Problem Set 3: Design Module for a Spacecraft Attitude Control System

Summary

The software module designed for this problem set calculated the disturbance torques on a satellite in a specified orbit, sized the required reaction wheels to counteract the disturbance torques, and sized the propulsion system required to dump angular momentum when the reaction wheels become saturated.

Results

The software module developed was tested with an orbit similar to the fictional FireSat satellite orbit from SMAD. The results from the software module were similar to those given in SMAD for the FireSat.

Useful References

Reaction Wheels

E. Ahronovich, M. Balling, *Reaction Wheel and Drive Electronics For LeoStar Class Space Vehicles,* 12th Annual USU Conference on Small Satellites, 1998, www.sdl.usu.edu/conferences/smallsat/proceedings/12/ssc98/1/ssci5.pdf
Dynacon Enterprises Limited, *Dynacon MicroWheel 200,* www.dynacon.ca/pdf/files/productpdf_6.pdf
Honeywell Aerospace Electronic Systems, *Constellation Series Reaction Wheels*, http://content.honeywell.com/dses/assets/datasheets/constellation_series_reaction_wheels.pdf.
Honeywell Aerospace Electronic Systems, *Miniature Reaction Wheels*, http://content.honeywell.com/dses/assets/datasheets/mini-wheel_reaction_wheel.pdf
Honeywell Aerospace Electronic Systems, *Honeywell Model HR 0610 Reaction Wheel*, http://content.honeywell.com/dses/assets/datasheets/hr0610_reaction_wheel.pdf
Teldix Space Product Group, *Momentum and Reaction Wheels 14-68 Nms with external Wheel Drive Electronics*, http://www.teldix.de/P22/RDR23-68.pdf
Teldix Space Product Group, *Momentum and Reaction Wheels 14-68 Nms with integrated Wheel Drive Electronics*, http://www.teldix.de/P22/RSI25-68.pdf
Teldix Space Product Group, *Momentum and Reaction Wheels 2-12 Nms with integrated Wheel Drive Electronics*, http://www.teldix.de/P22/RSI4-12.pdf
Teldix Space Product Group, *High motor torque Momentum and Reaction Wheels 14-68 Nms with integrated Wheel Drive Electronics*, http://www.teldix.de/P22/RSI18-68.pdf
Teldix Space Product Group, *Momentum and Reaction Wheels 0.04-0.12 Nms with integrated Wheel Drive Electronics*, http://www.teldix.de/P22/RSI01.pdf

Teldix Space Product Group, Momentum and Reaction Wheels 0.2-1.6 Nms with integrated Wheel Drive Electronics, http://www.teldix.de/P22/RSI02.pdf

The references listed above are for reaction wheels from Honeywell and Teldix. These two companies are some of the only companies that list their reaction wheel product specifications online. These reaction wheel specifications are useful for students that need real reaction wheel information to use in projects.

Atmospheric Model

Benson, Tom, http://www.grc.nasa.gov/WWW/K-12/airplane/atmosmet.html, *Earth Atmosphere Model, Metric Units*, NASA Glenn Research Center, 2002.

The website listed above is a good reference to an atmospheric model available online. This NASA developed atmospheric model is somewhat inaccurate, but it is a good starting point for a rough initial design.

ACS Equations

Wertz, James, and Larson, Wiley, *Space Mission Analysis and Design*, 2nd Ed., Microcosm, Inc., 1997.

SMAD, listed above, contains a great deal of ACS information for a first-order spacecraft design. The equations for determining disturbance torques as well as sizing reaction wheels and ACS propulsion systems are contained in SMAD as well.

Design Module for a Spacecraft Attitude Control System
Software Designed to Help Estimate Spacecraft Design Requirements

16.851 Satellite Engineering

Massachusetts Institute of Technology, Cambridge, MA
October 2003

Motivation

Spacecraft often have pointing requirements. Satellite antennas and optics will generally require that the spacecraft remain pointed at the desired target within a certain pointing tolerance. There are many environmental effects that disturb the spacecraft, and the attitude control systems (ACS) must be designed to account for this. The ACS system for this project will be designed for a three-axis stabilized spacecraft. This module will be useful for next semester when the x-ray telescope being designed will require three-axis stabilization.

Problem Statement

Design a tool that sizes attitude control system actuators for a three-axis stabilized spacecraft given disturbance torques created by various environmental effects. Environmental effects include gravity gradient, solar radiation, magnetic field, and aerodynamic forces. ACS actuators being investigated include momentum wheels and thrusters.

Introduction

MATLAB is used to evaluate the disturbance torques on the spacecraft and select and size appropriate ACS actuators to meet the pointing requirements of the spacecraft.

For spacecraft without large slewing requirements, momentum wheels are primarily used to counteract disturbance torques. Momentum wheels and the propulsion system to dump the excess momentum will be the primary focus of the software module.

The user will input information about the satellite including mass properties, physical dimensions, and orbit information (altitude, inclination, and eccentricity). The tool then examines the relevant environmental disturbance torques and finds the worst-case torque conditions for the specific spacecraft and orbit. Next, the tool will size the momentum wheels required to overcome worst-case disturbance torques. The masses of the momentum wheels are obtained from a database of available momentum wheel specifications. In addition, the module calculates the mass of the propulsion system required to dump momentum from the momentum wheels when they become saturated over the lifetime of the spacecraft.

Finally, the tool is executed for a test case in order to check the validity of the module. In addition, other test cases are run in order to construct parametric design figures that show trends of the output involving disturbance torques and ACS mass as a function of spacecraft size and orbit parameters.

Software Module

ACS Environmental Disturbance Torque Tool

Description of Code

This MATLAB software tool sizes ACS angular momentum storage and dumping devices for a specified vehicle in any orbit. The tool only addresses disturbance torques from the main environmental sources: atmospheric drag, solar radiation, magnetic field, and gravity gradient. The ACS sizing is only meant to account for disturbance torques and does not address the design needs for vehicle slewing. The tool requires input data structures to describe the vehicle, orbit, and planet (Table 1).

Table 1 Software tool inputs

Parameter	Description
`veh.dim`	[m], vehicle dimensions (x,y,z)
`veh.CG`	[m], vehicle CG offset from geometric center
`veh.mass`	[kg], vehicle mass
`veh.mat`	vehicle surface material code (see Appendix)
`veh.life`	[years], vehicle design life
`OE.a`	[m], orbit semi-major axis
`OE.e`	orbit eccentricity
`OE.i`	[deg], orbit inclination
`OE.Om`	[deg], argument of periapsis
`OE.om`	[deg], longitude of the ascending node
`planet.mu`	[m^3/s^2], earth gravity constant
`planet.r_pol`	[m], earth polar radius
`planet.r_equ`	[m], earth equatorial radius

The vehicle is modeled as a rectangular prism and described by the three edge lengths, mass, center of gravity, and exterior surface material. Using the dimensions and mass, the vehicle's moment of inertia can be determined using[1]:

$$I = \begin{bmatrix} \frac{1}{12}m(y^2+z^2) & 0 & 0 \\ 0 & \frac{1}{12}m(x^2+z^2) & 0 \\ 0 & 0 & \frac{1}{12}m(x^2+y^2) \end{bmatrix} \quad (1)$$

Using the orbital elements, the orbital period and inertial position and velocity are calculated[2]. Throughout the orbit, worst case environmental disturbance torques are calculated for the four environmental sources. Details of the torque calculations will be addressed in the following sections.

The total disturbance torque is then integrated for one complete orbit. This provides the worst case angular momentum imposed on the vehicle for one orbital period. Integrating the orbital torque instead of assuming an average from the maximum torque is advantageous because it incorporates the varying effects resulting from different orbit types. Two very different orbits might have the same maximum torque, but would in reality experience very different disturbance effects (i.e. circular orbit versus highly elliptic orbit).

For the angular momentum experienced by the vehicle, it is assumed that 80% is cyclic in nature and therefore only requires temporary storage during the orbit. The remaining 20% is secular and will continue to accumulate until the momentum dumping system applies an external torque and eliminates this momentum. This ratio was chosen because cyclic momentum loads are known to drive the ACS design and this ratio matched well with guidance provided in SMAD.

The cyclical angular momentum drives the momentum wheel design. The tool selects a momentum wheel system that can store the calculated cyclical momentum and then estimates the mass of the wheel.

Next, the tool selects the thruster system to dump the accumulating secular torque for the duration of the vehicle life. This determines the thruster system mass and thrust capability.

Finally, the tool outputs these calculated ACS specs for the worst case environmental disturbance torques. Table 2 shows the software tool output.

Table 2 Software tool output

Parameter	Units
Cyclical angular momentum (per orbit)	[Nms]
Secular angular momentum (per orbit)	[Nms]
ACS Momentum wheel mass	[kg]
ACS Thruster system mass	[kg]
ACS Thruster thrust	[N]

Program Execution

The ACS ENV tool is executed using MATLAB. First, various constant parameters are specified by the user within the "`ACS_env_ini.m`" file (see Table 1). Next, the tool is executed by entering at the Matlab command prompt:

```
>> [h_cyc,h_sec,wheel_mass,thr_mass,thr_force]
= ACS_env(OE,veh,planet);
```

The five program outputs are specified in Table 2. Copies of the initialization file and all program modules can be found in the appendix.

Aerodynamic Torque Module

Requirements

This MATLAB module, *torque_aero.m*, calculates the torque due to aerodynamic drag on the spacecraft. This drag is caused by the vehicle flying through the Earth's atmosphere during close approaches to the planet. An atmospheric model is used to approximate the density of the atmosphere at each position of the spacecraft. Combined with geometric and aerodynamic information, the torque caused by drag is determined.

Description of Code
The code uses an atmospheric model for the "upper atmosphere" from NASA. This model provides an approximation of the density of Earth's atmosphere for use in drag calculations on the spacecraft.[3]

The code also assumes the spacecraft is shaped as a rectangular parallelepiped of edge dimensions X, Y, and Z. The origin of the spacecraft is defined to be the geometric center of the 3-D shape. The coordinate system definition for the assumed spacecraft in the MATLAB module is shown below in Figure 1.

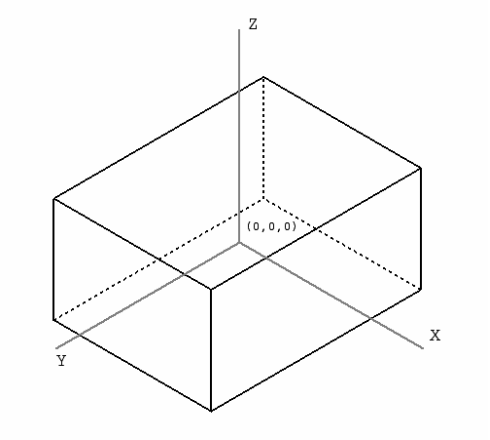

Figure 1 S/C coordinate system definition

In addition, the center of pressure, the location at which the aerodynamic drag is acting, is assumed to be at the center of one face of the spacecraft. It is assumed that the worst case drag the spacecraft experiences is the case in which the vehicle has one face directly facing the direction of travel though the atmosphere. In addition, it is assumed that the face on which the drag is acting has the largest surface area on the spacecraft. This results in the worst-case drag possible on the vehicle.

The worst-case of the aerodynamic drag torque also results from the worst-case moment arm of the force acting on the vehicle. This worst-case moment arm is determined from the absolute value of the maximum CG offset from the geometric center. The geometric center is assumed to be the coordinate system origin for the spacecraft. Since the center of pressure acting on each face is at the center of the face, the largest moment arm acting around the center of gravity is the maximum distance in one axis of the center of gravity from the geometric center.

Constants
The only constant used in this module was the coefficient of drag, C_D, which is assumed to be 2.2. This assumption was made due to a lack of experimental data about the drag and lift characteristics of this vehicle. Also, Chobotov recommends that a drag coefficient value of 2.2 be used as a conservative estimate.[4]

Inputs
size (m): This input was a vector containing the three edge lengths of the rectangular parallelepiped-shaped spacecraft.

V (m/s): This input is the instantaneous velocity of the spacecraft at any given time when the module is called by the main program.

h (m): This input is the altitude of the spacecraft at any time in which the module is called by the main program.

CG (m): This input is a vector of the center-of-gravity offset from the geometric center of the spacecraft. This is used to determine the worst-case moment arm on which the drag force is acting.

Outputs
Ta (N*m): This output is the disturbance torque on the spacecraft due to atmospheric drag in the spacecraft. This is sent back to the main program when this subroutine is called.

Theory & Equations
In order to estimate the atmospheric density at any given time during the orbit of the spacecraft, a density model provided by NASA was used. The model used was for the "upper atmosphere." This meant that it was acceptable for use above 25km altitude. The equations below were used for this approximation.[5]

$$T = -131.21 + .00299h \tag{2a}$$

$$p = 2.488\left[\frac{T + 273.1}{216.6}\right]^{-11.388} \tag{2b}$$

$$\rho = \frac{p}{.2869\left[T + 273.1\right]} \tag{2c}$$

In the above equations, T is the atmospheric temperature (degrees Celsius), p is the atmospheric pressure (KPa), and ρ is the atmospheric density (kg/m^3).

The force acting on the spacecraft due to drag was determined by the following equation.[6]

$$F = \tfrac{1}{2}\rho C_D A V^2 \tag{3}$$

The *A* in the above equation is the surface area of the face of the spacecraft normal to the airflow hitting the vehicle.

The torque due to aerodynamic drag on the spacecraft is calculated using the following equation.

$$T_a = F(\max(D_i)),\ where\ i = x, y, z \quad (4)$$

The torque, T_a, is calculated by multiplying the drag force on the spacecraft, F, by the maximum distance of the center of gravity from the geometric center out of the x, y, or z directions, D_i. This would allow for the largest moment arm and therefore would result in the highest disturbance torque due to aerodynamic drag.

Torque from Solar Radiation

Solar radiation pressure if found from the equation:

$$T_{sp} = \frac{F_s}{c} A_s (1+q)\cos i(cps - cg) \quad (5)$$

where *Fs* is the solar constant, 1367 W/m2, *c* is the speed of light, $3x10^8$ m/s, *As* is the surface area of the satellite facing the sun, *q* is the reflectance factor, *i* is the angle of incidence to the sun, *cps* is the location of the center of solar pressure, and *cg* is the center of gravity of the satellite. The units of solar radiation pressure are Newton-meters.

The worst case is for an incidence angle of zero degrees, so i=0 was always used in calculations. Also in a worst case situation, *As* would be the surface area of the largest face of the satellite. The value for q is based on the material of the satellite, which is specified on input.

Gravity-gradient Disturbance Torques

The gravity field of the Earth can place unwanted torques on a satellite. When a satellite is affected by the Earth's gravity field, the longitudinal axis becomes aligned toward the center of the Earth. The strength of the torque is a function of the distance of the satellite from the Earth. Also, the torque is symmetric about the nadir vector of the satellite.

The equation for finding the worst case gravity-gradient torque is:

$$T_g = \frac{3\mu}{2R^3}\left|I_z - I_y\right|\cos(2\theta) \quad (6)$$

where μ is the gravitational constant of the Earth, $3.986*10^{14}$ m^3/s^2, R is the orbit radius, I_z and I_y are the moments of inertial about the z and y axes of the satellite, and θ is the maximum deviation of the z-axis from the local vertical. To calculate the worst case scenario, θ is assumed to be 45^o.

Magnetic Field Disturbance Torques

The magnetic field of the earth can cause torques on a satellite. The strength of the magnetic field torque on a satellite is a function of the satellite's position. The magnetic field intensity B is found by the equation:

$$B = \frac{B_o}{R^3}\sqrt{1+\sin^2\lambda} \quad (7)$$

where B_0 is the magnetic field at the equator at the Earth's surface, 0.3 gauss, R is the radial distance from the Earth, and λ is the magnetic latitude. The worse case disturbance torque from the magnetic field on the satellite can then be found by multiplying B by the residual dipole of the satellite, D:

$$T_m = DB \quad (8)$$

The vehicle residual magnetic dipole (D) is assumed to be 1 Am^2 for this problem.

Mass in Reaction Wheels

Using the cyclic angular momentum calculated by the tool, the reaction wheel mass can be estimated. To determine mass of a reaction wheel, data was collected on many existing, commercial reaction wheels. A fourth-order polynomial curve was fit to this data using MATLAB, to compare momentum storage in reaction wheels and their weight in kilograms. From the curve, the approximate mass of a momentum wheel given its momentum storage can be determined. The figure below illustrates the data collected (red points) as well as the polynomial curve fit used (shown as the blue line).

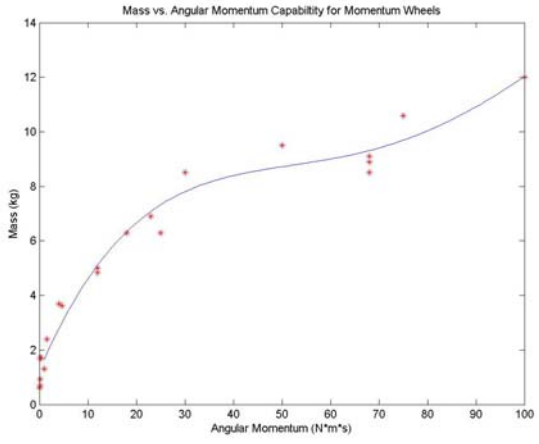

Figure 2 Mass vs. angular momentum capability for momentum wheels

Sources used for collecting data on reaction wheels are included in the references section of this paper.[7,8,9,10,11,12,13,14,15,16,17]

ACS Propulsion System Design Module

Requirements

This MATLAB module, called *prop_system.m*, calculated the thrust required to dump momentum at each momentum dump thruster firing as well as the total ACS propulsion system mass required for momentum dumping.

Description of Code

This code calculates the worst-case moment arm of an ACS thruster about a spin axis of the spacecraft and combines that moment arm with the required thrust for dumping momentum to size the propulsion subsystem.

The code assumes the spacecraft is shaped as a rectangular parallelepiped of edge dimensions X, Y, and Z, with thrusters arranged at locations shown in the figure below. The thrusters are the red triangles in the figure below.

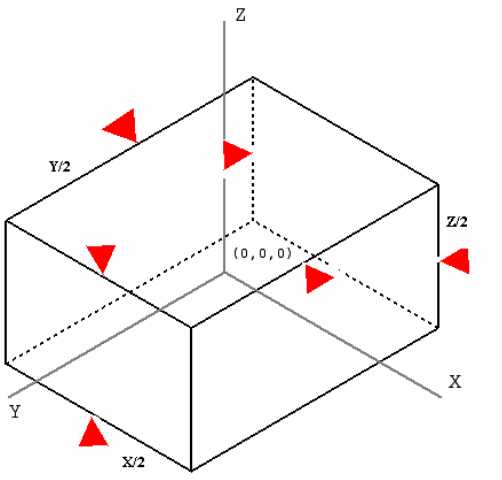

Figure 3 Assumed thruster locations on spacecraft

A Hydrazine monopropellant propulsion system is assumed to be used for momentum dumping for this spacecraft. Although the specific impulse, Isp, for Hydrazine can be as high as 245 seconds,[18] a conservative value for the Isp for Hydrazine is assumed in this module. This conservative Isp value is 200 seconds.

Although the thrusters and propulsion system plumbing are not directly sized, a conservative value for propulsion system mass is estimated. It is assumed that the mass of the propulsion system is 85% propellant.[19] This, by sizing the propellant, the total system mass can be determined by dividing by 0.85.

It should also be noted that it is assumed in this module that the time for each thruster firing is one second.

Constants

The only constant used in this module is the acceleration due to gravity at the Earth's surface, *g*. This is assumed to be 9.8 m/s^2.

Inputs

size (m): This input was a vector containing the three edge lengths of the rectangular parallelepiped-shaped spacecraft.

CG (m): This input is a vector of the center-of-gravity offset from the geometric center of the spacecraft. This is used to determine the worst-case moment arm on which the drag force is acting.

lifetime (years): This input is the lifetime of the satellite. It is used to determine how many thruster firings to dump momentum will be needed throughout the life of the spacecraft.

H (N*m*s): This input is the stored maximum momentum in any one momentum wheel. It is the momentum which will need to be dumped by using the thruster firing of the ACS system.

sat_rate (days/saturation): This input is the time it takes for the wheels of the spacecraft to become saturated with angular momentum. It is at this point that the momentum wheels have no more capacity to control the attitude of the spacecraft by spinning up any faster.

Outputs

F (N): This output is the force required for the thruster to impart on the spacecraft in order to dump the required momentum.

p_mass (kg): This output is the total mass of the propulsion system used to dump momentum when the momentum wheels become saturated throughout the lifetime of the spacecraft.

Theory & Equations

The force required to dump momentum is obtained by using the following equation.

$$F = \frac{H}{Lt} \tag{9}$$

In the above equations, L is the moment arm of the thruster to the required spin axis (meters) and t is the thruster firing time.

The propellant mass, m_P, and total propulsion system mass, p_mass, are then calculated using the following equations.

$$total\_pulses = \frac{3 * lifetime * 365.25}{sat\_rate} \quad (10)$$

$$m_P = \frac{F(total\_pulses)t}{I_{sp}g} \quad (11)$$

$$p\_mass = \frac{m_P}{0.85} \quad (12)$$

Total_pulses is the total number of thruster firings required throughout the lifetime of the spacecraft to counteract the environmental disturbance torques. The numerator of the *total_pulses* equation has a factor of 3 in it because it is assumed that all three wheels will need to be desaturated at each time momentum is required to be dumped. The I_{sp} in the equation above is for Hydrazine. The 0.85 factor is explained in the "Description of Code" section.

Results

Module test case

In order to verify that the MATLAB code is working properly, the module was used to determine the disturbance torques and calculate the required ACS mass for a test case. The test case used was the main spacecraft example in *Space Mission Analysis and Design*, by Wertz and Larson. This example is the FireSat satellite. The main parameters used to simulate the FireSat example to test this module with are shown below in Table 3.

Table 3 FireSat-like test case parameters

Parameter	Value	Description
`veh.dim`	[1.7 1 1.7]	[m], vehicle dimensions (x,y,z)
`veh.CG`	[0.2 0 0]	[m], vehicle CG offset from geometric center
`veh.mass`	200	[kg], vehicle mass
`veh.mat`	9	vehicle surface material code (corresponds to 0.63 reflectance)
`veh.life`	4	[years], vehicle design life
`OE.a`	7,078,000	[m], orbit semi-major axis
`OE.e`	0.0	orbit eccentricity
`OE.i`	45	[deg], orbit inclination
`OE.Om`	0	[deg], argument of periapsis
`OE.om`	0	[deg], longitude of the ascending node
`planet.mu`	3.986e14	[m^3/s^2], earth gravity constant
`planet.r_pol`	6,357,000	[m], earth polar radius
`planet.r_equ`	6,378,000	[m], earth equatorial radius

Based on the input parameters in the above table, the MATLAB module calculated the cyclical and secular angular momentum required to counteract disturbance torques on the spacecraft for one orbit, the required momentum wheel mass for the ACS system, the thrust required for each instance angular momentum of the spacecraft needs to be dumped, and the ACS propulsion system mass required for dumping angular momentum for the life of the spacecraft.

A plot of the orbit of the FireSat spacecraft example is shown below in Figure 4.

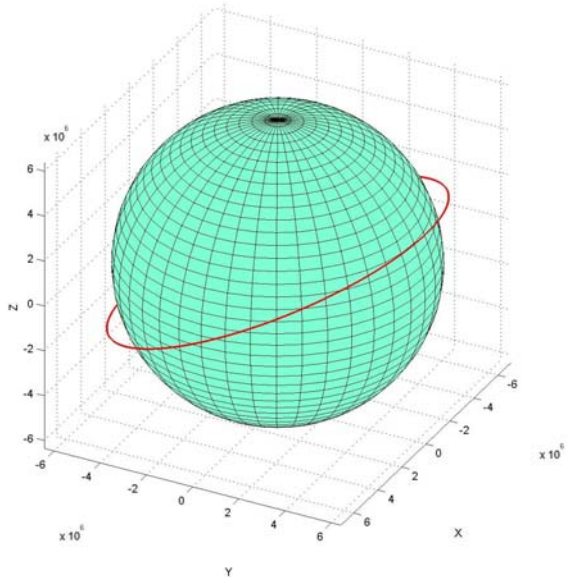

Figure 4 FireSat orbit

The disturbance torques imposed on the spacecraft are shown in the polar plot in following figure. Zero degrees on the plot corresponds to the orbital ascending node.

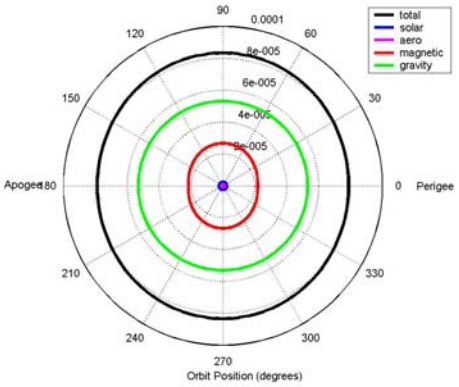

Figure 5 Disturbance Torques for FireSat Test Case

The curves in the plot in Figure 5 show the disturbance torque due to gravity gradient is the largest contributor to the overall disturbance torque on the spacecraft. The torque due to magnetic fields is the second-largest contributor to the total disturbance torque. Torque due to solar radiation and aerodynamic drag are the two minor contributors to overall disturbance torque. Solar pressure on the spacecraft is a weak force, which results in the torque from solar radiation being small. The torque due to drag is low because the spacecraft is in an 700km orbit. The Earth's atmosphere at an altitude of 700km is almost nonexistent.

A comparison of the results from SMAD and results from the MATLAB code is shown below in Table 4.

Table 4 FireSat-like test case solution vs. Module

Parameter	Value	SMAD Solution
Cyclical angular momentum (per orbit)	0.3845 [Nms]	0.4 [Nms]
Secular angular momentum (per orbit)	0.0961 [Nms]	N/A
ACS Momentum wheel mass	1.41 [kg]	N/A
ACS Thruster system mass	2.46 [kg]	2.43 [kg]

It can be seen in the above table that the values for cyclical angular momentum and ACS thruster system mass are nearly identical. The values for secular angular momentum and ACS momentum wheel mass are not included in SMAD and therefore can not be compared to the module solution.

Module output

The following section will display the capabilities of the MATLAB module by presenting data collected by running the module for various orbits and spacecraft sizes.

First, required angular momentum to counteract cyclic disturbance torques on the spacecraft is calculated for various orbit eccentricities and altitudes at perigee. The results are shown below in Figure 6.

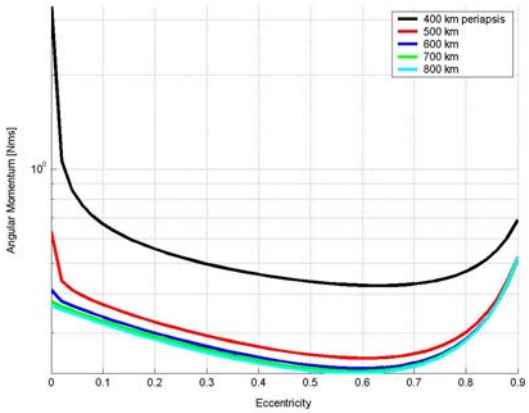

Figure 6 Cyclic angular momentum vs. eccentricity for various radii of perigee

The curves in the figure show several interesting trends which correspond with reality. First, the highly-eccentric orbits generally have lower required angular momentum storage capability. This is due to the fact that these orbits are far away from the Earth for most of the orbit. This nearly eliminates disturbance torques from drag, magnetic fields, and gravity gradient. The remaining torque, caused from solar radiation, is a small contributor to disturbance torque.

The reason the angular momentum increases slightly near the high end of eccentricity is most likely due to the fact that the satellite in those orbits will be passing by Earth at extremely high speeds compared to the smaller-eccentricity orbits. This high speed near Earth may contribute to much larger torque due to drag for that portion of the orbit.

Another interesting observation from the figure above is that the disturbance torque for satellites around 400 or 500km altitude is much greater than that of satellites in slightly higher low-Earth orbits of 600km. This shows a potential significant cost savings for an ACS system if a LEO satellite were placed in a 600km orbit as opposed to a 400km orbit, for example.

Figure 7 below graphs ACS mass versus orbit eccentricity for various radii of perigee.

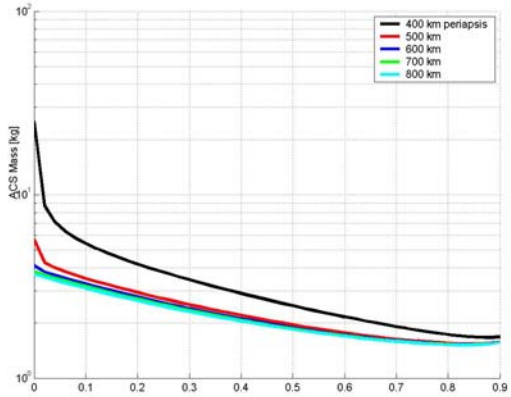

Figure 7 ACS mass vs. eccentricity for various radii of perigee

Figure 7 is interesting because it shows that the ACS mass for highly-eccentric orbits does not follow the exact same trend as that shown in the previous figure. The reason the ACS mass is reduced for the highly-eccentric orbits and does not continue with the same trend as the angular momentum is due to the fact that the highly-eccentric orbits have large orbit periods. Since the lifetime of the spacecraft is kept constant for these trend studies, a longer orbit period would result in fewer ACS angular momentum dumping situations during the lifetime of the satellite. It would take more time for a longer orbit for the attitude control system to become saturated with angular momentum and then require thrust to dump the momentum. This reduces the mass of the ACS propulsion system and produces the trends seen in the above figure.

The figure below plots angular momentum versus altitude for various inclinations for circular orbits.

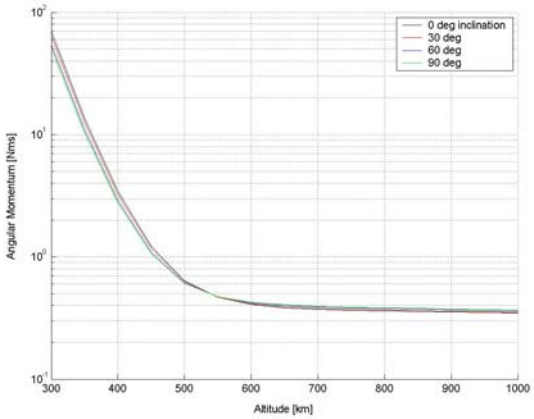

Figure 8 Angular momentum vs. altitude for various inclinations of circular orbits

Figure 8 shows a clear trend that for all circular orbit inclinations, there is an altitude at which the required angular momentum storage of the ACS system has reduced significantly and levels-off. This altitude is around 525km. This means there could be a significant benefit to putting LEO spacecraft in circular orbits above 525km in altitude in order to minimize the angular momentum capacity required for the ACS system.

In addition, there is a difference in angular momentum between various inclinations. It is especially noticeable at lower altitudes. For example, at 400km altitude, the angular momentum required for the 0 degree inclination (equatorial) orbit is approximately 20 N*m*s greater than that of the 90 degree inclination (polar) orbit. This trend also reverses itself after the 525km altitude mark and the polar orbit becomes the maximum angular momentum case.

The figure below shows the ACS mass versus orbit altitude for circular orbits of various inclinations.

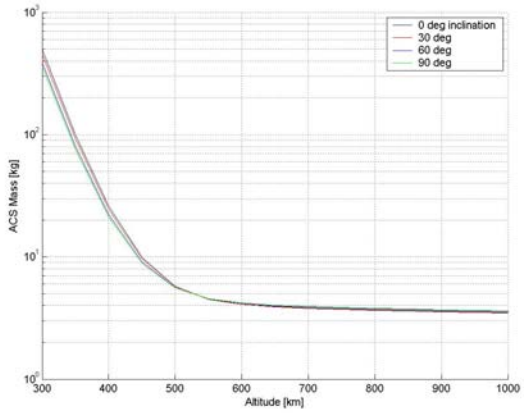

Figure 9 ACS mass vs. altitude for various inclinations of circular orbits

Figure 9 clearly exhibits the same trend of that shown in Figure 8. This shows that the ACS mass is directly related to the required angular momentum storage of the spacecraft for circular orbits.

The figure below shows the angular momentum storage required versus vehicle size for three different types of orbits. The vehicle density was held constant while volume and the corresponding mass were varied. The three investigated orbits are the Molniya orbit, a circular LEO orbit of 400km, and a GEO orbit.

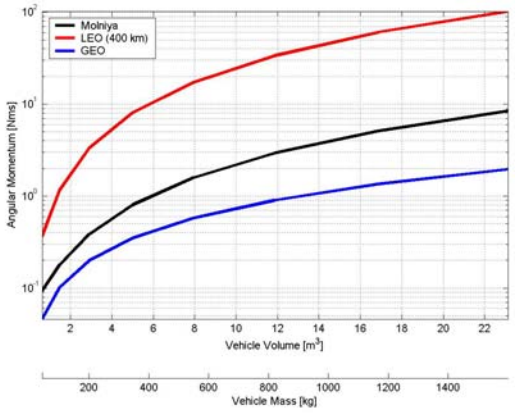

Figure 10 Angular momentum vs. vehicle volume for three orbit types

Although the vertical axis in the above figure is in a logarithmic scale, it can be seen that the angular momentum required for a 400km LEO orbit is much greater than that for a Molniya or a GEO orbit. The main reason for this is the fact that satellites in Molniya or GEO orbits spend all or most of an orbit period far from Earth. This means the spacecraft in those orbits will not experience much aerodynamic drag, will experience a reduced torque due to gravity gradient, and may experience less torque due to the Earth's magnetic fields.

Figure 10 also clearly shows that as the spacecraft grows in size, the angular momentum requirement increases as well. This is due to increased aerodynamic drag.

Figure 11 below shows the ACS mass versus vehicle volume for the same orbits as in the previous figure.

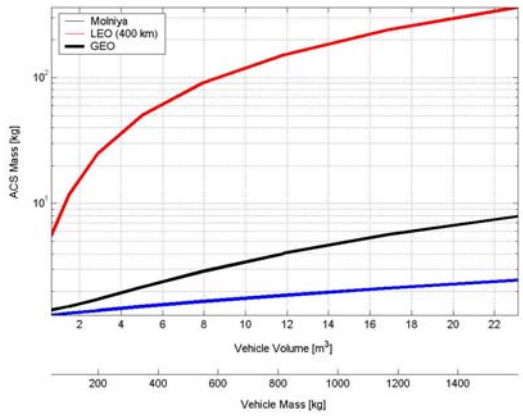

Figure 11 ACS mass vs. vehicle volume for three orbit types

Figure 11 above shows the trend that the ACS mass increases roughly at the same rate as the angular momentum requirement for orbits of the same type as vehicle volume is increased.

Conclusion

A MATLAB module was created which took inputs of orbital parameters, spacecraft dimensions, and the spacecraft environment and output cyclical angular momentum, secular angular momentum, ACS momentum wheel mass, and ACS thruster system mass. The module was checked by running the FireSat example. In addition, other cases were checked with the module and realistic data was output from the module.

The values of angular momentum and ACS mass are of use for preliminary design of an attitude control system. The designer can input preliminary information about a spacecraft design and get rough numbers for preliminary sizing of the ACS system and its impact on the spacecraft mass budget.

Future Work

The module can be expanded to include reaction wheels, control-moment gyros, and magnetic torquers.

In addition, the database of momentum wheels can be expanded to include more wheels and include other ACS actuator specifications as well.

Another area for future work is to validate the module by running a real life example of a satellite and comparing the results to experimental data.

Appendix A

FireSat-like Test Case Input

```
%################################
%## ACS Env
%##
%##   Initialization file
%##


  %# Vehicle Properties #%
    veh.dim = [1.7 1 1.7]; %[m], Length of edges on rectangular-prism shaped vehicle (length,
width, depth)
    veh.CG = [0.2 0 0]; %[m], center of gravity offset from geometric center
    veh.mass = 200; %[kg] (minus ACS system)
    veh.mat = 9; %Surface material code
    veh.life = 4; %[yrs], Vehicle lifespan


  %# Orbital Elements #%
    OE.a = 7078000; %[m], semi-major axis
    OE.e = 0.0; % eccentricity
    OE.i = 45 *pi/180; %[rad], inclination
    OE.Om = 0 *pi/180; %[rad], argument of periapsis (angle from ascending node to periapsis)
    OE.om = 0; % [rad],longitude of the ascending node (angle between x and asc. node)


  %# Planet Properties #%
    planet.mu = 3.986e14; %[m^3/s^2], Earth gravity constant
    planet.r_pol = 6357000; %[m], Polar radius
    planet.r_equ = 6378000; %[m], Equitorial radius
```

ACS Env Main Program

```
    % ## ACS sizing for Environmental Disturbance torques ##
    %
    %   Input:
    %     OE - Orbital elements for vehicle orbit
    %     veh - vehicle parameters
    %     planet - planet parameters
    %
    %   Output:
    %     ang_mom_cyc - cyclical ang momentum storage requirement [Nms]
    %     ang_mom_sec - secular ang momentum received per orbit [Nms]
    %     w_mass - spec for momentum wheel mass [kg]
    %     t_mass - spec for thruster mass [kg]
    %     thrust - spec for thruster thrust [N]
    %

    function [ang_mom_cyc, ang_mom_sec, w_mass, t_mass, thrust] = ACS_env(OE,veh,planet);

    %## Calculate vehicle moment of inertia ##%

      Ixx = veh.mass*(veh.dim(1)^2 + veh.dim(2)^2)/12; %[kg*m^2]
      Iyy = veh.mass*(veh.dim(1)^2 + veh.dim(3)^2)/12; %[kg*m^2]
      Izz = veh.mass*(veh.dim(2)^2 + veh.dim(3)^2)/12; %[kg*m^2]
      I = diag([Ixx Iyy Izz])


    %# Calculate time for one orbit
      t = 2*pi*sqrt(OE.a^3/planet.mu); %[sec]


    %# Calculate Solar radiation torques
      Ts = torque_solar(veh.dim, veh.CG, 0, veh.mat);


    ang_step = pi/50; %[rad]
    ang_range = [0:ang_step:2*pi];
    ii = 1;


    %## Calculate max disturbance torque around one complete orbit ##%
    for ang = ang_range


         %# Calculate orbital postion and velocity
          [r,v] = oe2rv([OE.a OE.e OE.i OE.Om OE.om ang], planet.mu);
          v_mag = norm(v); %[m/s], Calculate velocity magnitude
          R(ii,:) = r; %[m]
            V(ii,:) = v; %[m/s]
          V_MAG(ii,:) = v_mag;

          E = acos((OE.e + cos(ang))/(1+OE.e*cos(ang))); %Eccentric anomoly
          time(ii,:) = sqrt(OE.a^3/planet.mu)*(E-OE.e*sin(E)); %time to 'ang'

         %# Post-process orbit elevation (latitude)
          aa = sqrt(r(1)^2 + r(2)^2);
          lat = atan2(r(3), aa); %[rad]
          LAT(ii,:) = lat;



         %# Post-process altitude
          r_planet = (planet.r_pol*planet.r_equ)/...
                        sqrt(planet.r_pol^2*cos(lat)^2 + planet.r_equ^2*sin(lat)^2); %[m],
Calculate planet radius
                                                                                    assuming
                                                                                oblate
                                                                                sphereoid

           alt = norm(r)-r_planet; %[m], Subtract planet radius from vehicle position vector
magnitude
          ALT(ii,:) = alt;


         %# Calculate Aerodynamic torques
```

```
      Ta = torque_aero(veh.dim, v_mag, alt, veh.CG);
      TA(ii,:) = Ta;


     %# Calculate Magnetic torques
       Tm = torque_magnetic(lat, norm(r), r_planet);
      TM(ii,:) = Tm;


     %# Calculate Gravity torques
       Tg = torque_gravity(norm(r), planet.mu, I);
      TG(ii,:) = Tg;

      TS(ii,:) = Ts;


     %# Sum all disturbance torques
      T(ii,:) = Ts + Ta + Tm + Tg; %[Nm]


      ii=ii+1; %Increment counter

end


%## post-process time values ##%
  max_t = ceil(length(time)/2);
  for jj = [max_t+1: length(time)];
    time(jj) = 2*time(max_t)-time(jj);
  end


%## Integrate max torques around orbit to find total ang mom ##%
  ang_mom = trapz(time,T); %[Nms], total angular momentum around one complete orbit

  ang_mom_cyc = 0.8 * ang_mom; %[Nms], cyclical angular momentum per orbit
  ang_mom_sec = ang_mom - ang_mom_cyc; %[Nms], Secular angular momentum per orbit


%## Size ACS actuators for cyclical momentum storage##%
  wheel_data = get_wheel_data; %Loads Reaction wheel data (mass vs Nms)
  w_mass = polyval(wheel_data, ang_mom_cyc);


%## Size ACS thrusters for secular momentum dumping ##%
   orb_sat = 1; %[orbits/saturation]
   day_sat = orb_sat*t/86400; %[day/saturation]
   [thrust, t_mass] = prop_system(veh.dim, veh.CG, veh.life, ang_mom_sec, day_sat);


%## Plot results ##%
    plot_planet_3D(R,planet);

    figure(23);
    polar(ang_range', T, 'k')
    hold on
    polar(ang_range', TS, 'b');
    polar(ang_range', TA, 'm');
    polar(ang_range', TM, 'r');
    polar(ang_range', TG, 'g');
    hold off
    legend('total', 'solar', 'aero', 'magnetic', 'gravity');
   xlabel('[deg]')
```

torque_aero.m

```
% Bill Nadir
% 16.851 Satellite Engineering
% 10/11/2003


% Module for calculating external spacecraft torque caused by Aerodynamic forces

function Ta = torque_aero(size,V,h,CG)

% Here the force on the S/C, F, is calculated

% INPUTS
% size  = edge lengths of the S/C: vector (x,y,z) side lengths (meters)
% V     = S/C velocity (m/s)
% h     = S/C altitude (m)
% CG    = location of the center of gravity (x,y,z) for the S/C (assumed offset from the
%         geometric center of (0,0,0)) (m)

% OUTPUT
% Ta    = Torque on S/C due to aerodynamic drag (Nm)

% rho is the atmospheric density at the location of the S/C
% An atmospheric model for the upper atmosphere (h>25000m) is used to
% approximate the density of the upper atmosphere
% T is the atmospheric temperature, p is atmospheric pressure

T   = -131.21 + .00299*h; % in deg C
p   = 2.488*(((T + 273.1)/216.6)^-11.388); % pressure in KPa
rho = p / (.2869*(T + 273.1)); % in kg/m^3

% C_D is the drag coefficient of the cube-shaped S/C (assumed = 2.2)

C_D = 2.2;

% Cpa   = location of the center of aerodynamic pressure (x,y,z)
%         (assumed at the center of the face of one side of the cube which is
%         facing directly into the atmosphere = max drag)

% Here the surface areas of the sides of the S/C are determined
% This is used to assume the worst-case drag on the vehicle
% [x*z y*z x*y] => find max

area = [size(1)*size(3) size(2)*size(3) size(1)*size(2)];
max_area = max(area);
F = 0.5*rho*C_D*(max_area^2)*(V^2);

% here the external aerodynamic torque on the S/C is calculated
Ta = F*max(abs(CG));
```

torque_gravity.m

```
%   Disturbance torque from gravity gradient
%
%   Input:
%     r = vehicle radius [m]
%     mu = planet gravity constant [m^3/s^2]
%     I = vehicle moment of inertia [kg*m^2]
%
%   Output:
%     T_grav = gravity gradient torque [Nm]
%

function T_grav = torque_gravity(r, mu, I)

  %# Max moment
    Imax = max(diag(I)); %[kg*m^2]

  %# Min moment
    Imin = min(diag(I)); %[kg*m^2]

  %# Angle deviation from vertical
    theta = 45*pi/180; %[rad], worst case angle chosen

  %# Calc gravity gradient torque
    T_grav = 3*mu*sin(2*theta)*(Imax - Imin)/(2*r^3); %[N*m]
```

torque_magnetic.m

```
%   Disturbance torque from magnetic field
%
%   Input:
%     lat = vehicle latitude [rad]
%     r = vehicle position vector magnitude [m]
%     re = earth radius [m]
%
%   Output:
%     T_mag = magnetic field torque [Nm]
%

function T_mag = torque_magnetic(lat,r,re)

  %# Earth magnetic field (approx as dipole)
    B = (1 + sin(lat)^2)^(0.5) * 0.3/((r/re)^3); %[gauss]

    B_t = B*1e-4; %[tesla], [N/(A*m)]

  %# Vehicle residual dipole
    D = 1; %[A*m^2]

  %# Mag torque
    T_mag = B_t*D; %[Nm]
```

torque_solar.m

```
function T_solar = torque_solar(A, CG, i, mat)

% function to computer solar radiation pressure

% INPUTS
% A: vector describing size of object
% CG: distance from center of solar pressure to center of mass (m)
% i: angle of incidence of the Sun (radians)
% mat: ID of material on outside of craft

% OUTPUT
% T_solar: solar radiation pressure, in N*m

% some constants:
% speed of light, m/s
c = 3*10^8;

% solar constant, W/m^2
F_s = 1367;

% get reflectance, q, from file based on material used
tmp = xlsread('material_prop.xls','abs');
q = tmp(mat,3);

% find surface area of largest face of orbit
A_s = A(1)*A(2);
if(A(1)*A(3) > A_s)
    A_s = A(1)*A(2);
end

if(A(2)*A(3) > A_s)
    A_s = A(2)*A(3);
end

F = (F_s/c)*A_s*(1 + q)*cos(i);

T_solar = F*(max(abs(CG)));
```

get_wheel_data.m

```
    function p = get_wheel_data

    wheel(1) = struct('name', 'Teldix RSI 01-5/15', 'ang_moment', 0.04, 'mass', 0.6);
    wheel(2) = struct('name', 'Teldix RSI 01-5/28', 'ang_moment', 0.12, 'mass', 0.7);
    wheel(3) = struct('name', 'LeoStar', 'ang_moment', 4.7, 'mass', 3.628);
    wheel(4) = struct('name', 'Dyncon MicroWheel 200', 'ang_moment', 0.18, 'mass', 0.93);
    wheel(5) = struct('name', 'Honeywell HR12', 'ang_moment', 50, 'mass', 9.5);
    wheel(6) = struct('name', 'Honeywell HR14', 'ang_moment', 75, 'mass', 10.6);
    wheel(7) = struct('name', 'Honeywell HR16', 'ang_moment', 100, 'mass', 12);
    wheel(8) = struct('name', 'Honeywell Miniature Reaction Wheel', 'ang_moment', 1.0, 'mass',
1.3);
    wheel(9) = struct('name', 'Honeywell HR0610', 'ang_moment', 12, 'mass', 5.0);
    wheel(10) = struct('name', 'Teldix DR23-0', 'ang_moment', 23, 'mass', 6.9);
    wheel(11) = struct('name', 'Teldix RDR68-6', 'ang_moment', 68, 'mass', 9.1);
    wheel(12) = struct('name', 'Teldix RSI 25-75/60', 'ang_moment', 25, 'mass', 6.3);
    wheel(13) = struct('name', 'Teldix RSI 68-75/60x', 'ang_moment', 68, 'mass', 8.5);
    wheel(14) = struct('name', 'Teldix RSI 4-75/60', 'ang_moment', 4, 'mass', 3.7);
    wheel(15) = struct('name', 'Teldix RSI 12-75/60x', 'ang_moment', 12, 'mass', 4.85);
    wheel(16) = struct('name', 'Teldix RSI 18-220/45', 'ang_moment', 18, 'mass', 6.3);
    wheel(17) = struct('name', 'Teldix RSI 30-280/30', 'ang_moment', 30, 'mass', 8.5);
    wheel(18) = struct('name', 'Teldix RSI 68-170/60', 'ang_moment', 68, 'mass', 8.9);
    wheel(19) = struct('name', 'Teldix RSI 02-25/30', 'ang_moment', 0.2, 'mass', 1.7);
    wheel(20) = struct('name', 'Teldix RSI 04-25/60', 'ang_moment', 0.4, 'mass', 1.7);
    wheel(21) = struct('name', 'Teldix RSI 1.6-25/60', 'ang_moment', 1.6, 'mass', 2.4);

    for(i=1:length(wheel))
        %plot(wheel(i).ang_moment, wheel(i).mass, 'r*');
        %hold on;
        ang(i) = wheel(i).ang_moment;
        mass(i) = wheel(i).mass;
    end


    [p,s] = polyfit(ang, mass, 4);
    %f = polyval(p, ang);
    %plot(ang, f, 'g*');
```

oe2rv.m

```
%  CREDIT: Christopher D. Hall
%          http://www.aoe.vt.edu/~cdhall/
%
%  oe2rv.m  Orbital Elements to r,v
%
%  [r,v] = oe2rv(oe,mu)
%                 oe = [a e i Om om nu]
%                 r,v  expressed in  IJK  frame
%
% a = semi-major axis
% e = eccentricity
% i = inclination
% Om = argument of periapsis
% om = right ascension of the ascending node (longitude of ascending node)
% nu = true anomaly (at epoch). ***(location on orbit)***

function [ri,vi] = oe2rv(oe,mu)
   a=oe(1); e=oe(2); i=oe(3); Om=oe(4); om=oe(5); nu=oe(6);
   p = a*(1-e*e);
   r = p/(1+e*cos(nu));
   rv = [r*cos(nu); r*sin(nu); 0];                      % in PQW frame
   vv = sqrt(mu/p)*[-sin(nu); e+cos(nu); 0];
%
%  now rotate
%
   cO = cos(Om);  sO = sin(Om);
   co = cos(om);  so = sin(om);
   ci = cos(i);   si = sin(i);
   R  = [cO*co-sO*so*ci  -cO*so-sO*co*ci  sO*si;
            sO*co+cO*so*ci  -sO*so+cO*co*ci -cO*si;
            so*si            co*si           ci];
   ri = (R*rv)';
   vi = (R*vv)';
```

prop_system.m

```
    % Bill Nadir
    % 16.851 Satellite Engineering
    % 10/11/2003

    % Module for calculating spacecraft propulsion system mass for required
    % momentum dumping

    function [F, p_mass] = prop_system(size,CG,lifetime,H,sat_rate)

    % INPUTS
    % size      = edge lengths of the S/C: vector (x,y,z) side lengths (meters)
    % CG        = (x,y,z) coordinates of the location of the CG (offset from the
    %             geoometric center of the S/C)
    % lifetime  = required lifetime of the spacecraft [yrs]
    % H         = maximum stored momentum in any one momentum wheel (saturation
    %             point of a momentum wheel) [N*m*s]
    % sat_rate  = The rate of saturation of a momentum wheel (used to determine
    %             how often momentum needs to be dumped) [days/saturation]
    % OUTPUTS
    % p_mass    =  total mass of the propulsion subsystem which will provide
    %              momentum dumping capability for the spacecraft [kg]
    % F         = Thrust required to dump momentum

    % Hydrazine (monopropellant) is chosen as the fuel for this propulsion
    % system and a conservative specific impulse, Isp, is 200 seconds
    Isp = 200;

    % Here the earth's gravity constant is initialized (9.8 m/s^2)
    g = 9.8;

    % Here the impulse time, t, of the thruster firing is set
    % It is assumed that the thruster required for momentum dumping will fire
    % for 1 second
    t = 1;

    % Here the locations of the six required thrusters are initialized [x y z]
    % Each row is for a different thruster
    thruster = [0 size(2)/2 size(3)/2; 0 size(2)/2 -size(3)/2; size(1)/2 0 size(3)/2; -size(1)/2
0 size(3)/2; size(1)/2 -size(2)/2 0; -size(1)/2 -size(2)/2 0];

    % Here the moment arms for the six thrusters from the CG are determined

    % For X-thrusters (spin about X-axis), moment arm is in Y-direction (cols 1,2)
    % For Y-thrusters (spin about Y-axis), moment arm is in Z-direction (cols 3,4)
    % For Z-thrusters (spin about Z-axis), moment arm is in X-direction (cols 5,6)
    moment_arms = [abs(CG(2) - thruster(1,2)) abs(CG(2)- thruster(2,2)) abs(CG(3) -
thruster(3,3)) abs(CG(3) - thruster(4,3)) abs(CG(1) - thruster(5,1)) abs(CG(1) - thruster(6,1))];

    % Here we will assume the worst-case distance from the thruster to the CG
    % (shortest) which will require the largest thrust to impart the required
    % torque on the S/C for momentum dumping
    worst_moment_arm = min(moment_arms);


    % Here the thrust required to dump the momentum is calculated (per pulse)
    F = H / (worst_moment_arm * t);


    % Here the required propellant mass for this propulsion system is estimated
    total_pulses = (lifetime * 365.25) / sat_rate; % total thruster pulses required over lifetime
    m_prop = (F * total_pulses * t)/(Isp * g);  % mass in kg


    % Here the total propulsion system mass is determined by assuming that 85%
    % of the propulsion system mass is propellant (SMAD, p. 660) - conservative
    p_mass = m_prop / 0.85;  % mass in kg
```

***material_prop.xls*[20]**

NAME	Info	MaterialNumber	Absorptivity	Reflectance
OpticalSolarReflector	SSE	1	0.07	0.93
QuartzOverSilver	SMAD	2	0.077	0.923
SilvercoatedFEP	SSE	3	0.08	0.92
SilveredTeflon	SMAD	4	0.08	0.92
AluminizedTeflon	SMAD	5	0.163	0.837
WhiteEpoxy	Al.Substrate	6	0.248	0.752
WhiteEnamel	Al.Substrate	7	0.252	0.748
AluminizedFEP	SSE	8	0.16	0.84
SilverPaint	SSE	9	0.37	0.63
SolarCellFusedSilica	SMAD	10	0.805	0.195
BlackPaint	Al.Substrate	11	0.975	0.025
Titanium6AL4V	AsReceived	12	0.766	0.234
SteelAm350	AsReceived	13	0.567	0.433
Titanium6AL4V	Polished	14	0.448	0.552
AluminiumTape	SSE	15	0.21	0.79
Aluminum606T6	Polished	16	0.2	0.8
Gold	AsRolled	17	0.299	0.701
Aluminum606T6	AsReceived	18	0.379	0.621
GoldizedKapton	SSE	19	0.25	0.75
PolishedBeryllium	SSE	20	0.44	0.56

References

[1] Meriam, J.L., and Kraige, L.G., *Engineering Mechanic: Dynamic*, 4th Ed, John Wiley, Inc. 1997.
[2] Hall, Christopher D., *oe2rv.m* software tool, *http://www.aoe.vt.edu/~cdhall/*
[3] Benson, Tom, http://www.grc.nasa.gov/WWW/K-12/airplane/atmosmet.html, *Earth Atmosphere Model, Metric Units*, NASA Glenn Research Center, 2002.
[4] Chobotov, Vladimir A. (Editor), *Orbital Mechanics*, 2nd Ed., AIAA, 1996, p. 227
[5] Benson, Tom, http://www.grc.nasa.gov/WWW/K-12/airplane/atmosmet.html, *Earth Atmosphere Model, Metric Units*, NASA Glenn Research Center, 2002.
[6] Wertz, James, and Larson, Wiley, *Space Mission Analysis and Design*, 2nd Ed., Microcosm, Inc., 1997, p. 353.
[7] E. Ahronovich, M. Balling, *Reaction Wheel and Drive Electronics For LeoStar Class Space Vehicles*, 12th Annual USU Conference on Small Satellites, 1998, www.sdl.usu.edu/conferences/smallsat/proceedings/12/ssc98/1/ssci5.pdf
[8] Dynacon Enterprises Limited, *Dynacon MicroWheel 200*, www.dynacon.ca/pdf/files/productpdf_6.pdf
[9] Honeywell Aerospace Electronic Systems, *Constellation Series Reaction Wheels*, http://content.honeywell.com/dses/assets/datasheets/constellation_series_reaction_wheels.pdf.
[10] Honeywell Aerospace Electronic Systems, *Miniature Reaction Wheels*, http://content.honeywell.com/dses/assets/datasheets/mini-wheel_reaction_wheel.pdf
[11] Honeywell Aerospace Electronic Systems, *Honeywell Model HR 0610 Reaction Wheel*, http://content.honeywell.com/dses/assets/datasheets/hr0610_reaction_wheel.pdf
[12] Teldix Space Product Group, *Momentum and Reaction Wheels 14-68 Nms with external Wheel Drive Electronics*, http://www.teldix.de/P22/RDR23-68.pdf
[13] Teldix Space Product Group, *Momentum and Reaction Wheels 14-68 Nms with integrated Wheel Drive Electronics*, http://www.teldix.de/P22/RSI25-68.pdf
[14] Teldix Space Product Group, *Momentum and Reaction Wheels 2-12 Nms with integrated Wheel Drive Electronics*, http://www.teldix.de/P22/RSI4-12.pdf
[15] Teldix Space Product Group, *High motor torque Momentum and Reaction Wheels 14-68 Nms with integrated Wheel Drive Electronics*, http://www.teldix.de/P22/RSI18-68.pdf
[16] Teldix Space Product Group, *Momentum and Reaction Wheels 0.04-0.12 Nms with integrated Wheel Drive Electronics*, http://www.teldix.de/P22/RSI01.pdf
[17] Teldix Space Product Group, *Momentum and Reaction Wheels 0.2-1.6 Nms with integrated Wheel Drive Electronics*, http://www.teldix.de/P22/RSI02.pdf
[18] Sellers, Jerry Jon, *Understanding Space: An Introduction to Astronautics*, 2nd Ed., McGraw Hill, 2000, p. 570.
[19] Wertz, James, and Larson, Wiley, *Space Mission Analysis and Design*, 2nd Ed., Microcosm, Inc., 1997, p. 660.
[20] Fortescue, Peter, Stark, John, and Swinerd, Graham, *Spacecraft Systems Engineering*. John Wiley and Sons Ltd., Third Edition.

Problem Set 4: Efficient Orbit Transfer: Use of Electric Propulsion for Orbit Raising

Summary

A software module was developed to size the power and electric propulsion systems for a spacecraft based on spacecraft mass, initial and final orbit radii, and a desired transfer time, assuming constant tangential thrust. Several types of electric propulsion systems were investigated for use in orbit raising of satellites, such as Xenon ion propulsion, Xenon-Hall effect propulsion, and pulsed plasma thrusters (PPT).

Results

A test case was run for raising the orbit of a communications satellite from a LEO parking orbit into geosynchronous orbit. During the investigation of the various types of electric propulsion systems, it was learned that certain types of electric propulsion systems, such as pulsed plasma thrusters, do not have the capability for reasonable transfer times for such an orbit transfer. Therefore, only the Xenon ion and Xenon Hall thrusters were considered for this test case.

It took approximately 600 "orbits" in the spiral orbit transfer maneuver to raise the orbit of the satellite from a parking orbit to geosynchronous orbit. It was also noticed that this sort of orbit raising maneuver was just within the lifetime capabilities of the two types of electronic propulsion systems considered. This means that this type of electric propulsion could be a viable option for orbit raising to geosynchronous orbits for satellites that do not have to be urgently rushed into service in GEO.

Useful References

Electric Propulsion

Martinez-Sanchez, Manuel, *Spacecraft Electric Propulsion – An Overview*, Journal of Propulsion and Power, Vol. 14, No. 5, 9/98-10/98, p. 690.
Tajmar, Martin, *Advanced Space Propulsion Systems*, Springer Wien, New York, 2003, p. 76.

The references above contain detailed information about all available electric propulsion systems. They contain performance, mass, and additional information about each propulsion system.

Spiral Orbit Raising

Course notes, AA420, University of Washington Dept. of Aeronautical and Astronautical Engineering, 1999.

The course notes from this University of Washington engineering course contain a useful derivation for a simplified spiral maneuver using electric propulsion with constant thrust. More detailed calculations exist for optimal maneuvers using electric propulsion, but the derivation from this course is best for the scope of this problem set.

Problem Set 4 Solution
MEMORANDUM

16.851 Satellite Engineering

To: Professor David W. Miller
Col. John E. Keesee

From: 16.851 Students

Date: 29 October 2003

Subj: Use of Electric Propulsion for Orbit Raising: Orbits, Propulsion, and Power

cc: Marilyn Good

MOTIVATION

Electric propulsion systems offer the capability for mass-efficient orbit transfers. The specific impulse for electric propulsion is much higher than for chemical propulsion, ranging from 1,500 to 20,000 seconds;[1] however, electric propulsion provides much lower thrust than chemical propulsion. This results in much longer spacecraft maneuver times for a given change in velocity. If short maneuver times are not critical, electric propulsion may lend itself to be used as the propulsion system for orbit transfer maneuvers.

PROBLEM STATEMENT

Create a software module that sizes the power and electric propulsion systems for a spacecraft, given the spacecraft mass, initial and final circular orbit radii, and a specified transfer time, assuming constant tangential thrust. Use this module to characterize the dependence of propulsion and power system mass on orbit transfer requirements. Investigate this dependency for several different types of electric propulsion, such as pulsed plasma thrusters (PPT), Xenon ion propulsion, and Xenon Hall-effect propulsion.

SOLUTION

See attached.

[1] http://web.mit.edu/dept/aeroastro/www/labs/SPL/electric.htm, MIT Space Propulsion Lab Website, 2002.

Efficient Orbit Transfer: Use of Electric Propulsion for Orbit Raising

Software Designed to Provide Preliminary Sizing of Power and Propulsion Systems

16.851 Satellite Engineering

Massachusetts Institute of Technology, Cambridge, MA

October 2003

Motivation

Electric propulsion systems offer the capability for mass-efficient orbit transfers. The specific impulse for electric propulsion is much higher than for chemical propulsion, ranging from 1,500 to 20,000 seconds;[1] however, electric propulsion provides much lower thrust than chemical propulsion. This results in much longer spacecraft maneuver times for a given change in velocity. If short maneuver times are not critical, electric propulsion may lend itself to be used as the propulsion system for orbit transfer maneuvers.

Problem Statement

Create a software module that sizes the power and electric propulsion systems for a spacecraft, given the spacecraft mass, initial and final circular orbit radii, and a specified transfer time, assuming constant tangential thrust. Use this module to characterize the dependence of propulsion and power system mass on orbit transfer requirements. Investigate this dependency for several different types of electric propulsion, such as pulsed plasma thrusters (PPT), Xenon ion propulsion, and Xenon Hall-effect propulsion.

Introduction

Inputs to the software module include the initial and final orbit radii and the orbit transfer time. The module determines the constant propulsive force required to move the spacecraft from the initial orbit to the final orbit in the specified time. Using this constant force, and given the type of propulsion system, the propulsion system is sized by determining the total propulsion system mass required for the orbit transfer. For this propulsion system, the batteries and solar arrays required to support the maneuver are sized. The performance results are compared for several types of electric propulsion systems.

Software Module

Test Case Script

Requirements

The MATLAB script *electric_propulsion.m* is used to simplify the use of the primary software modules, and to run pre-configured test cases to produce the results presented in this report.

Description of the code

The script initializes the orbit transfer and spacecraft properties, calls each of the primary software modules in turn, and plots the results. In summary, the script performs the following functions:

- Set inputs: spacecraft mass, propulsion type, transfer time, initial radius, final radius.
- Get propulsion system properties from *propulsion_properties.m*.
- Determine the thrust and the orbit characteristics for the transfer maneuver using *ep_orbit.m*.
- Size the power and propulsion systems to provide the needed thrust and Δv, using *propulsion_power.m*.
- Plot the results.

Typing *electric_propulsion* at the MATLAB prompt runs the test case and produces the output shown in this report.

Propulsion Properties Module

Requirements

The MATLAB module *propulsion_properties.m* sets the propulsion system-specific values based on the type of propulsion system selected.

Description of the code

The desired type of propulsion system is passed to the module as a character string (e.g. ‘ion’). Propulsion system constants such as specific impulse and

efficiency for the specified type of electric propulsion system are returned in a data structure. This structure is a required input for the *electric_propulsion.m* and *propulsion_properties.m* modules.

Inputs

p_system: a string specifying which propulsion system is to be used. Valid values are 'ion' (Xenon Ion), 'hall' (Xenon Hall), and 'ppt' (Pulsed Plasma Thrusters).

Outputs

properties: a data structure containing descriptions of properties inherent to the propulsion system type, such as I_{sp}, efficiency, and lifetime.

Orbital Transfer Module

Requirements

The MATLAB module *ep_orbit.m* determines the constant tangential thrust required to expand an orbit using an electric propulsion system, and characterizes the expansion path, determining quantities such as the eclipse entry and exit times and the Δv applied during each orbit.

In order to maintain reasonable scope in the project, several assumptions were made with respect to the initial and final orbits and transfer path. First, all orbits are assumed to be circular and equatorial. This simplification applies to the initial and final orbits, as well as to all intermediate steps in the transfer path. The assumption of circularity during the transfer is reasonable both because the Δv imparted by the propulsion system during the maneuver is far smaller in magnitude than the orbital velocity, and because this Δv is applied continuously throughout the orbit, rather than at discrete points.

Description of the code

The code first verifies that all input quantities lie within valid ranges. The constant thrust is then calculated using Equation 14. The algorithm then steps through the transfer path one orbit at a time, calculating variable quantities such as mass, eclipse time, orbital radius, and Δv, and recording how these quantities change through the transfer maneuver.

Inputs

mass (kg): the initial mass of the spacecraft.

prop: a data structure containing propulsion system properties, as output by *propulsion_properties.m*.

time (s): the required orbital transfer time.

r0 (m): initial circular orbit radius.

r1 (m): final circular orbit radius.

Outputs

thrust [N]: constant thrust magnitude required to complete the specified maneuver.

radii [m]: history of orbit radii.

period [s]: history of orbital period.

eclipse [s]: history of eclipse entry and exit times.

deltav [m/s]: history of applied Δv.

Derivation of the spiral orbit Δv equation[2]

For thrust $T << mg$, the specific mechanical energy ε of the spacecraft in its orbit changes as:

$$\begin{aligned}\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} &= \frac{\mathrm{d}E}{m\mathrm{d}t} \\ &= \frac{\vec{T}\cdot\vec{v}}{m}\end{aligned} \tag{1}$$

For thrust applied in the velocity direction, this can be expressed using the magnitudes of T and v. The acceleration of the spacecraft is related to the specific mechanical energy as:

$$\begin{aligned}\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} &= \frac{Tv}{m} \\ &= av\end{aligned} \tag{2}$$

In a circular orbit, the specific mechanical energy and its derivative can also be expressed as:

$$\varepsilon = \frac{-\mu}{2r} \tag{3}$$

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} = \frac{\mu}{2r^2}\frac{\mathrm{d}r}{\mathrm{d}t} \tag{4}$$

Equating these two relations for the time derivative of the specific mechanical energy:

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} = \frac{\mu}{2r^2}\frac{\mathrm{d}r}{\mathrm{d}t} = av \tag{5}$$

Substituting in the velocity in a circular orbit

$$v = \sqrt{\frac{\mu}{r}} \tag{6}$$

leads to:

$$\frac{\mu}{2r^2}\frac{\mathrm{d}r}{\mathrm{d}t} = a\sqrt{\frac{\mu}{r}}$$
$$\frac{\mathrm{d}r}{\mathrm{d}t} = \frac{2ar^{3/2}}{\mu^{1/2}} \tag{7}$$
$$\frac{\mathrm{d}r}{r^{3/2}} = \frac{2a}{\sqrt{\mu}}\mathrm{d}t$$

This relationship can be integrated to give the change in r as a function of the change in time:

$$\int_{t_0}^{t_1}\frac{2a}{\sqrt{\mu}}\mathrm{d}t = \int_{r_0}^{r_1}\frac{\mathrm{d}r}{r^{3/2}}$$
$$\frac{2a}{\sqrt{\mu}}(t_1 - t_0) = \frac{-1}{\sqrt{r}}\Big|_{r_0}^{r_1} \tag{8}$$
$$= \frac{1}{\sqrt{r_1}} - \frac{1}{\sqrt{r_0}}$$

Rearranging this relationship reveals the Δv:

$$2a(t_1 - t_0) = \sqrt{\frac{\mu}{r_1}} - \sqrt{\frac{\mu}{r_0}}$$
$$= v_1 - v_0 \tag{9}$$
$$= \Delta v$$

Note that this result, although simple and apparently intuitive, is not generally true for orbital transfers using chemical propulsion. The Δv required to expand an orbit using impulsive burns is actually less than this amount, since the burns can be made at discreet optimal points during the orbit.

Derivation of thrust equations

The thrust T is related to the propellant mass flow rate and exit velocity c as

$$T = \dot{m}c \tag{10}$$

This relationship can be integrated for constant mass flow rate (i.e. constant thrust and exit velocity) over a time period t to obtain the propellant mass for constant T and c:

$$m_p = \int_{t_0}^{t_1}\dot{m}\,\mathrm{d}t$$
$$= \frac{Tt}{c} \tag{11}$$

The rocket equation provides a relationship between the initial spacecraft mass, propellant mass, Δv, and exit velocity:

$$\frac{m_p}{m_0} = 1 - e^{-\Delta V/c} \tag{12}$$

Combining these results gives:

$$T = m_0\frac{c}{t}\left(1 - e^{-\Delta V/c}\right) \tag{13}$$

Using the Δv equation previously derived, this equation can be solved for the velocity v as a function of time t.

$$T = m_0\frac{c}{t}\left(1 - e^{\frac{\left(\sqrt{\mu/r} - \sqrt{\mu/r_0}\right)}{c}}\right) \tag{14}$$

$$1 = e^{\frac{\left(\sqrt{\mu/r} - \sqrt{\mu/r_0}\right)}{c}} + \frac{tT}{m_0 c} \tag{15}$$

$$0 = \frac{1}{c}\left[\sqrt{\frac{\mu}{r}} - \sqrt{\frac{\mu}{r_0}}\right] - \ln\left[1 - \frac{tT}{m_0 c}\right] \tag{16}$$

$$\sqrt{\frac{\mu}{r}} = \left(c\ln\left[1 - \frac{tT}{m_0 c}\right] + \sqrt{\frac{\mu}{r_0}}\right) \tag{17}$$

This equation can then be solved for the radius r as a function of time:

$$r = \mu\left(c\ln\left[1 - \frac{tT}{m_0 c}\right] + \sqrt{\frac{\mu}{r_0}}\right)^{-2} \tag{18}$$

This equation can be used to calculate the radius at any time t during the transfer maneuver.

Propulsion and Power Sizing Module

Requirements

The MATLAB module *propulsion_power.m* calculates the mass of the electric propulsion system, the required solar array area, the required battery mass, and the mass of the solar arrays based on a given spiral-shaped orbit transfer. The module requires inputs of constant thrust during the orbit transfer, transfer time, time information which specifies when the spacecraft is in eclipse or sunlight during the orbit transfer, and the selection about which type of electric propulsion system is being investigated for the orbit transfer.

Description of the code

The code assumes that the starting point of the spiraling orbit transfer is at the moment when the satellite enters the earth's eclipse in the initial orbit. Each "orbit" of the spiral orbit transfer is considered to be the time from when the satellite enters the earth's eclipse until the next time it enters the eclipse. These orbits of the spiral orbit transfer are used to size the power system of the spacecraft. This is discussed in the Theory & Equations section for this module.

The code first checks to make sure that the input values from the user of initial orbit, final orbit, transfer time, and propulsion system type, are valid. For example, the transfer time must be less than the lifetime of the selected propulsion system.

Constants

The first constant used in this module is the gravitational acceleration constant, g. It is input into the module as being equal to 9.81 m/s^2.

The next sets of constants are specific for each propulsion system. Table 1 lists the basic constants used to describe the performance of each propulsion system, and Table 2 lists additional properties and constants.

Table 1. Propulsion system constants[3]

Propulsion System	I_{sp} (s)	Efficiency (%)	Lifetime
Xe Ion	2800	65	10000 hr
Xe Hall	1600	50	>7000 hr
PPT	1000	7	4000 N*s

Table 2. Additional propulsion system constants[4]

Propulsion System	Thrust Range[5] (N)	Thruster Mass (kg/kW)	PPU Mass (kg/kW)	Misc. Mass (kg/kW)
Xe Ion	.01 - .20	4.5	8	10
Xe Hall	.08 - .20	2.5	8	10
PPT	.001 - .10	120	110	small

Two additional constants used are used in the calculations for sizing the power system. These constants, Xe and Xd, are the efficiencies of the electrical paths from the solar arrays through the batteries to the loads and the path directly from the arrays to the loads, respectively.[6] Since it is assumed that a direct energy transfer power system is being used, the values for Xe and Xd are 0.65 and 0.85, respectively.

The final constant used in this module is the solar illumination intensity. This is assumed to be 1358 W/m^2.

Inputs

thrust (N): This input is the constant thrust required for the spacecraft to achieve its desired final orbit within the specified transfer time.

orbit (s): This input is an array of times during the orbit. The numbers specify the time during the orbit transfer when the spacecraft enters and exits the earth's eclipse.

Properties (s): the propulsion system properties, as output by *propulsion_properties.m.*

t_transfer (s): This input is the total transfer time specified for the spiral-shaped orbit raising maneuver.

Outputs

power.A_cells_req (m^2): This output is the total solar array area required to provide the necessary power for the propulsion system being used in the orbit transfer maneuver.

power.m_cells_req (kg): This output is the total mass of the solar arrays required to power the propulsion system as well as charge the batteries when in sunlight.

power.m_batt_req (kg): This output is the total mass of the batteries needed to provide power to the propulsion system during the worst case eclipse (longest duration).

propulsion.m_thruster (kg): This output is the mass of the thrusters in the chosen propulsion system.

propulsion.m_ppu (kg): This output is the mass of the power processing unit in the chosen propulsion system.

propulsion.m_misc (kg): This output is the mass of miscellaneous components in the chosen propulsion system.

propulsion.m_propellant (kg): This output is the mass of the propellant in the chosen propulsion system.

Theory & Equations

Based on the inputs to the module, the first desired quantity to be calculated is the required power for the propulsion system. In order to calculate the required power, the mass flow rate of the propellant must be first determined. This is done using Equation 19 below.[7]

$$\dot{m} = \frac{F}{I_{sp} g} \tag{19}$$

In the above equation, F is the propulsion system constant thrust, I_{sp} is the specific impulse of the propulsion system, and g is the gravitational acceleration constant.

Next, the required power for the propulsion system is determined in Equation 20.[8]

$$P = \frac{F^2}{2\dot{m}\eta} \tag{20}$$

In the above equation, η is the efficiency of the propulsion system, as reported in Table 1.

The next required quantity to be determined is the power required to be generated by the solar arrays for each orbit. This would include the power provided to the propulsion system as well as the power provided to the batteries for charging. The following equation is used to determine this required power for each "orbit" portion of the orbit transfer.

$$P_{sa_i} = \frac{P\left(\frac{T_{e_i}}{X_e} + \frac{T_{d_i}}{X_d}\right)}{T_{d_i}} \quad \text{where } i = 1,2,... \tag{21}$$

In Equation 21, Te and Td are the durations of the eclipse and daylight times for each orbit portion of the spiral orbit transfer. Each orbit is determined by the time at which the satellite enters the earth's eclipse until the time it next enters the eclipse. These orbits are denoted in the equation with the subscript *i*. It can be seen that only the eclipse and daylight times change with each successive orbit.

Normally, the numerator of Equation 21 has different power values for daytime and eclipse, but in this case, since the power required by the propulsion system does not vary from daylight to eclipse, this quantity is held constant at the value *P*, determined earlier.

Next, the power generated per area by the solar arrays is determined. This calculation is shown in Equation 22. It should be noted here that Gallium Arsenide solar cells are being used for the purposes of this software module. In addition, it should be noted here that it is assumed that beyond the inherent inefficiencies in the solar cells, no additional degradation is assumed for the solar cells during the orbit transfer. This assumption is made because the orbit transfer is relatively short compared to the life of the satellite and the arrays will likely not degrade significantly during this orbit raising maneuver. Therefore, the power determined in the following equation is assumed to be an end-of-life (EOL) power for the solar arrays as far as the orbit raising maneuver is concerned.

$$(p_0)_{EOL} = \eta_{cells}(1358\tfrac{W}{m^2})I_d \tag{22}$$

In the above equation, η_{cells} is the efficiency of the GaAs cells, which is approximately 18%. In addition, the quantity I_d is the inherent degradation of the solar cells. This is assumed to be 0.77.[9]

It should also be mentioned here that the solar arrays are assumed to track the sun and therefore it is assumed that the radiation from the sun is perfectly incident at all times on the solar arrays. This means that any reduced performance to any incidence angle to the solar radiation is ignored.

Next, the required solar array area for each orbit during the transfer maneuver is determined. This is shown in the equations below.

$$A_{cells_i} = \frac{P_{sa_i}}{(p_0)_{EOL}} \tag{23}$$

$$A\_cells\_req = \max(A_{cells_i}) \tag{24}$$

Equation 24 show how module determines the required solar array area to be the maximum of all the areas calculated for all orbit portions of the spiral orbit raising maneuver.

Next, the required solar array mass can be determined. This is done using the following equation from SMAD.[10]

$$m\_cells\_req = 0.04 * \max(P_{sa_i}) \tag{25}$$

Next, the battery capacity is determined to allow for sizing of the batteries for the spacecraft. It is assumed that NiH_2 batteries are being used with a specific energy density of approximately 50 W*hr/kg.[11] It is also assumed that only a single battery is being used. Equation 26 below is used to determine the battery capacity required for each orbit eclipse during the orbit transfer.

$$C_{r_i} = \frac{PT_{e_i}}{(DOD)n} \tag{26}$$

In the above equation, DOD is the depth-of-discharge of the nickel-hydrogen batteries. This assumed to be 75% since the number of cycles the batteries will need to be cycled for throughout the orbit transfer will not degrade the depth-of-discharge capability of the batteries to any significant amount.[12]

The quantity *n* is the power transfer efficiency to the batteries. This is assumed to be 0.9.[13]

Next, the required battery mass is determined. This is shown in the following two equations.

$$m\_batt_i = \frac{C_{r_i}}{50\frac{W*hr}{kg}} \tag{27}$$

$$m\_batt\_req = \max(m\_batt_i) \tag{28}$$

Finally, the propulsion system mass is determined using information from Table 2. The required figures from the table are multiplied by the propulsion system power to determine the masses of all the components of the propulsion system. In addition, the mass of the propellant is determined from the following equation.

$$m_{propellant} = \dot{m}*t\_transfer \tag{29}$$

Once the propellant mass is determined, the total propulsion system mass is calculated as the sum of the thruster, power processing unit, miscellaneous, and propellant masses.

Results

The software module was run for a specific test case of raising a satellite from a circular parking orbit at an altitude of 10,000 km to a geosynchronous orbit. In addition, the transfer time was varied by the user within the ranges allowed for Xenon ion and Xenon Hall propulsion systems.

It should be noted that the orbit number being used in many of the following charts is how the spiral orbit transfer is broken up into individual parts. Each orbit in the transfer is determined from the time the satellite enters the earth's eclipse until it reenters the eclipse.

In Figure 1, it can be seen how the orbital period changes with the orbit number, as the radius of the orbit increases.

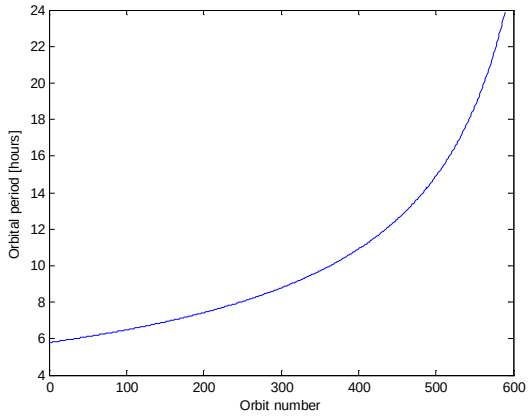

Figure 1. Orbit period vs. orbit number

The data in Figure 1 agree with what is expected for a spiraling orbit transfer, where the radius is constantly increasing. Figure 2 shows how the orbit radius varies with the orbit number for the test case.

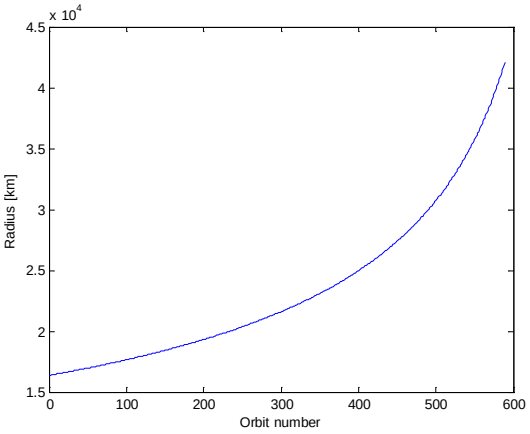

Figure 2. Orbit radius expansion with orbit number

It can be seen in Figure 2 that the orbit radius increases from the initial parking orbit until it reaches the orbit radius for geosynchronous orbit.

Figure 3 shows the time spent in eclipse during each orbit of the spiral transfer, and Figure 4 shows the fraction of each orbit during which the satellite is in eclipse.

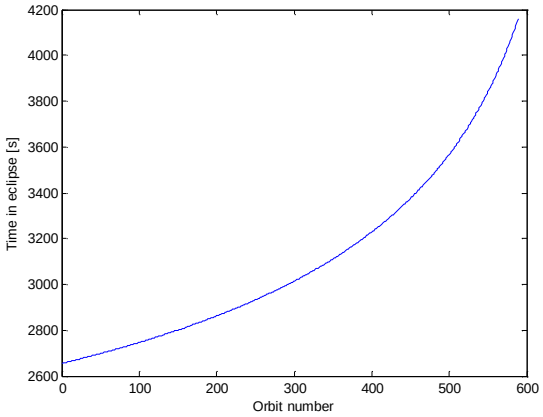

Figure 3. Eclipse time vs. orbit number

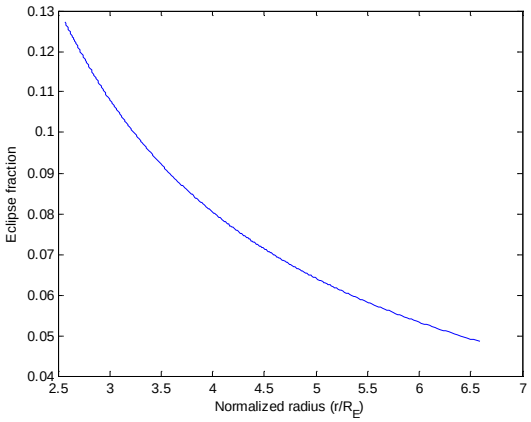

Figure 4. Eclipse fraction vs. normalized radius

As seen in Figure 3, as the radius increases, the time spent in eclipse should increase; however, as seen in Figure 4, the ratio of time spent in eclipse to the orbital period should decrease as the orbit radius increases. It should also be noted from Figure 4 that the orbit transfer ends when the orbit radius is 6.5 times the radius of the Earth, in geosynchronous orbit.

Figure 5 displays how the Δv per orbit provided by the propulsion system varies for each successive orbit during the transfer.

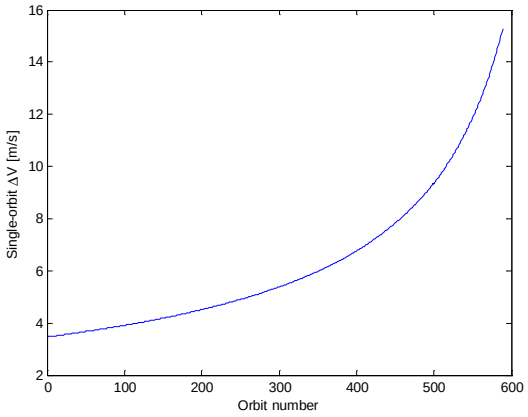

Figure 5. Single-orbit deltaV vs. orbit number

As expected, the Δv per orbit imparted by the electric propulsion system is continuously increasing as time elapses during the transfer orbit. This occurs because the orbital radius is constantly increasing throughout the orbit transfer due to the constant thrust provided by the electric propulsion system.

Figure 6 shows a perspective view of the complete path traversed during the spiral orbit transfer, from the initial orbit to the final orbit. The color of the spiral orbit transfer path varies from red at the 10,000 km altitude parking orbit to blue at the final, geosynchronous orbit.

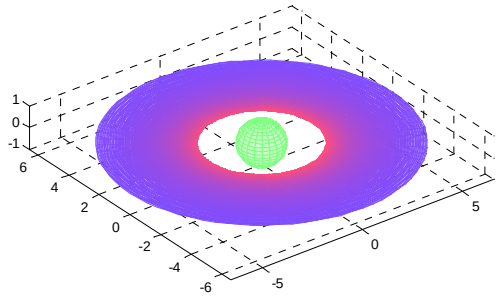

Figure 6. Orbit transfer path

Approximately 600 orbits are required to reach the final desired orbit, resulting in the densely-packed path shown above. Figure 7 shows a blown-up view of a portion of the spiral orbit track, in which individual spiral tracks are visible.

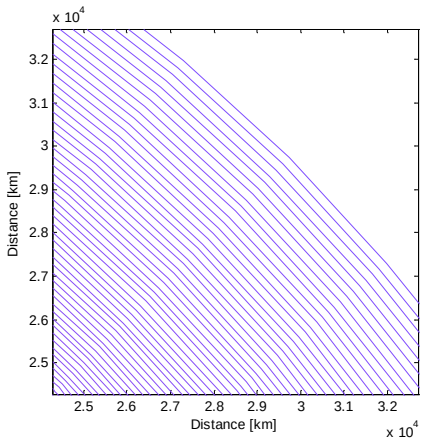

Figure 7. Blown-up view of spiral orbit path

It can be seen from Figure 7 that the change in orbital radius in each successive orbit is very small as compared to the orbital radius. This behavior is expected for low-thrust, long term maneuvers.

Figure 8 shows the required thrust from the propulsion system as a function of the user-specified transfer time.

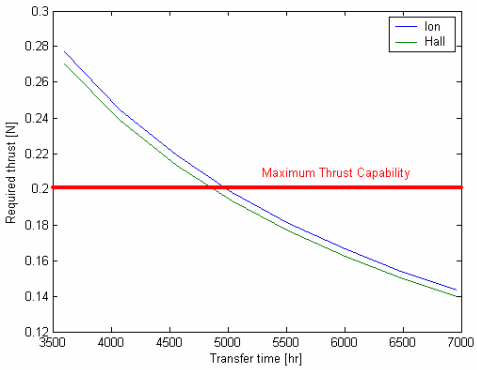

Figure 8. Dependency of thrust on transfer time

As expected, the thrust required to complete the transfer in the specified time decreases as the transfer time increases, for both the Xenon Ion and Hall propulsion systems. This makes sense because the required acceleration (and therefore thrust) should decrease as the transfer time increases.

It can also be seen in Figure 8 that the required thrust for each transfer time differs for the ion and Hall propulsion systems. This is due to the differences in mass flow rate of propellant out of the thrusters. The Xenon Hall thrusters have a lower I_{sp} than the Ion thrusters, which corresponds to a higher mass flow rate to produce a given thrust magnitude. The mass of the spacecraft decreases more rapidly with higher propellant mass flow rates, leading to more rapidly decreasing spacecraft mass, with the result that lower average thrust is requirement to achieve a given acceleration. This effect is likely due in part to the assumption that the initial spacecraft mass is independent of the type of propulsion system used.

Figure 9 shows how the total mass of the propulsion and power systems varies as a function of the specified orbit transfer time.

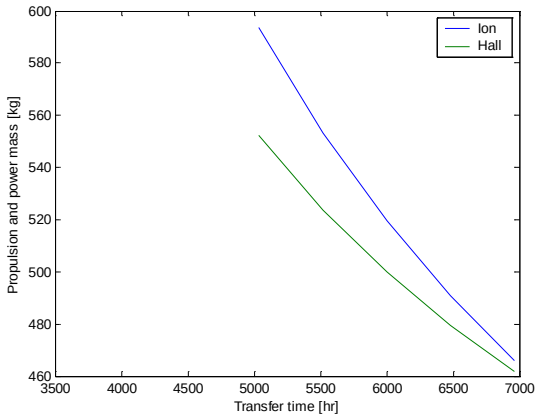

Figure 9. Total propulsion and power system mass as a function of transfer time

It can be seen in Figure 9 that no designs exist for transfer times less than approximately 5000 hours. This is due to the fact that neither the Xenon Ion nor the Hall propulsion systems are capable of producing high enough thrust to complete the orbit transfer maneuver in less time than 5000 hours. The maximum thrust capability of the two propulsion systems is shown explicitly in Figure 8 with a red horizontal line.

It should also be mentioned that the preceding figures do not show any data for the pulsed plasma propulsion system. It was determined during the testing of this software module that the PPT propulsion system does not have the capability to provide the ΔV necessary to perform the orbit transfer maneuver test case. Although PPT results are not shown, the software module is designed to handle such types of propulsion systems, and will return performance results given input requirements appropriate to the types of propulsion systems being considered.

Table 3 summarizes the final results for the sizing of the power and propulsion systems for the orbital transfer test case, using the Xenon Ion propulsion system. Table 4 summarizes the results for sizing the system using the Xenon Hall effect propulsion system.

Table 3. Sizing results using Xenon Ion propulsion

Solar cell area	19.0 m^2
Solar cell mass	142.7 kg
Battery mass	81.5 kg
Thruster mass	6.4 kg
PPU mass	20.4 kg
Misc. mass	25.5 kg
Propellant mass	223.3 kg
Total mass	*499.8 kg*

Table 4. Sizing results using Xenon Hall propulsion

Solar cell area	26.2 m^2
Solar cell mass	197.0 kg
Battery mass	112.6 kg
Thruster mass	15.8 kg
PPU mass	28.1 kg
Misc. mass	35.2 kg
Propellant mass	130.9 kg
Total mass	*519.5 kg*

Based on these results, the Ion and Hall propulsion systems appear to have similar performance for the test case transfer scenario. The Ion system is slightly less massive than the Hall-effect system, even though it requires 70% more propellant to accomplish the transfer. The mass savings is due to the lesser power requirement for the ion engine, which translates directly into smaller, less massive solar arrays and batteries.

Conclusion

A software module was created to size the propulsion and power systems for a spacecraft that uses electric propulsion for simple orbital transfers. The module requires the spacecraft mass, the initial and final circular orbit radii, a transfer time, and a propulsion system type. The software then determines the constant tangential thrust required to complete the orbit transfer in the specified time, the path of the satellite during the transfer maneuver, and the masses of the power and propulsion systems.

This tool is useful for preliminary sizing of propulsion and power requirements for a satellite when electric propulsion is used for orbital transfer. Although the tangential thrust approach to orbital transfer is sub-optimal, it is useful for rapid comparison of the relative performance of different types of propulsion system.

This tool can be easily extended to compare an arbitrary number of different types of electric propulsion system types. If the transfer scenario investigated is not within the capability of a propulsion system, results for that system are not displayed.

Future Work

One major task that could be undertaken in the future to improve and expand the capability of this software module would be to allow for plane changes during the orbit transfer. Another task would be to implement the equations for optimal thrust. Either one of these tasks would involve a significant amount of work (appropriate for Master's thesis-level research[14]), but would result in an extremely useful tool which could be used to provide detailed results for a greater number of orbit transfer scenarios.

Appendix A: MATLAB source code

The MATLAB files containing the implementations of the equations and relations described in this document are listed below. To run the test case, simply type *electric_propulsion* at the MATLAB prompt.

electric_propulsion.m

```
% constants
R_earth = 6378.1363; % [km] Earth average radius
geo = 42164.169637;  % [km] geostationary, from STK.
fignum = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% begin user-configurable inputs %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% initial spacecraft mass
m0  = 2000;

% propulsion system type ('ion', 'hall', 'ppt')
prop_type = 'ion';

% orbital transfer time
ttime = 3600*24*250;

% initial orbital radius
r0 = R_earth + 10000;

% final orbital radius
r1 = geo;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% end user-configurable inputs %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% get the propulsion system properties
prop = propulsion_properties(prop_type);

% get the orbital transfer properties
[thrust, radii, period, eclipse, deltav] = ep_orbit(m0, prop, ttime, r0*1000,
r1*1000);

% size the propulsion and power systems
[propulsion, power] = propulsion_power(thrust,eclipse,prop,ttime);

%%%%%%%%%%%%%%%%%%%%
% display results %
%%%%%%%%%%%%%%%%%%%%

radii  = radii/1000;                % change from [m] to [km]
orbits = 1:length(radii);           % orbit numbers
etime  = eclipse(2,:)-eclipse(1,:); % time spent in eclipse

fignum=fignum+1;
figure(fignum)
plot(orbits, period/3600);
xlabel('Orbit number')
ylabel('Orbital period [hours]');

fignum=fignum+1;
figure(fignum)
plot(orbits, radii);
xlabel('Orbit number');
```

```
ylabel('Radius [km]');

fignum=fignum+1;
figure(fignum)
plot(orbits, etime);
xlabel('Orbit number');
ylabel('Time in eclipse [s]');

fignum=fignum+1;
figure(fignum)
plot(orbits, deltav);
xlabel('Orbit number');
ylabel('Single-orbit {\Delta}V [m/s]');

fignum=fignum+1;
figure(fignum)
plot(radii/R_earth, etime./period);
xlabel('Normalized radius (r/R_E)');
ylabel('Eclipse fraction');

% plot all the orbit tracks around the Earth
fignum=fignum+1;
figure(fignum)
t = 0:pi/40:2*pi;
x = cos(t);
y = sin(t);
z = zeros(size(x));
[sx,sy,sz] = sphere(20);
colormap('white');
h1 = surf(sx,sy,sz);
set(h1,'EdgeColor',[.5 1.0 .5]);
set(h1,'FaceColor',[.8 1.0 .8]);
hold on
c = 'bgrcmyk';
scale = radii/R_earth;
for i=orbits
   h2 = plot3(scale(i)*x,scale(i)*y,scale(i)*z);
   r = 1.0 - i/orbits(end)/2;
   b = 0.5 + i/orbits(end)/2;
   set(h2, 'Color', [r 0.3 b]);
end
hold off
axis equal;

% zoom in to see some of the tracks closer up
fignum=fignum+1;
figure(fignum)
for i=orbits
   h3 = plot(radii(i)*x,radii(i)*y);
   r = 1.0 - i/orbits(end)/2;
   b = 0.5 + i/orbits(end)/2;
   set(h3, 'Color', [r 0.3 b]);
   hold on
end
hold off
axis equal;
maxrad = max(radii);
axis([maxrad/sqrt(3) 1.1*maxrad/sqrt(2) maxrad/sqrt(3) 1.1*maxrad/sqrt(2)]);
xlabel('Distance [km]')
ylabel('Distance [km]')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% trade over prop type and transfer time %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% propulsion system type ('ion', 'hall', ppt')
```

```
prop_types = {'ion', 'hall'};

% orbital transfer time
ttimes = 3600*24*[150:20:290];

data = [];
for i=1:length(prop_types)
   prop_type = prop_types{i};

   for j=1:length(ttimes)
      ttime = ttimes(j);

      % get the propulsion system properties
      prop = propulsion_properties(prop_type);

      % get the orbital transfer properties
      [thrust, radii, period, eclipse, deltav] = ...
         ep_orbit(m0, prop, ttime, r0*1000, r1*1000);

      % size the propulsion and power systems
      [propulsion, power] = propulsion_power(thrust,eclipse,prop,ttime);

      % save the interesting data
      data(i,j).thrust = thrust;
      data(i,j).time   = ttime;
      if ~isempty(propulsion) & ~isempty(power)
         data(i,j).mass = propulsion.m_thruster + propulsion.m_ppu + ...
            propulsion.m_misc + propulsion.m_propellant + ...
            power.m_cells_req + power.m_batt_req;
      else
         % no design was feasible for this case
         data(i,j).mass = NaN;
      end
   end
end

times   = [];
thrusts = [];
masses  = [];

% sort the data into standard vectors
for i=1:length(prop_types)
   times   = [times cat(2,data(i,:).time)'];
   thrusts = [thrusts cat(2,data(i,:).thrust)'];
   masses  = [masses cat(2,data(i,:).mass)'];
end

fignum=fignum+1;
figure(fignum)
plot(times/3600,thrusts);
xlabel('Transfer time [hr]');
ylabel('Required thrust [N]');
legend('Ion','Hall');

fignum=fignum+1;
figure(fignum)
plot(times/3600,masses);
xlabel('Transfer time [hr]');
ylabel('Propulsion and power mass [kg]');
legend('Ion','Hall');
```

propulsion_properties.m

```
function [properties] = propulsion_properties(p_system)
%PROPULSION_PROPERTIES  sets propulsion system properties
%
% [PROPERTIES] = PROPULSION_PROPERTIES(P_SYSTEM);
%   Returns a structure containing properties for the selected
%   type of propulsion system.  This structure is used as an
%   input to EP_ORBIT and PROPULSION_POWER.
%
% Inputs
%  P_SYSTEM    A string describing the type of propulsion system
%              to use.  Valid strings are 'ION', 'HALL', or 'PPT'.
%
% Outputs
%  PROPERTIES  A structure containing propulsion system
%              characteristics in the following fields:
%
%                 type       - descriptive string
%                 Isp        - specific impulse (s)
%                 eta        - efficiency
%                 lifetime   - lifetime (hours)
%                 th_mass    - thruster mass (kg/kW)
%                 ppu_mass   - power processor unit mass (kg/kW)
%                 misc_mass  - misc. prop. system mass (kg/kW)
%                 max_thrust - Maximum thrust allowed (N)
%                 min_thrust - Minimum thrust allowed (N)
%
%              An empty set is returned if P_SYSTEM is not a
%              valid string.

% William Nadir
% 16.851: Satellite Engineering
% 10/26/2003

properties = [];
switch lower(p_system)

   case 'ion'   % Xenon Ion propulsion system
      type        = 'ion';
      Isp         = 2800;     % specific impulse (s)
      eta         = 0.65;     % efficiency
      lifetime    = 10000;    % lifetime (hours)
      th_mass     = 4.5;      % thruster mass (kg/kW)
      ppu_mass    = 8;        % power processor unit mass (kg/kW)
      misc_mass   = 10;       % miscellaneous prop system mass (kg/kW)
      max_thrust  = .2;       % Maximum thrust allowed (N)
      min_thrust  = .01;      % Minimum thrust allowed (N)

   case 'hall'   % Xenon Hall propulsion system
      type        = 'hall';
      Isp         = 1600;     % specific impulse (s)
      eta         = 0.50;     % efficiency
      lifetime    = 7000;     % lifetime (hours)
      th_mass     = 2.5;      % thruster mass (kg/kW)
      ppu_mass    = 8;        % power processor unit mass (kg/kW)
      misc_mass   = 10;       % miscellaneous prop system mass (kg/kW)
      max_thrust  = .2;       % Maximum thrust allowed (N)
      min_thrust  = .08;      % Minimum thrust allowed (N)

   case 'ppt'   % Pulsed plasma (Teflon) propulsion system
      type        = 'ppt';
      Isp         = 1000;     % specific impulse (s)
      eta         = 0.07;     % efficiency
      lifetime    = 4000;     % lifetime (N*s)
```

```
      th_mass     = 120;      % thruster mass (kg/kW)
      ppu_mass    = 110;      % power processor unit mass (kg/kW)
      misc_mass   = 0;        % miscellaneous prop system mass (kg/kW)
      max_thrust  = .1;       % Maximum thrust allowed (N)
      min_thrust  = .001;     % Minimum thrust allowed (N)

   otherwise
      disp('Invalid input for p_system');
      return
end

properties.type      = type;
properties.Isp       = Isp;
properties.eta       = eta;
properties.lifetime  = lifetime;
properties.th_mass   = th_mass;
properties.ppu_mass  = ppu_mass;
properties.misc_mass = misc_mass;
properties.max_thrust= max_thrust;
properties.min_thrust= min_thrust;
```

ep_orbit.m

```
function [thrust, radii, per, eclipse, deltav] = ep_orbit(m0, prop, ttime, r0, r1)
%EP_ORBIT   Calculates circular orbit data for electric propulsion maneuvers
%
% [THRUST, RADII, PERIOD, ECLIPSE, DELTAV] = EP_ORBIT(MASS, PROP, TIME, R0, R1)
%
% Determines the constant tangential thrust needed to expand a circular orbit
% of radius R0 to radius R1 in the specified time TTIME.  Calculates the start
% and end times of each eclipse period, assuming an orbit in the plane of the
% ecliptic.  Solves for the constant direction (tangential), constant thrust
% case, rather than the optimal case, which is beyond the scope of this project.
%
% Inputs
%  MASS     [kg] Initial spacecraft mass
%  PROP     [-] Propulsion data structure (output of PROPULSION_PROPERTIES)
%  TIME     [s] Required orbital transfer time
%  R0       [m] Initial circular orbit radius
%  R1       [m] Final circular orbit radius
%
% Outputs
%  THRUST   [N] Constant thrust required to complete the maneuver
%  RADII    [m] History of orbit radii
%  PERIOD   [s] History of orbit period
%  ECLIPSE  [s] Eclipse enter and exit times for each orbit
%  DELTAV   [m/s] Delta V applied through each orbit

% constants
mu = 3.986004415e14; % [m^3/s^2]
R  = 6378136.3;      % [m] Earth radius
g  = 9.80665;        % [m/s^2] gravitational acceleration at Earth radius

% propellant specific impulse and exit velocity
Isp = prop.Isp;
c = Isp*g;

% check for invalid inputs
if (m0 <= 0)
   error('Initial mass must be positive');
end
if (Isp <= 0)
   error('Isp must be positive');
```

```
end
if (ttime <= 0)
   error('Transfer time must be positive');
end
if (r0 <= R)
   error(['Initial radius must be greater than Earth radius (' num2str(R) ' m)']);
end
if (r1 < r0)
   error('Final radius must be greater than or equal to initial radius');
end

% constant thrust required to perform this maneuver in the specified time
thrust = (m0*c/ttime)*(1-exp((sqrt(mu/r1)-sqrt(mu/r0))/c));

% initial values
timein  = 0;
timesun = 0;
orbit   = 0;
cdeltav = 0;
eclipse = [];
deltav  = [];
r = r0;
m = m0;

% iterate through the orbits, saving data
while (r<r1)
   orbit = orbit+1;

   % assume slow variation, so radius and mass are constant through each orbit
   radii(orbit) = r;
   mass(orbit)  = m;

   % time required to complete one orbit
   period = 2*pi*sqrt(r^3/mu);
   per(orbit) = period;

   % the delta V applied through this orbital period
   deltav(orbit) = 2*period*thrust/m;

   % angle off eclipsed sun line at which eclipse begins
   theta = asin(R/r);

   % time at which s/c crosses the eclipsed sun line
   if (orbit == 1)
      timesun = period*theta/(2*pi);
      timein  = 0;
   else
      timesun = timesun + period;
      timein  = timesun - period*theta/(2*pi);
   end

   % time at which s/c leaves eclipse
   timeout = timesun + period*theta/(2*pi);

   % save eclipse data
   eclipse(1,orbit) = timein;    % enter eclipse
   eclipse(2,orbit) = timeout;   % leave eclips

   % determine starting radius and mass for next iteration
   r = mu*(c*log(1-period*thrust/(m*c))+sqrt(mu/r))^-2;
   m = m-thrust*period/c;
end
```

propulsion_power.m

```
function [propulsion, power] = propulsion_power(thrust,orbit,properties,t_transfer)
% Here the basic information for the propulsion systems is input
%
% This module is used for sizing the electric propulsion system along with
% the required solar arrays and batteries for a circular orbit transfer
%
% Inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THRUST        = Constant thrust required to put complete the orbit
%                 transfer within the specific transfer time (N)
% ORBIT         = Data specifying times at which the satellite is in sun
%                 and eclipse during sprialing orbit transfer (times of
%                 entering and exiting eclipse)
% PROPERTIES    = a structure containing propulsion system properties,
%                 as created by the function PROPULSION_PROPERTIES.
% T_TRANSFER    = transfer time specified by user (seconds)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Outputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PROPULSION    = structure containing fields with propulsion system data
%                     m_thruster
%                     m_ppu
%                     m_misc
%                     m_propellant
%
% POWER         = structure containing fields with power system data
%                     A_cells_req = total required solar cell area (m^2)
%                     m_cells_req = total estimated solar array mass (kg)
%                     m_batt_req  = total battery mass required (kg)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% William Nadir
% 16.851: Satellite Engineering
% 10/26/2003

propulsion = [];
power      = [];

type        = properties.type;
Isp         = properties.Isp;
eta         = properties.eta;
lifetime    = properties.lifetime;
th_mass     = properties.th_mass;
ppu_mass    = properties.ppu_mass;
misc_mass   = properties.misc_mass;
max_thrust  = properties.max_thrust;
min_thrust  = properties.min_thrust;

% Here a check is performed to determine whether the input values of thrust
% and transfer time are acceptable for use with the chosen propulsion system
validity = 1;
switch type
   case {'ion', 'hall'}
      if t_transfer > lifetime*3600
         disp(['ERROR: transfer time (' num2str(t_transfer/3600) ...
               ' hr) exceeds life of propulsion system (' num2str(lifetime) ' hr).']);
         validity = 0;
      end

   case {'ppt'}
      if t_transfer > lifetime/thrust
         disp(['ERROR: transfer time (' num2str(t_transfer/3600)...
```

```
               ' hr) exceeds life of propulsion system ('
num2str(lifetime/thrust/3600)...
               ' hr) at ' num2str(thrust) ' N thrust.']);
         validity = 0;
      end
end

% is thrust within the allowed range?
if thrust > max_thrust
   disp('ERROR: thrust is over the capability range of propulsion system')
   validity = 0;
end
if thrust < min_thrust
   disp('ERROR: thrust is under the capability range of propulsion system')
   validity = 0;
end

% this lets all error messages display before returning
if (~validity)
   return
end

g = 9.81; % gravitational acceleration constant (m/s^2)

% This is the constant power required for the electric propulsion system to
% get the satellite from orbit A to orbit B
mass_flow_rate = thrust / (Isp * g); % (kg/s)

prop_power_continuous = (thrust^2) / (2 * mass_flow_rate * eta); % (Watts)

% Constants for direct energy power control system
Xe = 0.65;
Xd = 0.85;

eta_cells = 0.18;       % cell efficiency for GaAs cells (conservative = 18%)

I_degradation = 0.77;   % inherent degradation of solar cells

p_sun = 1358;           % solar input power density (W/m^2)

% Here we calculate the energy collection effectiveness of GaAs solar cells
% assuming no degradation over the time of orbit transfer (short time span)
power_per_area = eta_cells * p_sun * I_degradation; % (W/m^2)

% Initialize parameters for for loop to utilize 'orbit' vector data
[A, B] = size(orbit);

% The number of columns in the 'orbit' array is the number of orbits + 1
N_orbit = B - 1;

% Here the 'orbit' array is analyzed to determine the total eclipse and day
% times for each 'orbit' in the spiral orbit transfer
for I = 1:N_orbit

   eclipse_times(I) = orbit(2,I) - orbit(1,I); % (seconds)
   day_times(I) = orbit(1,I+1) - orbit(2,I);   % (seconds)

end

% Here the depth-of-discharge (DOD) for the battery is defined (NiH2
% battery)
DOD = .80;

% Here the battery charging efficiency is defined (90%)
charge_efficiency = 0.9;
```

```
% Here the battery specific energy density is defined for NiH2 batteries
sp_energy_density = 50; % (W*hr/kg)

% this for loop determines the power required for each portion of the
% spiral orbit transfer as well as the solar array area and battery mass
% required for each portion of the orbit transfer
for M = 1:N_orbit

   % P_sa is the required power for the propulsion system for one entire
   % orbit (Watts)
   P_sa(M) = (((prop_power_continuous * eclipse_times(M)) / Xe) + ...
      ((prop_power_continuous * day_times(M)) / Xd)) / day_times(M);

   A_cells(M) = P_sa(M) / power_per_area; % required area of solar arrays (m^2)

   m_cells(M) = 0.04 * P_sa(M); % estimated mass of solar arrays (kg)

   % Here the required battery capacity is calculated for one battery with
   % an efficiency of 0.9 for each eclipse during the spiral orbit
   % transfer (W*hr)
   Cr(M) = (prop_power_continuous * (eclipse_times(M)/3600)) / ...
      (DOD * charge_efficiency);

   % Here the required battery mass is determined
   m_batt(M) = Cr(M) / sp_energy_density; % (kg)

end

% required solar cell area to provide power for the orbit transfer (m^2)
A_cells_req = max(A_cells);

% required solar array mass (kg)
m_cells_req = max(m_cells);

% required battery mass to provide power for the orbit transfer (kg)
m_batt_req = max(m_batt);

m_thruster = th_mass * prop_power_continuous / 1000; % (kg)
m_ppu = ppu_mass * prop_power_continuous / 1000;     % (kg)
m_misc = misc_mass * prop_power_continuous / 1000;   % (kg)
m_propellant = mass_flow_rate * t_transfer;          % (kg)

% power output structure
power.A_cells_req       = A_cells_req;
power.m_cells_req       = m_cells_req;
power.m_batt_req        = m_batt_req;

% propulsion output structure
propulsion.m_thruster   = m_thruster;
propulsion.m_ppu        = m_ppu;
propulsion.m_misc       = m_misc;
propulsion.m_propellant = m_propellant;
```

References

[1] http://web.mit.edu/dept/aeroastro/www/labs/SPL/electric.htm, MIT Space Propulsion Lab Website, 2002.

[2] Course notes, AA420, University of Washington Dept. of Aeronautical and Astronautical Engineering, 1999.

[3] Martinez-Sanchez, Manuel, *Spacecraft Electric Propulsion – An Overview*, Journal of Propulsion and Power, Vol. 14, No. 5, 9/98-10/98, p. 690.

[4] ibid.

[5] Tajmar, Martin, *Advanced Space Propulsion Systems*, Springer Wien, New York, 2003, p. 76.

[6] Wertz, James, and Larson, Wiley, *Space Mission Analysis and Design*, 2nd Ed., Microcosm, Inc., 1997, p. 396.

[7] Sellers, Jerry Jon, *Understanding Space: An Introduction to Astronautics*, 2nd Ed., McGraw-Hill, 2000, p. 537.

[8] Tajmar, Martin, *Advanced Space Propulsion Systems*, Springer Wien, New York, 2003, p. 74.

[9] Wertz, James, and Larson, Wiley, *Space Mission Analysis and Design*, 2nd Ed., Microcosm, Inc., 1997, p. 397.

[10] ibid, p. 317.

[11] ibid, p. 403.

[12] ibid p. 404.

[13] ibid, p. 405.

[14] Kimbrel, Scott, *Optimization of Electric Propulsion Orbit Raising*, SM thesis, Massachusetts Institute of Technology, 2002.

Problem Set 5: Space Hotel Design: Preliminary Structural Design and Cost Estimation

Summary

A space hotel was designed based on a modular concept, the capability of modern launch vehicles, and various human factors. Finally, a cost model is used to estimate the cost of manufacturing, assembling, launching, and operating the space hotel. The software module created is used to investigate how the duration of each guest's stay at the hotel and the maximum occupancy of the hotel affect the total cost of the hotel.

Results

Based on the design constraints of the hotel design, the only variation in the structural design was the number of habitation modules around the outer ring of the hotel. This had a major cost impact on the hotel.

Another major cost impact on the hotel was the duration of the stay of each guest. The longer a guest stayed, the less frequently a vehicle would need to dock with the hotel to bring and take home guests. Launching people to the hotel is a significant cost.

Based on these constraints, it was found that the most economical space hotel design is one in which there are few guests and they stay for long durations of time. This is the lowest cost design for a space hotel.

Useful References

Human Factors

Conners, M.M., *et al., Living Aloft*, NASA, 1985.
Wieland, Paul., *Designing For Human Presence in Space*, NASA Marshall Space Flight Center , 1999, § 2.1.
Stafford, K.W., *et al., Advanced Life Support Systems Integration, Modeling, and Reference Missions Document*, NASA-Lyndon B. Johnson Space Center, 2001.
Sloan, James H., *Commercial Space Station Requirements*, AIAA-2000-5228, 2000.
Woodcock, Gordon, *Space Stations and Platforms*, Orbit Book Company, 1986.

The references listed above contain a great amount of useful information about what is required to support human life in space. Beyond these references, there are many AIAA papers that discuss manned space missions and what is required beyond unmanned space missions once humans are involved. Information such as shielding, food, water, and atmosphere is presented in these references. This information is key in estimating spacecraft volume and mass sizes.

Cost Modeling

Reynerson, Dr. Charles M., *A Space Tourism Platform Model for Engineering and Economic Feasibility*, 2000.

The reference listed above contains a useful cost model for estimating cost for space tourism missions. The main input to the cost model is mass, so if a realistic mass estimate can be made, it may be possible to calculate a reasonable first-order cost estimate based on the cost model presented in the reference.

Space Hotel Design: Preliminary Structural Design and Cost Estimation

Software Designed to Determine Preliminary Cost Estimate Based on Human Factors and Structural Design

16.851 Satellite Engineering

Massachusetts Institute of Technology, Cambridge, MA
November 2003

Motivation

Throughout history, people have been fascinated with exploring outer space. Until recently, only astronauts have had the privilege of being able to experience life in outer space. However, in 2001, the first space tourist, Dennis Tito, traveled to the International Space Station onboard a Russian Soyuz rocket.

The travels of Dennis Tito are just the beginning for space tourism. A new space tourism industry would be an entirely new commercial use of space and a huge potential new market. Space tourism may encourage other private investment in the use of space, which may in turn support significant future space exploration.

Problem Statement

Design a concept for a Space Hotel orbiting Earth. Create a CAD model of the hotel to "rough-out" the structural design as well as visualize the concept. The Space Hotel should provide all the amenities required by a tourist. These amenities should include gravity, power, food, water, and waste removal.

Design a MATLAB module to size the hotel structure as well as estimate the requirements for supporting human life. The user will set the number of hotel guests and the duration of the stay of each guest on the Space Hotel. These inputs will be the driving factors for the concept design. Based on the design concept for the hotel, estimate the costs involved in launching, assembling, and operating the hotel. The module will investigate cost with respect to the number of guests and the duration the stay of each guest.

Introduction

First, a conceptual design is created of the Space Hotel. A CAD model of the hotel is created with enough detail to present the rough conceptual design of the Space Hotel.

The spin rate of the hotel in order to produce artificial gravity is determined. Also, determine the other needs of the human guests of the hotel are determined. These needs will include electricity, food, water, waste removal, and crew.

Cost models are developed to estimate the cost for launch, assembly, and operation of the hotel. Using these cost models, the cost with respect to the number of guests and the duration of the stay of each guest is determined. In addition, trends are shown which illustrate how a Space Hotel can be operated in a cost-efficient manner.

Space Hotel Structural Design

Components

The Space Hotel consists of four main components: habitation modules, nodes, a center module, and connecting modules between the habitation modules and the center module. Figure 1 shows the configuration of the Space Hotel.

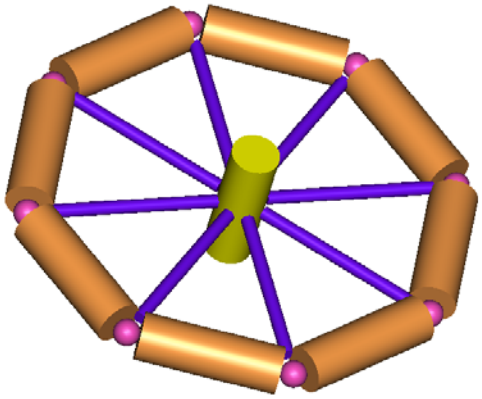

Figure 1 Isometric view of Space Hotel

All four components are visible in Figure 1, above. The large cylindrical components in the shape of a ring are the habitation modules. The spherical-shaped objects in between the habitation modules are the nodes. The long, thin, cylindrical components connecting the nodes to the center component can be seen as well. The large cylindrical component in the middle is the "center module."

Habitation Modules

The habitation modules are broken up into several compartments for hotel guests to live in. In addition, there is a large volume inside the module reserved for equipment to maintain the station as well as other storage space.

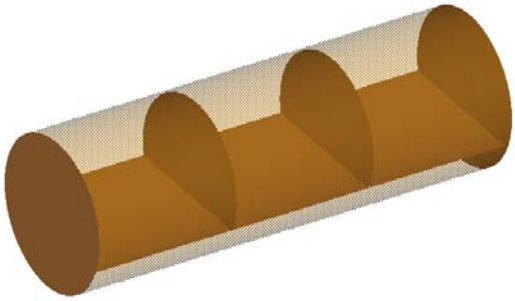

Figure 2 Habitation module internal compartments

The habitation modules are design in order to fit inside a realistic launch vehicle fairing. For the purposes of this project, an Atlas V, 5-meter fairing was chosen. The Atlas V is a likely launch vehicle to be used to launch components for a Space Hotel into orbit. The figure below shows one of the habitation modules fitting inside the launch vehicle fairing envelope.

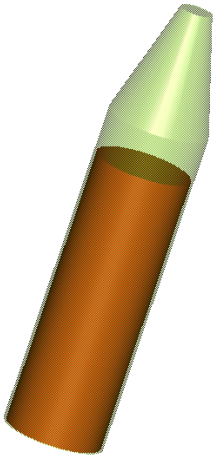

Figure 3 Habitation module inside Atlas V fairing

In addition, the size and shape of the habitation module is limited by the payload mass capability of the Atlas V launch vehicle. The estimated mass of one of the habitation modules, which is explained later in this paper, is roughly 10,000 kg. This mass is less than the payload capability of the Atlas V launch vehicle to many LEO orbits. For example, the Atlas V has the capability of launching 20,050 kg into a 185 km altitude, 28.5 degree inclination orbit.[1]

Nodes

The spherical shaped components between the habitation modules are simply interconnecting nodes which allow guests to transfer from one habitation module to another as well as to the center module. The spheres in the picture are simply placeholders for the components in a more detailed design.

Connecting Cylindrical Components

The long, thin, cylindrical components connecting the nodes to the center component are simply tubes which allow hotel guests to pass from the habitation module section of the hotel to the center module.

Center Module

The center module of the hotel is designed to roughly the same dimensions as the habitation modules in order for it to fit inside the same launch vehicle fairing. The purpose of the center module is to allow the hotel guests the experience of floating in a zero-gravity environment. Leisure activities can be held in this section of the hotel.

Life Support System

Our Space Hotel is situated in orbit around the earth, where the environment system for the passengers of the hotel is isolated except for the periodical supply by space transportation systems.

To support the life of passengers in the space environment, specific needs need to be met. The basic needs are the appropriate air composition, temperature, and humidity which should be maintained continuously. The consumables such as food and water should be supplied on passengers' demands. Wastes should be separately stored or removed periodically to maintain the optimal mass of the hotel and to keep the cleanliness.

In addition to those metabolic needs and effluents of humans, other supplies such as food preparation device, face/hand washing water, urinal flushing and etc should be also included. For a facility which stays in the

space environment for a long period, such as our Space Hotel, or International Space Station, functions such as O2 recovery and recycling of waste water and solid waste become important to keep the re-supply and storage expenses from becoming too high.

ECLSS

The current state-of-the-art system for long duration life support is embodied in the International Space Station (ISS) environmental control and life support system (ECLSS). ECLSS includes the function of providing a habitable environment, including clean air and water, plus solid waste processing, food processing, biomass production and thermal control, and supporting interfaces with other subsystems.

Considering the similar nature of the ISS and our Space Hotel, such as a durable structure for a long stay in the space environment, the choice was made to use ECLSS for our Space Hotel. In the following section, the details of each requirement and corresponding ECLSS device which satisfy the need will be presented.

Supporting System to meet Human Requirements

Interior Space

The ISS model is again utilized to estimate the personal volume necessary to experience a comfortable stay at our Space Hotel. In the limited space inside the structure, the astronauts rest, sleep (in a separate section), eat, shower, and also exercise on treadmills on the ISS. The volume of personal space on the ISS was calculated by dividing the volume of the habitat module by the number of the crewmembers, and was computed to be 645.6 (ft^3/person). This number was feasible considering that the minimum volume requirement calculated by Breeze (1961)[ii] was:

50ft^3/person (1-2 days)
260 ft^3/person (more than 1 or 2 months)
600 ft^3/person (for more than 2 months)

The mission length of the space station is between 3-6 months.

However, there are three concerns that we have to think about:

- The nature difference between ISS and Space Hotel. More space required for comfort
- Weightless state which allows passengers to utilize the space well is unavailable inside our Space Hotel. With the artificial gravity, more space is needed.
- Less space per person is needed as crew size increases

2. Thermal system

Although people can endure a relatively wide range of temperature and humidity conditions, the proper range in the habitat is important to maintain high work efficiency. For our hotel, it is crucial to provide comfort as a service. The ideal temperatures range from 18 to 27 C (65 to 80 F) and "ideal" humidity ranges from dew points of 4 to 16 C (40 to 60 F). Thermal management is divided into two systems, the internal and external thermal control systems. The former includes the avionic air assemblies which provide air-cooling for equipment, the common cabin air assemblies which control cabin air, condensate storage, and the water flow loops for hear transport. The external control system is included in the assessed cooling-mass penalty.

Food subsystem

Food will be provided in individual entrees from Earth. A mix of fresh, dehydrated, and full-water preserved, shelf-stable or frozen food will be used. This system required the significant amount of packaging. Supporting technology includes freezers and some food preparation equipment.

Air

In order to generate air conditions as close as the atmospheric configuration, oxygen, carbon dioxide, nitrogen, water vapor, trace contaminants, dust, and smoke particles are used as the components in the space habitats. Four separate systems, CO2 removal, CO2 reduction, O2 generation, and trace gas contaminants control systems, works to revitalize the air and maintain the quality of the air. Regenerative CO2 removal equipment based on molecular sieve technology, which does not require periodical replacement or storage space, is installed in the ISS ECLSS. CO2 Reduction is necessary to extract O required to generate O2. For a structure designed for the longer stay in the apace, the loss of the mass of CO2 leads to increased storage or re-supply requirements. O2 generation maintains sufficiently high oxygen partial pressure (21.4 kPa at near sea-level). Trace gas contaminants control systems is important in a closed structure like the Space Hotel because the volume is limited relative to containment sources. In addition, Atmosphere Control and Supply system is required to maintain proper composition and pressure of the air during the flight. The following chart shows the flow of air component in a recycle loop. This flowchart was extracted from a design report written by NASA.[iii]

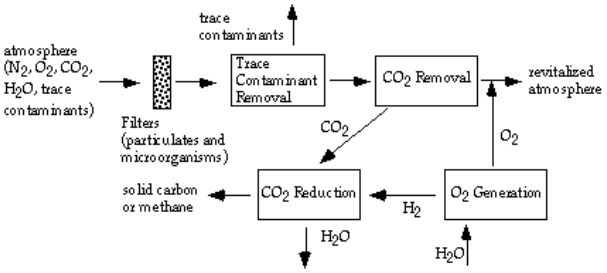

Figure 4 Atmosphere control and supply

Water

Ensuring a clean supply of potable water and water for bathing is essential. Water management consists of three parts, water storage and distribution, water recovery, and water quality monitoring.

For water recovery, urine is processed by vapor compression distillation, which claims 88 percent water recovery. The brine is either returned to Earth or dumped. The water processor deals with all unconfirmed water such as hygiene water, effluent from the vapor compression distillation, and condensation from dehumidification.

When recycled waste water is used, the potential for contamination is higher than when using stored water. Thus, process control water quality monitor provides water quality assurance. The following chart shows the flow of water management systems, extracted again from the report by NASA.[iv]

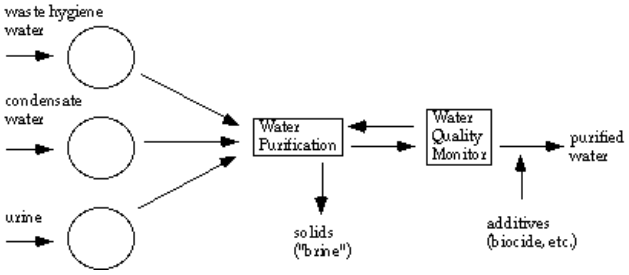

Figure 5 Water recovery and management

6. Waste Subsystems

The wastes generated on a space habitat can be classified into four general types: metabolic wastes consisting of moist solids including feces and vomit, other solid wastes, liquid wastes including urine and waste hygiene water, and gaseous wastes. For long duration missions this mass lost when the waste is dumped becomes prohibitive and methods are needed to recover to useable products as much mass as possible. On ISS, urine is recycled as explained in the Water section. Solid waste is stored and returned on the transfer vehicle or burned upon re-entry in an expendable re-supply vehicle. The toilet is also included under the subsystems.

Other Considerations

Although omitted in our design of the Space Hotel, we could aim to provide an even higher grade comfortable environment. For example, additional devices to decrease the level of odor and noise, or some decorative interior lighting could be installed. Other possibilities are to expand recreational facilities such as a plant growth facility, and improve safety devices such as fire detection and suppression systems.

Mass Estimation

Mass estimation for each component of human requirement was calculated using the actual data from the ISS. The optimization of ECLSS design in terms of the lowest launch cost was computed applying of Equivalent System Mass as related to the mass volume, power cooling and crew time needs.[v] Considering that masses of most of the components are proportional to the number of crews and the duration, the values shown in the following table is calculated by simple division with the number of crews and the duration time of ISS. These values are part of inputs for software module described in the following section.

Table 1 Mass estimation of human requirements

Consumables		[kg/CM-d]	
Supply	Air	0.84	
		(+ 0.29)	tank mass
	Food	1.37	
		(+ 0.24)	deposable packaging
	Thermal	0.003515	
	Water	7	drink, food preparation, hand/face washing, and urinal flushing
	Clothing	1.6	including EVA clothe
Waste	Waste	0.15	
Infrastructure		[kg/CM]	
ISS ECLSS technology		20366	including air tank, food freezers, CO2 removal device, EVA support

Software Module

Structural Sizing Module

Requirements

The MATLAB module *structure.m* determines the mass of the Space Hotel and the required spin rate of the hotel to maintain 1g of artificial gravity in the habitation modules of the spacecraft.

Description of Code

The code uses the inputs of the number of guests and the duration of the stay of each guest to calculate the numbers and sizes of the various components in the Space Hotel. The code calculates the masses of each component in the hotel and outputs the total structural mass of the hotel.

Constants

The first constant used in this module is the gravitational acceleration constant, g. It is input into the module as being equal to 9.81 m/s^2.

The remaining constants are the "free volumes" of the habitation and center modules in the Space Hotel. These volumes are determined from the fairing size of the Atlas V 5-meter fairing as well as the payload lift capability of the launch vehicle. Taking these constraints into consideration, the habitation modules were determined to have a "free volume," V_{hab}, of approximately 4500 ft^3.

The center module was determined to have a "free volume" of approximately 4500 ft^3.

Figure 6 Dimensions of habitation and center modules (inches)

Inputs

N_guests: This input is the total number of people living onboard the Space Hotel at any given time.

duration (days): This input is the time each person living on the Space Hotel will spend onboard before they head back to earth.

Outputs

str_mass (kg): This output is the total structural mass of the entire Space Hotel. This is a sum of all of the habitation modules, the interconnecting cylindrical modules, as well as the center structural module.

spin_rate (m/s): This output is the spin rate of the Space Hotel which is required to create artificial gravity of 1g in the habitation modules of the hotel.

Theory & Equations

The first step in determining the structural mass of the hotel is to determine how much volume each guest of the hotel will need during his/her stay. In 1961, Breeze noted that a person on a space station should need approximately 50, 260, and 600 cubic feet of volume for durations on the space station of 2, 30, and 60 days, respectively. This is discussed in the "Supporting System to Meet Human Requirements" section earlier in this paper.

Since this is a Space Hotel and should be somewhat luxurious and relaxing for the hotel guests, the numbers provided from Breeze are multiplied by a factor of 3 to result in volumes of 150, 780, and 1800 cubic feet for durations of 2, 30, and 60 days, respectively. In addition, since it has been estimated that the minimum volume for a space station is 700 cubic feet per person[vi] and it is unlikely that any person would stay on the hotel for a short time (i.e. less than one week), the volume per person estimated here is reasonable.

In order to determine guest required volumes for durations between the data points given, linear interpolation was done. This can be seen below in Figure 7.

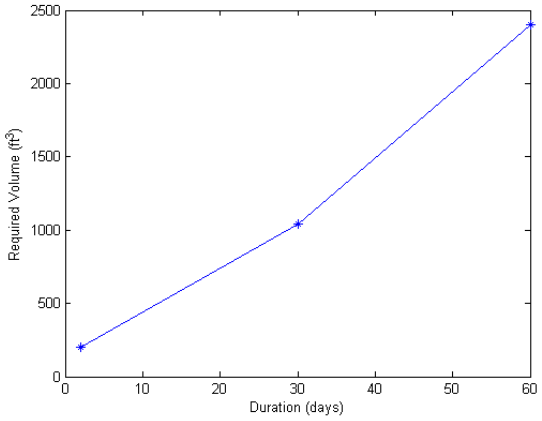

Figure 7 Linear interpolation of volume vs. duration

Next, the total required volume for the guests is calculated by multiplying the number of guests by the volume required per guest.

$$V_{total} = N\_guests * V_{guest} \quad (1)$$

The number of habitation modules is then determined from the following equation and then rounding up to the next highest integer.

$$N_{hab} = \frac{V_{total}}{V_{hab}} \quad (2)$$

Based on the sizing requirements given from the launch vehicle fairing constraints, the overall size of the habitation modules is then given. Next, the mass of the habitation modules is calculated. This is shown in the equation below.

$$m_{hab} = \rho_{hab} L\pi\left(r_{hab}^2 - (r_{hab} - t)^2\right) \quad (3)$$

In the above equation, the density of the habitation module is given to be 0.103 lb/in^3 for a material of Aluminum 2219. The wall thickness, *t*, of the module is given to be 0.4 inches.[vii]

The calculated mass of each habitation module is augmented with additional mass for welds, weld lands, and thickness tolerances. This adds an additional 1% to the mass of the habitation module.[viii]

Next, an additional 10% is added to the mass to take into account the internal, non-load-bearing structure of the habitation module. This is an extremely rough estimate.

In addition, the mass of the required shielding to protect the hotel from space debris is added to the weight calculation. A Whipple Shield[ix] is used for this purpose. This requires an additional thin Aluminum covering around the outside of the habitation module. This additional Aluminum piece is 0.080 inches thick. The design of the Whipple Shield can be seen in the figure below.

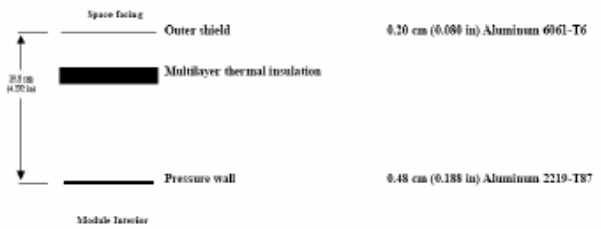

Figure 8 Example Whipple shield

Finally a factor of 1.5 is multiplied by the mass estimate due to the lack of design maturity and heritage.

The same procedure of calculating mass is done for the thin, cylindrical interconnecting structural members connecting the habitation modules to the center module. It is also done for the center module as well.

Finally, the spin rate of the hotel required to produce artificial gravity in the habitation modules is determined. The equation for centripetal acceleration is used. This is shown below.

$$a = \frac{V^2}{R} \quad (4)$$

In order to keep a simulated gravity of 1g in the habitation modules, the value of *a* is set to g, which is 9.81 m/s^2, and the value of R is the radius of the Space Hotel to the ends of each habitation module. See the figure below to illustrate this.

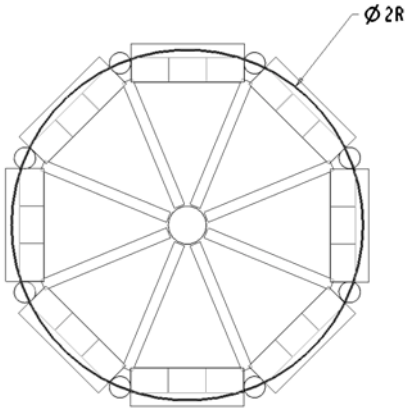

Figure 9 Radius of habitation module ring

In order to approximate the value of R for all sizes of the Space Hotel, the circumference of the ring of habitation modules is assumed to be equal to the lengths of the habitation modules plus the lengths of the nodes connecting the modules together. If this assumption is made, the radius can be determined form the equations 5a and 5b.

$$C = \pi D = N_{hab}(L + D_{node}) \quad (5a)$$

$$R = \frac{D}{2} = \frac{C}{2\pi} = \frac{N_{hab}(L + D_{node})}{2\pi} \quad (5b)$$

Rearranging equation 4 from above, the following equation produces the required spin rate, *V*, for the Space Hotel.

$$V = \sqrt{aR} \quad (6)$$

Cost Model Module

Requirements

The MATLAB module *cost_model.m* estimates the cost of the Space Hotel including the construction costs, logistics, and operation costs.

Description of Code

The code uses the inputs of the number of guests, the duration of the stay of each guest, the number of years in operation, and the weight of the Space Hotel to calculate total cost. The code calculates the individual costs such as the space structure, the ground support costs and logistics. These are summed together for total cost.

Inputs

Years(yrs): This input is number of years the Space Hotel expects to be in operation.

n_crew: This input is number of people staying onboard the Space Hotel at any time

duration (days): This input is the time each person living on the Space Hotel will spend onboard before they head back to earth.

w_f (kg): This input is the weight of Space Hotel structure.

Outputs

total_cos ($): This output is the total cost of constructing and operating a Space Hotel for a particular number of years, a maximum capacity of crew, and for a specified duration.

Theory & Equations

Various cost factors that result from space facility designs and an estimation of rough order of magnitude cost are included in this cost model. The required investment areas addressed include the space segment, launch vehicles, operations, and logistics.[x]

Space Segment Cost:

The space segment cost is calculated using:

$$S_c = S_{cf} * P_{cf} * R_{cf} * W_f \quad (7)$$

S_c: space segment cost ($)
S_{cf}: the price per kg of facility on orbit, for manned space programs the mean is 104 $K/kg
P_{cf}: the program cost normalized over the number of manned vehicles produced (non-dimensional)
R_{cf}: research, test, development, and engineering cost factor is used to compensate for new development cost. The R_{cf} should be 3 for new development programs, and 1 for a program based on existing hardware. We will use 2. (non_dimensional)
W_f: weight of facility (kg)

Launch Vehicle Cost:

The launch vehicle cost is calculated using:

$$L_c = L_{cf} * I_{cf} * W_f \quad (8)$$

L_c: launch vehicle cost ($)
L_{cf}: launch cost factor estimated using historical data and planned cost goals for future development, a mean of 15.2 $K/kg ($/kg)
I_{cf}: insurance cost factor, 1/3 of the launch vehicle cost, will use then 1.333. (non-dimensional)

Ground Operation and Support:

A good estimate for the purposes of this model is $80M per year for yearly operations and support costs.

Logistics:

The logistic cost was calculated using:

$$W_{cl} = 365 * ((\delta_{cs} + \delta_{crew} + \delta_{cg}) * (N_c/E_c) + \varepsilon) \quad (9)$$

W_{cl}: yearly crew logistics weight (kg)
δ_{cs}: equipment weight needed for crew support during the trip to and from orbit, assumed to be 2000 kg/person (kg/person)
δ_{crew}: weight/person (kg/person)
δ_{cg}: weight of gear/person (kg/person)

ε: consumption rate for the entire facility (kg/person-day)

$$W_{mm} = W_f * M_{mf} \quad (10)$$

W_{mm}: materials yearly delivery weight W_{mm} (kg)
M_{mf}: maintenance materials weight fraction, assumed to be 0.01 for this model (non-dimensional)

$$W_l = W_{cl} + W_{mm} \quad (11)$$
$$L_{gc} = L_{cf} * I_{cf} * W_l \quad (12)$$

L_{gc}: yearly logistics cost ($)

$$O_{sc} = N_y * (Y_{osc} + L_{gc}) \quad (13)$$

O_{sc}: total life cycle operations and support ($)
N_y: life cycle of station (yrs.)
Y_{osc}: ground operations and support cost, as above $80M/year ($)

$$\text{Total Investment} = S_c + L_c + O_{sc} \quad (14)$$

Results

Please refer to the figure below for the graph of results.

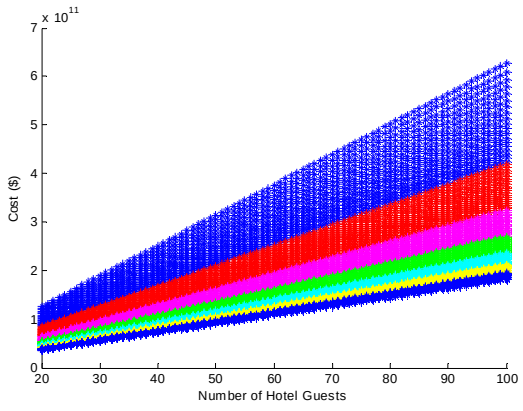

Figure 10 Cost vs. number of Space Hotel guests

In Figure 10, the different colors represent different durations of stay in increments of 20 days starting from the minimum of 30 days up to 170 days. The colors closer to the bottom of the graph represent long durations of stay. Here a trade was conducted in which the weight of the Space Hotel is calculated for the number of guests and duration and then cost is estimated based the above factors plus the weight of Space Hotel.

It can be seen in Figure 10 that the cost estimation is linear. This prevents any sort of minimization of the cost of building, launching, assembling, and running a Space Hotel. However, it can be seen that a Space Hotel business model which is designed around fewer guests staying for longer durations is a way to keep costs low.

Conclusion

A conceptual design for a Space Hotel was created and software was written to estimate the cost required to build, launch, assemble, and run the hotel. These costs were estimated based upon mass estimates for structure and environmental control systems required to support human life onboard the Space Hotel.

If we examine Figure 10, we notice that as duration of stay increases, costs decrease. This can be seen because the colored sections on the plot at the bottom are the longer guest stay durations. This makes sense because this lowers the logistical cost of shuttling people to and from the Space Hotel. Launches become prohibitively expensive if there are large numbers of guests and that are staying for short durations. As the number of guests increases, the smaller duration have a large effect on the cost. However, if the duration of stay is large, then duration has a smaller effect on costs because construction and ground support costs dominate. With long durations, it is possible to keep a large number of guests in space with a relatively small change in costs.

If one were looking to profit from this type of venture, it would be beneficial to require stays of up to 6 months and have 100 guests. The costs would be lower by a long duration and large revenues could possibly be seen due to the large number of guests staying at the hotel.

Future Work

A major area for future work would be to create a more detailed cost model for the Space Hotel. This enhanced cost model may result in a nonlinear distribution of costs, unlike the results shown in Figure 10. This may yield a minimum cost design for the Space Hotel.

In addition, a more detailed structural design of the Space Hotel could be created which would yield a more accurate mass estimate of the structure.

Appendix A: MATLAB source code

structure.m

```
% William Nadir
% 16.851 Satellite Engineering
% Problem Set 5
%
% Space Hotel Structural Design Software Module
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% INPUTS
%
% N_guests  = Number of people staying on board the Space Hotel at any time
% duration  = Duration of stay for guests of the Space Hotel (days)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% OUTPUTS
%
% str_mass  = Mass estimate of structure of Space Hotel (kg)
% spin_rate = Spin rate of hotel to produce artificial gravity in
%             habitation modules (m/s)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

function [str_mass,spin_rate] = structure(N_guests, duration)

% Here the volume required per passenger onboard the hotel is determined
time        = [2 30 60]; % duration of stay (days)
req_volume  = [150 780 1800]; % required "free volume" per passenger (ft^3)

if duration <= time(1)
    vol = req_volume(1);
elseif duration > time(1) && duration < time(2)
    vol = interp1(time,req_volume,duration,'linear');
elseif duration == time(2)
    vol = req_volume(2);
elseif duration > time(2) && duration < time(3)
    vol = interp1(time,req_volume,duration,'linear');
elseif duration == time(3)
    vol = req_volume(3);
end

% Input habitation module volume here
hab_free_volume = 4500; % (ft^3)

% Calculate how much free volume is required for the hotel guests
total_free_vol = N_guests * vol; % (ft^3)

% Determine how many habitation modules are required to house all the
% guests (rounding up)
N_habs = ceil(total_free_vol / hab_free_volume);
```

```
% Here we determine the mass of each hotel component
bulkhead_thickness = .4; % (in)
rho = .103; % Al 2219 (lb/in^3)

% Habitation modules
hab_dia     = 170; % (in)
hab_length  = 500; % (in)
node_dia    = 96;  % Nodes at the ends of each habitation module (in)

hab_mass = rho * hab_length * pi * (((hab_dia/2)^2) - ...
        (((hab_dia - 2*bulkhead_thickness)/2)^2));

% Here the mass for welds (1%) and internal structure (10%) are added
hab_mass = hab_mass + (hab_mass * .01) + (hab_mass * .1);

% Here the Whipple Shield mass is determined (radius is 4.2" larger than
% module)
whipple_density     = .0975; % Al 6061 (lb/in^3)
whipple_thk         = .08; % (in)
whipple_mass_hab    = whipple_density * hab_length * pi * ...
                    ((((hab_dia/2) + 4.2)^2) - ...
                    ((((hab_dia - 2*whipple_thk)/2) + 4.2)^2));

% Final habitation module mass plus 1.5 factor since calculations are very
% rough
hab_mass = (hab_mass + whipple_mass_hab)* 1.5; % (lb)

% Determine dimensions of overall hotel structure
% Assume that the circumference of the habitation module ring is roughly
% equivalent to the sum of the lengths of the habitation modules
hotel_dia = (N_habs * (hab_length + node_dia)) / pi; % (in)

% Interconnecting cylidrical structural elements
ic_dia      = 48; % (in)

% Assume the lengths of the interconnecting tubes is roughly equivalent to
% the radius of the hotel ring
ic_length   = hotel_dia / 2; % (in)

ic_mass = rho * ic_length * pi * (((ic_dia/2)^2) - ...
        (((ic_dia - 2*bulkhead_thickness)/2)^2));

% Here the mass for welds (1%) and internal structure (10%) are added
ic_mass = (ic_mass * .01) + (ic_mass * .1);

% Here we determine the Whipple Shield mass for the IC modules
whipple_mass_ic = whipple_density * ic_length * pi * ...
                (((ic_dia/2) + 4.2)^2) - ...
                ((((ic_dia - 2*whipple_thk)/2) + 4.2)^2));

% Final habitation module mass plus 1.5 factor since calculations are very
% rough
ic_mass = (ic_mass + whipple_mass_ic)* 1.5;

% Here we assume the center cylinder mass is equivalent to that of a habitation
% module
center_module_mass = hab_mass; % (lb)
```

```
% Here the total structural mass is calculated (lb)
str_mass = (hab_mass * N_habs) + (ic_mass * N_habs) + center_module_mass;

% Convert to kilograms from pounds
str_mass = str_mass * .454;  % (kg)

% Determine spin rate required to produce artificial gravity in habitation
% modules
g = 32.2; % (ft/s^2)
spin_rate = sqrt(g * (hotel_dia/2)); % (ft/s)

% Convert to meters/sec
spin_rate = spin_rate * .3048;  % (m/s)
```

cost_model.m

```
% Christopher Hynes
% 16.851 Satellite Engineering
% Problem Set 5
%
% Space Hotel Cost Software Module
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
% INPUTS
%
% years  = number of years in operation
% n_crew = Number of people staying on board the Space Hotel at any time
% duration  = Duration of stay for guests of the Space Hotel (days)
% w_f = weight of structure (kgs)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
% OUTPUTS
%
% total_cost = amount (dollars) of total investment required for
% construction, logistics, and operation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
function total_cost = cost_model(years, n_crew, duration, w_f)

%Please see paper for explanations
s_cf = 104e3; %space segment cost factor [$/kg]
p_cf = 1.0; %
r_cf = 2.0; % research and development cost factor

s_c = s_cf*p_cf*r_cf; %space cost

l_cf = 15.2e3; %launch cost factor
i_cf = 1.33; %insurance cost factor

l_c = l_cf*i_cf*w_f; %launch cost

y_osc = 80e6; %yearly operation cost

delta_cs = 2000; %crew support specific weight [kg/person]
```

```
delta_crew = 170; %crew specific weight [kg/person]
delta_gear = 72; %crew specific gear weight [kg/person]

consumption_rate = 9.453515; %rate of consumption [kg/person]

w_cl = 365*((delta_cs + delta_crew + delta_gear)*(n_crew/duration) + consumption_rate*n_crew);  %yearly crew
logistics weight

m_mf = 0.01; %maintenance materials weight fraction

w_mm = w_f*m_mf; %materials yearly delivery weight

w_l = w_cl + w_mm; %logistics weight
l_gc = l_cf*i_cf*w_l; %logistics cost (per year)

o_sc = years*(y_osc + l_gc);  % operational cost

total_cost = s_c + l_c + o_sc;
```

cost_modeltest.m

```
% Christopher Hynes
% 16.851 Satellite Engineering
% Problem Set 5
%
% Space Hotel Cost Software Module
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
% INPUTS
%
% years  = number of years in operation
% n_crew = Number of people staying on board the Space Hotel at any time
% duration  = Duration of stay for guests of the Space Hotel (days)
% w_f = weight of structure (kgs)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
% OUTPUTS
%
% total_cost = amount (dollars) of total investment required for
% construction, logistics, and operation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
% function total_cost = cost_model(years, n_crew, duration, w_f)

N_guests_min = 20;
N_guests_max = 100;
duration_min = 30;
duration_max = 180;

cost_matrix = zeros(N_guests_max - N_guests_min + 1, duration_max - duration_min + 1);

for N_guests = N_guests_min:N_guests_max
    for duration = duration_min:duration_max

        [str_mass,spin_rate] = structure(N_guests, duration);
        total_cost = cost_model(10, N_guests, duration, str_mass);
```

```
        cost_matrix(N_guests - N_guests_min + 1,duration - duration_min + 1) = total_cost;
    end
end

figure(1)
for i = 1:duration - duration_min + 1
    hold on
    duration = duration_min + i;
    if duration < 30
        plot([N_guests_min:N_guests_max],cost_matrix(:,i),'k*');
    elseif duration >= 30 && duration < 50
        plot([N_guests_min:N_guests_max],cost_matrix(:,i),'b*');
    elseif duration >=50 && duration < 70
        plot([N_guests_min:N_guests_max],cost_matrix(:,i),'r*');
    elseif duration >=70 && duration < 90
        plot([N_guests_min:N_guests_max],cost_matrix(:,i),'m*');
    elseif duration >= 90 && duration < 110
        plot([N_guests_min:N_guests_max],cost_matrix(:,i),'g*');
    elseif duration >= 110 && duration < 130
        plot([N_guests_min:N_guests_max],cost_matrix(:,i),'c*');
    elseif duration >= 130 && duration <= 150
        plot([N_guests_min:N_guests_max],cost_matrix(:,i),'y*');
    elseif duration >= 150 && duration <= 170
        plot([N_guests_min:N_guests_max],cost_matrix(:,i),'b*');
    end
end
ylabel('Cost ($)')
xlabel('Number of Hotel Guests')
```

References

[i] Isakowitz, Steven J., *International Reference Guide to Space Launch Systems*, AIAA, 1999, p. 55.
[ii] Conners, M.M., *et al., Living Aloft*, NASA, 1985, p60.
[iii] Wieland, Paul., *Designing For Human Presence in Space*, NASA Marshall Space Flight Center , 1999, § 2.1.
[iv] ibid, § 2.4.
[v] Stafford, K.W.*, et al*., *Advanced Life Support Systems Integration, Modeling, and Reference Missions Document*, NASA-Lyndon B. Johnson Space Center, 2001.
[vi] Sloan, James H., *Commercial Space Station Requirements*, AIAA-2000-5228, 2000, p. 5.
[vii] Woodcock, Gordon, *Space Stations and Platforms*, Orbit Book Company, 1986, p. 145.
[viii] ibid, p. 145.
[ix] Graves, R., *Space Station Meteoroid and Orbital Debris Survivability*, AIAA-2002-1607, 2002, p. 8.
[x] Reynerson, Dr. Charles M., *A Space Tourism Platform Model for Engineering and Economic Feasibility*, 2000.

Problem Set 6: Design of an Artificial Gravity Mars Mission

Summary

The purpose of this problem set was to investigate four possible vehicle designs to allow for a crew traveling to Mars to experience artificial gravity for the duration of the transit to and from Earth. The four designs investigated were a toroidal-shaped monolith structure, a two-module EMFF design, a multi-module, tethered spacecraft, and a two-module tethered spacecraft. The main focus of the project was on structures, human factors, and cost. Areas such as power systems and propulsion were investigated in a rough manner in order to have realistic numbers for those parts of the spacecraft.

A manned mission to Mars onboard a spacecraft which provides artificial gravity during the transit to Mars would allow the crew to be immediately productive as soon as they arrive at Mars. Without artificial gravity, the musculoskeletal deterioration of the crew during the transit to Mars would render the crew incapable of most tasks on Mars due to the .38g of gravity.

The cost model to determine the total program cost for each vehicle was made possible by a cost model that has been developed for just this purpose. In addition, the technology readiness levels (TRLs) of each spacecraft were estimated in order to scale the program costs to realistic values.

Results

It was found that the ideal crew size was approximately 9 people for a mission to Mars. This requirement drove the internal volumes of the vehicles which in turn sized the structure and determined the program cost. In addition, human factors such as gravity gradient and cross-coupled acceleration effects drove the sizing of the spacecraft.

Based on all of these issues, several MATLAB modules were written to size the spacecraft and estimate the program costs for each. The results showed that the toroidal-shaped monolith was the cheapest vehicle while the multi-module tethered spacecraft was the most expensive. This is most likely due to the relatively high TRL of the monolith vehicle. It was also found that the multiple-tethered spacecraft had the highest mass of the four designs. In fact, the other three designs had fairly similar masses but the multiple-tethered spacecraft had more than double the total mass.

Crew size was also varied to see its effect on the total program cost of each vehicle design. The cost for the multiple-tethered spacecraft increased at a significantly higher rate than that of the other three designs. Also, at a crew size of five and greater, the toroidal spacecraft is the cheapest of the four designs.

Finally, the cost of each design increased as the artificial gravity requirement was raised.

Useful References

Larson, Wiley, *Human Spaceflight: Mission Analysis and Design*, McGraw-Hill, 1999.

For this project, I found many useful human factors references dealing with humans in space for long duration missions. The most useful of all the references was HSMAD. This book contains information about all types of human spaceflight, including long duration manned missions to Mars.

In addition to using HSMAD, several useful AIAA papers were found which dealt with long duration human spaceflight. The most useful of these papers dealt with the implications of crew size on long duration missions in extreme environments. This paper investigated the interactions of the crew on many space missions as well as long duration expeditions to places such as the Antarctic. The paper found that more heterogeneous and larger crews ended up working better together than smaller, more homogeneous crews. This reference is shown below.

Dudley-Rowley, Marilyn, et. al., *Crew Size, Composition, and Time: Implications for Exploration Design,* AIAA 2002-6111, AIAA, 2002.

Design of an Artificial Gravity Mars Mission

Mission Design Considering Human Factors, Structures, and Cost

16.851 Satellite Engineering

Massachusetts Institute of Technology, Cambridge, MA
December 2003

Motivation

Mars exploration is one of the main directions of NASA. One overarching goal is to someday have a manned mission to Mars. This is the next major step for space exploration beyond the moon and out into the solar system.

A manned mission to Mars poses several significant technological challenges for engineers. One such challenge is to minimize the physiological impact on the astronauts during prolonged spaceflight. A possible solution to this is using artificial gravity. Once the crew arrives on Mars, they will almost immediately be able to begin useful scientific research, rather than spending significant time rehabilitating due to problems like bone decalcification. Using current or planned technology, artificial gravity almost certainly requires some sort of spinning spacecraft.

The extended mission to Mars also poses psychological challenges for the crew. The psychological well-being of the crew may depend on the number of astronauts, the gender makeup of the crew, the ages of the crew members, and the "free volume" available per astronaut. Many of these human factors will contribute to the design of the spacecraft used to transport humans to Mars.

Problem Statement

Create a tool to evaluate the feasibility of an artificial gravity Mars mission. The tool should output the cost for four designs: a large monolithic station, a tethered multi-spacecraft station, a tethered two-spacecraft system, and an EMFF system (see Figure 1). In addition, the tool will determine how Mars mission inputs such as number of crew members affect the design of each system. A systems engineer using the tool will be able to vary these parameters to fit a launch or cost constraint.

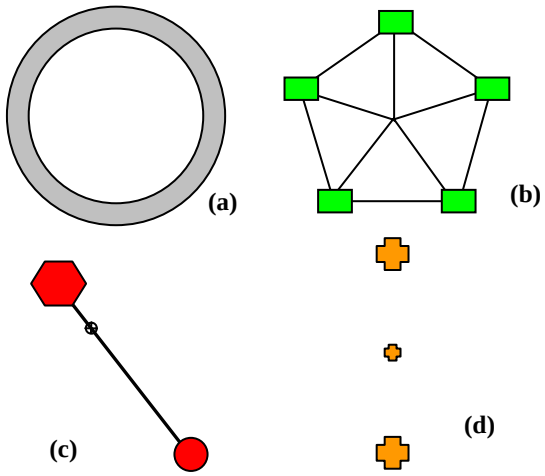

Figure 1 Four designs: (a) monolithic station (Toroid), (b) multi-spacecraft tethered, (c) two-spacecraft tethered, and (d) EMFF.

Introduction

This study analyzes human, structural, and cost aspects of the various spacecraft types in order to determine their feasibility. In addition, the power and propulsion systems of these spacecraft are modeled. The number of crew members is treated as a variable in order to analyze the effects of this parameter on the designs.

Some of the human factors that are considered, in addition to the area issues, are requirements for reducing motion sickness (since the spacecraft will be spinning) and support systems and maintenance (such as food, waste management, thermal and power needs etc.).

The structural aspects are dictated by the design configuration and human factors. For instance, in order to prevent motion sickness, a minimum distance to the spin axis is required. Similarly, the space/area needs for the crew are imposed requirements on the habitable volume of the spacecraft. In addition, volume must be allocated for the Mars science payload, equipment, and spacecraft subsystems. This tool assumes an Earth

return vehicle already exists on Mars for the astronauts to use. The design of each particular system has unique structural requirements.

Total cost of each type of spacecraft is evaluated based on the structural and mass requirements of the design. Factors such as launch and operations are included.

Figure 2 summarizes the map of how the various subsystems relate to each other.

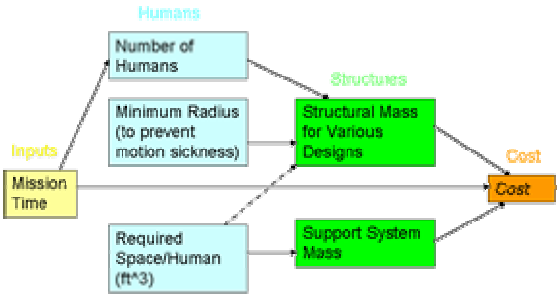

Figure 2. Relationships between the various subsystems of the tool.

Crew Size and Composition

A long journey such as the one to and from Mars would put any crew under extreme stress. The size and composition of the crew for a manned Mars mission are factors that can be controlled in such a way as to minimize the stress during such a mission.

The human-human interface is the most important with respect to the psychological and sociological aspects of the extreme environment of a manned Mars mission. The success of the mission depends on the ability of the crew to effectively work together to accomplish their mission objectives.

Based on an ongoing study of this human-human interface in extreme environments, several important observations have been documented. First, larger crews tend to have lower rates of deviance and conflict. Second, deviance and conflict tend to decline with increasing length of mission. Third, heterogeneous crews have lower rates of deviance and conflict.[1]

Although it was found that a larger crew had fewer incidents of deviance and conflict, a maximum value for crew size needs to be set. In the study previously mentioned, it was found that the least dysfunction of any crew studied was a crew of nine people.[2] This favorable crew size of 9 and the fact that a manned mission to Mars could take as long as nine months, a crew size of nine was set for a mission length of nine months.

The other end of the spectrum, a shorter mission, needs to have a limit for crew size as well. As the duration of the mission gets shorter, the "extremeness" of the environment decreases. This is because the crew knows that they will not be as far from home as they might be on a longer mission and they are closer to reality than a nine-month expedition to Mars. This lessening of the "extremeness" of the trip makes it plausible for a crew of two members to run a mission for duration of approximately one month. Several manned missions to Mars even suggest using a crew of two.[3] Therefore, it is reasonable to assume a crew of two could handle a month-long space mission.

Based on these two limits, linear interpolation is used to estimate the crew sizes for mission durations between one and nine months. However, since a worst-case scenario is assumed in which the crew must stay in the vehicle and return to Earth without landing on Mars, the mission durations are doubled for the same estimated crew size. This effectively places a cap of 9 as the crew size for a mission to Mars using a Hohmann transfer (roughly 9 months transit time each way). These crew size estimates are shown below in Figure 3.

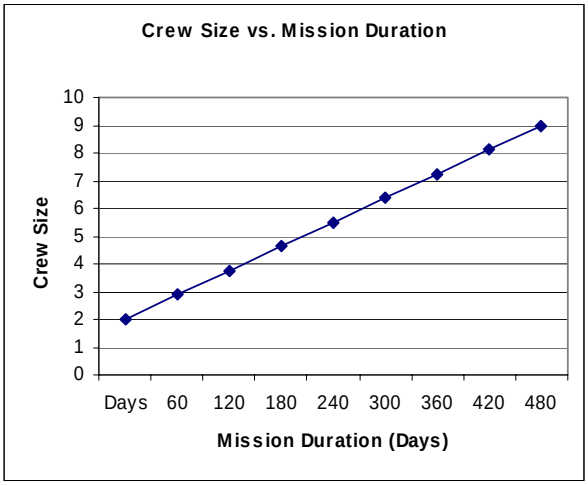

Figure 3 Crew size vs. mission duration

In addition to the crew size, the gender, ethnic, and cultural makeup of the crew plays a large role in the performance of the crew during the mission. It was found that more heterogeneous crews begin a mission with some deviance, conflict, and dysfunction, but these problems seem to decline as the mission progresses. On the other hand, a more homogeneous crew tends to begin a mission without much, if any, deviance, conflict, or dysfunction, but these problems tend to increase throughout the duration of the mission.[4] Therefore, a heterogeneous crew, most likely half men

and half women, with a mix of various ethnicities and cultures, would tend to produce a more effective crew for an extreme mission such as a manned mission to Mars.

Human Factors

Interior "Free" Space for Crew

The long duration of a mission to Mars requires that extra comfort be given to the crew than that given to astronauts on a one or two week mission to low Earth orbit. Significant comfort can be given to the crew in the form of increased interior volume to use for work and leisure activities. This would result in improved mental health of the crew at the time of their arrival at Mars.

Breeze (1961) estimated that a crew on a space mission would require a minimum volume of 600 ft^3 per crew member for space missions longer than two months.[5] Sloan,[6] on the other hand, estimates the minimum volume per crew member for life on a space station to be approximately 700 ft^3. Being conservative, a value of 700 ft^3 is assumed for the free volume required per crew member for a manned Mars exploration mission.

Life Support System Equipment Volume and Mass

Crew Systems

The crew systems onboard the spacecraft for a manned mission to Mars contain equipment such as galley and food system, waste collection system, personal hygiene, clothing, recreational equipment, housekeeping, operational supplies, maintenance, sleep provisions, and health care. HSMAD contains a detailed breakdown on the mass and volume requirements of crew systems specifically designed for a manned Mars mission.[7] By dividing the numbers provided in HSMAD by the estimated mission duration and specified crew size, a normalized crew systems mass and volume per crew member per day can be determined. These values are shown below in Table 1.

Table 1 Crew systems normalized volume and mass.

Crew Systems Mass (kg/CM-d)	7.55
Crew Systems Vol. (ft^3/CM-d)	1.51

ECLSS Atmosphere Management

The Environmental Control Life Support System (ECLSS) manages the air, water, waste, and other systems onboard the spacecraft which support human life in space. The portion of the ECLSS which manages the atmosphere onboard the spacecraft utilizes physio-chemical (P/C) technology in order to remove carbon dioxide from the air, control trace contaminants, and provide oxygen to the crew. An atmosphere management system suggested by HSMAD is used for the purposes of this study. This suggestion is a triple-redundant system of three different types of P/C atmospheric management systems.

The three types of P/C systems used in this manned Mars mission spacecraft are 4BMS (4-bed molecular sieve), TCCS, and Sabatier P/C atmosphere management systems.[8] A basic flow chart of the method used to manage the atmosphere on board the spacecraft is shown in the figure below.

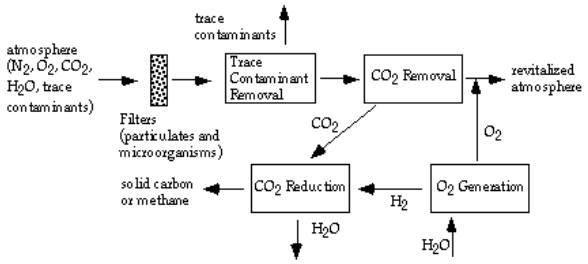

Figure 4 Atmosphere control and supply[9]

Based on the mass and volume requirements provided in HSMAD, the mass per crew member of these environmental support systems could be estimated. The three types of atmospheric management systems were summed and multiplied by a factor of three for redundancy. These values are shown below in Table 2.

Table 2 ECLSS atmosphere management mass and volume per crew member[10]

ECLSS Atm. Mass (kg/CM)	255
ECLSS Atm. Vol. (ft^3/CM)	35.3

ECLSS Water Management

Based on the manned Mars mission design example in HSMAD, the ECLSS water management system design for this project was estimated. HSMAD assumes a P/C water management system of vapor compression distillation (VCD) for use on the spacecraft. A basic flow chart detailing the process of water recovery and management is shown in the figure below.

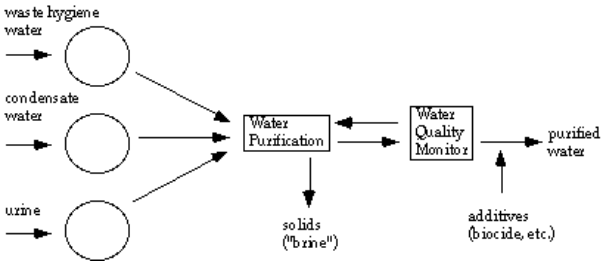

Figure 5 Water recovery and management[11]

This technology requires a mass of approximately 25 kg per crew member and a volume of 3.53 ft^3 per crew member.[12] A redundancy of two water management systems is assumed,[13] which brings the total mass per crew member to 50 kg and total volume per crew member to 7.1 ft^3.

Artificial Gravity

A manned mission to Mars requires that the crew be subjected to the space environment for a significant period of time. A travel time of nearly one year would result in significant musculoskeletal deterioration of the crew members if the transit period were completely zero-g.[14] This would result in the crew members being physically incapable of performing much work, if any, when they arrive at Mars.

The downtime as a result of the crew's required physical rehabilitation would dramatically reduce the available time on Mars for the crew to perform valuable research activities.

In order for the crew to be productive when they reach Mars, an artificial gravity of 0.38g, the magnitude of gravity on Mars, is created on board the spacecraft during the transit to Mars. The artificial gravity is set to Mars gravity because it is unnecessary to provide artificial gravity of 1g if the crew will need to adjust to Mars gravity of 0.38g when they arrive. Also, a smaller artificial gravity requirement reduces the propellant required to spin-up and spin-down the spacecraft, as well as the structural requirements on the spinning spacecraft (and tethers).

Gravity Gradient, Coriolis, and Cross-coupled Acceleration Effects

Due to the fact that the centrifugal acceleration resulting from the spin of the spacecraft varies with radial distance from the spin center, a different level of gravity will exist between various levels of the structure as well as throughout the human body. If this gravity gradient is too large, it could become uncomfortable for the crew members.[15]

In addition, crew members will experience pseudo weight changes depending on their direction of motion due to radial and tangential Coriolis effects. When walking parallel to the spacecraft spin axis, crew members will feel heavier when walking in the direction of the spin and lighter when walking in the opposite direction. Tangential Coriolis effects will be felt by crew members walking moving radially about the spacecraft (possible in the Toroid spacecraft). They will feel a push in the direction of the spacecraft spin when climbing towards or away from the spacecraft's center of motion.

Another potential uncomfortable result of the spinning spacecraft, cross-coupled angular acceleration effects, can be felt by crew members. This occurs when a crew member moves his/her head in directions transverse to the axis of rotation and the direction of flight of the spacecraft. Interior design of the spacecraft may help to alleviate this problem. In addition, researchers at Slow Rotating Room in Pensacola, Florida, found that human test subjects in a room rotating at speeds up to 10 rpm could be trained to adapt to the rotating environment.

These potential impacts to the human crew for the manned Mars mission result in design requirements in order to minimize the impacts of these potential problems and create a safer, healthier, and more enjoyable environment for the crew during their long journey to Mars. The two requirements imposed on the spacecraft design are a maximum spin rate and a minimum spin radius.

Stone (1970) and Thompson (1965) recommend a rotation radius greater than 14.6 meters and spin rate less than 6 rpm, while Shipov (1997) thinks a minimum radius of 20 meters is appropriate. In order to be conservative, a minimum spin radius of 30 meters and a rotation rate of 6 rpm were used for the purposes of this project.

Radiation Design Considerations

During the journey from Earth to Mars, the crew will not enjoy the protection of the Earth's atmosphere from high energy particles from the Sun. Solar particle events (SPEs) cause large numbers of these high energy particles to emanate from the Sun. These particles may impact the spacecraft and could result in harmful health effects for the crew.

Background radiation in space, such as galactic cosmic rays, may also affect the crew during transit to Mars.

In order to design a spacecraft to provide reasonable protection for the crew from radiation, the thickness of the aluminum hull of the spacecraft must be designed with a minimum thickness. This thickness is

determined from the maximum allowable radiation dose for crews. This is given to be 1 Gy.[16]

This allowable radiation exposure for crews is compared to the dose the crew would receive based on the aluminum hull thickness to obtain the minimum thickness. The data table used to make this decision is shown below.

Table 3 Radiation dose from an unusually large solar particle event

Shielding Depth (cm Al)	Dose (Gy)
0.5	4.68
1.0	1.95
1.5	1.02
2.0	0.59
2.5	0.37

Since an acceptable dose for the crew is 1 Gy, a minimum hull thickness of 1.5cm of aluminum is chosen for this spacecraft.

Spacecraft Power

In order to obtain an estimate for the power system for the Earth-Mars cruise spacecraft, a rough approximation of spacecraft power per crew member was required. Several opinions exist as to exactly how much power per crew member is required for the Earth-Mars cruise phase of a manned Mars mission.

HSMAD assumes 20kW for a six-crew member mission to mars. This normalizes to 3.33kW per crew member.[17] In addition, Sloan notes that 2kW per crew member is required purely for life support.[18]

It is realistic to assume that more power will be required than the minimum for life support. Research and other activities will require additional power beyond life support. Therefore, it is assumed for the purposes of this project that 3.3kW is required per crew member for the Earth-Mars cruise phase of a manned Mars mission.

Structures

Cylinder

A cylindrical pressure vessel is used as the structure for the tethered multiple spacecraft, two tethered spacecraft and EMFF spacecraft designs. A cylindrical habitat module was chosen because they have a high TRL and can fit easily into a launch vehicle. The diameter of the launch vehicle was used as the diameter of the cylinder. Given the required volume and the number of spacecraft in the array, the length of the cylinder was then determined. The volume is equally distributed among the spacecraft. The material selected for the cylinder was Aluminum 606a-T6, based on the design for a habitat module in HSMAD (Chapter 21). The thickness of the cylinder can be determined by the Hoop stress (since the hoop stress is greater than the longitudinal stress)

$$f_h = \frac{pr}{t} \le F_{tu} \tag{0.1}$$

where f_h is the Hoop stress, p is the pressure, r is the radius of the cylinder, t is the thickness of the cylinder, and F_{tu} is the allowable tensile ultimate stress for aluminum. The thickness is set as 0.015 cm if it is found to be less than that because of radiation shielding requirements. The maximum internal pressure of 0.1096 times a safety factor of 2 is used as the pressure inside the cylinder (based on HSMAD). Finally the mass of the cylinder including the two ends is found by

$$mass = 2 \cdot \pi r^2 t \rho_{AL} + 2rlt\rho_{AL} \tag{0.2}$$

The dry mass of each spacecraft includes the structural mass plus the solar array mass (see Power Module section) and the life support equipment mass.

Toroid

The toroid for the monolithic system is found in a similar fashion as the cylindrical case. The inner radius of the toroid, r_t, is found by the following

$$V = 2\pi^2 r_t^2 R \tag{0.3}$$

where V is the required volume and R is the radius of rotation. The minimum radius is set as 3 feet (0.9144 m) if the radius, r_t, is found to be less than that. The thickness is found using the hoop stress requirement. The mass of the toroid is found by the following

$$mass_{toroid} = (2\pi r_t)(2\pi R)t\rho_{AL} \tag{0.4}$$

The dry mass for the monolithic system includes spacecraft includes the toroid mass plus the solar array mass (see Power Module section) and the life support equipment mass.

EMFF Coil Mass

The superconducting EMFF coils are used to rotate the two habitat modules for the EMFF design by creating torque at a distance[17]. The EMFF system assumes that spin-up of the array has occurred and the reaction wheels will not saturate during the steady state spin by rephrasing the array to dump momentum.[19]

To determine the mass of the coils, the force generated by the coils must equal the centripetal force from steady state rotation as seen in Equation (0.5) where R is the coil radius, I_t is the total current, S is the array baseline, ω is the rotation rate, and m_{tot} is the total mass of a habitat module.

$$F = \frac{3}{2}\mu_o \pi I_t^2 R^4 \left(\frac{1}{\left(\frac{S}{2}\right)^4} + \frac{1}{S^4} \right) = \frac{m_{tot} S \omega^2}{2} \quad (0.5)$$

For further clarification, Equation (0.5) uses a three identical satellite system, where the two outer spacecraft are the habitat modules, and the center spacecraft contains only the EMFF coil as shown in Figure 6. The reason for this design is to increase the amount of electromagnetic force in the system. The center spacecraft increases the electromagnetic force by 17 times the force produced by the outer spacecraft. The result is that the three spacecraft design contains EMFF coils that are $17^{-0.5}$ times lighter than those in a two spacecraft design.

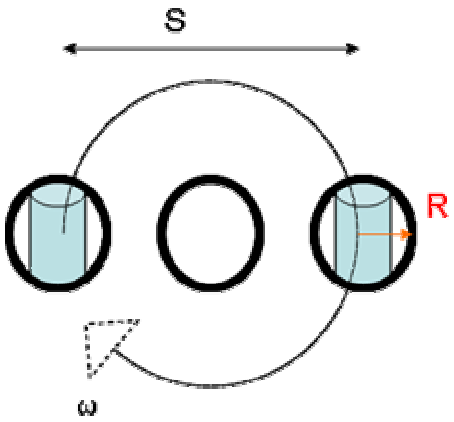

Figure 6 EMFF System Layout

To find the mass of the EMFF coils, the left hand side of Equation (0.5) can be rearranged to include the mass of the coil, M_c, and the wire current density over the wire density, I_c/p_c

$$M_c = \frac{\omega}{R} \frac{1}{\frac{I_c}{p_c}} \sqrt{\frac{m_{tot} S^5}{3 \cdot 17 \cdot 10^{-7}}} \quad (0.6)$$

For a high temperature superconducting coil, the I_c/p_c from the EMFF lecture is 16250 A m/kg.

Tether Sizing

Tethers are required for two of the system designs, so their mass is calculated based on the requirements of each design. For the single tether setup, the radius from the habitation module to the center of rotation is calculated according to the following equation.

$$r_{des} = \frac{g_{des}}{\omega_{\max}^2} \quad (0.7)$$

Here r_{des} is the radius that is 'desired' by the system given the desired force, g_{des}, and the maximum allowable rotation rate, ω_{max} (ω_{max} is determined by human factors). If the calculated radius is larger than the minimum radius allowed by human factors, then r_{des} can be used to calculate the tether length. If not, the minimum radius is used and the rotation rate must be slowed accordingly. Assuming that the two payloads are equal mass, the tether length for this system is then twice the desired radius. The tension in the tether may be calculated as:

$$T = m r_{des} \omega^2 \quad (0.8)$$

Here T is the tension in the tether, m is the mass of one payload, and r_{des} and ω are as above. This is simply an expression of Newton's Second Law, where the radial acceleration is calculated as the radius times the square of the angular velocity. The axial stress equation can then be used to calculate the required cross-sectional area to support the tension T, given the ultimate tensile strength of the chosen material.

$$A = \frac{T}{\sigma_{uts}} \quad (0.9)$$

For this analysis, two materials were considered, as shown in Table 4.[20]

Table 4 Tether material properties

Material	σ_{uts} *(GPa)*	*P (kg/m^3)*
Kevlar	3.6	1440
Spectra	2.6	970

Since the area, length, and density of the tether are now known, the mass can be easily calculated as follows.

$$m = lA\rho \quad (0.10)$$

The multiple tether design requires a different calculation of tether length, but similar techniques are used to compute the final mass. From Figure 1 it is clear that the multiple tether system is overconstrained.

For a system of tethers in tension this is not necessarily a bad thing, it simply makes analysis more tedious. For example, the multiple tether system would maintain its shape (while spinning) if it consisted of *either* the spokes or the rim tethers alone. However, if only the spoke tethers were used, there would be a small risk of collision between the pods during spin up, etc, so it might be wise to include the rim tethers. The solution to this problem is to analyze these systems separately and add the results.

Allowing the angle between two spokes to be α, and the angle between any spoke and its adjacent rim tether to be β, it is clear that:

$$\alpha = \tfrac{2\pi}{n}$$
$$\beta = \tfrac{1}{2}(\pi - \alpha) \qquad (0.11)$$

Here n is the number of spokes. The pods are each at distance r_{des} (or r_{min}, whichever is larger), as calculated above, so the total length of the spokes, l_s, is n times r. The total length of the rim tether, l_r, can be calculated as:

$$l_r = 2nr\sin(\tfrac{\alpha}{2}) \qquad (0.12)$$

The tension in the spokes is calculated without the rim tethers in place as:

$$T_s = mr\omega^2 \qquad (0.13)$$

The tension in the rim tethers is calculated without the spokes as:

$$T_r = \frac{T_s}{2}\cos\beta \qquad (0.14)$$

The area and mass of the rim and spoke tethers can then be calculated as before for each type of tether, and the results added for total tether mass. The following table gives examples of total tether mass for various systems. Systems labeled 'tether' are single tether systems, while systems labeled 'mult-n' are multiple tether systems with n pods. The subscript r represents the rim tethers, and the subscript s represents the spoke tethers. To get the total tether mass for a mult-n system, add the spoke mass to the rim mass. For single tether systems, the mass of both pods is 70,000 kg, and for mult-n systems the individual pod mass is taken as 40,000 kg.

Table 5 Example tether properties

System	Material	Force kN	Area (mm^2)	Length (m)	Mass (kg)
Tether	Kevlar	261	72.5	75.5	39.4
Tether	Spectra	261	100.4	75.5	36.8
Mult-5_r	Kevlar	127	35.2	222	56.3
Mult-5_s	Kevlar	149	41.4	189	56.3
Mult-5_r	Spectra	127	48.8	222	52.5
Mult-5_s	Spectra	149	57.4	189	52.5

In general, it is not wise to base a design on the ultimate tensile strength of a material, so a factor of safety of five is included in the implementation of these equations. It was found that the tether is a small portion of the total system mass, so this factor does not have a large impact. In addition, it helps to account for other tether properties that have been ignored, such as coatings against atomic oxygen, connecting hardware, etc.

Spacecraft Propulsion

A major contributor to total system mass is the propulsion system. While propulsion is not the main focus of this analysis, it is recognized that the fuel required by these spacecraft will be a significant portion of total system mass. Several parts of the required propulsion are treated in some detail, and others are left for a future study. The current analysis is of a high-risk, one-shot Mars approach. Enough fuel is provided to initiate the Mars transfer and the spin required for artificial gravity. It is assumed that the mission will succeed spectacularly. That is, the landing craft will reach Mars interface with hyperbolic velocity, and perform an aerocapture-assisted entry and descent phase. The astronauts will return on a vehicle that is already in place at their landing site. There is no provision for deceleration on Mars approach, for Earth-entry in case of an aborted mission, or other margin of any kind. This is obviously no way to design a manned mission to Mars, but since the primary thrusts of this analysis are cost, structures, and human factors, this greatly simplified propulsion model has been used.

Mars Transfer

In terms of fuel, the cheapest trip to Mars on high-impulse chemical thrusters is a Hohmann transfer. The Hohmann transfer assumes that the transfer orbit is tangent to both the initial and final circular orbits, so it is very efficient and easy to analyze.[21] Knowing the radii of the initial and final orbits, r_1 and r_2 respective, the semi-major axis, a, of the transfer orbit can be calculated as follows.

$$a = \tfrac{r_1 + r_2}{2} \qquad (0.15)$$

The velocity of the spacecraft in the transfer orbit, but at the point of making the impulsive injection maneuver, can then be calculated from the energy integral as:

$$v = \sqrt{\mu\left(\frac{2}{r_1} - \frac{1}{a}\right)} \tag{0.16}$$

Here v is the required velocity of the spacecraft in the transfer orbit, μ is the gravitational parameter for the central body (the Sun), and r_1 and a are as above. The required change in velocity can then be calculated as:

$$\Delta V = v - v_c \tag{0.17}$$

Where ΔV is the required change in velocity, v is the required spacecraft velocity above, and v_c is the circular velocity that the spacecraft already has due to the orbital motion of the Earth, given by:

$$v_c = \sqrt{\frac{\mu}{r_1}} \tag{0.18}$$

The required ΔV in the solar frame is only half of the calculation, however, for it takes extra fuel to escape from the Earth's sphere of influence. The ΔV required in the solar frame can be considered the "hyperbolic excess velocity" that is required in the Earth-centered frame, or the speed that the spacecraft has with respect to Earth as it leaves the sphere of influence. The change in velocity required from a low Earth parking orbit can be found by first calculating:

$$v_i = \sqrt{v_\infty + \frac{2\mu}{r_c}} \tag{0.19}$$

Here v_i is the insertion velocity that is required from LEO, v_∞ is the hyperbolic excess speed that is required in the heliocentric frame (the ΔV solved for above), and r_c is the radius of the parking orbit where the spacecraft is holding until departure. For this study, r_c was taken to be 200km. The required ΔV is then calculated as above, where v_c is recalculated for the parking orbit in the Earth frame.

For the Hohmann transfer from Earth to Mars studied here, the required change in velocity in the solar frame (v_∞) is calculated as 2.942 km/s. Using a circular parking orbit at 200 km, the burn required for the spacecraft is calculated to be 3.61 km/s.

The required ΔV can be used to calculate the fuel required to get the spacecraft to Mars. Using the classic rocket equation, the fuel mass is seen to be a function of required change in velocity, spacecraft dry mass, and the efficiency of the thruster (or specific impulse, I_{sp}).

$$m_p = m_o\left[1 - e^{-(\Delta V / I_{sp} g)}\right] \tag{0.20}$$

Here m_p is the mass of the propellant, m_o is the dry mass of the vehicle, I_{sp} is the specific impulse of the chosen thruster, and g is the acceleration due to gravity at the Earth's surface (where I_{sp} is defined). For this study, a value of 350 seconds is assumed for the specific impulse, a typical value for a bipropellant chemical thruster [see Ref 23, pg 692]. Thus, for a given spacecraft dry mass, the required fuel mass can be determined for each structural design. Note that the required fuel masses to insert the desired payloads into Mars orbit are quite large, so it is a reasonable assumption to ignore the thruster hardware at this stage of analysis. Additionally, for potentially massive components such as fuel tanks, all of the systems under consideration will have similarly scaled components so the relative error here is not significant. Finally, the question of thruster location is not specifically addressed here. It is assumed that the Mars transfer burn will be performed *before* the various designs have initiated their spin. Thus, all tethers will be retracted, and the EMFF system will be docked into a single unit. This way, the whole system can be started on the transfer orbit as a unit, and then the rotations can be initiated en route.

Spacecraft Rotation

A unique feature of the EMFF system is that it does not require fuel to initiate and maintain the nominal spin rate. All other designs, however, will require some manner of external thrust to start spinning. It will be assumed that the monolith structure has a pair of thrusters on opposite sides of the wheel (i.e. at the ends of a line of diameter) to create a couple. The single and multiple tether systems will have a single thruster on each individual pod, oriented to create a pure moment with no net force. The selected thruster has the same specific impulse, 350 s, as the primary thruster.

Under these assumptions, the fuel required to spin up can be calculated from the rocket equation above, noting that:

$$\Delta V = \Delta\omega\, r \tag{0.21}$$

Here $\Delta\omega$ is the change in angular velocity, and r is the radius from the center of rotation to the thruster. When calculating the fuel requirement from the rocket equation, the total fuel requirement is the dry mass of each individual spacecraft times the number of spacecraft. The following table shows example fuel requirements for spin up for the monolith, single tether, and multiple tether systems. In this table, both 'mass' and 'fuel mass' represent the individual spacecraft

masses, and should be multiplied by the total number of spacecraft if total system mass is desired.

Table 6 Propellant mass for system designs

System	Dry Mass (kg)	Fuel Mass (kg)
Monolith	86,774	56,704
Tether	44,344	28,971
Multiple	43,027	28,117

Cost Estimation

A first order model was developed to estimate approximate costs of the manned Mars mission. A detailed work breakdown structure (WBS) was not created since this study considers high-level concept designs that focus only on certain aspects of the mission, *i.e.* human factors, and structural configurations. The cost model utilized a mix of *analogy based estimation*, and *parametric estimation* in determining the costs of the various segments.

The cost model determines the mission cost in FY03$ by evaluating the required expenses in the following categories:

Space Segment

This is driven by the space segment cost factor (S_{cf}), the program level cost factor (P_{cf}), the heritage cost factor (H_{cf}), and the space system mass, M.

The space segment cost, S_c, in dollars is calculated, after adjusting the relationship given by Reynerson, as:

$$S_c = \left(S_{cf} H_{cf} M\right) + P_{cf} \qquad (0.22)$$

The S_{cf} ($/kg) is the price per kg of facility on orbit. For government run, manned space programs it ranges from 38 to 157 $K/kg, with the mean being 104 $K/kg.[22] The maximum value of 157 $K/kg is used in the model in order to get a conservative estimate.

The P_{cf} ($) accounts for the program level costs such as contractor costs for system engineering, management, quality assurance, and other costs that cannot be directly assigned to individual hardware or software components. The P_{cf} was determined from the parametric cost estimation data provided in table 20-4 in SMAD.

The heritage cost factor, H_{cf}, is a dimensionless quantity and accounts for the technology readiness level (TRL) costs. SMAD discusses the development heritage factor in space segment cost (pg 798) as a multiplicative factor. It defines heritage as the percentage of a subsystem that is identical to one or more previous spacecraft, by mass. This idea is applied in the cost model by assuming that the TRL can be considered as the system's heritage. A TRL of 3 is thus considered to have a heritage of only 30%, and the basic RDT&E cost estimate is increased by 70% to account for additional costs that will be accrued due to the development required for the new technology. This assumption provides a means to roughly estimate effects of different design TRLs on the cost. Note that the heritage factors are more appropriate to consider at the subsystem level, and it would be more accurate if they were considered when determining costs of specific subsystems. However, in this study only structures and human life support systems were considered in detail. Therefore, a blanket 'heritage factor' to the whole system cost estimate in this model really means an application to only these two subsystems.

The mass, M (kg), of the system is the total mass of the facility in space. The mass is often the primary cost driver of space systems.[23] The model used in the study also uses the facility's mass as a chief factor in the cost.

Launch Segment

The launch segment costs are determined by using the launch cost factor, L_{cf}, the insurance cost factor, I_{cf}, and the mass of the system, M, to be placed in orbit.

The launch cost is determined as:

$$L_c = L_{cf} I_{cf} M \qquad (0.23)$$

The launch cost factor, L_{cf}, is based on historical data and planned future cost goals. It is the cost per kilogram of placing a payload in LEO orbit. Table 20-14 in SMAD lists the cost per kg to LEO for various launch vehicles in FY00$. The average value for US launch vehicles comes out to be 14.66 $K/kg. Only US launch vehicles were considered since it is assumed that the mars mission will be a government run program. L_{cf} was taken as 15.4 $K/kg (after converting the dollar value from FY00 to FY03).

The insurance cost factor, I_{cf}, was used to account for insurance related expenses associated with launch. For commercial launches, the insurance is a third of the launch costs and I_{cf} is typically 1.33. The cost model in this study assumes a value of 1.5 to account for somewhat higher insurance costs that would probably be involved for a new type of mission. Furthermore, a higher factor would give a conservative estimate.

The mass, M (kg) used in this cost portion is the same facility mass that was used in determining the space segment cost.

Ground Operations and Support

The ground operations and support cost is usually much smaller than the space segment and launch cost. For missions that extend over long periods of time however, this cost can become quite significant. The ground segment costs are normally evaluated by considering the requirements for the ground station facilities such as square footage, equipment, personnel, *etc*. However since such details are not available at concept level studies, an analogy-based estimation was done to determine the operations and support cost for the manned mars mission. The International Space Station has a $13 billion operations budget for its ten-year life. The yearly operations costs are therefore earmarked as $1.3 billion. The cost model in this study uses a value of $1.5 billion per year for operations cost.

The total cost is obtained by summing the space segment, launch and operations cost of the mission.

Software Modules

Volume and Equipment Mass Module

Requirements

The MATLAB module *constants.m* determines the number of crew required for the mission as well as the volume and mass of the vehicle, life support equipment, as well as the required power for the spacecraft.

Description of Code

The code uses the input of the mission duration to calculate the number of crew members required for the mission. The number of crew members combined with the mission duration is used to size the free volume of the vehicle along with the life support system and power requirements.

Constants

The constants used in this module are the values for spacecraft volume, mass, and power which are given and explained in the "human factors" and "spacecraft power" sections of this document.

Inputs

duration (days): This input is the mission duration from Earth to Mars.

Outputs

crew: This output is the total number of crew required for the mission to Mars.

free_vol (ft^3): This output is the total required "free volume" in the spacecraft.

cs_vol (ft^3): This output is the total required volume for the crew systems equipment.

cs_mass (ft^3): This output is the total required mass of the crew systems equipment.

ls_air_vol (ft^3): This output is the total required volume for the atmosphere management equipment.

ls_air_mass (kg): This output is the total mass of the required atmosphere management equipment.

ls_water_vol (ft^3): This output is the total required volume of the water recovery and management equipment.

ls_water_mass (kg): This output is the total mass of the required water recovery and management equipment.

power (W): This output is the total power required for the spacecraft during the Earth-Mars transit. This is purely based on the number of crew members in the spacecraft.

Power Module

Requirements

The MATLAB module *SolarArrays.m* determines mass of the solar array and the area of the solar array. This module was used by Kwon, Vaughan, and Siddiqi in Problem Set 5.

Description of Code

The code uses the required power to determine the mass of the solar array. Multijunction arrays with no ellipse periods were used in the calculation

Constants

The constants used in this module are the specific powers for each type of solar array design.

Inputs

Average_power (W): This input is the required power that the solar arrays need to deliver.

Ellipse_fraction (0-1): This input is the fraction of the orbit spent in eclipse.

type (number): This input is selects the solar array type (1, 2, or 3 for Si, GaAs, or multijunction respectively).

Mission_duration (years): This input is the mission duration in years.

Outputs

Mass_solarArray (kg): This output is the mass of the solar array..

Area_solarArray (m^3): This output is the area of the solar array.

Structures Module

Requirements

The MATLAB module *structures.m* determines mass of the structure given the total volume required, the number of vehicles, and the type of architecture. The architecture options are a toroidal monolithic spinning spacecraft, a tethered multiple spacecraft, two tethered spacecraft, or two EMFF spacecraft.

Description of Code

The code uses the required volume and calculates the dimensions of a cylindrical pressure vessel. The diameter of the cylinder is constrained by the launch vehicle diameter. For the toroidal monolith system, it is assumed that the toroid is cut into sections while it is in the launch vehicle. The total volume is divided equally between the number of spacecraft for the design. Additionally the radius of rotation is used to determine the length of the cylinder. Once the dimensions of the structure are determined, its mass is calculated and outputted.

Constants

The constants used in this module are the values for density and allowable tensile ultimate stress for Aluminum 6061-T6 and the maximum internal pressure for design of the pressure vessel. These values are given and explained in the "structures" section of this document.

Inputs

V (m^3): This input is the total volume required for the structure to contain.

D (m): This input is the launch vehicle diameter.

N (number): This input is the number of vehicles in the array.

R (m): This input is the radius of rotation.

w (rad/s): This input is the rotation rate of the system.

design ('text'): This input is the desired design, options include 'monolith', 'multiple', 'tether', and 'emff'.

Outputs

Mass (kg): This output is the total mass of the structure.

Tether Mass Module

Requirements

The MATLAB module *tether_mass.m* determines mass of the tether given the type of architecture, number of vehicles, dry mass, tether material, and desired acceleration.

Description of Code

The code takes the system architecture and decides how to calculate the tether length and tension. For the monolith and EMFF, there is no tether. For the single and multiple tether systems, the values are computed appropriately as described above. Mass of the tether is then calculated from the material properties of the tether and the required length and tension.

Constants

The constants used in this module are the values for maximum allowable spin rate and minimum allowable radius, as defined by human factors.

Inputs

AG_type ('text'): This input is the desired design, options include 'monolith', 'multiple', 'tether', and 'emff'.

n (number): This input is the number of vehicles in the array.

w (rad/s): This input is the rotation rate of the system.

dry_mass (kg): This is the mass of the spacecraft. For the single tether, an array of 2 masses (can be unique). For the multiple tether, one mass is provided and the modules are assumed to be identical.

tether_mat ('text'): Input describes what material to use for the tether. Current options are 'spectra' and 'kevlar'.

g_des (m/s^2): This desired acceleration at the rim.

Outputs

Mass (kg): This output is the total mass of the tether(s).

Propulsion Module

Requirements

The MATLAB module *propulsion.m* determines mass of the required fuel for orbit transfer and spin-up, given the type of architecture, number of vehicles, dry mass, tether material, and the moment arm to the thruster.

Description of Code

The code calculates the change in velocity required for a Hohmann transfer to Mars, and then solves the Earth-centered problem for ΔV required from a LEO parking orbit. The rocket equation is then used for an assumed thruster to find the fuel mass for the transfer. Given the type of system and the thruster moment arm, the rocket equation is used again to find the propellant required for spin-up. This function calls several auxiliary functions that are included and commented in Appendix A, namely: *ic_circ.m, hohmann.m, p_conic.m,* and *r_equation.m*.

Constants

The constants used in this module are the gravitational constants for the Earth and Sun, the radius of the Earth, the Earth-Sun distance, the parking orbit radius, and the Mars-Sun distance.

Inputs

AG_type ('text'): This input is the desired design, options include 'monolith', 'multiple', 'tether', and 'emff'.

n (number): This input is the number of vehicles in the array.

w (rad/s): This input is the rotation rate of the system.

dry_mass (kg): This is the mass of the spacecraft. For the single tether, an array of 2 masses (can be unique). For the multiple-tethered spacecraft, one mass is provided and the modules are assumed to be identical.

r_outer (m): The moment arm for the thruster.

Outputs

Mass (kg): This output is the mass of the propulsion system per pod.

EMFF Module

Requirements

The MATLAB module *emff.m* determines mass of the superconducting EMFF coils needed to rotation rate for a given amount of artificial gravity.

Description of Code

The code uses the size of the cylinder as the size of the coils, the total dry mass each satellite, the radius of rotation, and the rotation rate to determine the mass of the coils for a three spacecraft collinear array. The equation used for this is explained in the "emff coil mass" section.

Constants

The constant used in this module is the Superconducting coil current density divided by the wire density as given in the EMFF Lecture.

Inputs

V (m^3): This input is the total volume required for the structure to contain.

D (m): This input is the launch vehicle diameter.

R (m): This input is the radius of rotation.

w (rad/s): This input is the rotation rate of the system.

Mass_tot (kg): This input is the total dry mass of one of the satellites.

Outputs

Mass_coil (kg): This output is the total mass of the EMFF coil.

Cost Module

The MATLAB module *cost.m* calculates the total cost of a manned mission based on the system mass, technology readiness level of the system, and mission duration.

Inputs

mass (kg): This is the total mass of the system /facility in space.

TRL: The Technology Readiness Level of the system

duration (days): The mission duration from Earth to Mars.

Outputs

TotalCost ($): This is the total cost of the mission in $FY03. It is the sum of all the cost segments that are also given out by the module.

SpaceSegCost ($): The space segment cost in $FY03

LVCost ($): Launch cost in $FY03

OpSupCost ($): Operations support cost in $FY03

Results

The total program costs for a 1.5 year mission for the different designs are shown in the figure below. It is seen that the cheapest design option is the monolith while the multiple tether configuration is the most expensive.

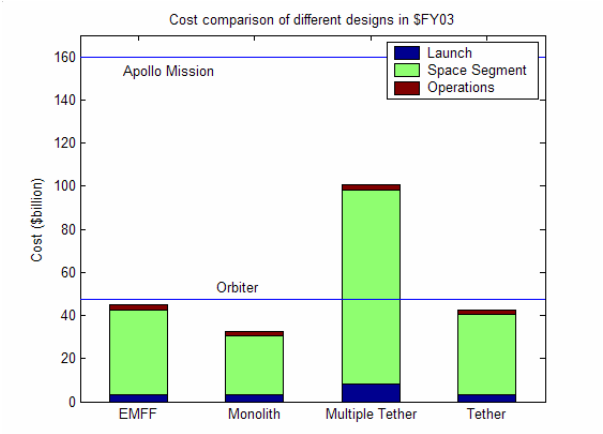

Figure 7 Cost of different designs

The cost breakdown shows that the space segment cost is by far the largest portion as compared to launch and operation costs. A comparison with Apollo and Orbiter costs show that the model estimates lie within a reasonable range.

Since the cost model is driven primarily by the system mass, an analysis of the mass of the different designs shows a trend that matches with the cost results. The figure below illustrates the total mass estimates obtained for the different designs.

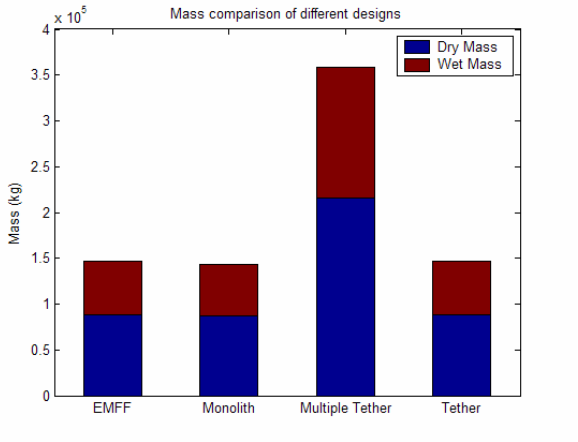

Figure 8 Mass comparison of different designs

Although the mass of the EMFF, monolith and tether designs are in the same range, the monolith is cheaper than the other two designs due to a higher TRL value. EMFF and tether designs have lower TRLs (the model assumed 3 and 4 respectively), therefore they cost more than the monolith. The dry mass in each design included the power subsystem, the structural mass, and crew and life support equipment mass. It also included mass of subsystems that were unique to each particular configuration, for instance the dry mass of the EMFF design includes the mass of coils while in the multiple tether and tether options it includes the mass of tethers. From these results it appears that the monolith design offers the lightest and cheapest option.

Varying Crew Size

One interesting plot is the change in total program cost versus the number of crew used in the mission to Mars. As the number of crew increases, the required structure volume increases, which in turn increases the mass and cost. The results for the four designs considered are shown in the figure below.

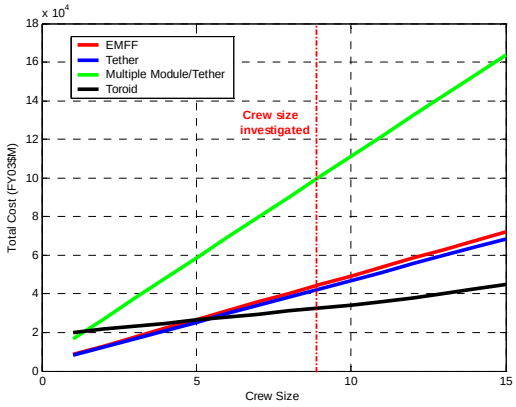

Figure 9 Cost vs. crew size

The above figure shows significant differences in the rate of change of cost as the crew size changes for the various spacecraft designs. The single tether and EMFF designs have nearly identical curves in the figure as well. This is due to the fact that they both have the same basic structural design: each design has two main modules and little or no mass connecting the two modules. Also, these two designs have nearly identical TRL values.

The other two designs, the toroid and the multiple-tethered module, are radically different designs than the previous two. It can be seen that the cost of the monolith increases much less dramatically than the cost of the multiple-tethered module vehicle. This difference is mainly the result of the fact that the TRL of the toroid is much higher than that of the other three designs, especially the multiple-tethered module.

Finally, it can be seen in Figure 9 that the cost of the Toroid spacecraft becomes the most cost effective design for crew sizes greater than five. Based on this information, a Toroid may be the most cost effective design for a large crew of approximately nine members for a manned mission to Mars.

Effect of Varying Artificial Gravity

The artificial gravity is created by rotation of the vehicle(s). A higher artificial gravity results in a higher rotation rate, given a fixed radius of rotation. For the tethered two spacecraft, multiple spacecraft, and monolith systems, a higher rotation rate results in a larger ΔV needed for spin-up and results in more propellant. For the EMFF system, the EMFF coil mass is directly related to the rotation rate as seen in equation (0.6). Figure 10 illustrates these results for the four different systems. Each of the systems shown an increase as the Earth's gravity is approached. None of the curves overlap and the multiple-tethered spacecraft shows the highest mass while the two tethered spacecraft is the least massive option.

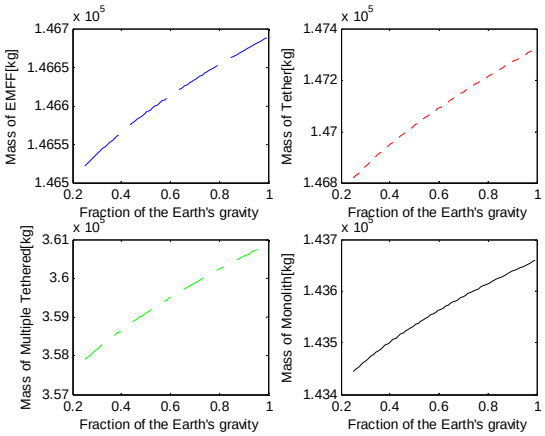

Figure 10 Effect of varying the fraction of Earth's gravity on total system mass

Figure 11 illustrates the effect of varying the fraction of Earth's gravity on the total system cost. Since the cost varies directly with the mass, these results show an expected trend; the cost shows an increase as the Earth's gravity is approached.

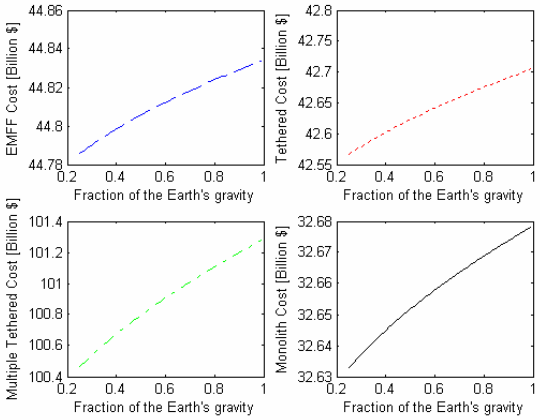

Figure 11 Effect of varying the fraction of Earth's gravity on cost

Conclusion

The preceding design of an artificial gravity Mars mission demonstrates that the mission has feasibility in terms of cost since the cost is less than the Apollo program. The mass of the systems are high mainly due to the significant propellant mass, but more advanced propulsion systems could help decrease this. The monolith system is currently the most favorable design for cost and mass, and for large crew sizes. The tether and EMFF designs may become more favorable with further development of their technology boosting their TRL.

Future Work

This is an exciting project with much potential for future work. An improvement that is immediately obvious is to allow different mission durations and to evaluate the effect of that change on mission cost and mass. Currently, a Hohmann transfer from Earth to Mars is specified, but other orbits should be examined such as faster one-tangent burns, or perhaps longer orbits with free-return trajectories. Changing the mission duration will impact the number of desired crew-members as well as the required propellant for transfer, and so could have a large impact on mass and cost.

Another improvement would be to add a detailed propulsion system model to this analysis. Current all propulsion system hardware is neglected, along with any propellant margin, corrective maneuvers, terminal rendezvous burn, or mission-abort scenarios. All of these things could be added to increase the fidelity of the overall model. Certainly, including these things will increase the total mass and cost of the systems.

There are many other systems that could be added as well. Power, while mentioned in this study, could be investigated in a much more thorough fashion. Issues could be addressed relating to human needs such as thermal controls, debris and radiation mitigation, and communications. Each of these improvements could greatly enhance the quality of the analysis and make this an even more valuable tool for future use.

Appendix A: MATLAB source code

main.m

```
%Constants
g_des = 1/3 * 9.81;
rmin = 30; %meters
wmax = 6; %rpm
w = wmax * pi/30;

D = 5;  %m, Launch vehicle width

mission_duration = 1.5;   %years

%first call constants
%Get the required volume and average_power
[crew, free_vol, cs_vol, cs_mass, ls_air_vol, ls_air_mass, ls_water_vol, ...
        ls_water_mass, power] = constants(mission_duration*365);
V_ft = free_vol + cs_vol + ls_air_vol + ls_water_vol; %Total Volume, ft^3
V = V_ft * 2.83168*10^-2;  %Total Volume, conversion from ft^3 to m^3

M_systems = cs_mass + ls_air_mass + ls_water_mass; %Mass of crew and support systems

%Find Power Mass
[M_power, Area_power] = SolarArrays (power,0,3,mission_duration);

r_des = g_des/w/w;   % calculate 'desired' radius to get desired acceleration
if (r_des < rmin);
    r_des = rmin;
    w = sqrt(g_des/r_des);
end

%Now For Each Design
%EMFF%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N=2;
%Find structural mass
%mass of each cylinder (note that there are two for emff, tether)
M_struct_emff = structure(V, D, N, r_des, w,'emff');
%Find Propulsion system Mass
Mass_dry_emff = M_struct_emff + M_power + M_systems;
M_coil_emff = emff(V, D,r_des, w, Mass_dry_emff);
M_wet_emff=propulsion('emff', N, w, Mass_dry_emff+M_coil_emff, r_des);
%Compute the total mass
M_total_emff = Mass_dry_emff + M_coil_emff+M_wet_emff;
%Computer system mass
M_system_emff = N * M_total_emff;
%Find cost
[TotalCost_emff, SpaceSegCost_emff, LVCost_emff,
OpSupCost_emff]=cost(M_system_emff,3,mission_duration);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Tether%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N=2;
%Find structural mass
%mass of each cylinder (note that there are two for emff, tether)
M_struct_tether = structure(V, D, N, r_des, w,'tether');
%Find Propulsion system Mass for each spacecraft
Mass_dry_tether = M_struct_tether + M_power + M_systems;
M_wet_tether=propulsion('tether', N, w, Mass_dry_tether, r_des);
M_tether = tether_mass('tether', N, w, [(Mass_dry_tether+M_wet_tether)
(Mass_dry_tether+M_wet_tether)], 'kevlar', g_des);
%Compute the total mass
M_total_tether = Mass_dry_tether + M_wet_tether;
M_system_tether = N * M_total_tether + M_tether;
%Find cost
[TotalCost_tether, SpaceSegCost_tether, LVCost_tether,
OpSupCost_tether]=cost(M_system_tether,4,mission_duration);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Multiple Tethered 5 spacecraft%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N=5;
%Find structural mass
%mass of each cylinder (note that there are two for emff, tether)
M_struct_multiple = structure(V, D, 5, r_des, w,'multiple');
%Find Propulsion system Mass
Mass_dry_multiple = M_struct_multiple + M_power + M_systems;
M_wet_multiple=propulsion('multiple', N, w, Mass_dry_multiple, r_des);
```

```
M_tether_multiple = tether_mass('multiple', 5, w, Mass_dry_multiple+M_wet_multiple, 'kevlar',
g_des);
%Compute the total for each spacecraft
M_total_multiple = Mass_dry_multiple + M_wet_multiple;
%Computer system mass
M_system_multiple = N * M_total_multiple + M_tether_multiple;
%Find cost
[TotalCost_multiple, SpaceSegCost_multiple, LVCost_multiple,
OpSupCost_multiple]=cost(M_system_multiple,4,mission_duration);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Monolithic Spacecraft%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Find structural mass
%mass of each cylinder (note that there are two for emff, tether)
M_struct_monolith = structure(V, D, 1, r_des, w,'monolith');
%Find Propulsion system Mass
Mass_dry_monolith = M_struct_monolith + M_power + M_systems;
M_wet_monolith=propulsion('monolith', N, w, Mass_dry_monolith, r_des);
%Compute the total mass
M_total_monolith = Mass_dry_monolith + M_wet_monolith;
%Find cost
[TotalCost_monolith, SpaceSegCost_monolith, LVCost_monolith,
OpSupCost_monolith]=cost(M_total_monolith,8,mission_duration);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

constants.m

```
% William Nadir
% 16.851 Satellite Enginnering
% Module to estimate Mars mission crew size and vehicle volume and mass requirements
%
% INPUTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% duration      = Mission duration (days)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% OUTPUTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% crew                    = Crew size (number of people)
% free_vol               = Amount of total "free" volume required for crew (ft^3)
% cs_vol                  = Amount of total volume required for crew systems (ft^3)
% cs_mass              = Mass of crew systems (kg)
% ls_air_vol             = Life support equpment (air) total volume (ft^3)
% ls_air_mass         = Mass of life support equipment (air) (kg)
% ls_water_vol         = Life support equpment (water) total volume (ft^3)
% ls_water_mass    = Mass of life support equipment (water) (kg)
% power                       = Required total spacecraft power (W)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [crew,free_vol, cs_vol, cs_mass, ls_air_vol, ls_air_mass, ls_water_vol, ...
        ls_water_mass, power] = constants(duration)

% Here the required crew size is determined based on the duration of the
% mission to Mars
crew = ceil((.0292 * duration + (9/8))/2); % (No. of crew members)

free_vol = 700 * crew; % Total free volume required (ft^3)

cs_vol = 1.512* crew * duration; % Total crew systems volume required (ft^3)
cs_mass = 7.55 * crew * duration; % Total crew systems mass (kg)

ls_air_vol = 35.3 * crew; % Total ECLSS air control system volume (ft^3)
ls_air_mass = 255 * crew; % Total ECLSS air control system mass (kg)

ls_water_vol = 7.1 * crew; % Total ECLSS water control system volume (ft^3)
ls_water_mass = 50 * crew; % Total ECLSS water control system mass (kg)

power = 3300 * crew; % required S/C power (W)
```

SolarArrays.m

```
%This function calculates the mass, cost, and size of a given type of solar array.
%The inputs are eclipse fraction, average power, shadow fraction, mission duration, and cell
type.


function [Mass_solarArray,Area_solarArray]=SolarArrays
(average_power,eclipse_fraction,type,mission_duration)
```

```
%P_solarArray: power produced by solar arrays
%Pe           :power required during eclipse period
%Pd           : power required during daylight period
%Xe           :path efficiency from solar array, through battery to loads
%Xd           :path efficiency from solar array to loads
%Xcell        :cell efficiency
%Isolar       :solar illumination intensity
%Id           :Inherent degredation
%DSi etc      : degredation/yr
%Ld           :lifetime degredation

%*******************************
% Constants
Xe = 0.65;
Xd = 0.85;
XSi = 0.148;
XGaAs= 0.185;
Xmulti= 0.22;
Isolar = 1367;
Id = 0.77;
DSi = 0.0375;
DGaAs = 0.0275;
Dmulti = 0.005;
SpSi = 0.55; %16.89 design doc kg/m^2
SpGaAs = 0.85; %kg/m^2
Spmulti= 0.85; %kg/m^2
%*******************************

Pe = average_power;
Pd = average_power;

%Te = orbit_period*eclipse_fraction; %eclipse time
%Td = orbit_period-Te;       %daylight time
%P_solarArray = ((Pe*Te )/(Xe*Td)) + (Pd/Xd) %power produced by solar arrays

P_solarArray = Pd/Xd;

%Silicon is type 1, GaAs is type2, and multijunction is 3

if type == 1
    Xcell = XSi;
    Degredation = DSi;
MassPerArea = SpSi;
SpecificPower = 25;
%SpecificCost = SpCostSi;
end

if type == 2
    Xcell = XGaAs;
    Degredation = DGaAs;
MassPerArea = SpGaAs;
SpecificPower = 60; %ref:
http://lheawww.gsfc.nasa.gov/docs/balloon/2nd_tech_workshop/Loyselle.pdf
%SpecificCost = SpCostGaAs;
end

if type ==3

    Xcell = Xmulti;
 Degredation = Dmulti;
MassPerArea = Spmulti;
SpecificPower = 66; %assumed based on info in SMAD
%SpecificCost = SpCostmulti;
end

%Power out of solar cell assuming sun rays are normal to solar panels
Pout = Xcell * Isolar;


%Power at begining of life

Pbol = Pout *Id;
Ld = (1-Degredation)^mission_duration;

%Power at end of life

Peol = Pbol*Ld;

Area_solarArray = P_solarArray/Peol;

Mass_solarArray = P_solarArray/SpecificPower;

%Cost = SpecificCost * Mass_solarArray;
```

```
%Cost = NaN;
```

structure.m

```
function [mass] = structure(V, D, N, R, w,design)

% V = 85;             %m^3
% D = 5;              % Launch vehicle diameter
% N = 2;              %number of vehicles
% R = 20;             % distance from center of rotation to floor(really ceiling)
% design = 'monolith';    %choices: monolith, multiple, tether, emff
% w = 6;              %rotation rate in rpm
%Material Selection: AL 6061-T6
rho = 2.85 * 10^3;  %kg/m^3, density
Ftu = 290 * 10^6;   %Pa, Allowable Tensile Ultimate Stress

% Design Factors
Pmax = 0.1096 * 10^6;   %Pa, Maximum Internal Pressure
SF = 2.0;           %Safety Factor

Pu = SF * Pmax;     %Design Ultimate Internal Pressure

switch lower(design)
    case {'emff', 'tether'}
        r = D/2;            %m, Radius of the cylinder
        l = (V/N)/(pi*r^2);     %m, Length of the Cylinder
        % Thin-Walled pressure cylinder thickness
        t = Pu * r / (Ftu); %m

        if t < 0.015
            t = 0.015;      %minimum thickness necessary for radiation dosage
        end

        %Calculating the mass of the cylinder structure
        Mcyl = 2 * r * l * t * rho;
        Mends = pi * r^2 * t * rho;
        mass = 2 * Mends + Mcyl;

    case {'multiple'}
        Vi = V/N;
        r = D/2;            %m, Radius of the cylinder
        l = Vi/(pi*r^2);     %m, Length of the Cylinder

        % Thin-Walled pressure cylinder thickness
        t = Pu * r / (Ftu); %m

        if t < 0.015
            t = 0.015;      %minimum thickness necessary for radiation dosage
        end

        %Calculating the mass of the cylinder structure
        Mcyl = 2 * r * l * t * rho;
        Mends = pi * r^2 * t * rho;
        mass = (2 * Mends + Mcyl);
    case {'monolith'}
        %Given V
        %R is set from minimum radius needed for artifical gravity

        rt = 1/pi * sqrt(V/(2*R));    %inner toroid radius
        %if rt < some height, then rt = minimum height
        if rt < 1.8288/2 % if height is less than six feet
            rt = 1.8288/2;    %meters
        end

        t = Pu * rt / Ftu;  %m
        if t < 0.015
            t = 0.015;      %minimum thickness necessary for radiation dosage
        end

        %Calculating the mass of the toroid structure
        Mass_toroid = rho * t * (2*pi*rt) * (2*pi*R);

        %Calculating the mass of the center spherical shell
        %         rs = 1/10 * R;
        %         t_shell = 0.015;
        %         Mass_shell = 4*pi*rs^2 * t_shell*rho;
        %
        %         %Calculating the mass of the beams
        %         Num_beams = 4;
        %         rb = R - rs;
        %         h = 0.1;    %beam width
        %         Mass_beam = rb * h^2 *rho;

        %Calculating total monolith mass
```

```
        %         mass = Mass_toroid + Mass_shell + Mass_beam * Num_beams;
        mass = Mass_toroid;
    otherwise
        disp('Unknown method.')
end
```

tether_mass.m

```
function [mass]=tether_mass(AG_type, n, w, dry_mass, tether_mat, g_des)

% Function "tether_mass.m" takes parameters of the system and returns the
% calculated mass of the tether.
%
% Inputs:
%    AG_type [] - The type of system in question (monolith, multiple, tether, emff)
%    n [] - The number of spacecraft, only applicable for the pinwheel design
%    w [rad/s] - The desired rotational rate of the system [to be hardcoded?]
%    dry_mass [kg] - The dry mass of the spacecraft.  This should be a row
%       vector with appropriate dimensions as follows:
%           * Monolith: N/A
%           * EMFF: N/A
%           * Single Tether: [(habitation module)  (other mass)]    ###[1x2]
%           * Pinwheel: (mass of nodes, assumed uniform)     ###[1x1]
%    tether_mat [] - Indicates what tether material should be used
%    g_des [m/s^2] - The desired acceleration at the habitation module
%
% Outputs:
%    mass [kg] - The mass of the tether

% Constants

rmin = 30; %meters
wmax = 6; %rpm
verbose = 1;

% /Constants

% Tether

% source for this stuff 'http://callisto.my.mtu.edu/my4150/props.html' is ok

if strcmp(tether_mat, 'kevlar')
    sig_uts = 3.6e9;     % [Pa]
    density = 1440;    % [kg/m^3]
elseif strcmp(tether_mat, 'spectra')
    sig_uts = 2.6e9;     % [Pa]
    density = 970;    % [kg/m^3]
else
    fprintf('Error :: tether_mass :: Unknown tether material!\r')
end


% /Tether

wmax = wmax*pi/30;  % rad/s
if (w > wmax)
    fprintf('Warning :: tether_mass :: Spacecraft spinning too fast!\r')
end

if strcmp(AG_type, 'monolith')
    mass=0;
elseif strcmp(AG_type, 'emff')
    mass=0;
elseif strcmp(AG_type, 'tether')
    m1=dry_mass(1);
    m2=dry_mass(2);
    r_des = g_des/w/w;   % calculate 'desired' radius to get desired acceleration
    if (r_des < rmin);
        r_des = rmin;
        w = sqrt(g_des/r_des);
        fprintf(['Warning :: tether_mass :: Calculated radius is below minimum, used minimum
radius of ' num2str(rmin) ' [m].\r'])
        fprintf(['                          Angular rate should be less than ' num2str(w*30/pi) '
[rpm].\r'])
    end
    tlength=r_des*(m1+m2)/m2;    % calculate tether length based on desired radius and relative
masses
    F=m1*r_des*w*w;   % calculate tension in the tether
    A=F/sig_uts;   % required tether area is tension/ultimate tensile strength

    mass=density*tlength*A * 5;    % calculate tether mass using factor of safety of 5

    if (verbose == 1)
```

```
        fprintf(' \r')
        fprintf(['Designed a ''' AG_type ''' tether of ' tether_mat '.\r'])
        fprintf(['Desired radius: ' num2str(r_des) ' [m].\r'])
        fprintf(['Tether length:  ' num2str(tlength) ' [m].\r'])
        fprintf(['Tether tension: ' num2str(F) ' [N].\r'])
        fprintf(['Tether area:    ' num2str(A) ' [m^2].\r'])
        fprintf(['Tether mass:    ' num2str(mass) ' [kg].\r'])
        fprintf(' \r')
    end

elseif strcmp(AG_type, 'multiple')
    m=dry_mass(1);
    r_des = g_des/w/w;   % calculate 'desired' radius to get desired acceleration
    if (r_des < rmin);
        r_des = rmin;
        w = sqrt(g_des/r_des);
        fprintf(['Warning :: tether_mass :: Calculated radius is below minimum, used minimum
radius of ' num2str(rmin) ' [m].\r'])
        fprintf(['                          Angular rate should be less than ' num2str(w*30/pi) '
[rpm].\r'])
    end
    alpha=2*pi/n;  % angle between the spokes
    beta=.5*(pi-alpha);  % angle betwen spokes and outer strands
    tlength1=n*r_des;    % calculate length for the spokes
    tlength2=n*2*r_des*sin(alpha/2);  % calculate length for outer strands

    F1=m*r_des*w*w;   % calculate tension in the spoke tethers
    F2=F1/2/cos(beta);   % calculate tension in the rim tethers

    A1=F1/sig_uts;    % required tether area is tension/ultimate tensile strength
    A2=F2/sig_uts;

    mass1=density*tlength1*A1 * 5;   % calculate spoke tether masses using factor of safety of 5
    mass2=density*tlength2*A2 * 5;   % calculate rim tether masses using factor of safety of 5

    mass=mass1+mass2;

    if (verbose == 1)
        fprintf(' \r')
        fprintf(['Designed a ''' AG_type ''' tether of ' tether_mat '.\r'])
        fprintf(['Desired radius: ' num2str(r_des) ' [m].\r'])
        fprintf(['Spoke tether length (total):  ' num2str(tlength1) ' [m].\r'])
        fprintf(['Outer tether length (total):  ' num2str(tlength2) ' [m].\r'])
        fprintf(['Spoke tether tension: ' num2str(F1) ' [N].\r'])
        fprintf(['Outer tether tension: ' num2str(F2) ' [N].\r'])
        fprintf(['Spoke tether area:    ' num2str(A1) ' [m^2].\r'])
        fprintf(['Outer tether area:    ' num2str(A2) ' [m^2].\r'])
        fprintf(['Spoke tether mass:    ' num2str(mass1) ' [kg].\r'])
        fprintf(['Outer tether mass:    ' num2str(mass2) ' [kg].\r'])
        fprintf(['Total tether mass:    ' num2str(mass) ' [kg].\r'])
        fprintf(' \r')
    end

else
    fprintf('Error :: tether_mass :: Unknown spacecraft type!\r')
end
```

propulsion.m

```
function mass=propulsion(AG_type, n, w, dry_mass, r_outer)

% function 'proplusion.m' calculates the required fuel mass for both
% "spin-up" and initiation of the interplanetary transfer orbit.

% dry_mass should be the dry mass of a single 'pod'

% Inputs:
%    AG_type [] - The type of system in question (monolith, multiple, tether, emff)
%    n [] - The number of spacecraft, only applicable for the pinwheel design
%    w [rad/s] - The desired rotational rate of the system [to be hardcoded?]
%    dry_mass [kg] - The dry mass of the spacecraft.  This should be a row
%       vector with appropriate dimensions as follows:
%           * Monolith: N/A
%           * EMFF: N/A
%           * Single Tether: [(habitation module)  (other mass)]    ###[1x2]
%           * Pinwheel: (mass of nodes, assumed uniform)     ###[1x1]
%    r_outer [m] - moment arm to the thruster
%
% Outputs:
%    mass [kg] - The mass of the propulsion system per pod

MU_s=1.327e20;  %m^3/s^2, gravitational constant for the sun
MU_e=3.986e14;   %m^3/s^2, gravitational constant for the earth
```

```
res=1.5e11;    %m, earth-sun distance
re_mag=6.38e6;   %m, earth radius

parking=200e3;  % parking orbit ALTITUDE in km

planet1 = 3;  % choose Earth as the origin
planet2 = 4;  % choose Mars as the destination

planet(1)=0.3871;    % define planetary radii for future use
planet(2)=0.7233;
planet(3)=1;
planet(4)=1.524;
planet(5)=5.203;
planet(6)=9.519;
planet(7)=19.28;
planet(8)=30.17;
planet(9)=39.76;

planet = planet * res;   % put planet distances in [m]

[r, v] = ic_circ(MU_s, planet(planet1));   % get circular initial conditions for the Earth
[dv, t_trans]=hohmann(MU_s, r, v, planet(planet2));  % calculate dV and transfer time for hohmann
to mars

[re, ve] = ic_circ(MU_e, re_mag+parking);   % get circular initial conditions for circular 200km
parking orbit
[eta, dv_earth, t_soi] = p_conic(MU_e, MU_s, norm(dv), re, ve, res);  % find the dV required to
Mars from the parking orbit

i_mass=r_equation(dry_mass, dv_earth);   % find the mass required for insertion

% s_mass is the spin-up propulsion system mass

if strcmp(AG_type, 'monolith')
    % assumes a pair of thrusters at the rim of the craft, to set up a couple
    dv_req=w*r_outer;
    s_mass=r_equation(dry_mass, dv_req);
elseif strcmp(AG_type, 'emff')
    s_mass=0;
elseif strcmp(AG_type, 'tether')
    dv_req=w*r_outer;
    s_mass=r_equation(dry_mass, dv_req);
elseif strcmp(AG_type, 'multiple')
    dv_req=w*r_outer;
    s_mass=r_equation(dry_mass, dv_req);
end

mass=i_mass + s_mass;
```

ic_circ.m

```
function [r, v] = ic_circ(MU, r_init)

% calculate circular initial conditions (position and velocity) given
%   a central body and an initial radius

r = [r_init 0 0];
vc = sqrt(MU/r_init);
v = [0 vc 0];
```

hohmann.m

```
function [dv, t_trans]=hohmann(MU, r, v, r_target)

% Function ip_hohmann takes the current (sun-centered inertial) position
%    and velocity vectors, verifies an initial circular orbit, and then
%    calculates the delta-V required at that instant to enter a
%    minimum-energy (Hohmann) transfer to a given radius (scalar).  The
%    function also returns the time of transfer, which is half the period
%    of the transfer orbit.

% MU is the gravitational parameter of the central body (m^3/s^2)
% r is the radius vector to the spacecraft (sun-centered)
% v is the velocity vector of the spacecraft (sun-centered)
% r_target is the orbital radius of the target planet

h = cross(r,v);       % angular momentum vector
h_mag = norm(h);
p = h_mag*h_mag/MU;     % orbit parameter
a = -1/((norm(v)^2)/MU-2/norm(r));     % semimajor axis
e = sqrt(1-p/a);
```

```
if ( e > .01)
    fprintf('Error :: IC :: Initial orbit is not circular!\r')
    fprintf(['Orbital eccentricity is ' num2str(e) '!\r'])
end

r_mag = norm(r);

a_trans = (r_mag + r_target)/2;
v_trans_i = sqrt(MU*(2/r_mag - 1/a_trans));

dv_trans = v_trans_i - norm(v);
dv = dv_trans*v/norm(v);

t_trans=2*pi*sqrt((a_trans^3)/MU)/2;
```

p_conic.m

```
function [eta, dv, t_soi] = p_conic(MU1, MU2, v_inf, r, v, rps)

% inputs:
%   MU1: MU for the primary body, i.e. escaping from Earth orbit
%   MU2: MU for the contending body in the SOI problem, typically the sun
%   v_inf: the scalar velocity required at r_inf to enter the desired
%     helicentric transfer.  Found by solving heliocentric problem.
%   r: the radius vector to the spacecraft
%   v: the velocity vector of the spacecraft
%   rps: the distance between the two SOI bodies, i.e. the Earth and Sun

% returns:
%   eta: the angle between the velocity vector of primary body and radius vector to s/c
%   dv: the scalar change in velocity required to get the desired v_inf
%   t_soi: the time required to reach the sphere of influence (SOI)

rc = norm(r);
vc = norm(v);

v1 = sqrt(v_inf^2 + 2*MU1/rc);    % required velocity at the parking orbit radius

energy = (v_inf^2)/2;
h = rc*v1;
ei = sqrt(1 + 2*energy*h^2/MU1^2);   % eccentricity of the transfer orbit


eta = acos(-1/ei);    % angle between the velocity vector of primary body and radius vector to
s/c
dv = v1 - vc;   % the required delta-v

p = 2*rc;
r_soi=(rps)*(MU1/MU2)^(2/5);    % the radius of the sphere of influence

f = acos((p/r_soi-1)/ei);   % the true anomaly there

H = 2*atanh(sqrt((ei-1)/(ei+1))*tan(.5*f));   % the hyperbolic anomaly there

N = ei*sinh(H) - H;   % kepler's equation for hyperbolas, N ~~ Mean anomaly


a = p/(1-ei^2);   % the 'semi-major axis' of the hyperbola (<0!)

t_soi = N/sqrt(MU1/(-a)^3);   % the time to reach the sphere of influence (r = r_inf, v = v_inf)
```

r_equation.m

```
function mass=r_equation(dry_mass, dV)

% !!!The Rocket Equation!!!

% function 'r_equation.m' takes the dry mass of the vehicle and the
%   required dV to gain the transfer orbit, and computes the mass of
%   fuel required for injection (given a particular thrusting system)
%
% Inputs:
%   dry_mass [kg] - The mass of the spacecraft w/o fuel
%   dV [m/s] - The change in velocity required to gain the transfer orbit
%
% Outputs:
%   mass [kg] - The mass of the fuel required
```

```
% Assume typical bipropellant chemical thruster w/ Isp ~ 350s

Isp = 350;
g = 9.81;

mass = dry_mass * (1-exp(-(dV/(Isp*g))));
```

emff.m

```
function [mass_coil] = emff(V, D,R, w, Mass_tot)
%The code uses the size of the cylinder as the size of the coils,
%the total dry mass each satellite, the radius of rotation,
%and the rotation rate to determine the mass of the coils
%for a three spacecraft collinear array.

r = D/2;            %m, Radius of the cylinder
l = V/(pi*r^2);     %m, Length of the Cylinder
Ic_pc = 16250;      %Superconducting coil current density divided by wire density.
w_rad = w * 2*pi/60;%converting rpm to rad/sec

mass_coil = w_rad/(l*Ic_pc)*sqrt(Mass_tot * (R + r)^5 /(3 * 17 * 10^-7));
```

cost.m

```
%This module determines cost of a mars transfer vehicle based on vehicle
%weight.

%References:
% 1. Reynerson, C. "Human Space System Modeling: A tool for designing
%inexpensive Moon and Mars exploration missions"

% 2. SMAD

function [TotalCost, SpaceSegCost, LVCost, OpSupCost]=cost(mass,TRL,duration)

%mass: is total mass of vehicle in kg
%TRL: is technology readiness level and should range from 1 to 9.
%duration: is duration of mission in years

Infl = 1.052; %inflation factor to convert from FY00$ to FY03$ [2]

%Space segment costs

%Space Segment Cost Factor
Scf = 157e3; % ($/Kg) we use maximum value to obtain conservative estimate [1]

%Program level Cost Factor
Pcf = Infl * 1.963*(523e6)^0.841; %($) Program level cost estimated from table 20-4 [2]

%RTDECF of 1 means program based on existing hardware,
%3 means new development program, 2 is somewhere in between [1]
%Rcf = 2;

%Heritage Cost Factor: assume that a TRL of 9 means 90% heritage,
%therefore 10% extra needs to be spent in RDTE, [2] pg 798.
Hcf = 2-TRL/10;

%SpaceSegCost = Scf*Rcf*Hcf*Mass+Pcf;
SpaceSegCost = Scf*Hcf*mass+Pcf;
%*****************************************************************

%Launch Vehicle Cost

%Launch Vehicle cost factor
Lcf = 15.4e3; %($/Kg) [1]

%Insurance cost factor
Icf = 1.5; %for commercial launches it is 1.33, for govt. we are assuming a bit higher number [1]

LVCost = Lcf*Icf*mass;

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

%Ground Operations and Support Costs

OpSupCost = 1.5e9*(duration); %($) ISS operational budget is $13 billion for 10 years [1]


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

TotalCost = SpaceSegCost+LVCost+OpSupCost;
```

References

[1] Dudley-Rowley, Marilyn, et. al., *Crew Size, Composition, and Time: Implications for Exploration Design*, AIAA 2002-6111, AIAA, 2002, p. 4.
[2] ibid, p. 13.
[3] Zubrin, Robert, *Athena: A Potential First Step in a Program of Human Mars Exploration*, AIAA-96-4465, AIAA, 1996, p. 1.
[4] Dudley-Rowley, Marilyn, et. al., *Crew Size, Composition, and Time: Implications for Exploration Design*, AIAA 2002-6111, AIAA, 2002, p. 13.
[5] Conners, M.M., *et al., Living Aloft*, NASA, 1985, p60.
[6] Sloan, James, *Commercial Space Station Requirements*, AIAA-2000-5228, AIAA, 2000, p. 5.
[7] Larson, Wiley, *Human Spaceflight: Mission Analysis and Design*, McGraw-Hill, 1999, Table 18-8.
[8] ibid, p. 554.
[9] Wieland, Paul., *Designing For Human Presence in Space*, NASA Marshall Space Flight Center , 1999, § 2.1.
[10] ibid, Table 17-9, p. 554.
[11] Wieland, Paul., *Designing For Human Presence in Space*, NASA Marshall Space Flight Center , 1999, § 2.4.
[12] ibid, p. 558.
[13] ibid, p. 998.
[14] Borowski, Stanley, and Dudzinsky, Leonard, *Artificial Gravity Design Option for NASA's Human Mars Mission Using "Bimodal" NTR Propulsion*, AIAA-99-2545, AIAA, 1999, p. 3.
[15] ibid, p. 4.
[16] ibid, p. 71.
[17] ibid, p. 994.
[18] Sloan, James, *Commercial Space Station Requirements*, AIAA-2000-5228, AIAA, 2000, p. 5.
[19] Miller, David, Course lecture notes, *Electromagnetic Formation Flight*, Presented 10/2002.
[20] Properties of Selected Materials, http://callisto.my.mtu.edu/my4150/props.html.
[21] Vaughan, A., Figgess, A., *Interaction between mission orbit, space environment, and human needs*, course project, 2003.
[22] Reynerson, C., *Human Space System Modeling: A Tool for Designing Inexpensive Moon and Mars Exploration Missions*, AIAA 2000-5240, AIAA, 2000.
[23] Wertz, J. R. and Larson, W. J. (editors), *Space Mission Analysis and Design, 3rd Edition*, 1999 Microcosm Press, El Segundo California.