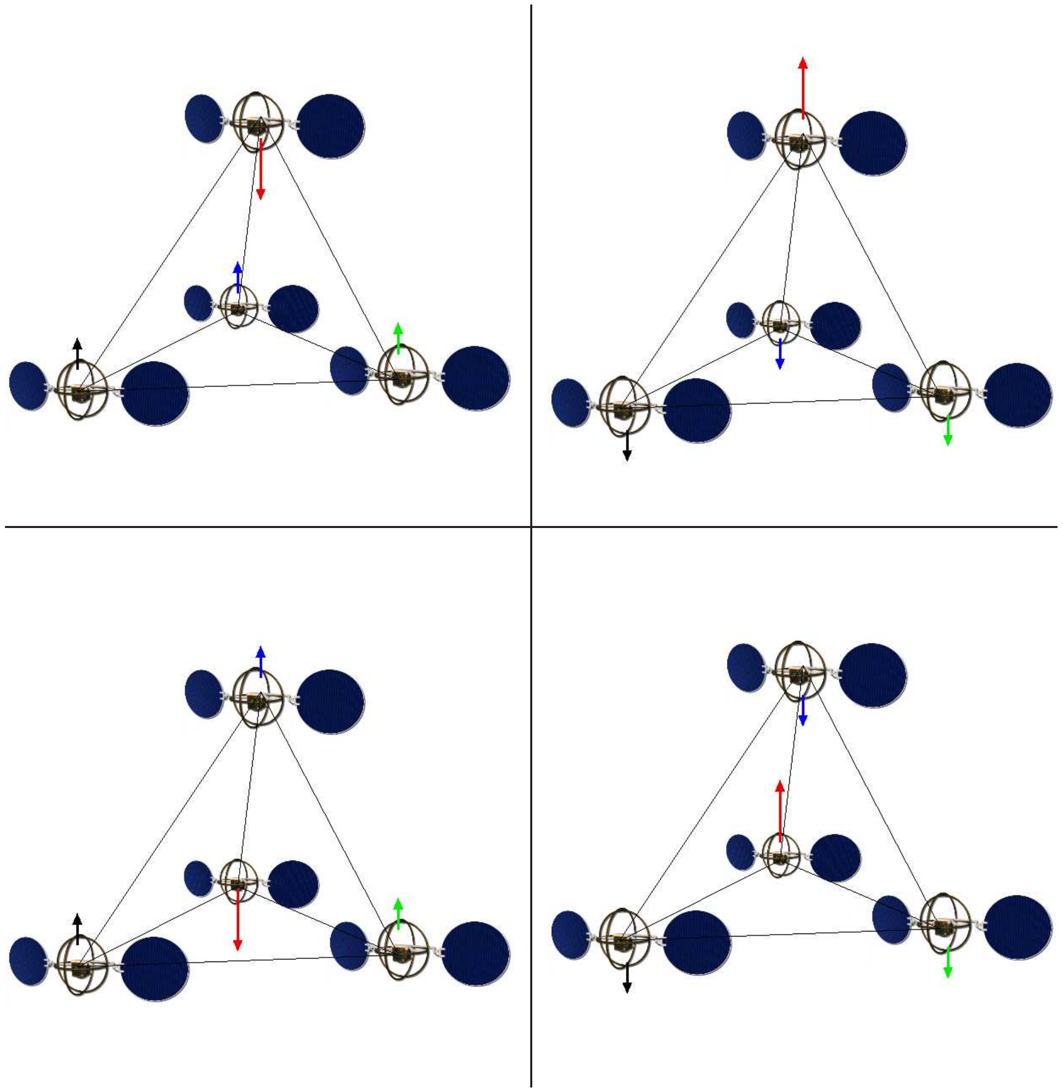


Figure 4.20: Tetrahedral configurations:
Top left: Thrusting satellite steered inward, as in figs. 4.21 through 4.27.
Top right: Thrusting satellite steered outward, as in fig. 4.28.
Bottom left: Thrusting satellite steered out and to the side, as in figs. 4.29 through 4.32.
Bottom right: Thrusting satellite steered in and to the side, as in see figs. 4.33 through 4.36.

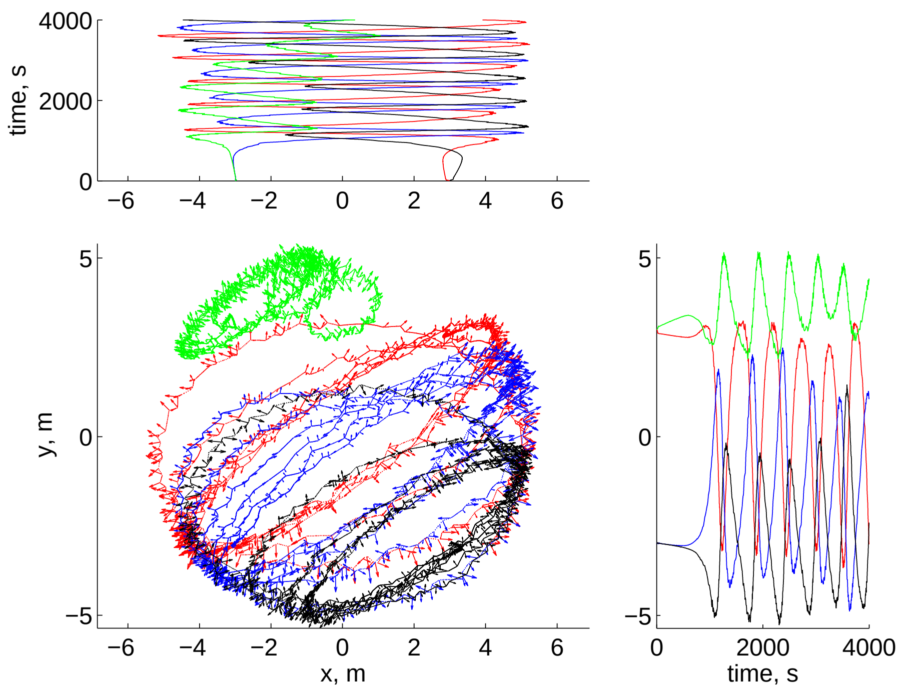


Figure 4.21: X-Y plane, tetrahedral formation

4.3.4 Tetrahedron

The most complicated test done with the paired-satellite technique was holding a particular formation. In this case, the simplest four-satellite formation was used: a regular tetrahedron, with one satellite at each vertex. As before, one of the four satellites had an acceleration imposed on it.

No attempt was made to maintain the orientation of the tetrahedron; indeed, it was not maintained, except under very specific thrust direction. However, in every case, the satellites remained in a tetrahedron, after a brief initial spike in separation distance (see fig. 4.27). The motion of the four satellites in the three planes are shown in figs. 4.21, 4.22, and 4.23.

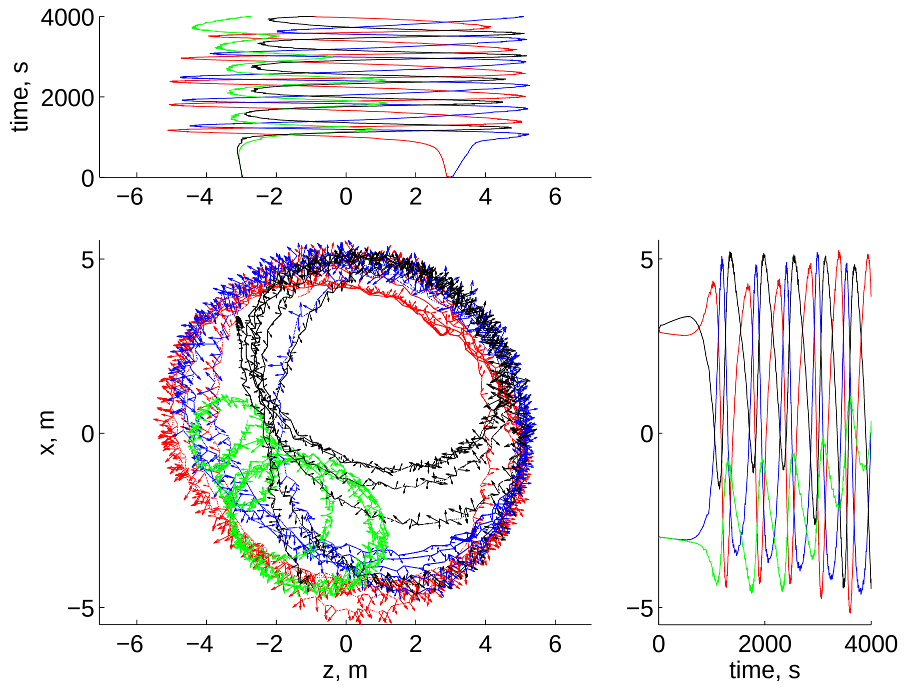


Figure 4.22: X-Z plane, tetrahedral formation

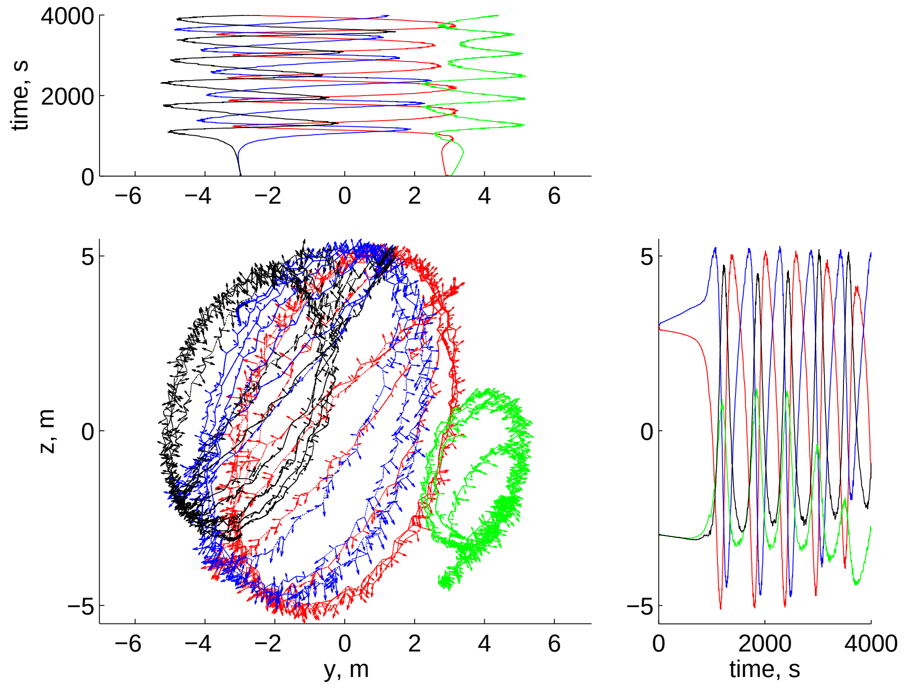


Figure 4.23: Y-Z plane, tetrahedral formation

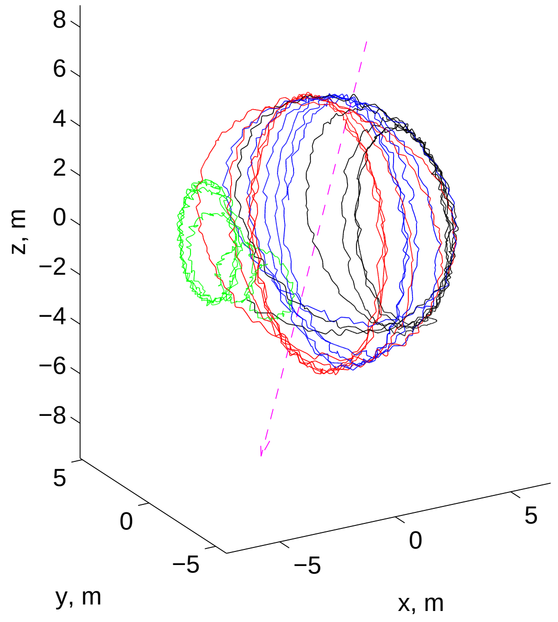


Figure 4.24: Rotation of the tetrahedron over time. The lower left node in the red satellite's 'orbit' points in the (-1,-1,-1) direction, which is the direction of thrust.

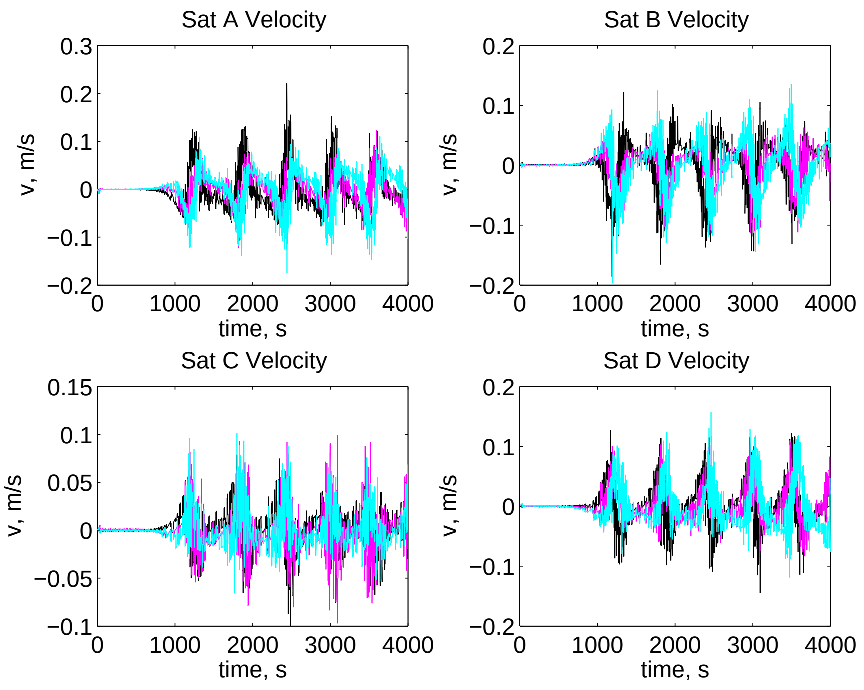


Figure 4.25: Satellite velocities, tetrahedral formation, relative to system center of mass. Cyan, pink, and black correspond to v_x, v_y, and v_z, respectively. Spikes correspond to reversals of direction and the extrema of the loops made by the satellites.

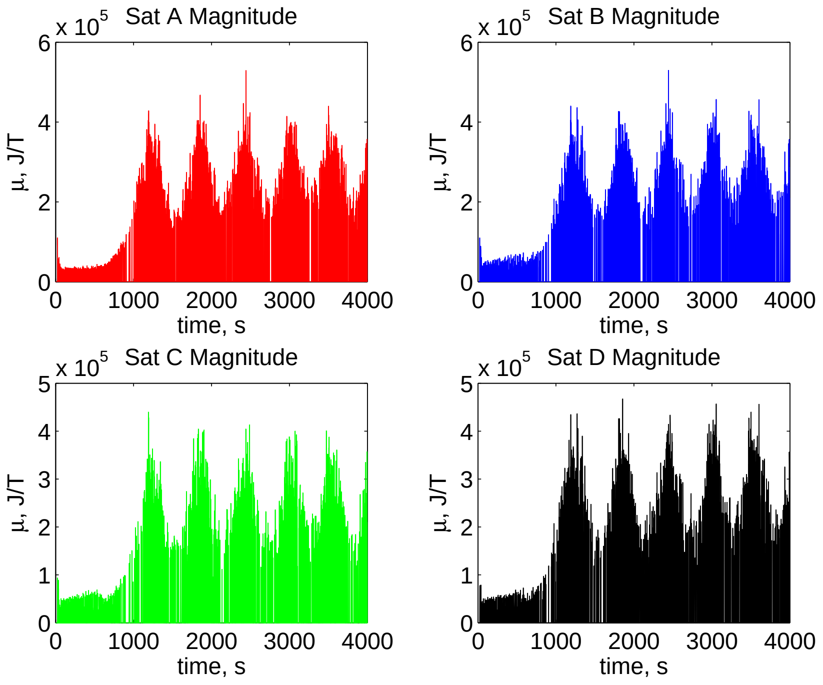


Figure 4.26: Satellite dipole magnitudes, tetrahedral formation. Interactions are intermittent, but happen very frequently, with spikes at equal frequency to the loops.

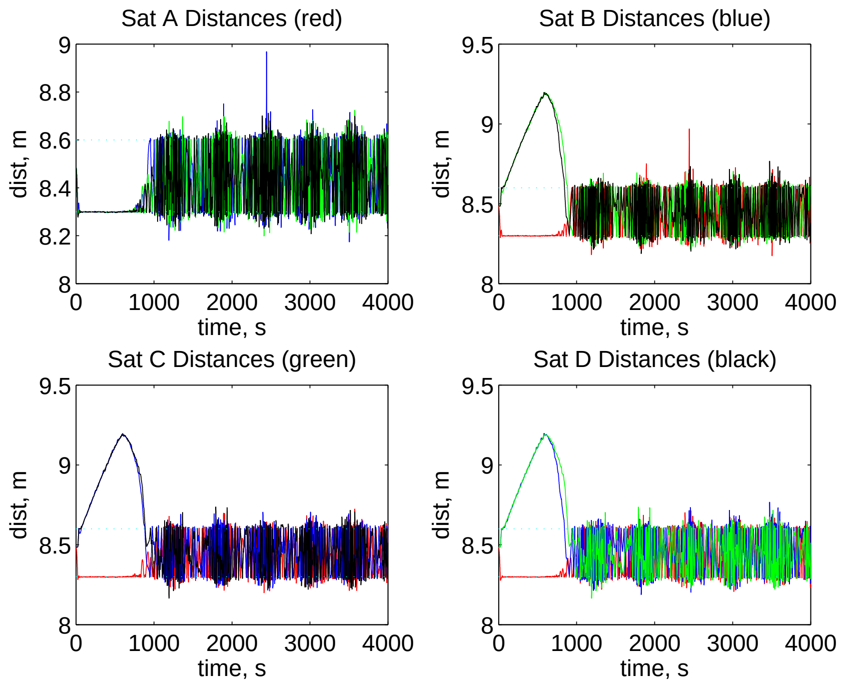


Figure 4.27: Satellite distances, tetrahedral formation. After some initial divergence, satellites stay within the allotted distances.

If the leading satellite, A, is steered inward, the end result is not a smooth translation of the formation, but an in-formation tumble. The four satellites move about as on the surface of a sphere, with the satellite A tracing out a great circle including the point on a line from the center of the tetrahedron in the direction of thrust (as shown in fig. 4.24), and the other three tracing out different latitude lines. Velocities and dipole magnitudes tend to come in wide pulses, as shown in fig. 4.25 and fig. 4.26.

If the driven satellite is steered directly away from the center of the tetrahedron, on the other hand, the shape and orientation are maintained nearly exactly, as in fig. 4.28. The three following satellites B, C, and D, pair with satellite A in rapid succession, thus pulling themselves forward. As they are all chasing satellite A, they are also moving toward each other; once they get too close, occasional follower-to-follower repulsive pulses are mixed in with the leader-follower attractive pulses, keeping everything stable.

In the case where satellite A starts out on the side of the formation, rather than the front, the situation is rather different. Rather than making a loop, or staying in their orientation, the satellites travel in C-shaped arcs. Two cases were examined; one in which the satellite was driven out and to the side (see figs. 4.29 – 4.32), and the other where the satellite was driven in and to the side (see figs. 4.33 – 4.36). In the out-and-to-the-side case, the arc was noticeably shorter than the in-and-to-the-side case.

Comparing all four cases, it can bee seen that the satellite thrusting will tend to follow a circular arc where at the midpoint, the thrust is directed directly away from the center point, and with the starting point at one end of the arc. Thus, when the driven satellite starts out at the front, there is no arc, when it starts out at the back, it makes a great circle, and when it starts out on the side, it makes a C-shape.

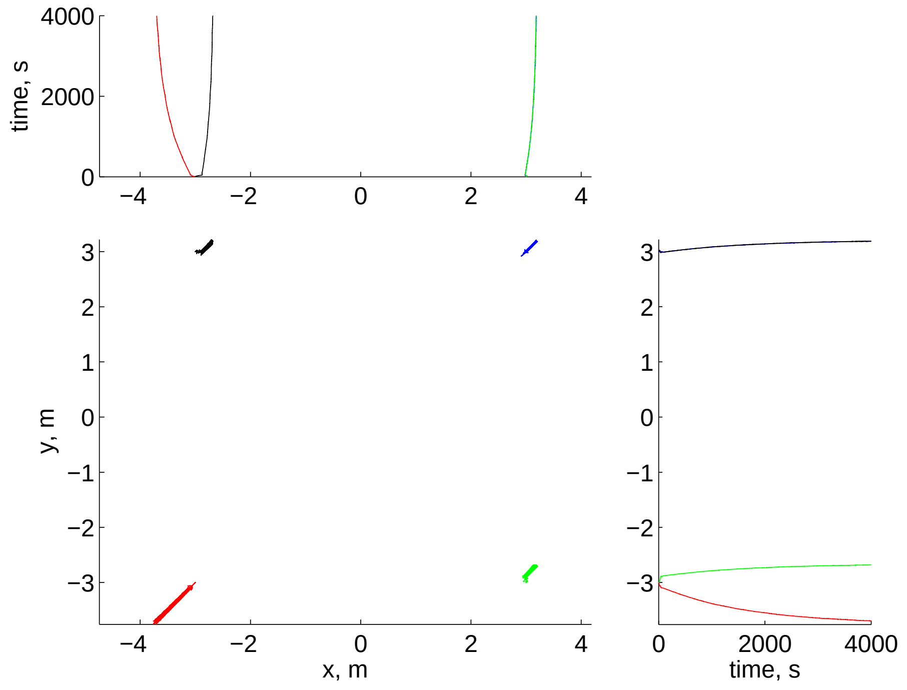


Figure 4.28: X-Y plane, tetrahedral formation, outward thrust. Satellites move to the maximum separation distances, and stay there.

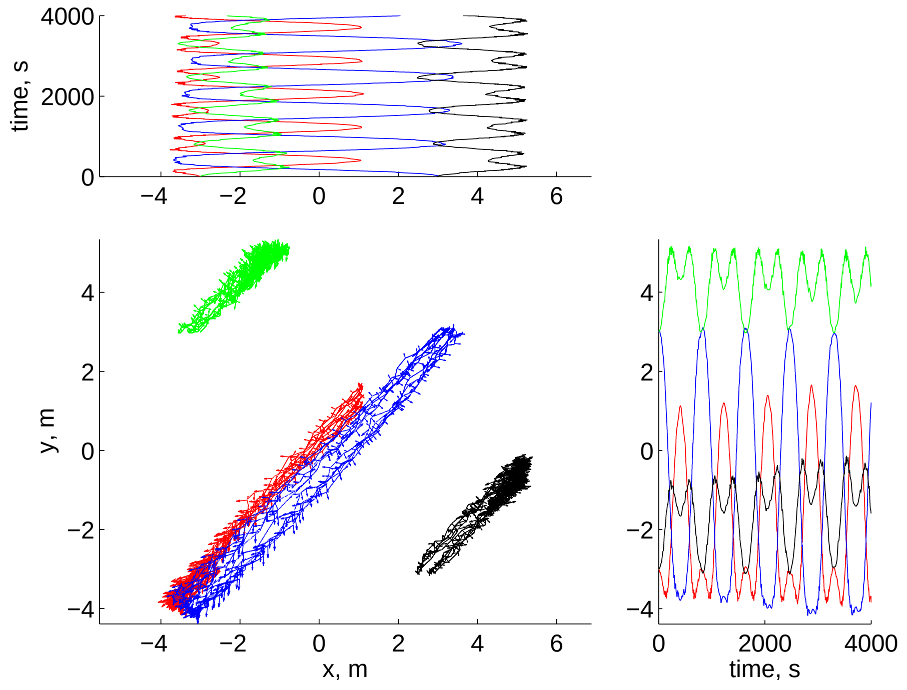


Figure 4.29: X-Y plane, tetrahedral formation, thrust out to the side. Satellites travel in C-shapes. The small retrograde motion visible in the time graphs corresponds to the satellites curving back to the end of the arc before turning around.

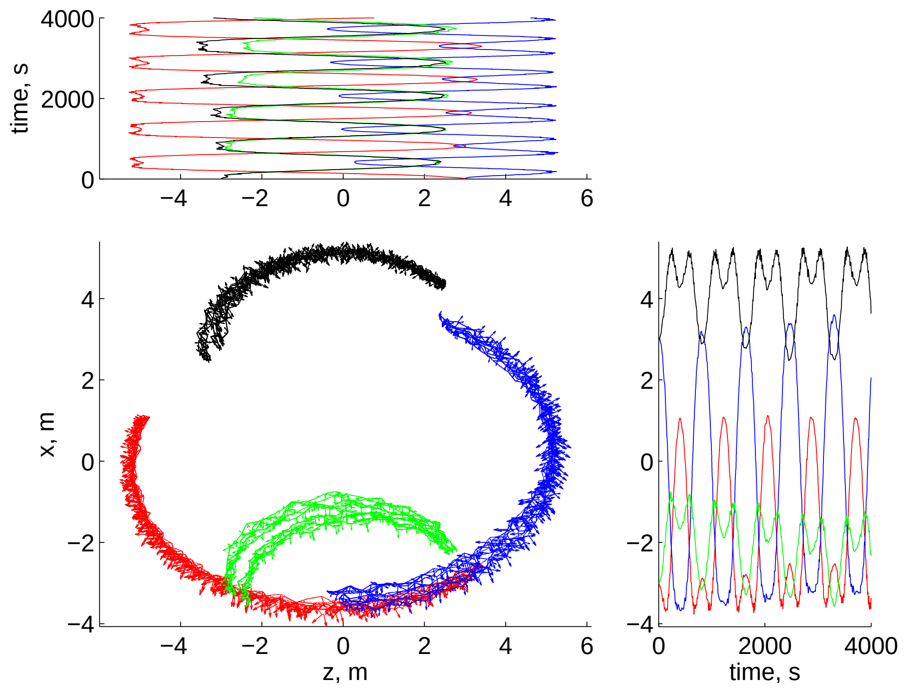


Figure 4.30: X-Z plane, tetrahedral formation, thrust out to the side

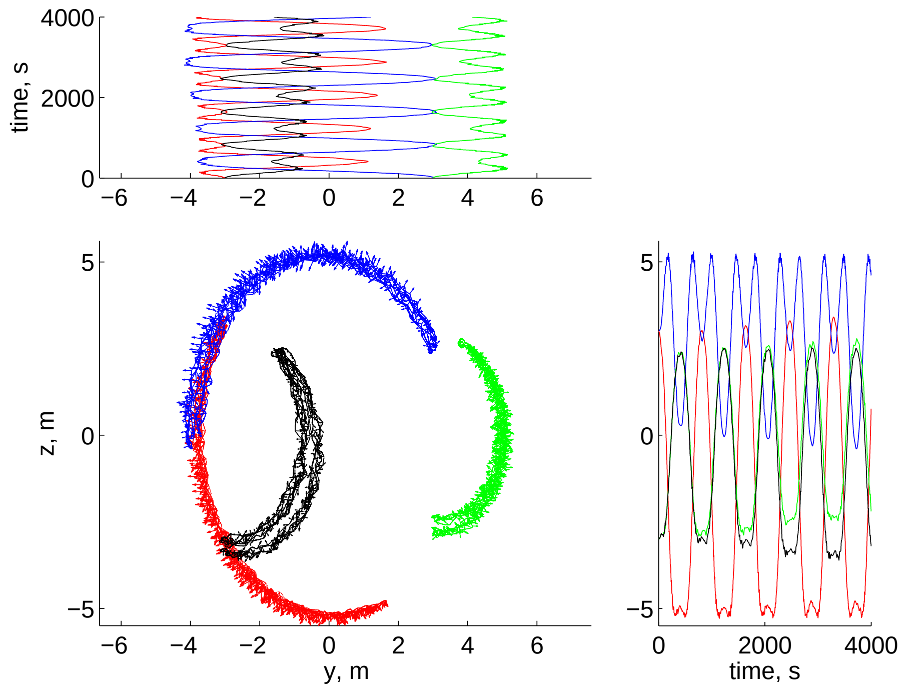


Figure 4.31: Y-Z plane, tetrahedral formation, thrust out to the side

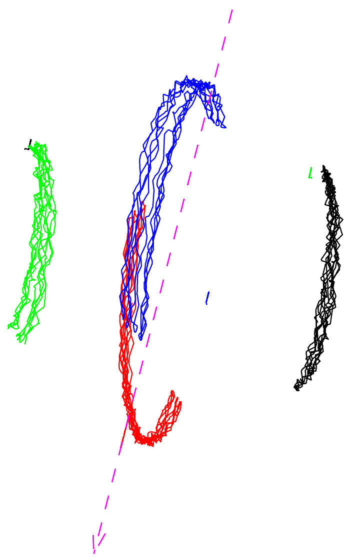


Figure 4.32: Rotation of tetrahedron over time, thrust out to the side. The midpoint of the red arc points in the direction of thrust, (-1,-1,-1).

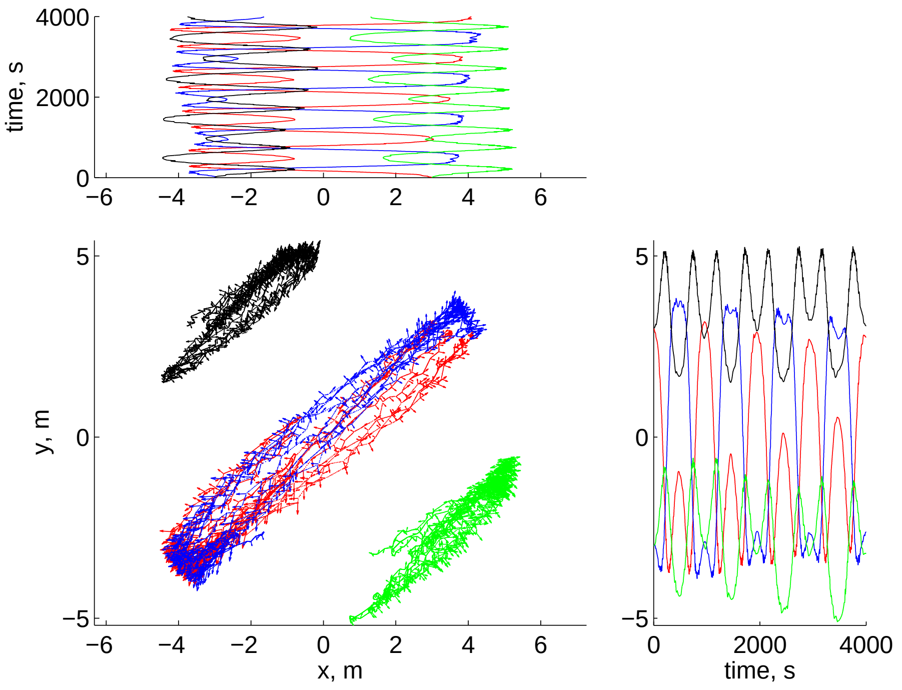


Figure 4.33: X-Y plane, tetrahedral formation, thrust in from the side. As before, but with larger arcs.

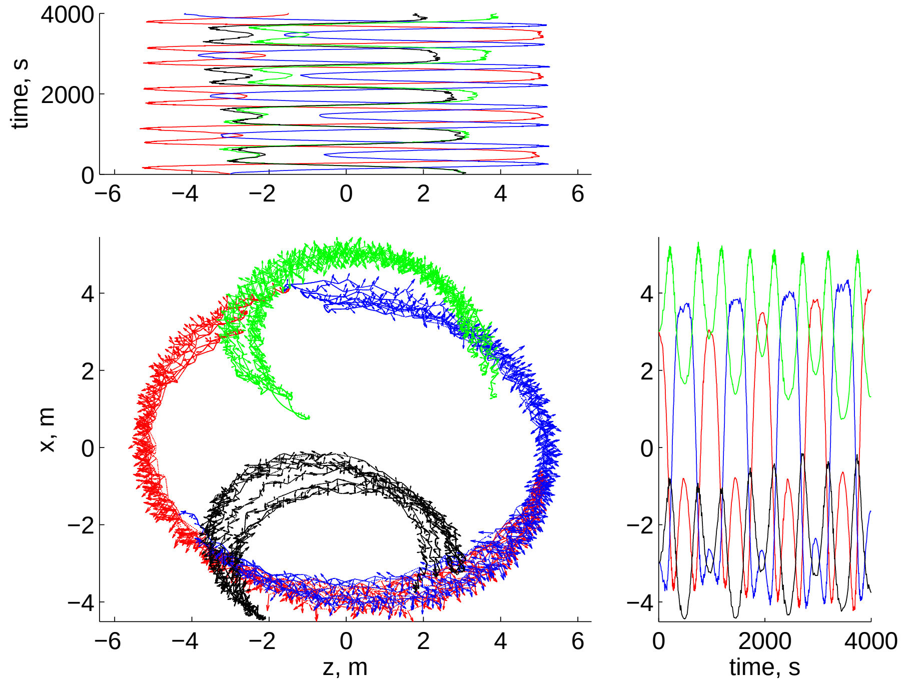


Figure 4.34: X-Z plane, tetrahedral formation, thrust in from the side

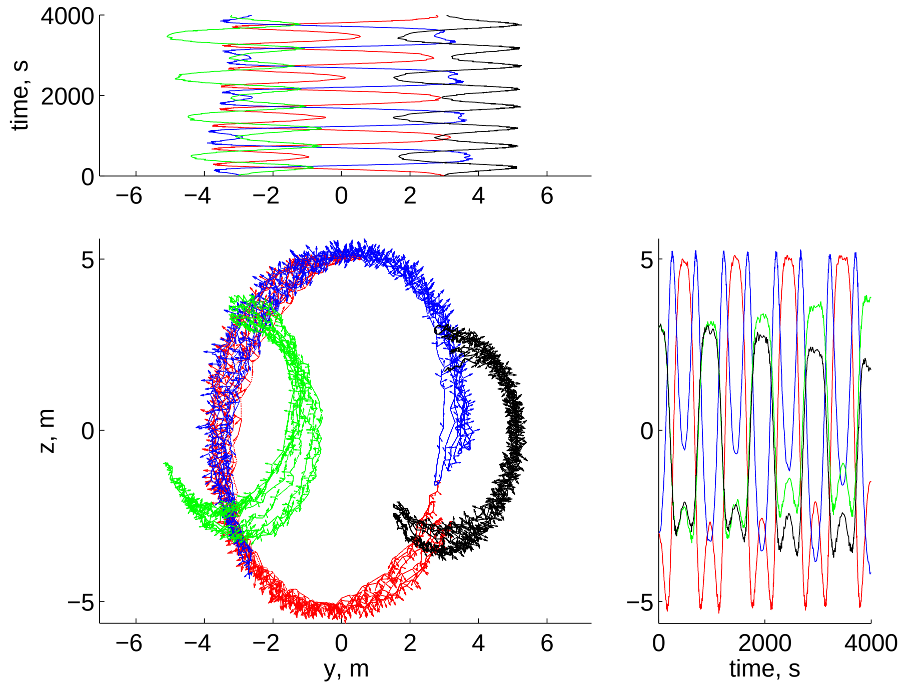


Figure 4.35: Y-Z plane, tetrahedral formation, thrust in from the side

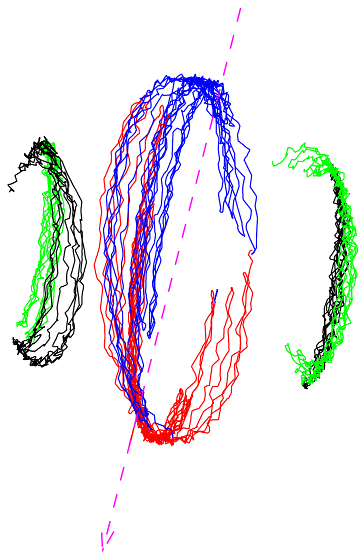


Figure 4.36: Rotation of tetrahedron over time, thrust in from the side. Again, the red satellite's arc is centered on the direction of thrust.

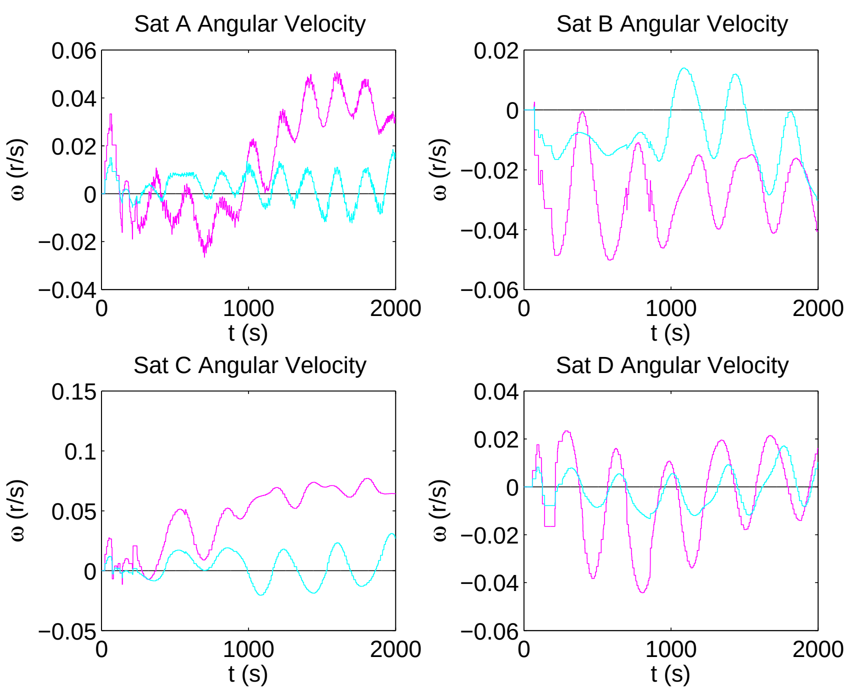


Figure 4.37: Angular velocity, general case. Cyan, pink, and black correspond to ω_x, ω_y, and ω_z, respectively.

4.4 Interaction with the Earth's Dipole

Because, in the paired satellite technique, all interactions are on-axis, there is no torque generated between satellites in normal operation, except for tiny amounts due to roundoff errors. However, if the satellites are orbiting the Earth, the terrestrial magnetic field will interact in some way with the satellite's dipole, most likely off-axis. Therefore, the Earth's own dipole will impart some amount of torque to the satellites.

When the Earth's dipole is integrated into `tstepSat.m`, the angular momentum buildup can be calculated. When `pairedsats.m` and `tetrasats.m` are run, their angular velocities generally stay within 0.1 $\frac{rad}{s}$, as can be seen in figure 4.37, which is well within the capacities of reaction wheels to take care of. For these purposes, the moments of inertia used were 100 kg·m^2, and the masses, 50 kg, selected as round numbers, on the order of magnitude of a micro-EMFF system.

4.5 Conclusion

It has been now demonstrated that the paired satellites technique can successfully keep four satellites continuously within specified boundaries. It can also maintain a tetrahedral formation.

In this research, any capacity of the paired satellite technique to maintain an orientation of the formation was not demonstrated, nor was any ability to perform complex maneuvers.

There may be some significant impracticalities in some of these scenarios simulated in this section. Notably, the dipoles, in some cases, are on nearly continuously. In a real system, this would tend to overheat the electrical system. Also, in some cases, the dipole moment strengths may be higher than practical. However, the scale of the simulation was arbitrarily chosen, and is not intended to be exclusively representative of operational conditions.

Chapter 5

Conclusion

5.1 Summary

Optimization

The first problem addressed was finding the lowest-mass design of a small-scale (on a satellite massing less than around 100 kg) Electromagnetic Formation Flight system. For micro-EMFF systems of various spacecraft bus masses and separation distances, optimal designs were found. Wire radius, number of turns, and current in the wire were varied, which in turn affected battery size, solar panel size, and wire insulation. The key variable was found to be wire radius. When the mass or separation distances changed, it tended to be the wire radius that changed most to accommodate the acceleration requirements for the new conditions.

Power Transfer

In the next chapter, an attempt was made to use the coils intended for EMFF to transmit and receive power via strongly coupled magnetic resonances. The coils were analyzed, and inductances and capacitances were found; these, in turn, were used to calculate the power transfer efficiencies. These efficiencies increased with larger numbers of turns, and fell off with increasing distances, but within a few meters of each other, the coils are quite efficient for power transfer.

For the mass-optimal designs selected in the optimization chapter, all those which were designed to operate within five meters were found to be highly efficient at power transmission.

Paired Satellites

The final section sought to identify whether it is feasible to operate an EMFF system by only activating two satellite dipoles at a time, maneuvering just one pair of satellites.

The ability of the paired satellites technique to keep four satellites together, and even to hold a simple formation, was demonstrated. This bodes well for the possibility of using only two dipoles at a time, thus simplifying the control systems for the satellite cluster.

Four scenarios were studied: one with four satellites in a line, one with four satellites starting in a square, one with four satellites in a random configuration, and one with four satellites in a tetrahedron, each with one satellite of the four being accelerated.

In the linear case, when the driven satellite was in the middle position, the cluster did not stay together, since the driven satellite, and those in front, would tend to get stuck in collision-avoidance mode. However, when there was enough multi-satellite interaction, each satellite had a chance to maneuver.

In the square case, as soon as symmetry was broken, the four satellites would begin to oscillate within the appointed range of each other.

The general case, with random initial locations, would tend to stabilize to an uneven tetrahedron, rotating about the direction of thrust, with the driven satellite out front.

The tetrahedral case, much like the general case, would remain tetrahedral; its behavior depended on the initial position of the driven satellite, but it would stay in formation, oscillating in a controlled fashion.

5.2 Future Work

One of the results from the optimization was that the cooling time on the wires needs to be several times longer than the time the dipoles are active. This information has not yet been included in EMFF dynamics and control simulations.

While the power transfer efficiencies have been found for the mass-optimized configurations, the power transfer system has not yet been integrated into the optimization. Wireless power transfer could replace the solar panels, and could easily increase the optimal number of turns of wire; these effects remain to be studied.

The power transfer code itself has been partially validated with the source paper (Kurs *et al.*, 2007), but not completely. The remaining differences need to be resolved.

Any capacity of the paired satellite technique to maintain an orientation of the formation was not demonstrated, nor was any ability to perform complex maneuvers. Also, this technique has not yet been applied to clusters of any number other than four satellites. This should be demonstrated in order to further prove the effectiveness of the paired satellite technique.

5.3 Contributions

A mass optimization was found for a micro-EMFF spacecraft, taking into account the maximum time the coils can be active without overheating, and the necessary cooling time afterwards. A thermal analysis was made on the electromagnetic coils, and the required cooling times were identified.

The coils were further examined to determine if they would be useful for coupled magnetic resonance power transfer. It was found that power transfer would be quite efficient at close range (within a few meters), particularly when there are many turns of wire in the coils.

A scheme for controlling satellites through electromagnetic dipoles was developed and tested, in which two satellites at a time were active, with the dipoles aligned with each other on-axis. This system was shown to keep clusters of four satellites within given boundaries, and to maintain a tetrahedral formation.

Appendix A

MATLAB Code — Optimization

A.1 timetoheat.m

Calculates the time a system can be on before the insulation begins to melt.

```
function answer = timetoheat(N, rw, th, I)
%answer = timetoheat(N,rw,th,I)
%
%   N  -- Number of turns of wire in loop
%   rw -- Wire radius in meters (a rather small number)
%   th -- Insulation thickness in meters (a smaller number yet)
%   I  -- Current through each loop of wire in amps
%
%   answer -- time, in seconds, for a coil of given parameters to reach a
%             temperature of 360K
%
%   Global variable T contains the final temperature of the wire across its
%cross-section.

global T gPaperMode
rl = 1;
t_on = 5;

layers = round(.5+sqrt(12*N-3)/6)-1;%Number of layers, not including the
                                    %innermost, in the nearest centered
                                    %hexagonal number to N.
era = 2.82e-8;  %Electrical resistivity of aluminum
err = 0;%1e13;  %Electrical resistivity of rubber -- set to zero as an
roa = 2700;     %Density of aluminum                "ignore" signal
ror = 1500;     %Density of rubber
cpa = 897;      %Heat capacity of aluminum
cpr = 2000;     %Heat capacity of rubber
tra = 1/220;    %Thermal resistivity of aluminum
trr = 1/0.16;   %Thermal resistivity of rubber

%arrays of the various properties through the cross-section of the bundle
thik = [rw/2 rw/2 th repmat([th rw/2 rw/2 rw/2 rw/2 th],[1 layers])];
eres = [era era err repmat([err era era era era err],[1 layers])];
```

```
rhos = [roa roa ror repmat([ror roa roa roa roa ror],[1 layers])];
cpes = [cpa cpa cpr repmat([cpr cpa cpa cpa cpa cpr],[1 layers])];
tres = [tra tra trr repmat([trr tra tra tra tra trr],[1 layers])];
rdus = cumsum(thik);    %distance from center of bundle to layer
area = 4*pi^2*rl*rdus;  %outside surface area per unit length of the layer

dstep = length(thik);
eps = .11;%?
sig=5.6704e-8;
alpha = 1/tra/roa/cpa;
alphr = 1/trr/ror/cpr;
Tc = 250;
T = 295*ones(1,dstep);

dt = min([t_on/2500 min([th^2/alphr (rw/2)^2/alpha])]);
t=0;
while max(T)≤360
    t = t + 1;          %heat transfer:
    qrad = [zeros(1,dstep-1) eps*sig*area(end)*(Tc^4-T(end)^4)];%radiative
    qcnd = (T([1 1:(dstep-1)]) - T)./tres([1 1:(dstep-1)]) + ...%conductive
        (T([2:end dstep])-T)./tres;
    qres = I^2*rl/rw./thik.*eres;                        %resistive heating
    qdot = qrad + qcnd + qres;                           %total heat transfer
    DT = dt*qdot./rhos./cpes./area./thik;                %change in temperature
    T = T + DT;                                          %new temperature\
    if t*dt ≥ 50000 %if it takes more than about fifteen hours to heat up
        break;        %beyond the maximum temperature, cycle time is going
    end               %to be no problem; cut off the loop.
end

answer = (t)*dt;

if gPaperMode
    figure(1)
    plot(rdus,T)
    xlabel('Distance from center of coil (m)','FontSize',14)
    ylabel('T (K)','FontSize',14)
    set(gca,'FontSize',14)
end
end
```

A.2 timetocool.m

Calculates the time a system must be off after reaching the point in appendix A.1 in order to cool down.

```
function answer = timetocool(N, rw, th, I)
%answer = timetocool(N,rw,th,I)
%
%See timetoheat.m for more information
%
```

```
%Can use global variable T, from timetoheat.m, which contains the final
%temperature of the wire across its cross-section after heating.

global T gPaperMode
rl = 1;
t_on = 5;

layers = round(.5+sqrt(12*N-3)/6)-1;%Number of layers, not including the
                                    %innermost, in the nearest centered
                                    %hexagonal number to N.
era = 2.82e-8;  %Electrical resistivity of aluminum
err = 0;%1e13;  %Electrical resistivity of rubber -- set to zero as an
roa = 2700;     %Density of aluminum                 "ignore" signal
ror = 1500;     %Density of rubber
cpa = 897;      %Heat capacity of aluminum
cpr = 2000;     %Heat capacity of rubber
tra = 1/220;    %Thermal resistivity of aluminum
trr = 1/0.16;   %Thermal resistivity of rubber

%arrays of the various properties through the cross-section of the bundle
thik = [rw/2 rw/2 th repmat([th rw/2 rw/2 rw/2 rw/2 th],[1 layers])];
eres = [era era err repmat([err era era era era err],[1 layers])];
rhos = [roa roa ror repmat([ror roa roa roa roa ror],[1 layers])];
cpes = [cpa cpa cpr repmat([cpr cpa cpa cpa cpa cpr],[1 layers])];
tres = [tra tra trr repmat([trr tra tra tra tra trr],[1 layers])];
rdus = cumsum(thik);    %distance from center of bundle to layer
area = 4*pi^2*rl*rdus;  %outside surface area per unit length of the layer

dstep = length(thik);
eps = .11;%?
sig=5.6704e-8;
alpha = 1/tra/roa/cpa;
alphr = 1/trr/ror/cpr;
Tc = 250;
dt = min([t_on/2500 min([th^2/alphr (rw/2)^2/alpha])]);
if isempty(T)%heating for five seconds, if there is no initial state given
    T = 295*ones(1,dstep);
    steps = min([round(t_on/dt) 50000]);
    for t=1:(steps-1)                         %radiative cooling
        qrad = [zeros(1,dstep-1) eps*sig*area(end)*(Tc^4-T(end)^4)];
        qcnd = (T([1 1:(dstep-1)]) - T)./tres([1 1:(dstep-1)]) + ...
            (T([2:end dstep])-T)./tres;       %conductive cooling
        qres = I^2*rl/rw./thik.*eres;         %resistive heating
        qdot = qrad + qcnd + qres;            %total heat transfer
        DT = dt*qdot./rhos./cpes./area./thik; %change in temperature
        T = T + DT;                           %new temperature
    end
else
    steps = 1;
    t = 0;
end
while max(T)>295
    t = t + 1;
    qrad = [zeros(1,dstep-1) eps*sig*area(end)*(Tc^4-T(end)^4)];%radiative
    qcnd = (T([1 1:(dstep−1)]) − T)./tres([1 1:(dstep−1)]) + ...%conductive
        (T([2:end dstep])−T)./tres;
    qdot = qrad + qcnd;                                 %total heat transfer
    DT = dt*qdot./rhos./cpes./area./thik;               %change in temperature
    T = T + DT;                                         %new temperature
end

answer = (t+1−steps)*dt;

if gPaperMode
    figure(2)
    plot(rdus,T)
    xlabel('Distance from center of coil (m)','FontSize',14)
    ylabel('T (K)','FontSize',14)
    set(gca,'FontSize',14)
end
end
```

A.3 findbestconfig.m

Script for creating array of thermal characteristics, and identifying the optimal design point. Also includes parameter plots.

```
global gPaperMode
if isempty(gPaperMode)
    gPaperMode = 0;
end
if gPaperMode
    set(0,'defaultTextFontSize',14)
    sizet = 14;
else
    clear all
    gPaperMode = 0;
    sizet = 10;
end

if 0
    if 0    %Warning: calculating bigthermalmatrix.mat takes several days
        hmax = 6;   %when complete, change "if 1" to "if 0"
        imax = 8;
        jmax = 8;

        iterations = hmax*imax*jmax;
        thisone = 0;
        sofar = 0;
        A = zeros(imax,jmax,hmax);
        C = zeros(imax,jmax,hmax);
        for h=1:6
            I = 5*h+20;
            for i=1:imax
                n = 20*i;
                for j = 1:jmax
                    r = .0005*j;
                    inches = r*2/.0254;                        %\ Calculation of
                    awg = 36-39*log(inches/.005)/log(92);      %| insulation
                    mils = 10^(0.518-awg/44.8);                %| thickness from
                    th = max([mils*.0254/1000 1e-6]);          %/ wire gauge
                    thisone = thisone + 1;
                    tic
                    fprintf([num2str(thisone) '/' num2str(iterations)...
                        ': heating... '])
                    A(i,j,h) = timetoheat(n,r,th,I);
                    fprintf(' cooling...  ')
                    C(i,j,h) = timetocool(n,r,th,I);
                    thistime = toc; sofar = sofar + thistime;
                    fprintf([colontimefromsec(thistime) ' sec; '...
                        colontimefromsec(sofar) ' elapsed, about '...
                        colontimefromsec(sofar*((iterations-thisone)...
                        /thisone)) ' to go.\n'])%See page 99
                end
            end
            save intermediatethermal.mat
        end

        harray = (1:hmax)*5+20;%amps
        iarray = (1:imax)*20;%turns
        jarray = (1:jmax)*.0005;%m radius

        save bigthermalmatrix.mat
    else
        load bigthermalmatrix.mat
    end
else
    if 1   %when complete, change "if 1" to "if 0"; took only an hour
        hmax = 6;
        imax = 8;
        jmax = 8;

        iterations = hmax*jmax;
        thisone = 0;
        sofar = 0;
        As = zeros(1,jmax,hmax);
        Cs = zeros(1,jmax,hmax);
        n = 1;
        for h=1:6
            I = 5*h+20;
            for j = 1:jmax
                r = .0005*j;
                inches = r*2/.0254;                        %\ Calculation of
                awg = 36-39*log(inches/.005)/log(92);      %| insulation
                mils = 10^(0.518-awg/44.8);                %| thickness from
                th = max([mils*.0254/1000 1e-6]);          %/ wire gauge
                thisone = thisone + 1;
                tic
                fprintf([num2str(thisone) '/' num2str(iterations)...
                    ': heating... '])
                As(1,j,h) = timetoheat(n,r,th,I);
                fprintf(' cooling...  ')
                Cs(1,j,h) = timetocool(n,r,th,I);
                thistime = toc; sofar = sofar + thistime;
                fprintf([colontimefromsec(thistime) ' sec; '...
                    colontimefromsec(sofar) ' elapsed, about '...
                    colontimefromsec(sofar*((iterations-thisone)...
                    /thisone)) ' to go.\n'])%See page 99
            end
        end

        A = repmat(As,[imax,1,1]);
        C = repmat(Cs,[imax,1,1]);

        harray = (1:hmax)*5+20;%amps
        iarray = (1:imax)*20;%turns
        jarray = (1:jmax)*.0005;%m radius

        save littlethermalmatrix.mat
    else
        load littlethermalmatrix.mat
    end
end

%%%%%%%%%%%%%%%%
%Changable Part%
sat_mass = 50; %
s_apart  = 10; %
%%%%%%%%%%%%%%%%

rl = 1;
dvmin = 3.4213*10^-9*s_apart;%minimum allowable acceleration for J2
res= 2.82*10^-8;

tableofdoom = zeros(hmax*imax*jmax,16);
index = 0;

for h=1:hmax
    I = 5*h+20;
    for i=1:imax
        N = 20*i;
        for j = 1:jmax
            rw = .0005*j;
            index = index + 1;

            R = N*rl*res/(rw^2);
            V = I*R;
            t_on  = A(i,j,h);
            t_off = C(i,j,h);

            %insulation mass
            inches = rw*2/.0254;
            awg = 36-39*log(inches/.005)/log(92);
```

```
            mils = 10^(0.518-awg/44.8);
            th_ins = max([mils*.0254/1000 1e-6]);
            ins_den = 1500;
            m_ins = 4*pi^2*rl*((rw+th_ins)^2-rw^2)*ins_den;

            %wire mass
            den = 2750;
            m_wire = (den*rl*2*pi*N*pi*rw^2);

            %battery mass
            on_time_shadowed = t_on*ceil(2000/(t_on+t_off));
            bat_den = 2e5; %J/kg
            m_bat = V^2/R*on_time_shadowed/bat_den;

            %solar panel mass
            sol_den = 1/25;
            m_sol = (V^2/R*(t_on-on_time_shadowed)/(t_off+t_on)*sol_den);

            %mass of emff system
            mass = m_wire + m_bat + m_sol + m_ins; %wire mass + battery
                                 %mass + solar panel mass + insulation mass
            %system impulse
            accel = (t_on/((t_off+t_on)*(sat_mass+mass)))*pi^2*6e-7*...
                N^2*(V/R)^2*rl^4/s_apart^4; %acceleration, averaged

            %update table of doom
            tableofdoom(index,:) = [rl,N,rw,V,I,R,t_on,t_off,mass,...
                accel*1e6,accel/dvmin*100,accel*1e6/mass,m_wire,m_bat,...
                m_sol,m_ins];
        end
    end
end

if 0
disp(['     Loop    Number      Wire    Voltage   Current  Resistance '...
    '  Time      Time     System     Accel   Percent    Accel'])
disp(['    radius  of turns    radius                               '...
    '   on       off       mass              DeltaV    per kg'])
disp(['     (m)                 (m)       (V)       (A)      (ohms)   '...
    '  (s)       (s )      (kg)      (um/s^2)          (um/kgs^2)'])
%%%%%[%    1.0000   20.0000    0.0005  112.8000    50.0000    2.2560    '
   %' 0.0000    0.0000    0.3657    0.0001    0.1245    0.0000
sortedtable = sortrows(tableofdoom,9);
disp(sortedtable(:,1:12))
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Plots by current level
if 1 || gPaperMode
cur25 = tableofdoom(  1:64, :);
cur30 = tableofdoom( 65:128,:);
cur35 = tableofdoom(129:192,:);
cur40 = tableofdoom(193:256,:);
```

```
cur45 = tableofdoom(257:320,:);
cur50 = tableofdoom(321:384,:);

figure(1);clf
plot(cur25(:,9),cur25(:,10),'.',cur30(:,9),cur30(:,10),'.',cur35(:,9),...
    cur35(:,10),'.',cur40(:,9),cur40(:,10),'.',cur45(:,9),cur45(:,10),...
    '.',cur50(:,9),cur50(:,10),'.',[0 200],[dvmin dvmin]*1e6,':k')
xlabel('Mass (kg)','FontSize',sizet)
ylabel('Accel (\mu m/s^2)','FontSize',sizet)
legend('25 A','30 A','35 A','40 A','45 A','50 A','Min. Accel.',...
    'Location','SouthEast')
if gPaperMode
    set(gca,'FontSize',14)
end
x = 0:200;
y = dvmin*1e6./x;
figure(2);clf
plot(cur25(:,9),cur25(:,12),'.',cur30(:,9),cur30(:,12),'.',cur35(:,9),...
    cur35(:,12),'.',cur40(:,9),cur40(:,12),'.',cur45(:,9),cur45(:,12),...
    '.',cur50(:,9),cur50(:,12),'.',x,y,':k')
xlabel('Mass (kg)','FontSize',sizet)
ylabel('Accel per kilo (\mu m/kg s^2)','FontSize',sizet)
legend('25 A','30 A','35 A','40 A','45 A','50 A','Min. Accel.')
if gPaperMode
    set(gca,'FontSize',14)
end
%figure(3)
%plot(cur25(:,3),cur25(:,7)/60,'.',cur30(:,3),cur30(:,7)/60,'.',...
%    cur35(:,3),cur35(:,7)/60,'.',cur40(:,3),cur40(:,7)/60,'.',...
%    cur45(:,3),cur45(:,7)/60,'.',cur50(:,3),cur50(:,7)/60,'.')
%xlabel('Wire radius (m)','FontSize',sizet)
%ylabel('Max on time (min)','FontSize',sizet)
%legend('25 A','30 A','35 A','40 A','45 A','50 A')
%if gPaperMode
%    set(gca,'FontSize',14)
%end

figure(4);clf
plot(cur25(:,2).*cur25(:,3).^2,cur25(:,9),'.',cur30(:,2).*cur30(:,3).^2,...
    cur30(:,9),'.',cur35(:,2).*cur35(:,3).^2,cur35(:,9),'.',...
    cur40(:,2).*cur40(:,3).^2,cur40(:,9),'.',cur45(:,2).*cur45(:,3).^2,...
    cur45(:,9),'.',cur50(:,2).*cur50(:,3).^2,cur50(:,9),'.')
ylabel('Masses (kg)','FontSize',sizet)
xlabel('Total Wire Cross-section (m^2)','FontSize',sizet)
legend('25 A','30 A','35 A','40 A','45 A','50 A','Location','NorthWest')
if gPaperMode
    set(gca,'FontSize',14)
end
figure(5);clf
plot(cur25(:,2).*cur25(:,3).^2,cur25(:,10),'.',...
    cur30(:,2).*cur30(:,3).^2,cur30(:,10),'.',cur35(:,2).*cur35(:,3).^2,...
    cur35(:,10),'.',cur40(:,2).*cur40(:,3).^2,cur40(:,10),'.',...
    cur45(:,2).*cur45(:,3).^2,cur45(:,10),'.',cur50(:,2).*cur50(:,3).^2,...
    cur50(:,10),'.',[0 3e-3],[dvmin dvmin]*1e6,':k')
```

```
ylabel('Accel (m/s^2)','FontSize',sizet)
xlabel('Total Wire Cross-section (m^2)','FontSize',sizet)
legend('25 A','30 A','35 A','40 A','45 A','50 A','Min. Accel.',...
    'Location','SouthEast')
if gPaperMode
    set(gca,'FontSize',14)
end
%figure(6)
%plot(cur25(:,2).*cur25(:,3).^2,(cur25(:,7)+cur25(:,8))/60,'.',...
%    cur30(:,2).*cur30(:,3).^2,(cur30(:,7)+cur30(:,8))/60,'.',...
%    cur35(:,2).*cur35(:,3).^2,(cur35(:,7)+cur35(:,8))/60,'.',...
%    cur40(:,2).*cur40(:,3).^2,(cur40(:,7)+cur40(:,8))/60,'.',...
%    cur45(:,2).*cur45(:,3).^2,(cur45(:,7)+cur45(:,8))/60,'.',...
%    cur50(:,2).*cur50(:,3).^2,(cur50(:,7)+cur50(:,8))/60,'.')
%ylabel('Maximum time on (min)','FontSize',sizet)
%xlabel('Total Wire Cross-section (m^2)','FontSize',sizet)
%if gPaperMode
%    set(gca,'FontSize',14)
%end
end

%single-row/column/pillar plots
if 1 || gPaperMode
rad4mm = tableofdoom(8:8:384,:);
num160 = tableofdoom([57:64 121:128 185:192 249:256 313:320 377:384],:);
varcur = num160(8:8:48,:);%160 turns, 4-mm wire
varnum = rad4mm(41:48,:);%50 A, 4-mm wire
varrad = num160(41:48,:);%50 A, 160 turns

figure(10)
plot(varcur(:,5),varcur(:,7),'.-')
xlabel('Current (A)','FontSize',sizet)
ylabel('Time on (s)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
figure(11)
plot(varcur(:,5),varcur(:,8),'.-')
xlabel('Current (A)','FontSize',sizet)
ylabel('Time off (s)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
figure(12)
plot(varcur(:,5),varcur(:,9),'.-',varcur(:,5),varcur(:,13),'x-',...
    varcur(:,5),varcur(:,14),'o-',varcur(:,5),varcur(:,15),'+-',...
    varcur(:,5),varcur(:,16),'*-')
xlabel('Current (A)','FontSize',sizet)
ylabel('Mass (kg)','FontSize',sizet)
legend('Total mass','Wire mass','Battery mass','Solar panel mass',...
    'Insulation mass','Location','East')
if gPaperMode
    set(gca,'FontSize',14)
end
```

```
figure(13)
plot(varcur(:,5),varcur(:,10),'.-')
xlabel('Current (A)','FontSize',sizet)
ylabel('Accel (m/s^2)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
figure(14)
plot(varcur(:,5),varcur(:,7)./(varcur(:,7)+varcur(:,8)),'.-')
xlabel('Current (A)','FontSize',sizet)
ylabel('Fraction of cycle time on','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end

figure(20)
plot(varnum(:,2),varnum(:,7),'.-')
xlabel('Number of turns','FontSize',sizet)
ylabel('Time on (s)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
figure(21)
plot(varnum(:,2),varnum(:,8),'.-')
xlabel('Number of turns','FontSize',sizet)
ylabel('Time off (s)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
figure(22)
plot(varnum(:,2),varnum(:,9),'.-',varnum(:,2),varnum(:,13),'x-',...
    varnum(:,2),varnum(:,14),'o-',varnum(:,2),varnum(:,15),'+-',...
    varnum(:,2),varnum(:,16),'*-')
xlabel('Number of turns','FontSize',sizet)
ylabel('Mass (kg)','FontSize',sizet)
legend('Total mass','Wire mass','Battery mass','Solar panel mass',...
    'Insulation mass','Location','NW')
if gPaperMode
    set(gca,'FontSize',14)
end
figure(23)
plot(varnum(:,2),varnum(:,10),'.-')
xlabel('Number of turns','FontSize',sizet)
ylabel('Accel (m/s^2)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
figure(24)
plot(varnum(:,2),varnum(:,7)./(varnum(:,7)+varnum(:,8)),'.-')
xlabel('Number of turns','FontSize',sizet)
ylabel('Fraction of cycle time on','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
```

```
figure(30)
plot(varrad(:,3),varrad(:,7),'.−')
xlabel('Wire radius (m)','FontSize',sizet)
ylabel('Time on (s)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
figure(31)
plot(varrad(:,3),varrad(:,8),'.−')
xlabel('Wire radius (m)','FontSize',sizet)
ylabel('Time off (s)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
figure(32)
plot(varrad(:,3),varrad(:,9),'.−',varrad(:,3),varrad(:,13),'x−',...
    varrad(:,3),varrad(:,14),'o−',varrad(:,3),varrad(:,15),'+−',...
    varrad(:,3),varrad(:,16),'*−')
xlabel('Wire radius (m)','FontSize',sizet)
ylabel('Mass (kg)','FontSize',sizet)
legend('Total mass','Wire mass','Battery mass','Solar panel mass',...
    'Insulation mass','Location','NW')
if gPaperMode
    set(gca,'FontSize',14)
end
figure(33)
plot(varrad(:,3),varrad(:,10),'.−')
xlabel('Wire radius (m)','FontSize',sizet)
ylabel('Accel (m/sˆ2)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
figure(34)
plot(varrad(:,3),varrad(:,7)./(varrad(:,7)+varrad(:,8)),'.−')
xlabel('Wire radius (m)','FontSize',sizet)
ylabel('Fraction of cycle time on','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%3−d line plots
if 1 && ¬gPaperMode
figure(40)
plot3(tableofdoom(:,2),tableofdoom(:,3),tableofdoom(:,5))
zlabel('Current (A)')
xlabel('Number of Turns')
ylabel('Wire Radius (m)')
figure(41)
plot3(tableofdoom(:,2),tableofdoom(:,3),tableofdoom(:,9))
zlabel('Masses (kg)')
```

```
xlabel('Number of Turns')
ylabel('Wire Radius (m)')
figure(42)
plot3(tableofdoom(:,2),tableofdoom(:,3),tableofdoom(:,10))
zlabel('Accel (um/sˆ2)')
xlabel('Number of Turns')
ylabel('Wire Radius (m)')
figure(43)
plot3(tableofdoom(:,2),tableofdoom(:,3),tableofdoom(:,7))
zlabel('Time On (s)')
xlabel('Number of Turns')
ylabel('Wire Radius (m)')
figure(44)
plot3(tableofdoom(:,2),tableofdoom(:,3),tableofdoom(:,8))
zlabel('Time Off (s)')
xlabel('Number of Turns')
ylabel('Wire Radius (m)')

figure(45)
plot3(tableofdoom(:,5),tableofdoom(:,3),tableofdoom(:,7))
zlabel('Time On (s)')
xlabel('Current (A)')
ylabel('Wire Radius (m)')
figure(46)
plot3(tableofdoom(:,5),tableofdoom(:,3),tableofdoom(:,8))
zlabel('Time Off (s)')
xlabel('Current (A)')
ylabel('Wire Radius (m)')
end

%plots by number of turns
if 0 && ¬gPaperMode
num020series = [ 1:8   65:72  129:136 193:200 257:264 321:328];
num040series = [ 9:16  73:80  137:144 201:208 265:272 329:336];
num060series = [17:24  81:88  145:152 209:216 273:280 337:344];
num080series = [25:32  89:96  153:160 217:224 281:288 345:352];
num100series = [33:40  97:104 161:168 225:232 289:296 353:360];
num120series = [41:48 105:112 169:176 233:240 297:304 361:368];
num140series = [49:56 113:120 177:184 241:248 305:312 369:376];
num160series = [57:64 121:128 185:192 249:256 313:320 377:384];

num020 = tableofdoom(num020series,:);
num040 = tableofdoom(num040series,:);
num060 = tableofdoom(num060series,:);
num080 = tableofdoom(num080series,:);
num100 = tableofdoom(num100series,:);
num120 = tableofdoom(num120series,:);
num140 = tableofdoom(num140series,:);
num160 = tableofdoom(num160series,:);

figure(50)
plot(num020(:,3),num020(:,7),'+',num040(:,3),num040(:,7),'o',...
    num060(:,3),num060(:,7),'*',num080(:,3),num080(:,7),'.',...
    num100(:,3),num100(:,7),'x',num120(:,3),num120(:,7),'s',...
    num140(:,3),num140(:,7),'d',num160(:,3),num160(:,7),'^')
xlabel('Wire radius (m)')
ylabel('Time on (s)')
legend('20 turns','40 turns','60 turns','80 turns','100 turns',...
    '120 turns','140 turns','160 turns')
figure(51)
plot(num020(:,5),num020(:,7),'+',num040(:,5),num040(:,7),'o',...
    num060(:,5),num060(:,7),'*',num080(:,5),num080(:,7),'.',...
    num100(:,5),num100(:,7),'x',num120(:,5),num120(:,7),'s',...
    num140(:,5),num140(:,7),'d',num160(:,5),num160(:,7),'^')
xlabel('Current(A)')
ylabel('Time on (s)')
legend('20 turns','40 turns','60 turns','80 turns','100 turns',...
    '120 turns','140 turns','160 turns')

figure(52)
plot(num020(:,3),num020(:,8),'+',num040(:,3),num040(:,8),'o',...
    num060(:,3),num060(:,8),'*',num080(:,3),num080(:,8),'.',...
    num100(:,3),num100(:,8),'x',num120(:,3),num120(:,8),'s',...
    num140(:,3),num140(:,8),'d',num160(:,3),num160(:,8),'^')
xlabel('Wire radius (m)')
ylabel('Time on (s)')
legend('20 turns','40 turns','60 turns','80 turns','100 turns',...
    '120 turns','140 turns','160 turns')
figure(53)
plot(num020(:,5),num020(:,8),'+',num040(:,5),num040(:,8),'o',...
    num060(:,5),num060(:,8),'*',num080(:,5),num080(:,8),'.',...
    num100(:,5),num100(:,8),'x',num120(:,5),num120(:,8),'s',...
    num140(:,5),num140(:,8),'d',num160(:,5),num160(:,8),'^')
xlabel('Current(A)')
ylabel('Time on (s)')
legend('20 turns','40 turns','60 turns','80 turns','100 turns',...
    '120 turns','140 turns','160 turns')

figure(54)
plot(num020(:,3),num020(:,12),'+',num040(:,3),num040(:,10),'o',...
    num060(:,3),num060(:,12),'*',num080(:,3),num080(:,10),'.',...
    num100(:,3),num100(:,12),'x',num120(:,3),num120(:,10),'s',...
    num140(:,3),num140(:,12),'d',num160(:,3),num160(:,10),'^')
xlabel('Wire radius (m)')
ylabel('Accel (m/s^2)')
legend('20 turns','40 turns','60 turns','80 turns','100 turns',...
    '120 turns','140 turns','160 turns')
figure(55)
plot(num020(:,5),num020(:,12),'+',num040(:,5),num040(:,10),'o',...
    num060(:,5),num060(:,12),'*',num080(:,5),num080(:,10),'.',...
    num100(:,5),num100(:,12),'x',num120(:,5),num120(:,10),'s',...
    num140(:,5),num140(:,12),'d',num160(:,5),num160(:,10),'^')
xlabel('Current(A)')
ylabel('Accel (m/s^2)')
legend('20 turns','40 turns','60 turns','80 turns','100 turns',...
    '120 turns','140 turns','160 turns')
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%plots by radius
if 0 && ¬gPaperMode
rad05 = tableofdoom(1:8:377,:);
rad10 = tableofdoom(2:8:378,:);
rad15 = tableofdoom(3:8:379,:);
rad20 = tableofdoom(4:8:380,:);
rad25 = tableofdoom(5:8:381,:);
rad30 = tableofdoom(6:8:382,:);
rad35 = tableofdoom(7:8:383,:);
rad40 = tableofdoom(8:8:384,:);

figure(60)
plot(rad05(:,2),rad05(:,7),'+',rad10(:,2),rad10(:,7),'o',rad15(:,2),...
    rad15(:,7),'*',rad20(:,2),rad20(:,7),'.',rad25(:,2),rad25(:,7),...
    'x',rad30(:,2),rad30(:,7),'s',rad35(:,2),rad35(:,7),'d',...
    rad40(:,2),rad40(:,7),'ˆ')
xlabel('Number of turns')
ylabel('Time on (s)')
legend('.5 mm','1 mm','1.5 mm','2 mm','2.5 mm','3 mm','3.5 mm','4 mm')
figure(61)
plot(rad05(:,5),rad05(:,7),'+',rad10(:,5),rad10(:,7),'o',rad15(:,5),...
    rad15(:,7),'*',rad20(:,5),rad20(:,7),'.',rad25(:,5),rad25(:,7),...
    'x',rad30(:,5),rad30(:,7),'s',rad35(:,5),rad35(:,7),'d',...
    rad40(:,5),rad40(:,7),'ˆ')
xlabel('Current(A)')
ylabel('Time on (s)')
legend('.5 mm','1 mm','1.5 mm','2 mm','2.5 mm','3 mm','3.5 mm','4 mm')

figure(62)
plot(rad05(:,2),rad05(:,8),'+',rad10(:,2),rad10(:,8),'o',rad15(:,2),...
    rad15(:,8),'*',rad20(:,2),rad20(:,8),'.',rad25(:,2),rad25(:,8),...
    'x',rad30(:,2),rad30(:,8),'s',rad35(:,2),rad35(:,8),'d',...
    rad40(:,2),rad40(:,8),'ˆ')
xlabel('Number of turns')
ylabel('Time on (s)')
legend('.5 mm','1 mm','1.5 mm','2 mm','2.5 mm','3 mm','3.5 mm','4 mm')
figure(63)
plot(rad05(:,5),rad05(:,8),'+',rad10(:,5),rad10(:,8),'o',rad15(:,5),...
    rad15(:,8),'*',rad20(:,5),rad20(:,8),'.',rad25(:,5),rad25(:,8),...
    'x',rad30(:,5),rad30(:,8),'s',rad35(:,5),rad35(:,8),'d',...
    rad40(:,5),rad40(:,8),'ˆ')
xlabel('Current(A)')
ylabel('Time on (s)')
legend('.5 mm','1 mm','1.5 mm','2 mm','2.5 mm','3 mm','3.5 mm','4 mm')

figure(64)
plot(rad05(:,2),rad05(:,10),'+',rad10(:,2),rad10(:,10),'o',rad15(:,2),...
    rad15(:,10),'*',rad20(:,2),rad20(:,10),'.',rad25(:,2),rad25(:,10),...
    'x',rad30(:,2),rad30(:,10),'s',rad35(:,2),rad35(:,10),'d',...
    rad40(:,2),rad40(:,10),'ˆ')
xlabel('Number of turns')
ylabel('Accel (m/sˆ2)')
```

```
legend('.5 mm','1 mm','1.5 mm','2 mm','2.5 mm','3 mm','3.5 mm','4 mm')
figure(65)
plot(rad05(:,5),rad05(:,10),'+',rad10(:,5),rad10(:,10),'o',rad15(:,5),...
    rad15(:,10),'*',rad20(:,5),rad20(:,10),'.',rad25(:,5),rad25(:,10),...
    'x',rad30(:,5),rad30(:,10),'s',rad35(:,5),rad35(:,10),'d',...
    rad40(:,5),rad40(:,10),'^')
xlabel('Current(A)')
ylabel('Accel (m/s^2)')
legend('.5 mm','1 mm','1.5 mm','2 mm','2.5 mm','3 mm','3.5 mm','4 mm')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if 0 && ¬gPaperMode
figure(70)
plot(tableofdoom(:,9),tableofdoom(:,10),'.',[0 200],[dvmin dvmin]*1e6,'r')
xlabel('Mass (kg)','FontSize',sizet)
ylabel('Accel (um/s^2)','FontSize',sizet)
x = 0:200;
y = dvmin*x*1e6;
figure(71)
plot(tableofdoom(:,9),tableofdoom(:,12),'.',x,y)
xlabel('Mass (kg)','FontSize',sizet)
ylabel('Accel per kilo (um/kg s^2)','FontSize',sizet)
figure(72)
plot(tableofdoom(:,9),(tableofdoom(:,7)+tableofdoom(:,8))/60,'.')
xlabel('Mass (kg)','FontSize',sizet)
ylabel('Total Cycle Time (min)','FontSize',sizet)

figure(73)
plot(tableofdoom(:,2).*tableofdoom(:,3).^2,tableofdoom(:,9),'.')
ylabel('Masses (kg)','FontSize',sizet)
xlabel('Total Wire Cross-section (m^2)','FontSize',sizet)
figure(74)
plot(tableofdoom(:,2).*tableofdoom(:,3).^2,tableofdoom(:,10),'.',...
    [0 3e-3],[dvmin dvmin]*1e6,'r')
ylabel('Accel (m/s^2)','FontSize',sizet)
xlabel('Total Wire Cross-section (m^2)','FontSize',sizet)
figure(75)
plot(tableofdoom(:,2).*tableofdoom(:,3).^2,(tableofdoom(:,7)+...
    tableofdoom(:,8))/60,'.')
ylabel('Total Cycle Time (min)','FontSize',sizet)
xlabel('Total Wire Cross-section (m^2)','FontSize',sizet)
end
```

A.3.1 colontimefromsec.m

```
%This function is used only by findbestconfig.m.  It is used only in the
%initial creation of the thermal profile matrix, and used only as a means
%to keep track of the progress of the calculations, which can take several
%days to complete, depending on processor speed.
```

```
function result = colontimefromsec(seconds)
    minutes = floor(seconds/60);
    seconds = floor(mod(seconds,60)*100)/100;
    hours   = floor(minutes/60);
    minutes = mod(minutes,60);
    days    = floor(hours/24);
    hours   = mod(hours,24);
    result = '';
    if days>0
        result = [result num2str(days) ':'];
    end
    if hours>0
        if minutes ≥ 10
            result = [result num2str(hours) ':'];
        else
            result = [result num2str(hours) ':0'];
        end
    end
    if seconds ≥ 10
        result = [result num2str(minutes) ':' num2str(seconds)];
    else
        result = [result num2str(minutes) ':0' num2str(seconds)];
    end
end
```

A.4 findmultibest.m

Finds optimal design points for an array of spacecraft masses and average separation distances.

```
global gPaperMode
if isempty(gPaperMode)
    gPaperMode = 0;
end
if gPaperMode
    set(0,'defaultTextFontSize',14)
    sizet = 14;
else
    clear all
    gPaperMode = 0;
    sizet = 10;
end

%load bigthermalmatrix.mat
load littlethermalmatrix.mat

sepfac = 4; %number of points checked per meter
minima = zeros(10*sepfac−sepfac+1,10,3);

rl = 0.5; %%%%%loop radius is preset here
res= 2.82*10ˆ−8;
```

```
muo = 4e-7*pi;
light = 299792458;
sig = 37.8e6;
eps = 1/muo/light^2;

for sat_mass = [10 50 100];

    switch sat_mass
        case 10
            whichmass = 1;
        case 50
            whichmass = 2;
        otherwise
            whichmass = 3;
    end

    for sepdist = 1:(10*sepfac+1-sepfac)
        s_apart = sepdist/sepfac+(1-1/sepfac);
        Dvmin = 3.4213*10^-9*s_apart;

        tableofdoom = zeros(hmax*imax*jmax,10);
        index = 0;

        for h=1:hmax
            I = 5*h+20;
            for i=1:imax
                N = 20*i;
                for j = 1:jmax
                    rw = .0005*j;
                    index = index + 1;

                    R = N*rl*res/(rw^2);
                    V = I*R;
                    t_on  = A(i,j,h);
                    t_off = C(i,j,h);

                    %insulation mass
                    inches = rw*2/.0254;
                    awg = 36-39*log(inches/.005)/log(92);
                    mils = 10^(0.518-awg/44.8);
                    th_ins = max([mils*.0254/1000 1e-6]);
                    ins_den = 1500;
                    m_ins = 4*pi^2*rl*((rw+th_ins)^2-rw^2)*ins_den;

                    %wire mass
                    den = 2750;
                    m_wire = (den*rl*2*pi*N*pi*rw^2);

                    %battery mass
                    on_time_shadowed = t_on*ceil(2000/(t_on+t_off));
                    bat_den = 2e5; %J/kg
                    m_bat = V^2/R*on_time_shadowed/bat_den;
                    %solar panel mass
                    sol_den = 1/25;
                    m_sol = (V^2/R*t_on/(t_off+t_on)*sol_den);

                    %mass of emff system
                    mass = m_wire + m_bat + m_sol + m_ins;
                     %masses of wire + battery + solar panel + insulation

                    %system impulse
                    △v = (t_on/((t_off+t_on)*(sat_mass+mass)))*pi^2*...
                        6e−7*N^2*(V/R)^2*rl^4/s_apart^4; %△−v, averaged

                    %update table of doom
                    tableofdoom(index,:) = [rl,N,rw,V,I,R,mass,...
                        △v*1e6,△v/Dvmin*100,t_on+t_off];
                end
            end
        end

        tableofdoom = sortrows(tableofdoom,7);
        thebest = find(tableofdoom(:,9)>100);
        thebest = thebest(1,1);
        minima(sepdist,:,whichmass) = tableofdoom(thebest,:);

    end
    s_apart = 1:1/sepfac:10;
    if 1 && ¬gPaperMode
        disp(['System mass = ' num2str(sat_mass)])
        disp(['  Separation   Loop    Number      Wire    Voltage  '...
            ' Current  Resistance  System    DeltaV'])
        disp(['    distance   radius  of turns    radius            '...
            '                      mass'])
        disp(['      (m)      (m)                 (m)       (V)    '...
            '   (A)      (ohms)     (kg)     (um/s)'])
        %%%%%%%    1.0000    0.5000   20.0000    0.0005   56.4000  '...
        %   ' 50.0000    1.1280    0.2003    1.0607  710.4068    0.9277
        disp([s_apart' minima(:,1:8,whichmass)])
    end
end

save('multibest.mat','minima','s_apart')

figure(1)
plot(1:1/sepfac:10,minima(:,7,1),'x:',1:1/sepfac:10,minima(:,7,2),'o:',...
    1:1/sepfac:10,minima(:,7,3),'+:')
legend('bus mass = 10','bus mass = 50','bus mass = 100','Location','NW')
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('Minimum propulsion mass (kg)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
else
    title('Minimum Masses')
end
```

```
figure(2)
plot(1:1/sepfac:10,minima(:,7,1)./(minima(:,7,1)+10),'x:',1:1/sepfac:10,...
    minima(:,7,2)./(minima(:,7,2)+50),'o:',1:1/sepfac:10,minima(:,7,3)./...
    (minima(:,7,3)+100),'+:')
legend('bus mass = 10','bus mass = 50','bus mass = 100','Location','NW')
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('Propulsion mass fraction','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
else
    title('Minimum Masses')
end


figure(3)
loglog(1:1/sepfac:10,minima(:,7,1),'x:',1:1/sepfac:10,minima(:,7,2),...
    'o:',1:1/sepfac:10,minima(:,7,3),'+:')
legend('bus mass = 10','bus mass = 50','bus mass = 100','Location','NW')
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('Minimum propulsion mass (kg)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
else
    title('Minimum Masses')
end


figure(4)
subplot(2,2,1)
plot(1:1/sepfac:10,minima(:,3,1),'.-',1:1/sepfac:10,minima(:,3,2),'.-',...
    1:1/sepfac:10,minima(:,3,3),'.-')
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('Wire radius (m)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end

subplot(2,2,2)
plot(1:1/sepfac:10,minima(:,2,1),'.-',1:1/sepfac:10,minima(:,2,2),'.-',...
    1:1/sepfac:10,minima(:,2,3),'.-')
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('Number of turns','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end

subplot(2,2,3)
plot(1:1/sepfac:10,minima(:,5,1),'.-',1:1/sepfac:10,minima(:,5,2),'.-',...
    1:1/sepfac:10,minima(:,5,3),'.-')
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('Current (A)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
```

```
end

subplot(2,2,4)
plot(1:1/sepfac:10,minima(:,4,1),'.-',1:1/sepfac:10,minima(:,4,2),'.-',...
    1:1/sepfac:10,minima(:,4,3),'.-')
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('Voltage (V)','FontSize',sizet)
if gPaperMode
    set(gca,'FontSize',14)
end


figure(5)
plot(1:1/sepfac:10,minima(:,10,1)/60,'x:',1:1/sepfac:10,minima(:,10,2)/...
    60,'o:',1:1/sepfac:10,minima(:,10,3)/60,'+:')
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('Cycle time (min)','FontSize',sizet)
legend('bus mass = 10','bus mass = 50','bus mass = 100','Location','NW')
if gPaperMode
    set(gca,'FontSize',14)
end
```

Appendix B

MATLAB Code — Power Transfer

B.1 inductionarray.m

Calculates coupling coefficient κ and efficiency η_{max} for various separation distances and numbers of turns in the coil.

```
function bigarray = inductionarray()
global gPaperMode
if isempty(gPaperMode)
    gPaperMode = 0;
end
if gPaperMode
    set(0,'defaultTextFontSize',14)
    sizet = 14;
else
    clear all
    gPaperMode = 0;
    sizet = 10;
end

if 0 %warning: takes several days to run
    kappa  = zeros(8,10);
    etaMax = zeros(8,10);
    rl = 0.5;
    rw = .003;
    for loopnum=1:8
        N = 20*loopnum;
        height = sqrt(N)*rw;
        sections = ceil(120*N);
        leng = N*2*pi*rl;
        %Compute Inductance and Capacitance
        disp([num2str(loopnum) '/8: '])
        it = GenerateHelix(leng,N,rl,rw,height,sections);
        save([num2str(N) 'loops.mat'],'it')
        freq = 1/2/pi/sqrt(it.L*it.C);  %resonant frequency
        w = 2*pi*freq;                  %angular frequency
        for distnum=1:10
```

```
            dist = distnum;
            %Compute Mutual inductance
            fprintf([num2str(dist) '/10: '])
            M=ComputeM(it,it,dist,w); disp(['M = ' num2str(M)])
            kappa(loopnum,distnum)=w*abs(M)/2/sqrt(it.L^2);
        end
        %Compute eta
        coupling = kappa(loopnum,:).^2/it.Gamma^2;
        ratio = sqrt(it.Gamma^2+kappa(loopnum,:).^2)/it.Gamma;
        etaMax(loopnum,:) = ratio.*coupling./((1+ratio).*coupling+...
            (1+ratio).^2);
        save(['LCarray' num2str(loopnum) '.mat'],'kappa','etaMax')
    end
    bigarray = [kappa etaMax];
    save('LCarray.mat','kappa','etaMax')
else
    load LCarray.mat
end

if ¬gPaperMode
distances = repmat((1:10),      8, 1);
numbers   = repmat((20:20:160)',1,10);
figure(1);clf
mesh(distances,numbers,kappa)
xlabel('Distance (m)')
ylabel('Number of turns')
zlabel('\kappa')
figure(2);clf
mesh(distances,numbers,etaMax)
xlabel('Distance (m)')
ylabel('Number of turns')
zlabel('\eta_{max}')
end

loops = 20:20:160;
dists = 1:10;
figure(3);clf
plot(loops,kappa(:,1))
xlabel('Number of turns','FontSize',sizet)
ylabel('\kappa','FontSize',sizet)
legend('d_{sep} = 1 m','Location','SE')
if gPaperMode
    set(gca,'FontSize',14)
end
figure(4);clf
plot(loops,etaMax(:,1))
xlabel('Number of turns','FontSize',sizet)
ylabel('\eta_{max}','FontSize',sizet)
legend('d_{sep} = 1 m','Location','SE')
if gPaperMode
    set(gca,'FontSize',14)
end
figure(5);clf
plot(loops,kappa(:,10))
```

```
xlabel('Number of turns','FontSize',sizet)
ylabel('\kappa','FontSize',sizet)
legend('d_{sep} = 10 m','Location','SE')
if gPaperMode
    set(gca,'FontSize',14)
end
figure(6);clf
plot(loops,etaMax(:,10))
xlabel('Number of turns','FontSize',sizet)
ylabel('\eta_{max}','FontSize',sizet)
legend('d_{sep} = 10 m','location','SE')
if gPaperMode
    set(gca,'FontSize',14)
end

figure(7);clf
plot(dists,kappa(1,:))
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('\kappa','FontSize',sizet)
legend('N = 20')
if gPaperMode
    set(gca,'FontSize',14)
end
figure(8);clf
plot(dists,etaMax(1,:))
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('\eta_{max}','FontSize',sizet)
legend('N = 20')
if gPaperMode
    set(gca,'FontSize',14)
end
figure(9);clf
plot(dists,kappa(8,:))
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('\kappa','FontSize',sizet)
legend('N = 160')
if gPaperMode
    set(gca,'FontSize',14)
end
figure(10);clf
plot(dists,etaMax(8,:))
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('\eta_{max}','FontSize',sizet)
legend('N = 160','Location','SW')
if gPaperMode
    set(gca,'FontSize',14)
end

load multibest.mat
distpoints = zeros(1,10);
for i=1:10
    distpoints(i) = find(s_apart==i,1,'last');
end
turns = zeros(3,10);
```

```
turns(1,:) = minima(distpoints,2,1);
turns(2,:) = minima(distpoints,2,2);
turns(3,:) = minima(distpoints,2,3);
turns = turns/20;

etae = zeros(3,10);
for i=1:10
    etae(1,i) = etaMax(turns(1,i),i);
    etae(2,i) = etaMax(turns(2,i),i);
    etae(3,i) = etaMax(turns(3,i),i);
end

figure(11);clf
plot(1:10,etae(1,:),'x:',1:10,etae(2,:),'o:',1:10,etae(3,:),'+:')
xlabel('Separation distance (m)','FontSize',sizet)
ylabel('\eta_{max}','FontSize',sizet)
legend('bus mass = 10','bus mass = 50','bus mass = 100','Location','SW')
if gPaperMode
    set(gca,'FontSize',14)
end
end


function answer = ComputeM(Source,Device,dist,w)
    mu0=pi*4e-7;c=299792458;epsilon0=1/mu0/c^2;
    Device.Pos(:,3) = Device.Pos(:,3)+dist;
    m=0;
    for i=1:(Source.N)
        for j=1:(Device.N)
            dM=sin(pi*Source.CumLen(i)/Source.leng)*sin(pi*...
                Device.CumLen(j)/Device.leng)*Source.LenSeg(i)*...
                Device.LenSeg(j)*dot(Source.Seg(i,:),Device.Seg(j,:))...
                /norm(Source.Pos(i,:)-Device.Pos(j,:));
            dM2=(Source.leng/(pi*w^2))*sin(pi*Source.CumLen(i)/...
                Source.leng)*cos(pi*Device.CumLen(j)/Device.leng)*...
                Source.LenSeg(i)*Device.LenSeg(j)*dot(Source.Seg(i,:),...
                (Source.Pos(i,:)-Device.Pos(j,:)))/norm(Source.Pos(i,:)...
                -Device.Pos(j,:))^3;
            m = m + mu0*dM/(4*pi) + dM2/(4*pi*epsilon0);
        end
    end
    answer = m;
end

function it = GenerateHelix(leng,turns,radius,rw,h,N)
    dtheta=2*pi*turns/N;
    dz=h/N;
    theta    = (0:dtheta:dtheta*(N-1))';
    currentz = (0:dz:dz*(N-1))';

    Pos      = [radius*cos(theta) radius*sin(theta) currentz];
    Seg      = [-radius*cos(theta) radius*sin(theta) repmat(dz,N,1)]./...
        sqrt(radius^2+dz^2);
    LenSeg   = repmat(sqrt((dtheta*radius)^2+dz^2),N,1);
    CumLen   = cumsum(LenSeg);

    %Now make the current and charge distributions along the loop
    Curr  = Seg.*sin(pi.*[CumLen CumLen CumLen]./leng);
    qDens = (pi./leng).*cos(pi.*CumLen./leng);

    it = struct('Pos',Pos,'Seg',Seg,'LenSeg',LenSeg,'CumLen',CumLen,...
        'leng',leng,'Curr',Curr,'qDens',qDens,'rw',rw,'N',N,'L',0,'C',...
        0,'Gamma',0);

    it.L = ComputeL(it,0); disp(['L = ' num2str(it.L)])
    it.C = ComputeC(it,0); disp(['C = ' num2str(it.C)])

    muo = pi*4e-7;light=299792458;eps=1/muo/light^2;
    res = 1.72e-8;%copper resistivity%2.82e-8%aluminum resistivity%
    freq = 1/2/pi/sqrt(it.L*it.C);                     %resonant frequency
    w = 2*pi*freq;                                      %angular frequency
    Ro = sqrt(muo*w*res/2)*leng/4/pi/rw;                %ohmic resistance
    Rr = sqrt(muo/eps)*(pi/12*turns^2*(w*radius/light)^4+2/3/pi^3*...
        (w*h/light)^2);  %radiative resistance
    it.Gamma = (Ro+Rr)/2/it.L;
end

function L=ComputeL(it,IntTech)
    rw     = it.rw;
    N      = it.N;
    Curr   = it.Curr;
    Pos    = it.Pos;
    LenSeg = it.LenSeg;

    mu0 = pi*4e-7;

    l=0;
    for i=1:N
        for j=1:N
            if IntTech==1%SimpsonRuleIntegration())
                dL11 = Curr(i-1) * Curr(j-1) * LenSeg(i-1)* LenSeg(j-1) ...
                    /norm((Pos(i-1,:) + offset1) - (Pos(j-1,:) + offset2));
                dL12 = Curr(i-1) * Curr(j)   * LenSeg(i-1)* LenSeg(j)
...
                    /norm((Pos(i-1,:) + offset1) - (Pos(j,:)   + offset2));
                dL21 = Curr(i)   * Curr(j-1) * LenSeg(i)  * LenSeg(j-1) ...
                    /norm((Pos(i,:)   + offset1) - (Pos(j-1,:) + offset2));
                dL22 = Curr(i)   * Curr(j)   * LenSeg(i)  * LenSeg(j)
...
                    /norm((Pos(i,:)   + offset1) - (Pos(j,:)   + offset2));
                dL=(dL11+dL12+dL21+dL22)/4;
            else%(IntTech==SimpleIntegration())
                dL = dot(Curr(i,:),Curr(j,:)) * LenSeg(i)*LenSeg(j)/...
                    max([norm(Pos(i,:)-Pos(j,:)) rw/2]);
                if norm(Pos(i,:)-Pos(j,:))≤rw/2
                    dL = 0;
                end
            end
            l = l + mu0*dL/(4*pi);
        end
    end
    L = l;
end

function C=ComputeC(it,IntTech)
    leng   = it.leng;
    CumLen = it.CumLen;
    LenSeg = it.LenSeg;
    Pos    = it.Pos;
    rw     = it.rw;
    N      = it.N;

    epsilon0 = 1/(pi*4e-7)/299792458^2;

    OneOverC=0;
    for i=1:N
        for j=1:N
            if IntTech==1 %(SimpsonRuleIntegration)
                dC11 = (pi/leng)^2*cos(pi*CumLen(i-1)/leng)*cos(pi*...
                    CumLen(j-1)/leng)*LenSeg(i-1)*LenSeg(j-1)/norm((...
                    Pos(i-1,:)+offset1)-(Pos(j-1,:)+offset2));
                dC21 = (pi/leng)^2*cos(pi*CumLen(i)  /leng)*cos(pi*...
                    CumLen(j-1)/leng)*LenSeg(i)  *LenSeg(j-1)/norm((...
                    Pos(i,:)  +offset1)-(Pos(j-1,:)+offset2));
                dC12 = (pi/leng)^2*cos(pi*CumLen(i-1)/leng)*cos(pi*...
                    CumLen(j)  /leng)*LenSeg(i-1)*LenSeg(j)  /norm((...
                    Pos(i-1,:)+offset1)-(Pos(j,:)  +offset2));
                dC22 = (pi/leng)^2*cos(pi*CumLen(i)  /leng)*cos(pi*...
                    CumLen(j)  /leng)*LenSeg(i)  *LenSeg(j)  /norm((...
                    Pos(i,:)  +offset1)-(Pos(j,:)  +offset2));
                dC = (dC11+dC12+dC21+dC22)/4;
            else%(IntTech==SimpleIntegration)
                dC = (pi/leng)^2*cos(pi*CumLen(i)/leng)*cos(pi*CumLen(j)...
                    /leng)*LenSeg(i)*LenSeg(j)/max([norm(Pos(i,:)-...
                    Pos(j,:)) rw/2]);
                if norm(Pos(i,:)-Pos(j,:))≤rw/2
                    dC = 0;
                end
            end
            OneOverC = OneOverC + dC/(4*pi*epsilon0);
        end
    end
    C = 1/OneOverC;
end
```

Appendix C

MATLAB Code — Paired Satellites

C.1 tstepSat.m

Kinematics file for paired satellite simulation.

```
function newplacenewspeed = tstepSat(oldplaceoldspeed,masses,...
    moments,dipoles,dt)
%newplacenewspeed = tstepSat(oldplaceoldspeed,masses,moments,dipoles,dt)
%
%  Inputs are of the form [A B C D] with each satellite's data in a column.
%                         [A B C D]
%                         [A B C D]
%                         [A B C D]
%                         [  &c.  ]
%
%  oldplaceoldspeed:
%      rows 1:3 -- position (x;y;z) [m]
%      rows 4:6 -- velocity (vx;vy;vz) [m/s]
%      rows 7:9 -- angular velocity (wx;wy;wz) [rad/s]
%
%  masses  -- array of satellite masses [mA mB mC mD] in kilograms
%
%  moments -- array of satellite moments of inertia [kg m^2], in the form
%             [Ixx;Iyy;Izz;Iyz;Ixz;Ixy], with each column corresponding to
%             a satellite, as before.
%
%  dipoles -- magnetic dipoles (ux;uy;uz) in ampere-meters-squared.
%
%  dt      -- time step in seconds
%
%The output, newplacenewspeed, is in the same format as oldplaceoldspeed.
%
%Subfunctions:
%
%   Bforce(m1,m2,r1,r2)
%       Using dipole moments m1 and m2, with positions r1 and r2, the force
%       between them is found using the dipole force equation.
%
%   Btorque(m1,m2,r1,r2)
%       Using dipole moments m1 and m2, with positions r1 and r2, the
%       torque between them is found using the dipole torque equation.
%
%   tensorify(list)
%       Takes the list of moments passed as a parameter to tstepSat() and
%       turns them into the moment of inertia tensor.
%
%   newposition(p,v,f,dt,m)
%       Uses basic kinematic equations to find the new position and
%       velocity given the old position and velocity (p and v), the force
%       (f), the time interval (dt), and the mass (m).
%
%   newattitude(w,t,dt,I)
%       Uses basic rotational kinematics to find the new angular velocity
%       given the old velocity (w), torque (t), time (dt), and the moment
%       of inertia tensor (I).

global gEarthDipoleOn
if isempty(gEarthDipoleOn)
    gEarthDipoleOn = 1;
end

position = oldplaceoldspeed(1:3,:);
velocity = oldplaceoldspeed(4:6,:);
angvel   = oldplaceoldspeed(7:9,:);

uA = dipoles(:,1);%magnetic dipole moments
uB = dipoles(:,2);
uC = dipoles(:,3);
uD = dipoles(:,4);
mA = masses(1);
mB = masses(2);
mC = masses(3);
mD = masses(4);
iA = tensorify(moments(:,1));%moments of inertia
iB = tensorify(moments(:,2));
iC = tensorify(moments(:,3));
iD = tensorify(moments(:,4));
pA = position(:,1);%positions
pB = position(:,2);
pC = position(:,3);
pD = position(:,4);
vA = velocity(:,1);%velocities
vB = velocity(:,2);
vC = velocity(:,3);
vD = velocity(:,4);
wA = angvel(:,1);%angular velocities
wB = angvel(:,2);
wC = angvel(:,3);
wD = angvel(:,4);
```

```
uE = gEarthDipoleOn*[0;0;8e22];%dipole of the Earth's magnetic field
height = 500;%orbit height (km)    if gEarthDipoleOn = 0, dipole is off.
pE = [-1000*(6378+height);0;0];%location of the Earth's center.

%Forces from Magnets
fMagA = -Bforce(uA,uB,pA,pB) + -Bforce(uA,uC,pA,pC) + ...
    -Bforce(uA,uD,pA,pD) + -Bforce(uA,uE,pA,pE);
fMagB = -Bforce(uB,uA,pB,pA) + -Bforce(uB,uC,pB,pC) + ...
    -Bforce(uB,uD,pB,pD) + -Bforce(uB,uE,pB,pE);
fMagC = -Bforce(uC,uA,pC,pA) + -Bforce(uC,uB,pC,pB) + ...
    -Bforce(uC,uD,pC,pD) + -Bforce(uC,uE,pC,pE);
fMagD = -Bforce(uD,uA,pD,pA) + -Bforce(uD,uB,pD,pB) + ...
    -Bforce(uD,uC,pD,pC) + -Bforce(uD,uE,pD,pE);

%Torques from Magnets
tMagA = Btorque(uA,uB,pA,pB) + Btorque(uA,uC,pA,pC) + ...
    Btorque(uA,uD,pA,pD) + Btorque(uA,uE,pA,pE);
tMagB = Btorque(uB,uA,pB,pA) + Btorque(uB,uC,pB,pC) + ...
    Btorque(uB,uD,pB,pD) + Btorque(uB,uE,pB,pE);
tMagC = Btorque(uC,uA,pC,pA) + Btorque(uC,uB,pC,pB) + ...
    Btorque(uC,uD,pC,pD) + Btorque(uC,uE,pC,pE);
tMagD = Btorque(uD,uA,pD,pA) + Btorque(uD,uB,pD,pB) + ...
    Btorque(uD,uC,pD,pC) + Btorque(uD,uE,pD,pE);

%actual incrementation of state

pvA = newposition(pA,vA,fMagA,dt,mA);
pvB = newposition(pB,vB,fMagB,dt,mB);
pvC = newposition(pC,vC,fMagC,dt,mC);
pvD = newposition(pD,vD,fMagD,dt,mD);
qwA = newattitude(wA,tMagA,dt,iA);
qwB = newattitude(wB,tMagB,dt,iB);
qwC = newattitude(wC,tMagC,dt,iC);
qwD = newattitude(wD,tMagD,dt,iD);

newplace = [pvA pvB pvC pvD];
newpoint = [qwA qwB qwC qwD];
newplacenewspeed = [newplace;newpoint];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function B = Btorque(u1,u2,r1,r2)
    R = r2-r1;
    r = norm(R);
    muo = pi*4e-7;
    if r==0
        B = cross(u1,2/3*muo*u2);
    else
        B = cross(u1,muo/(32*pi*r^3)*(3*dot(u2,R/r)*R/r - u2));
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function F = Bforce(u1,u2,r1,r2)
    R = r2-r1;
    r = norm(R);
    muo = pi*4e-7;
    if r == 0
        F = 0;
    else
        F = 3*muo/64/pi/r^4*(dot(u1,u2)*R/r+dot(u1,R/r)*u2+...
            dot(u2,R/r)*u1-5*dot(u1,R/r)*dot(u2,R/r)*R/r);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function moment = tensorify(list)
    moment = [list(1) list(6) list(5);
              list(6) list(2) list(4);
              list(5) list(4) list(3)];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function pv = newposition(p,v,f,dt,m)
    P = p + v*dt + f/m/2*dt^2;%new position
    V = v + f/m*dt;%new velocity
    pv = [P;V];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function W = newattitude(w,t,dt,I)
    W = I\(t-cross(w,I*w))*dt+w;%changed angular rate
end
```

C.2 plotSat.m

Output file for paired satellite simulation.

```
function plotSat(A,B,C,D,dt,plon,dScale)
%plotSat(A,B,C,D,dt,plon,dScale)
%
%   A,B,C,D -- arrays giving the state of the four satellites through time;
%              columns are time steps, rows 1-3 are x,y,z coordinates, rows
%              4-6 are dipole vectors, rows 7-9 are velocity vectors, and
%              rows 10-12 are angular velocity vectors.
%
%   dt      -- time step
%
%   plon    -- 12-cell boolean array telling which of the following outputs
%              to activate:
```

```
%
%      1: Main plot -- x-y plane, showing dipole vectors at each moment.
%         Side plots -- x and y axes over time.
%      2: Main plot -- x-z plane, showing dipole vectors at each moment.
%         Side plots -- x and z axes over time.
%      3: Main plot -- y-z plane, showing dipole vectors at each moment.
%         Side plots -- y and z axes over time.
%
%             (In the preceeding three plots, the two side plots are
%              arranged so that the axis in question is lined up with the
%              main plot, and time increases perpendicularly to the main
%              plot.)
%
%      4: 3-d plot of the satellites through time.  (Formerly showing
%         dipole moments at each point; to save rendering time, only the
%         locations are now shown.)
%      5: 3-d dipole moments of each satellite through time; satellite
%         positions are not shown.
%      6: Angular velocities of each satellite in time, in four subplots.
%      7: Satellite velocities in time; vx is red, vy is blue, vz is green;
%         in four subplots.
%      8: Satellite dipole magnitudes in time, in four subplots.  Y-axis
%         labels show magnitude scaled to the dipole strength at the outer
%         bound.
%      9: Distances of each satellite to the other three.  For example, the
%         A graph shows distances AB, AC, and AD; the B graph shows
%         distances BA, BC, and BD, and so on.  Specific scenario scripts
%         can call figure(9) and draw in lines showing the distances at
%         which the dipoles activate.
%     10: Plots of x vs. vx, y vs. vy, and z vs. vz.
%     11: Distances of the four satellites over time to the CG of the
%         system, along with the mean of the four distances (in pink).
%     12: Text output of the state of each satellite in time (mainly for
%         debugging purposes.)
%
%             (When multiple satellites have outputs in one plot, satellite
%              A (which has the additional thrust) is red, B is blue, C is
%              green, and D is black.)
%
%   dScale  -- Dipole magnitude scaling factor for plots 1-4.  Most input
%              files use the nominal dipole strength at the outer limit of
%              allowable distance, which makes most dipole magnitudes max
%              out in the general vicinity of 0.75.

global gPaperMode  %how big will the text be?
if isempty(gPaperMode) || gPaperMode == 0
    tsize = 10; %normal size if it's regular debugging
else
    tsize = 14; %larger size if it's for the paper
end

times = 0:dt:dt*size(A,2)-dt;%vector of times at each time step
wun =  ones(size(times));
A(4:6,:)=A(4:6,:)/dScale;%In order to make most graphs readable, the dipole
```

```
B(4:6,:)=B(4:6,:)/dScale;%moments need to be on the same order of magnitude
C(4:6,:)=C(4:6,:)/dScale;%as the distances; in the case of the test values,
D(4:6,:)=D(4:6,:)/dScale;%that order is one.  Thus, it is scaled to dScale.

if plon(1)%X—Y Plane
figure(1)
clf
title('XY Plane')
subplot(3,3,[4 5 7 8])%Main subplot —— plane, with dipole vectors shown
hold on
quiver(A(1,:),A(2,:),A(4,:),A(5,:),0,'r')
quiver(B(1,:),B(2,:),B(4,:),B(5,:),0,'b')
quiver(C(1,:),C(2,:),C(4,:),C(5,:),0,'g')
quiver(D(1,:),D(2,:),D(4,:),D(5,:),0,'k')
plot(A(1,:),A(2,:),':r',B(1,:),B(2,:),':b',...
    C(1,:),C(2,:),':g',D(1,:),D(2,:),':k')
xlabel('x (m)','FontSize',tsize)
ylabel('y (m)','FontSize',tsize)
hold off
set(gca,'FontSize',tsize)
axis equal
limx = xlim;
limy = ylim;
subplot(3,3,[6 9])%Side plot showing Y location over time
hold on
plot(times,A(2,:),'r')
plot(times,B(2,:),'b')
plot(times,C(2,:),'g')
plot(times,D(2,:),'k')
xlabel('t (s)','FontSize',tsize)
ylim(limy)
hold off
set(gca,'FontSize',tsize)
subplot(3,3,[1 2])%Side plot showing X location over time
hold on
plot(A(1,:),times,'r')
plot(B(1,:),times,'b')
plot(C(1,:),times,'g')
plot(D(1,:),times,'k')
ylabel('t (s)','FontSize',tsize)
xlim(limx)
hold off
set(gca,'FontSize',tsize)
end

if plon(2)%X—Z Plane
figure(2)
clf
title('XZ Plane')
subplot(3,3,[4 5 7 8])%Main subplot —— plane, with dipole vectors shown
hold on
quiver(A(3,:),A(1,:),A(6,:),A(4,:),0,'r')
quiver(B(3,:),B(1,:),B(6,:),B(4,:),0,'b')
quiver(C(3,:),C(1,:),C(6,:),C(4,:),0,'g')
quiver(D(3,:),D(1,:),D(6,:),D(4,:),0,'k')
plot(A(3,:),A(1,:),':r',B(3,:),B(1,:),':b',...
    C(3,:),C(1,:),':g',D(3,:),D(1,:),':k')
xlabel('z (m)','FontSize',tsize)
ylabel('x (m)','FontSize',tsize)
hold off
set(gca,'FontSize',tsize)
axis equal
limx = xlim;
limy = ylim;
subplot(3,3,[6 9])%Side plot showing X location over time
hold on
plot(times,A(1,:),'r')
plot(times,B(1,:),'b')
plot(times,C(1,:),'g')
plot(times,D(1,:),'k')
xlabel('t (s)','FontSize',tsize)
ylim(limy)
hold off
set(gca,'FontSize',tsize)
subplot(3,3,[1 2])%Side plot showing Z location over time
hold on
plot(A(3,:),times,'r')
plot(B(3,:),times,'b')
plot(C(3,:),times,'g')
plot(D(3,:),times,'k')
ylabel('t (s)','FontSize',tsize)
xlim(limx)
hold off
set(gca,'FontSize',tsize)
end

if plon(3)%Y-Z Plane
figure(3)
clf
title('YZ Plane')
subplot(3,3,[4 5 7 8])%Main subplot -- plane, with dipole vectors shown
hold on
quiver(A(2,:),A(3,:),A(5,:),A(6,:),0,'r')
quiver(B(2,:),B(3,:),B(5,:),B(6,:),0,'b')
quiver(C(2,:),C(3,:),C(5,:),C(6,:),0,'g')
quiver(D(2,:),D(3,:),D(5,:),D(6,:),0,'k')
plot(A(2,:),A(3,:),':r',B(2,:),B(3,:),':b',...
    C(2,:),C(3,:),':g',D(2,:),D(3,:),':k')
xlabel('y (m)','FontSize',tsize)
ylabel('z (m)','FontSize',tsize)
hold off
set(gca,'FontSize',tsize)
axis equal
limx = xlim;
limy = ylim;
subplot(3,3,[6 9])%Side plot showing Z location over time
hold on
plot(times,A(3,:),'r')
plot(times,B(3,:),'b')
plot(times,C(3,:),'g')
plot(times,D(3,:),'k')
xlabel('t (s)','FontSize',tsize)
ylim(limy)
hold off
set(gca,'FontSize',tsize)
subplot(3,3,[1 2])%Side plot showing Y location over time
hold on
plot(A(2,:),times,'r')
plot(B(2,:),times,'b')
plot(C(2,:),times,'g')
plot(D(2,:),times,'k')
ylabel('t (s)','FontSize',tsize)
xlim(limx)
hold off
set(gca,'FontSize',tsize)
end

if plon(4)%3-d space plot
figure(4)
clf
hold on
%Quiver plot showing locations and dipole moments at each point
%--Takes forever to render in large graphs
%quiver3(A(1,:),A(2,:),A(3,:),A(4,:),A(5,:),A(6,:),0,'r')
%quiver3(A(1,:),A(2,:),A(3,:),A(7,:),A(8,:),A(9,:), ':r')
%quiver3(B(1,:),B(2,:),B(3,:),B(4,:),B(5,:),B(6,:),0,'b')
%quiver3(B(1,:),B(2,:),B(3,:),B(7,:),B(8,:),B(9,:), ':b')
%quiver3(C(1,:),C(2,:),C(3,:),C(4,:),C(5,:),C(6,:),0,'g')
%quiver3(C(1,:),C(2,:),C(3,:),C(7,:),C(8,:),C(9,:), ':g')
%quiver3(D(1,:),D(2,:),D(3,:),D(4,:),D(5,:),D(6,:),0,'k')
%quiver3(D(1,:),D(2,:),D(3,:),D(7,:),D(8,:),D(9,:), ':k')

%Alternative plot 4: same plot, no vectors shown
%--much faster to render; not much useful information lost
plot3(A(1,:),A(2,:),A(3,:),'r')
plot3(B(1,:),B(2,:),B(3,:),'b')
plot3(C(1,:),C(2,:),C(3,:),'g')
plot3(D(1,:),D(2,:),D(3,:),'k')
xlabel('x (m)','FontSize',tsize)
ylabel('y (m)','FontSize',tsize)
zlabel('z (m)','FontSize',tsize)
hold off
set(gca,'FontSize',tsize)
axis equal
end

if plon(5)%3-d dipole plot
figure(5) %Arrows point in the direction of the dipole.  Starting points
clf       %of each vector are a bit more complicated: the X-location shows
hold on   %time.  Y and Z locations, and colors, indicate the satellite:
quiver3(times, wun, wun,A(4,:),A(5,:),A(6,:),0,'r')%Sat A at (t,+1,+1)
quiver3(times, wun,-wun,B(4,:),B(5,:),B(6,:),0,'b')%Sat B at (t,+1,-1)
```

```
quiver3(times,-wun, wun,C(4,:),C(5,:),C(6,:),0,'g')%Sat C at (t,-1,+1)
quiver3(times,-wun,-wun,D(4,:),D(5,:),D(6,:),0,'k')%Sat D at (t,-1,-1)
plot3(times,wun,wun,'r',times,wun,-wun,'b',...
    times,-wun,wun,'g',times,-wun,-wun,'k')
xlabel('time','FontSize',tsize)
hold off          %colors are the standard A-red, B-blue, C-green, D-black
set(gca,'FontSize',tsize)
end

if plon(6)%angular velocity plot
figure(6)%one subplot per satellite
subplot(2,2,1)%cyan -- wx; magenta -- wy; black -- wz
plot(times,A(12,:),'k',times,A(11,:),'m',times,A(10,:),'c')
title('Sat A Angular Velocity','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('\omega (r/s)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,2)
plot(times,B(12,:),'k',times,B(11,:),'m',times,B(10,:),'c')
title('Sat B Angular Velocity','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('\omega (r/s)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,3)
plot(times,C(12,:),'k',times,C(11,:),'m',times,C(10,:),'c')
title('Sat C Angular Velocity','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('\omega (r/s)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,4)
plot(times,D(12,:),'k',times,D(11,:),'m',times,D(10,:),'c')
title('Sat D Angular Velocity','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('\omega (r/s)','FontSize',tsize)
set(gca,'FontSize',tsize)
end

if plon(7)%velocity plot
figure(7)%one subplot per satellite
subplot(2,2,1)%cyan -- vx; magenta -- vy; black -- vz
plot(times,A(9,:),'k',times,A(8,:),'m',times,A(7,:),'c')
title('Sat A Velocity','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('v (m/s)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,2)
plot(times,B(9,:),'k',times,B(8,:),'m',times,B(7,:),'c')
title('Sat B Velocity','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('v (m/s)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,3)
plot(times,C(9,:),'k',times,C(8,:),'m',times,C(7,:),'c')
title('Sat C Velocity','FontSize',tsize)
```

```
xlabel('t (s)','FontSize',tsize)
ylabel('v (m/s)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,4)
plot(times,D(9,:),'k',times,D(8,:),'m',times,D(7,:),'c')
title('Sat D Velocity','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('v (m/s)','FontSize',tsize)
set(gca,'FontSize',tsize)
end

if plon(8)%dipole magnitude plot
figure(8)%one subplot per satellite
subplot(2,2,1)
plot(times,sqrt(sum(A(4:6,:).^2,1))*dScale,'r')
title('Sat A Magnitude','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('u (J/T)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,2)
plot(times,sqrt(sum(B(4:6,:).^2,1))*dScale,'b')
title('Sat B Magnitude','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('u (J/T)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,3)
plot(times,sqrt(sum(C(4:6,:).^2,1))*dScale,'g')
title('Sat C Magnitude','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('u (J/T)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,4)
plot(times,sqrt(sum(D(4:6,:).^2,1))*dScale,'k')
title('Sat D Magnitude','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('u (J/T)','FontSize',tsize)
set(gca,'FontSize',tsize)
end

if plon(9)%distance plots
figure(9)
dr = sqrt(sum((A(1:3,:)-B(1:3,:)).^2,1));%first, distances between each
de = sqrt(sum((A(1:3,:)-C(1:3,:)).^2,1));% pair of satellites are found
dl = sqrt(sum((A(1:3,:)-D(1:3,:)).^2,1));
re = sqrt(sum((B(1:3,:)-C(1:3,:)).^2,1));
rl = sqrt(sum((B(1:3,:)-D(1:3,:)).^2,1));
el = sqrt(sum((C(1:3,:)-D(1:3,:)).^2,1));
subplot(2,2,1)%AB AC and AD
plot(times,dr,'b',times,de,'g',times,dl,'k')
title('Sat A Distances (red)','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('dist (m)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,2)%AB BC and BD
```

```
plot(times,dr,'r',times,re,'g',times,rl,'k')
title('Sat B Distances (blue)','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('dist (m)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,3)%AC BC and CD
plot(times,de,'r',times,re,'b',times,el,'k')
title('Sat C Distances (green)','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('dist (m)','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,4)%AD BD and CD
plot(times,dl,'r',times,rl,'b',times,el,'g')
title('Sat D Distances (black)','FontSize',tsize)
xlabel('t (s)','FontSize',tsize)
ylabel('dist (m)','FontSize',tsize)
set(gca,'FontSize',tsize)
end

cm = (A(1:3,:)+B(1:3,:)+C(1:3,:)+D(1:3,:))/4;%center of mass
Aoff = A(1:3,:)−cm;%separation distances from center of mass for each sat
Boff = B(1:3,:)−cm;
Coff = C(1:3,:)−cm;
Doff = D(1:3,:)−cm;

if plon(10)%position−velocity graphs
figure(10)%distance from cm plotted against velocity −− stable if it loops
subplot(2,2,1)%X−Vx
plot(Aoff(1,:),A(7,:),'r',Boff(1,:),B(7,:),'b',...
    Coff(1,:),C(7,:),'g',Doff(1,:),D(7,:),'k')
xlabel('x','FontSize',tsize)
ylabel('vx','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,2)%Y−Vy
plot(Aoff(2,:),A(8,:),'r',Boff(2,:),B(8,:),'b',...
    Coff(2,:),C(8,:),'g',Doff(2,:),D(8,:),'k')
xlabel('y','FontSize',tsize)
ylabel('vy','FontSize',tsize)
set(gca,'FontSize',tsize)
subplot(2,2,3)%Z−Vz
plot(Aoff(3,:),A(9,:),'r',Boff(3,:),B(9,:),'b',...
    Coff(3,:),C(9,:),'g',Doff(3,:),D(9,:),'k')
xlabel('z','FontSize',tsize)
ylabel('vz','FontSize',tsize)
set(gca,'FontSize',tsize)
end

if plon(11)%distances from center of mass over time
figure(11)
Aoff = sqrt(Aoff(1,:).^2+Aoff(2,:).^2+Aoff(3,:).^2);
Boff = sqrt(Boff(1,:).^2+Boff(2,:).^2+Boff(3,:).^2);
Coff = sqrt(Coff(1,:).^2+Coff(2,:).^2+Coff(3,:).^2);
Doff = sqrt(Doff(1,:).^2+Doff(2,:).^2+Doff(3,:).^2);
offness = (Aoff+Boff+Coff+Doff)/4;%average distace from center of mass
```

```
plot(times,offness,'.m',times,Aoff,'r',...
    times,Boff,'b',times,Coff,'g',times,Doff,'k')
xlabel('Time (s)','FontSize',tsize)       %(average distance is useful for
ylabel('Distance from CG','FontSize',tsize)% determining if the trend is to
set(gca,'FontSize',tsize)                  % move together or apart)
end

if plon(12)%numerical dump to command window.  Useful mainly for debugging.
disp(A)
disp(B)
disp(C)
disp(D)
end
```

C.3 linearsats.m

Setup and control file for satellites in a linear formation.

```
function linearsats(arrangement)
%linearsats(arrangement)
%  arrangement -- the number, 1-4, corresponding to satellite A's position
%                 in the line.
%
%  Runs simulation of EMFF satellites starting in a linear formation
%  See tstepSat.m for state calculation; see plotSat.m for output
%  This script contains the initial setup, the control laws, additional
%  thrust on satellite A, and the J2 perturbation on all satellites.
%
%  Figure 9 is modified showing activation distances, inner and outer, for
%  the dipoles.
%
%  An additional plot is made (fig. 12) showing position vs. time.

global gEarthDipoleOn gPaperMode
if isempty(gPaperMode)
    gPaperMode = 0;
end
if ¬gPaperMode  %text size for graphs:
    tsize = 10; %normal size if it's regular debugging
else
    tsize = 14; %larger size if it's for the paper
end
gEarthDipoleOn = 0;

%Satellite characteristics
mA = 50;
mB = 50;
mC = 50;
mD = 50;

iA = [100;100;100;0;0;0];
```

```
iB = [100;100;100;0;0;0];
iC = [100;100;100;0;0;0];
iD = [100;100;100;0;0;0];

%parameters
iter = 10000;
dt = .03125;
outon = 8;
inon = 0.20;

mTrgt = max([mA mB mC mD]);
dTrgt = outon;
strength = sqrt(8*dTrgt^4*mTrgt/3e-7);
innerfac = (outon/inon)^4;

rocket = .001;

%initial setup
switch arrangement
    case 1
        pA = [0;0;0];
        pB = [0;0;-1];
        pC = [0;0;-2];
        pD = [0;0;-3];
    case 2
        pA = [0;0;-1];
        pB = [0;0;0];
        pC = [0;0;-2];
        pD = [0;0;-3];
    case 3
        pA = [0;0;-2];
        pB = [0;0;0];
        pC = [0;0;-1];
        pD = [0;0;-3];
    otherwise
        pA = [0;0;-3];
        pB = [0;0;0];
        pC = [0;0;-1];
        pD = [0;0;-2];
end

vA = [0;0;0];
vB = [0;0;0];
vC = [0;0;0];
vD = [0;0;0];

wA = [0;0;0];
wB = [0;0;0];
wC = [0;0;0];
wD = [0;0;0];

bA = [0;0;0];
bB = [0;0;0];
bC = [0;0;0];
```

```
bD = [0;0;0];

A = zeros(12,iter+1);
B = zeros(12,iter+1);
C = zeros(12,iter+1);
D = zeros(12,iter+1);

%actual program
A(:,1) = [pA;bA;vA;0;0;0];
B(:,1) = [pB;bB;vB;0;0;0];
C(:,1) = [pC;bC;vC;0;0;0];
D(:,1) = [pD;bD;vD;0;0;0];

for i=1:iter
    distances = [norm(pA-pB);
                 norm(pA-pC);
                 norm(pA-pD);
                 norm(pB-pC);
                 norm(pB-pD);
                 norm(pC-pD)];
    bA = [0;0;0];
    bB = [0;0;0];
    bC = [0;0;0];
    bD = [0;0;0];
    for j=6:-1:1%Collision avoidance phase
        if ordinal(distances,j)<inon%starting with the closest pair, check
            mdist = ordinal(distances,j);%to see if they are:
            switch mdist                 %1. too close, and
                case distances(1)        %2. moving toward each other
                    if togethering(pA,pB,vA,vB)
                        vrel = norm(vA-vB);
                        both = lineup(pA,pB)*(mdist/inon)^-3*sqrt(vrel);
                        bA = strength/innerfac*both; %if so, activate the
                        bB = -strength/innerfac*both;%dipoles opposite each
                    end                              %other, for repulsion.
                case distances(2)
                    if togethering(pA,pC,vA,vC)
                        vrel = norm(vA-vC);
                        both = lineup(pA,pC)*(mdist/inon)^-3*sqrt(vrel);
                        bA = strength/innerfac*both;
                        bC = -strength/innerfac*both;
                    end
                case distances(3)
                    if togethering(pA,pD,vA,vD)
                        vrel = norm(vA-vD);
                        both = lineup(pA,pD)*(mdist/inon)^-3*sqrt(vrel);
                        bA = strength/innerfac*both;
                        bD = -strength/innerfac*both;
                    end
                case distances(4)
                    if togethering(pB,pC,vB,vC)
                        vrel = norm(vB-vC);
                        both = lineup(pB,pC)*(mdist/inon)^-3*sqrt(vrel);
                        bB = strength/innerfac*both;
                        bC = −strength/innerfac∗both;
                    end
                case distances(5)
                    if togethering(pB,pD,vB,vD)
                        vrel = norm(vB−vD);
                        both = lineup(pB,pD)∗(mdist/inon)ˆ−3∗sqrt(vrel);
                        bB = strength/innerfac∗both;
                        bD = −strength/innerfac∗both;
                    end
                otherwise
                    if togethering(pC,pD,vC,vD)
                        vrel = norm(vC−vD);
                        both = lineup(pC,pD)∗(mdist/inon)ˆ−3∗sqrt(vrel);
                        bC = strength/innerfac∗both;
                        bD = −strength/innerfac∗both;
                    end
            end
        end
        if any([bA;bB;bC;bD])
            break
        end
    end
    for j=1:6
        if (ordinal(distances,j)>outon) && (¬any([bA;bB;bC;bD]))
            mdist = ordinal(distances,j);%starting with the furthest pair,
            switch mdist                 %1. check if they're too far apart
                case distances(1)        %2. & moving away from each other
                    if ¬togethering(pA,pB,vA,vB)
                        vrel = norm(vA−vB);
                        both = lineup(pA,pB)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bB = strength∗both;
                    end
                case distances(2)
                    if ¬togethering(pA,pC,vA,vC)
                        vrel = norm(vA−vC);
                        both = lineup(pA,pC)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bC = strength∗both;
                    end
                case distances(3)
                    if ¬togethering(pA,pD,vA,vD)
                        vrel = norm(vA−vD);
                        both = lineup(pA,pD)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bD = strength∗both;
                    end
                case distances(4)
                    if ¬togethering(pB,pC,vB,vC)
                        vrel = norm(vB−vC);
                        both = lineup(pB,pC)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bB = strength∗both;
                        bC = strength∗both;
                    end
                case distances(5)
                    if ¬togethering(pB,pD,vB,vD)
                        vrel = norm(vB−vD);
                        both = lineup(pB,pD)*(mdist/outon)ˆ4*sqrt(vrel);
                        bB = strength*both;
                        bD = strength*both;
                    end
                otherwise
                    if ¬togethering(pC,pD,vC,vD)
                        vrel = norm(vC−vD);
                        both = lineup(pC,pD)*(mdist/outon)ˆ4*sqrt(vrel);
                        bC = strength*both;
                        bD = strength*both;
                    end
            end
        end
        if any([bA;bB;bC;bD])
            break
        end
    end

    placespeed = [pA pB pC pD;vA vB vC vD;wA wB wC wD];
    placespeed = tstepSat(placespeed,[mA mB mC mD],[iA iB iC iD],...
        [bA bB bC bD],dt);

    %update position
    pA = placespeed(1:3,1);
    pB = placespeed(1:3,2);
    pC = placespeed(1:3,3);
    pD = placespeed(1:3,4);

    %J2 perturbation
    cm = (pA+pB+pC+pD)/4;
    j2A = (pA−cm)*3.4213e−9*dt;
    j2B = (pB−cm)*3.4213e−9*dt;
    j2C = (pC−cm)*3.4213e−9*dt;
    j2D = (pD−cm)*3.4213e−9*dt;

    %update velocity, including J2 and rocket thrust
    vA = placespeed(4:6,1)+[0;0; rocket*dt*.75]+j2A;
    vB = placespeed(4:6,2)+[0;0;−rocket*dt*.25]+j2B;
    vC = placespeed(4:6,3)+[0;0;−rocket*dt*.25]+j2C;
    vD = placespeed(4:6,4)+[0;0;−rocket*dt*.25]+j2D;
    %though theoretically, only satellite A is accelerated, the other three
    %satellites are here given an acceleration as well so that the center
    %of mass remains in one place, for ease of comparison.

    %update angular velocity
    wA = placespeed(7:9,1);
    wB = placespeed(7:9,2);
    wC = placespeed(7:9,3);
    wD = placespeed(7:9,4);

    %update state matrix
```

```
    A(1:3,i+1) = pA;
    B(1:3,i+1) = pB;
    C(1:3,i+1) = pC;
    D(1:3,i+1) = pD;
    A(4:6,i+1) = bA;
    B(4:6,i+1) = bB;
    C(4:6,i+1) = bC;
    D(4:6,i+1) = bD;
    A(7:9,i+1) = vA;
    B(7:9,i+1) = vB;
    C(7:9,i+1) = vC;
    D(7:9,i+1) = vD;
    A(10:12,i+1) = wA;
    B(10:12,i+1) = wB;
    C(10:12,i+1) = wC;
    D(10:12,i+1) = wD;
end

if gPaperMode
    plotSat(A,B,C,D,dt,[0 0 0 0 0 0 0 1 0 0 0 0],strength);
else
    plotSat(A,B,C,D,dt,[0 0 1 0 0 0 1 1 1 1 1 0],strength);

    figure(9)
    subplot(2,2,1)
    hold on
    plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
    hold off
    subplot(2,2,2)
    hold on
    plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
    hold off
    subplot(2,2,3)
    hold on
    plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
    hold off
    subplot(2,2,4)
    hold on
    plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
    hold off
end %¬gPaperMode
figure(12)
clf
hold on
times = 0:dt:dt*iter;
plot(times,A(3,:)+1.5,'r')
plot(times,B(3,:)+1.5,'b')
plot(times,C(3,:)+1.5,'g')
plot(times,D(3,:)+1.5,'k')
xlabel('t','FontSize',tsize)
ylabel('dist from cg (m)','FontSize',tsize)
legend('Satellite A','Satellite B','Satellite C','Satellite D',...
    'Location','Best')
set(gca,'FontSize',tsize)
```

```
hold off
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function R = lineup(p1,p2)
    r = p1-p2;
    R = r/norm(r);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function whether = togethering(p1,p2,v1,v2)
    r12 = p1-p2;
    r21 = p2-p1;
    v21 = v1-v2;
    v12 = v2-v1;
    whether = (dot(r12,v12)>0)|(dot(r21,v21)>0);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function result = ordinal(vector,number)
    list = sort(vector,'descend');
    result = list(number);
end
```

C.4 squaresats.m

Setup and control file for satellites in a square formation.

```
function squaresats()
%squaresats()
%  Runs simulation of EMFF satellites starting in a square formation
%  See tstepSat.m for state calculation; see plotSat.m for output
%  This script contains the initial setup, the control laws, additional
%  thrust on satellite A, and the J2 perturbation on all satellites.
%
%  Figure 9 is modified showing activation distances, inner and outer, for
%  the dipoles.
%
%  Two additional plots are made (fig. 12 & 13) showing position vs. time,
%  first y-t and then z-t.

global gEarthDipoleOn gReverseThrust gSideThrust gPaperMode
if isempty(gPaperMode)
    gPaperMode = 0;
end
if isempty(gReverseThrust)
    gReverseThrust = 0;
end
```

```
if isempty(gSideThrust)
    gSideThrust = 0;
end
if ¬gPaperMode
    tsize = 10; %text size for graphs:
    sizet = 10; %normal size if it's regular debugging
else
    tsize = 14; %larger size if it's for the paper
    sizet = 20; %even larger for some graphs
end
gEarthDipoleOn = 0;

%Satellite characteristics
mA = 50;
mB = 50;
mC = 50;
mD = 50;

iA = [100;100;100;0;0;0];
iB = [100;100;100;0;0;0];
iC = [100;100;100;0;0;0];
iD = [100;100;100;0;0;0];

%parameters
if gSideThrust
    iter = 80000;
else
    if gReverseThrust
        iter = 5000;
    else
        iter = 100000;
    end
end
dt = .03125;
outon = 8;
inon = 0.60;

mTrgt = max([mA mB mC mD]);
dTrgt = outon;
strength = sqrt(8*dTrgtˆ4*mTrgt/3e−7);
innerfac = (outon/inon)ˆ4;

if gReverseThrust
    rocket = −.001;
else
    rocket = .001;
end

%initial setup
if gSideThrust
    pA = [0;1;0];
    pB = [0;0;1];
else
    pA = [0;0;1];
    pB = [0;1;0];
end
pC = [0;0;-1];
pD = [0;-1;0];

vA = [0;0;0];
vB = [0;0;0];
vC = [0;0;0];
vD = [0;0;0];

wA = [0;0;0];
wB = [0;0;0];
wC = [0;0;0];
wD = [0;0;0];

bA = [0;0;0];
bB = [0;0;0];
bC = [0;0;0];
bD = [0;0;0];

A = zeros(12,iter+1);
B = zeros(12,iter+1);
C = zeros(12,iter+1);
D = zeros(12,iter+1);

%actual program
A(:,1) = [pA;bA;vA;0;0;0];
B(:,1) = [pB;bB;vB;0;0;0];
C(:,1) = [pC;bC;vC;0;0;0];
D(:,1) = [pD;bD;vD;0;0;0];

for i=1:iter
    distances = [norm(pA-pB);
                 norm(pA-pC);
                 norm(pA-pD);
                 norm(pB-pC);
                 norm(pB-pD);
                 norm(pC-pD)];
    bA = [0;0;0];
    bB = [0;0;0];
    bC = [0;0;0];
    bD = [0;0;0];
    for j=6:-1:1%Collision avoidance phase
        if ordinal(distances,j)<inon%starting with the closest pair, check
            mdist = ordinal(distances,j);%to see if they are:
            switch mdist                %1. too close, and
                case distances(1)       %2. moving toward each other
                    if togethering(pA,pB,vA,vB)
                        vrel = norm(vA-vB);
                        both = lineup(pA,pB)*(mdist/inon)^-3*sqrt(vrel);
                        bA = strength/innerfac*both;
                        bB = -strength/innerfac*both;
                    end
                case distances(2)
                    if togethering(pA,pC,vA,vC)
                        vrel = norm(vA–vC);
                        both = lineup(pA,pC)*(mdist/inon)^–3*sqrt(vrel);
                        bA = strength/innerfac*both; %if so, activate the
                        bC = –strength/innerfac*both;%dipoles opposite each
                    end                              %other, for repulsion.
                case distances(3)
                    if togethering(pA,pD,vA,vD)
                        vrel = norm(vA–vD);
                        both = lineup(pA,pD)*(mdist/inon)^–3*sqrt(vrel);
                        bA = strength/innerfac*both;
                        bD = –strength/innerfac*both;
                    end
                case distances(4)
                    if togethering(pB,pC,vB,vC)
                        vrel = norm(vB–vC);
                        both = lineup(pB,pC)*(mdist/inon)^–3*sqrt(vrel);
                        bB = strength/innerfac*both;
                        bC = –strength/innerfac*both;
                    end
                case distances(5)
                    if togethering(pB,pD,vB,vD)
                        vrel = norm(vB–vD);
                        both = lineup(pB,pD)*(mdist/inon)^–3*sqrt(vrel);
                        bB = strength/innerfac*both;
                        bD = –strength/innerfac*both;
                    end
                otherwise
                    if togethering(pC,pD,vC,vD)
                        vrel = norm(vC–vD);
                        both = lineup(pC,pD)*(mdist/inon)^–3*sqrt(vrel);
                        bC = strength/innerfac*both;
                        bD = –strength/innerfac*both;
                    end
            end
        end
        if any([bA;bB;bC;bD])
            break
        end
    end
    for j=1:6
        if (ordinal(distances,j)>outon) && (¬any([bA;bB;bC;bD]))
            mdist = ordinal(distances,j);%starting with the furthest pair,
            switch mdist                 %1. check if they're too far apart
                case distances(1)        %2. & moving away from each other
                    if ¬togethering(pA,pB,vA,vB)
                        vrel = norm(vA–vB);
                        both = lineup(pA,pB)*(mdist/outon)^4*sqrt(vrel);
                        bA = strength*both;
                        bB = strength*both;
                    end
                case distances(2)
                    if ¬togethering(pA,pC,vA,vC)
                        vrel = norm(vA–vC);
                        both = lineup(pA,pC)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bC = strength∗both;
                    end
                case distances(3)
                    if ¬togethering(pA,pD,vA,vD)
                        vrel = norm(vA−vD);
                        both = lineup(pA,pD)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bD = strength∗both;
                    end
                case distances(4)
                    if ¬togethering(pB,pC,vB,vC)
                        vrel = norm(vB−vC);
                        both = lineup(pB,pC)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bB = strength∗both;
                        bC = strength∗both;
                    end
                case distances(5)
                    if ¬togethering(pB,pD,vB,vD)
                        vrel = norm(vB−vD);
                        both = lineup(pB,pD)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bB = strength∗both;
                        bD = strength∗both;
                    end
                otherwise
                    if ¬togethering(pC,pD,vC,vD)
                        vrel = norm(vC−vD);
                        both = lineup(pC,pD)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bC = strength∗both;
                        bD = strength∗both;
                    end
            end
        end
        if any([bA;bB;bC;bD])
            break
        end
    end

    placespeed = [pA pB pC pD;vA vB vC vD;wA wB wC wD];
    placespeed = tstepSat(placespeed,[mA mB mC mD],...
        [iA iB iC iD],[bA bB bC bD],dt);

    %update position
    pA = placespeed(1:3,1);
    pB = placespeed(1:3,2);
    pC = placespeed(1:3,3);
    pD = placespeed(1:3,4);

    %J2 perturbation
    cm = (pA+pB+pC+pD)/4;
    j2A = (pA−cm)∗3.4213e−9∗dt;
    j2B = (pB−cm)∗3.4213e−9∗dt;
    j2C = (pC−cm)∗3.4213e−9∗dt;
```

```
    j2D = (pD–cm)*3.4213e–9*dt;

    %update velocity, including J2 perturbation and rocket thrust
    vA = placespeed(4:6,1)+[0;0; rocket*dt*.75]+j2A;
    vB = placespeed(4:6,2)+[0;0;–rocket*dt*.25]+j2B;
    vC = placespeed(4:6,3)+[0;0;–rocket*dt*.25]+j2C;
    vD = placespeed(4:6,4)+[0;0;–rocket*dt*.25]+j2D;
    %though theoretically, only satellite A is accelerated, the other three
    %satellites are here given an acceleration as well so that the center
    %of mass remains in one place, for ease of comparison.

    %update angular velocity
    wA = placespeed(7:9,1);
    wB = placespeed(7:9,2);
    wC = placespeed(7:9,3);
    wD = placespeed(7:9,4);

    %update state matrix
    A(1:3,i+1) = pA;
    B(1:3,i+1) = pB;
    C(1:3,i+1) = pC;
    D(1:3,i+1) = pD;
    A(4:6,i+1) = bA;
    B(4:6,i+1) = bB;
    C(4:6,i+1) = bC;
    D(4:6,i+1) = bD;
    A(7:9,i+1) = vA;
    B(7:9,i+1) = vB;
    C(7:9,i+1) = vC;
    D(7:9,i+1) = vD;
    A(10:12,i+1) = wA;
    B(10:12,i+1) = wB;
    C(10:12,i+1) = wC;
    D(10:12,i+1) = wD;
end

if gPaperMode
    plotSat(A,B,C,D,dt,[0 0 0 0 0 0 0 1 1 0 0 0],strength);
else
    plotSat(A,B,C,D,dt,[0 0 1 0 0 0 1 1 1 1 1 0],strength);
end %gPaperMode

figure(9)
subplot(2,2,1)
hold on
plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
hold off
subplot(2,2,2)
hold on
plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
hold off
subplot(2,2,3)
hold on
plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
```

```
hold off
subplot(2,2,4)
hold on
plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
hold off

figure(12)
clf
hold on
times = 0:dt:dt*iter;
plot(times,A(2,:),'r')
plot(times,B(2,:),'b')
plot(times,C(2,:),'g')
plot(times,D(2,:),'k')
xlabel('t (s)','FontSize',sizet)
ylabel('y dist from cg (m)','FontSize',sizet)
%legend('Satellite A','Satellite B','Satellite C','Satellite D')
hold off
set(gca,'FontSize',sizet)
figure(13)
clf
hold on
plot(times,A(3,:),'r')
plot(times,B(3,:),'b')
plot(times,C(3,:),'g')
plot(times,D(3,:),'k')
xlabel('t (s)','FontSize',sizet)
ylabel('z dist from cg (m)','FontSize',sizet)
%legend('Satellite A','Satellite B','Satellite C','Satellite D',...
%    'Location','Best')
hold off
set(gca,'FontSize',sizet)
figure(14)
clf
plot(A(2,:),A(3,:),'r',B(2,:),B(3,:),'b',...
     C(2,:),C(3,:),'g',D(2,:),D(3,:),'k')
axis equal
xlabel('y (m)','FontSize',tsize)
ylabel('z (m)','FontSize',tsize)
hold on
if gSideThrust
    plot(1,0,'.r')
    text(1,0,'A')
    plot(0,1,'.b')
    text(0,1,'B')
else
    plot(0,1,'.r')
    text(0,1,'A')
    plot(1,0,'.b')
    text(1,0,'B')
end
plot(0,-1,'.g')
text(0,-1,'C')
plot(-1,0,'.k')
```

```
text(-1,0,'D')
hold off
set(gca,'FontSize',tsize)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function R = lineup(p1,p2)
    r = p1-p2;
    R = r/norm(r);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function whether = togethering(p1,p2,v1,v2)
    r12 = p1-p2;
    r21 = p2-p1;
    v21 = v1-v2;
    v12 = v2-v1;
    whether = (dot(r12,v12)>0)|(dot(r21,v21)>0);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function result = ordinal(vector,number)
    list = sort(vector,'descend');
    result = list(number);
end
```

C.5 pairedsats.m

Setup and control file for satellites in a random formation.

```
function pairedsats()
%pairedsats()
%  Runs simulation of EMFF satellites starting in a random formation
%  See tstepSat.m for state calculation; see plotSat.m for output
%  This script contains the initial setup, the control laws, additional
%  thrust on satellite A, and the J2 perturbation on all satellites.
%
%  Figure 9 is modified showing activation distances, inner and outer, for
%  the dipoles.

global gEarthDipoleOn gPaperMode
if isempty(gPaperMode)
    gPaperMode = 0;
end
gEarthDipoleOn = 1;

%Satellite characteristics
mA = 50;
```

```
mB = 50;
mC = 50;
mD = 50;

iA = [100;100;100;0;0;0];
iB = [100;100;100;0;0;0];
iC = [100;100;100;0;0;0];
iD = [100;100;100;0;0;0];

%parameters
iter = 8000;
dt = .25;
outon = 8;
inon = 0.5;

mTrgt = max([mA mB mC mD]);
dTrgt = outon;
strength = sqrt(8*dTrgt^4*mTrgt/3e-7);
innerfac = (outon/inon)^4;

rocket = .01;

%initial setup
pB = random('unif',-4,4,3,1);
pC = random('unif',-4,4,3,1);
pD = random('unif',-4,4,3,1);
pA = -(pB+pC+pD);%random('unif',-4,4,3,1);

vB = random('unif',-.04,.04,3,1);
vC = random('unif',-.04,.04,3,1);
vD = random('unif',-.04,.04,3,1);
vA = -(vB+vC+vD);%random('unif',-.04,.04,3,1);

wA = [0;0;0];
wB = [0;0;0];
wC = [0;0;0];
wD = [0;0;0];

bA = [0;0;0];
bB = [0;0;0];
bC = [0;0;0];
bD = [0;0;0];

A = zeros(12,iter+1);
B = zeros(12,iter+1);
C = zeros(12,iter+1);
D = zeros(12,iter+1);

%actual program
A(:,1) = [pA;bA;vA;0;0;0];
B(:,1) = [pB;bB;vB;0;0;0];
C(:,1) = [pC;bC;vC;0;0;0];
D(:,1) = [pD;bD;vD;0;0;0];
```

```
for i=1:iter
    distances = [norm(pA-pB);
                 norm(pA-pC);
                 norm(pA-pD);
                 norm(pB-pC);
                 norm(pB-pD);
                 norm(pC-pD)];
    bA = [0;0;0];
    bB = [0;0;0];
    bC = [0;0;0];
    bD = [0;0;0];
    for j=6:-1:1%Collsion avoidance phase
        if ordinal(distances,j)<inon%starting with the closest pair, check
            mdist = ordinal(distances,j);%to see if they are:
            switch mdist                 %1. too close, and
                case distances(1)        %2. moving toward each other
                    if togethering(pA,pB,vA,vB)
                        vrel = norm(vA-vB);
                        both = lineup(pA,pB)*(mdist/inon)^-3*sqrt(vrel);
                        bA = strength/innerfac*both; %if so, activate the
                        bB = -strength/innerfac*both;%dipoles opposite each
                    end                              %other, for repulsion.
                case distances(2)
                    if togethering(pA,pC,vA,vC)
                        vrel = norm(vA-vC);
                        both = lineup(pA,pC)*(mdist/inon)^-3*sqrt(vrel);
                        bA = strength/innerfac*both;
                        bC = -strength/innerfac*both;
                    end
                case distances(3)
                    if togethering(pA,pD,vA,vD)
                        vrel = norm(vA-vD);
                        both = lineup(pA,pD)*(mdist/inon)^-3*sqrt(vrel);
                        bA = strength/innerfac*both;
                        bD = -strength/innerfac*both;
                    end
                case distances(4)
                    if togethering(pB,pC,vB,vC)
                        vrel = norm(vB-vC);
                        both = lineup(pB,pC)*(mdist/inon)^-3*sqrt(vrel);
                        bB = strength/innerfac*both;
                        bC = -strength/innerfac*both;
                    end
                case distances(5)
                    if togethering(pB,pD,vB,vD)
                        vrel = norm(vB-vD);
                        both = lineup(pB,pD)*(mdist/inon)^-3*sqrt(vrel);
                        bB = strength/innerfac*both;
                        bD = -strength/innerfac*both;
                    end
                otherwise
                    if togethering(pC,pD,vC,vD)
                        vrel = norm(vC-vD);
                        both = lineup(pC,pD)*(mdist/inon)^-3*sqrt(vrel);
                    bC = strength/innerfac∗both;
                    bD = –strength/innerfac∗both;
                end
            end
        end
        if any([bA;bB;bC;bD])
            break
        end
    end
    for j=1:6  %making sure they aren't too far apart
        if (ordinal(distances,j)>outon) && (¬any([bA;bB;bC;bD]))
            mdist = ordinal(distances,j);%starting with the furthest pair,
            switch mdist                 %1. check if they're too far apart
                case distances(1)        %2. & moving away from each other
                    if ¬togethering(pA,pB,vA,vB)
                        vrel = norm(vA–vB);
                        both = lineup(pA,pB)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bB = strength∗both;
                    end
                case distances(2)
                    if ¬togethering(pA,pC,vA,vC)
                        vrel = norm(vA–vC);
                        both = lineup(pA,pC)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bC = strength∗both;
                    end
                case distances(3)
                    if ¬togethering(pA,pD,vA,vD)
                        vrel = norm(vA–vD);
                        both = lineup(pA,pD)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bD = strength∗both;
                    end
                case distances(4)
                    if ¬togethering(pB,pC,vB,vC)
                        vrel = norm(vB–vC);
                        both = lineup(pB,pC)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bB = strength∗both;
                        bC = strength∗both;
                    end
                case distances(5)
                    if ¬togethering(pB,pD,vB,vD)
                        vrel = norm(vB–vD);
                        both = lineup(pB,pD)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bB = strength∗both;
                        bD = strength∗both;
                    end
                otherwise
                    if ¬togethering(pC,pD,vC,vD)
                        vrel = norm(vC–vD);
                        both = lineup(pC,pD)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bC = strength∗both;
                        bD = strength∗both;
                end
            end
        end
        if any([bA;bB;bC;bD])
            break
        end
    end

    placespeed = [pA pB pC pD;vA vB vC vD;wA wB wC wD];
    placespeed = tstepSat(placespeed,[mA mB mC mD],...
        [iA iB iC iD],[bA bB bC bD],dt);

    %update position
    pA = placespeed(1:3,1);
    pB = placespeed(1:3,2);
    pC = placespeed(1:3,3);
    pD = placespeed(1:3,4);

    %J2 perturbation
    cm = (pA+pB+pC+pD)/4;
    j2A = (pA-cm)*3.4213e-9*dt;
    j2B = (pB-cm)*3.4213e-9*dt;
    j2C = (pC-cm)*3.4213e-9*dt;
    j2D = (pD-cm)*3.4213e-9*dt;

    %update velocity, including J2 and rocket thrust
    vA = placespeed(4:6,1)+[0;0; rocket*dt*.75]+j2A;
    vB = placespeed(4:6,2)+[0;0;-rocket*dt*.25]+j2B;
    vC = placespeed(4:6,3)+[0;0;-rocket*dt*.25]+j2C;
    vD = placespeed(4:6,4)+[0;0;-rocket*dt*.25]+j2D;
    %though theoretically, only satellite A is accelerated, the other three
    %satellites are here given an acceleration as well so that the center
    %of mass remains in one place, for ease of comparison.

    %update angular velocity
    wA = placespeed(7:9,1);
    wB = placespeed(7:9,2);
    wC = placespeed(7:9,3);
    wD = placespeed(7:9,4);

    %update state matrix
    A(1:3,i+1) = pA;
    B(1:3,i+1) = pB;
    C(1:3,i+1) = pC;
    D(1:3,i+1) = pD;
    A(4:6,i+1) = bA;
    B(4:6,i+1) = bB;
    C(4:6,i+1) = bC;
    D(4:6,i+1) = bD;
    A(7:9,i+1) = vA;
    B(7:9,i+1) = vB;
    C(7:9,i+1) = vC;
    D(7:9,i+1) = vD;
    A(10:12,i+1) = wA;
    B(10:12,i+1) = wB;
    C(10:12,i+1) = wC;
    D(10:12,i+1) = wD;
end

if gPaperMode
    plotSat(A,B,C,D,dt,[1 1 1 0 0 1 1 1 1 1 0 0],strength);
else
    plotSat(A,B,C,D,dt,[1 1 1 1 1 1 1 1 1 1 1 0],strength);

    %draw in upper and lower bounds on distance plots
    figure(9)
    subplot(2,2,1)
    hold on
    plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
    hold off
    subplot(2,2,2)
    hold on
    plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
    hold off
    subplot(2,2,3)
    hold on
    plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
    hold off
    subplot(2,2,4)
    hold on
    plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
    hold off
end %gPaperMode

%plot motion for approximately one loop
slices = [(iter-iter/10);iter];
slice(A,B,C,D,slices,1,dt)

end %function

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function R = lineup(p1,p2)
    r = p1-p2;
    R = r/norm(r);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function whether = togethering(p1,p2,v1,v2)
    r12 = p1-p2;
    r21 = p2-p1;
    v21 = v1-v2;
    v12 = v2-v1;
    whether = (dot(r12,v12)>0)|(dot(r21,v21)>0);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function result = ordinal(vector,number)
    list = sort(vector,'descend');
    result = list(number);
end
```

C.5.1 slice.m

Output file supplement to show the relative positions of the satellites at a specified time.

```
function slice(A,B,C,D,slices,planes,dt)
%provides graphs of the positions of the four satellites at given times on
%a single plane.  Documentation will be completed by the final revision

global gPaperMode
if isempty(gPaperMode)
    gPaperMode = 0;
end
if ¬gPaperMode
    tsize = 10; %text size for graphs:
else
    tsize = 20; %larger size if it's for the paper
end

switch planes
    case 1    %xy
        alfa = 1;
        beta = 2;
        gama = 7;
        dlta = 8;
        labelx = 'x (m)';
        labely = 'y (m)';
    case 2    %yz
        alfa = 2;
        beta = 3;
        gama = 8;
        dlta = 9;
        labelx = 'y (m)';
        labely = 'z (m)';
    otherwise %xz
        alfa = 1;
        beta = 3;
        gama = 7;
        dlta = 9;
        labelx = 'x (m)';
        labely = 'z (m)';
end

for i = 1:size(slices,2)
    figure(i+14);clf
    if slices(2,i) == 0
```

```
        slice = slices(1,i);
        hold on
        quiver(A(alfa,slice),A(beta,slice),A(gama,slice),A(dlta,slice),'r')
        quiver(B(alfa,slice),B(beta,slice),B(gama,slice),B(dlta,slice),'b')
        quiver(C(alfa,slice),C(beta,slice),C(gama,slice),C(dlta,slice),'g')
        quiver(D(alfa,slice),D(beta,slice),D(gama,slice),D(dlta,slice),'k')
        text(A(alfa,slice),A(beta,slice),'A','FontSize',tsize)
        text(B(alfa,slice),B(beta,slice),'B','FontSize',tsize)
        text(C(alfa,slice),C(beta,slice),'C','FontSize',tsize)
        text(D(alfa,slice),D(beta,slice),'D','FontSize',tsize)
        hold off
        xlabel(labelx,'FontSize',tsize)
        ylabel(labely,'FontSize',tsize)
        title(['t = ' num2str(slice*dt) ' s'],'FontSize',tsize)
        axis equal
        set(gca,'FontSize',tsize)
    else
        slice = slices(1,i):slices(2,i);
        plot(A(alfa,slice),A(beta,slice),'r',B(alfa,slice),...
            B(beta,slice),'b',C(alfa,slice),C(beta,slice),'g',...
            D(alfa,slice),D(beta,slice),'k')
        title(['t = ' num2str(slice(1)*dt) ' - ' num2str(slice(end)*dt) ...
            ' s'],'FontSize',tsize)
        slice = slices(2,i);
        text(A(alfa,slice),A(beta,slice),'A','FontSize',tsize)
        text(B(alfa,slice),B(beta,slice),'B','FontSize',tsize)
        text(C(alfa,slice),C(beta,slice),'C','FontSize',tsize)
        text(D(alfa,slice),D(beta,slice),'D','FontSize',tsize)
        xlabel(labelx,'FontSize',tsize)
        ylabel(labely,'FontSize',tsize)
        axis equal
        set(gca,'FontSize',tsize)
    end
end
```

C.6 tetrasats.m

Setup and control file for satellites in a tetrahedral formation.

```
function tetrasats()
%tetrasats()
%  Runs simulation of EMFF satellites starting in a tetrahedral formation
%  See tstepSat.m for state calculation; see plotSat.m for output
%  This script contains the initial setup, the control laws, additional
%  thrust on satellite A, and the J2 perturbation on all satellites.
%
%  Figure 9 is modified showing activation distances, inner and outer, for
%  the dipoles.

global gEarthDipoleOn gReverseThrust gSideThrust gPaperMode
if isempty(gPaperMode)
```

```
    gPaperMode = 0;
end
if isempty(gReverseThrust)
    gReverseThrust = 0;
end
if isempty(gSideThrust)
    gSideThrust = 0;
end
gEarthDipoleOn = 1;

%Satellite characteristics
mA = 5;
mB = 5;
mC = 5;
mD = 5;

iA = [10;10;10;0;0;0];
iB = [10;10;10;0;0;0];
iC = [10;10;10;0;0;0];
iD = [10;10;10;0;0;0];

%parameters
iter = 8000;
dt = .5;
outon = 8.6;
inon = 8.3;

mTrgt = max([mA mB mC mD]);
dTrgt = outon;
strength = sqrt(8*dTrgt^4*mTrgt/3e-7);
innerfac = (outon/inon)^4;

rocket = -.001;

%initial setup
if gSideThrust
    pA = 3*[-1;-1; 1];
    pB = 3*[ 1; 1; 1];
else
    pA = 3*[ 1; 1; 1];
    pB = 3*[-1;-1; 1];
end
pC = 3*[-1; 1;-1];
pD = 3*[ 1;-1;-1];

if gReverseThrust
    pA = -1*pA;
    pB = -1*pB;
    pC = -1*pC;
    pD = -1*pD;
end

vA = [0;0;0];
vB = [0;0;0];
```

```
vC = [0;0;0];
vD = [0;0;0];

wA = [0;0;0];
wB = [0;0;0];
wC = [0;0;0];
wD = [0;0;0];

bA = [0;0;0];
bB = [0;0;0];
bC = [0;0;0];
bD = [0;0;0];

A = zeros(12,iter+1);
B = zeros(12,iter+1);
C = zeros(12,iter+1);
D = zeros(12,iter+1);

%actual program
A(:,1) = [pA;bA;vA;0;0;0];
B(:,1) = [pB;bB;vB;0;0;0];
C(:,1) = [pC;bC;vC;0;0;0];
D(:,1) = [pD;bD;vD;0;0;0];

for i=1:iter
    distances = [norm(pA-pB);
                 norm(pA-pC);
                 norm(pA-pD);
                 norm(pB-pC);
                 norm(pB-pD);
                 norm(pC-pD)];
    bA = [0;0;0];
    bB = [0;0;0];
    bC = [0;0;0];
    bD = [0;0;0];
    for j=6:-1:1%Collision avoidance phase
        if ordinal(distances,j)<inon%starting with the closest pair, check
            mdist = ordinal(distances,j);%to see if they are:
            switch mdist                 %1. too close, and
                case distances(1)        %2. moving toward each other
                    if togethering(pA,pB,vA,vB)
                        vrel = norm(vA-vB);
                        both = lineup(pA,pB)*(mdist/inon)^-3*sqrt(vrel);
                        bA = strength/innerfac*both; %if so, activate the
                        bB = -strength/innerfac*both;%dipoles opposite each
                    end                              %other, for repulsion.
                case distances(2)
                    if togethering(pA,pC,vA,vC)
                        vrel = norm(vA-vC);
                        both = lineup(pA,pC)*(mdist/inon)^-3*sqrt(vrel);
                        bA = strength/innerfac*both;
                        bC = -strength/innerfac*both;
                    end
                case distances(3)
                    if togethering(pA,pD,vA,vD)
                        vrel = norm(vA−vD);
                        both = lineup(pA,pD)∗(mdist/inon)ˆ−3∗sqrt(vrel);
                        bA = strength/innerfac∗both;
                        bD = −strength/innerfac∗both;
                    end
                case distances(4)
                    if togethering(pB,pC,vB,vC)
                        vrel = norm(vB−vC);
                        both = lineup(pB,pC)∗(mdist/inon)ˆ−3∗sqrt(vrel);
                        bB = strength/innerfac∗both;
                        bC = −strength/innerfac∗both;
                    end
                case distances(5)
                    if togethering(pB,pD,vB,vD)
                        vrel = norm(vB−vD);
                        both = lineup(pB,pD)∗(mdist/inon)ˆ−3∗sqrt(vrel);
                        bB = strength/innerfac∗both;
                        bD = −strength/innerfac∗both;
                    end
                otherwise
                    if togethering(pC,pD,vC,vD)
                        vrel = norm(vC−vD);
                        both = lineup(pC,pD)∗(mdist/inon)ˆ−3∗sqrt(vrel);
                        bC = strength/innerfac∗both;
                        bD = −strength/innerfac∗both;
                    end
            end
        end
        if any([bA;bB;bC;bD])
            break
        end
    end
    for j=1:6
        if (ordinal(distances,j)>outon) && (¬any([bA;bB;bC;bD]))
            mdist = ordinal(distances,j);%starting with the furthest pair,
            switch mdist                 %1. check if they're too far apart
                case distances(1)        %2. & moving away from each other
                    if ¬togethering(pA,pB,vA,vB)
                        vrel = norm(vA−vB);
                        both = lineup(pA,pB)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bB = strength∗both;
                    end
                case distances(2)
                    if ¬togethering(pA,pC,vA,vC)
                        vrel = norm(vA−vC);
                        both = lineup(pA,pC)∗(mdist/outon)ˆ4∗sqrt(vrel);
                        bA = strength∗both;
                        bC = strength∗both;
                    end
                case distances(3)
                    if ¬togethering(pA,pD,vA,vD)
                        vrel = norm(vA−vD);
                        both = lineup(pA,pD)*(mdist/outon)^4*sqrt(vrel);
                        bA = strength*both;
                        bD = strength*both;
                    end
                case distances(4)
                    if ¬togethering(pB,pC,vB,vC)
                        vrel = norm(vB−vC);
                        both = lineup(pB,pC)*(mdist/outon)^4*sqrt(vrel);
                        bB = strength*both;
                        bC = strength*both;
                    end
                case distances(5)
                    if ¬togethering(pB,pD,vB,vD)
                        vrel = norm(vB−vD);
                        both = lineup(pB,pD)*(mdist/outon)^4*sqrt(vrel);
                        bB = strength*both;
                        bD = strength*both;
                    end
                otherwise
                    if ¬togethering(pC,pD,vC,vD)
                        vrel = norm(vC−vD);
                        both = lineup(pC,pD)*(mdist/outon)^4*sqrt(vrel);
                        bC = strength*both;
                        bD = strength*both;
                    end
            end
        end
        if any([bA;bB;bC;bD])
            break
        end
    end

    placespeed = [pA pB pC pD;vA vB vC vD;wA wB wC wD];
    placespeed = tstepSat(placespeed,[mA mB mC mD],...
        [iA iB iC iD],[bA bB bC bD],dt);

    %update position
    pA = placespeed(1:3,1);
    pB = placespeed(1:3,2);
    pC = placespeed(1:3,3);
    pD = placespeed(1:3,4);

    %J2 perturbation
    cm = (pA+pB+pC+pD)/4;
    j2A = (pA−cm)*3.4213e−9*dt;
    j2B = (pB−cm)*3.4213e−9*dt;
    j2C = (pC−cm)*3.4213e−9*dt;
    j2D = (pD−cm)*3.4213e−9*dt;

    %update velocity, including J2 and rocket thrust
    vA = placespeed(4:6,1)+rocket*dt*.75/sqrt(3)*[1;1;1]+j2A;
    vB = placespeed(4:6,2)−rocket*dt*.25/sqrt(3)*[1;1;1]+j2B;
    vC = placespeed(4:6,3)−rocket*dt*.25/sqrt(3)*[1;1;1]+j2C;
    vD = placespeed(4:6,4)−rocket*dt*.25/sqrt(3)*[1;1;1]+j2D;
```

```
    %though theoretically, only satellite A is accelerated, the other three
    %satellites are here given an acceleration as well so that the center
    %of mass remains in one place, for ease of comparison.

    %update angular velocity
    wA = placespeed(7:9,1);
    wB = placespeed(7:9,2);
    wC = placespeed(7:9,3);
    wD = placespeed(7:9,4);

    %update state matrix
    A(1:3,i+1) = pA;
    B(1:3,i+1) = pB;
    C(1:3,i+1) = pC;
    D(1:3,i+1) = pD;
    A(4:6,i+1) = bA;
    B(4:6,i+1) = bB;
    C(4:6,i+1) = bC;
    D(4:6,i+1) = bD;
    A(7:9,i+1) = vA;
    B(7:9,i+1) = vB;
    C(7:9,i+1) = vC;
    D(7:9,i+1) = vD;
    A(10:12,i+1) = wA;
    B(10:12,i+1) = wB;
    C(10:12,i+1) = wC;
    D(10:12,i+1) = wD;
end

if gPaperMode
    plotSat(A,B,C,D,dt,[1 1 1 1 0 0 1 1 1 0 0 0],strength);
else
    plotSat(A,B,C,D,dt,[1 1 1 1 1 1 1 1 1 1 1 0],strength);
end

figure(4)
hold on
quiver3(5,5,5,-12,-12,-12,'--m')%clearly show thrust direction
hold off

%draw in upper and lower bounds on distance plots
figure(9)
subplot(2,2,1)
hold on
plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
hold off
subplot(2,2,2)
hold on
plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
hold off
subplot(2,2,3)
hold on
plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
hold off
```

```
subplot(2,2,4)
hold on
plot([0 dt*iter],[outon outon],':c',[0 dt*iter],[inon inon],':c')
hold off
end %function

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function R = lineup(p1,p2)
    r = p1-p2;
    R = r/norm(r);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function whether = togethering(p1,p2,v1,v2)
    r12 = p1-p2;
    r21 = p2-p1;
    v21 = v1-v2;
    v12 = v2-v1;
    whether = (dot(r12,v12)>0)|(dot(r21,v21)>0);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function result = ordinal(vector,number)
    list = sort(vector,'descend');
    result = list(number);
end
```

References

Elias, Laila Mireille. 2004. *Dynamics of Multi-Body Space Interferometers Including Reaction Wheel Gyroscopic Stiffening Effects: Structurally Connected and Electromagnetic Formation Flying Architectures.* M.S. Thesis, Massachusetts Institute of Technology.

Jaluria, Yogesh, & Torrance, Kenneth E. 2003. *Computational Heat Transfer*. 2 edn. Taylor & Francis.

Kurs, André, Karalis, Aristeidis, Moffatt, Robert, Joannopoulos, J. D., Fisher, Peter, & Soljačić, Marin. 2007. Wireless Power Transfer via Strongly Coupled Magnetic Resonances. *Science*, July.

Kwon, Daniel W. 2004. *Electromagnetic Formation Flight of Satellite Arrays.* M.S. Thesis, Massachusetts Institute of Technology.

Mark, James E. (ed). 1996. *Physical Properties of Polymers Handbook.* Woodbury, NY: AIP Press.

Pines, Darryl. 2007. *ENAE 602.* Class Notes.

Sakaguchi, Aya. 2007. *Micro-Electromagnetic Formation Flight of Satellite Systems.* M.S. Thesis, Massachusetts Institute of Technology.

Schweighart, Samuel A. 2005. *Electromagnetic Formation Flight Dipole Solution Planning.* Ph.D. thesis, Massachusetts Institute of Technology.

Sedwick, Raymond J., Miller, David W., Elias, Laila M., Schweighart, Samuel A., Neave, Matthew D., Kwon, Daniel, Ahsun, Umair, & Lee, Sang-il. 2005 (August). *Electromagnetic Formation Flight.* Tech. rept. Massachusetts Institute of Technology.

Sullivan, Charles R. 1999 (March). Optimal Choice for Number of Strands in a Litz-Wire Transformer Winding. *Pages 283–291 of: IEEE Transactions on Power Electronics*, vol. 14.

Weisstein, Eric W. 2008. *Hex Number.* From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/HexNumber.html.