16.851 - SATELLITE ENGINEERING MEMORANDUM

TO: 16.851 FACULTY

FROM: STUDENTS

SUBJECT: PROBLEM SET #4 FINAL REPORT

DATE: 6/21/2004

Subject: Telemetry, Communications, and Power

Motivation: A satellite being able to communicate with the ground station is an essential part of the spacecraft mission. The telemetry, communication, and power subsystem collectively establish the satellite's communication link. Various design parameters of Telemetry and Communication impose requirements on the Power subsystem. We will therefore create a tool that aids in making design decisions about optimum frequency and optimum data rate requirements.

Problem Statement: Determine the optimum communication frequency and optimum data rate that minimizes the combined mass requirements of telemetry, communications, and power subsystems.

Approach:

We will perform the following trade-offs:

1. Given frequencies and data rates calculate transmitter power required. Assuming a particular power system, derive total power subsystem mass associated with the required power output. Choose antenna type and size to minimize power required.

2. Assuming a particular power system, derive total power subsystem mass associated with the required power output. Choose the optimum power subsystem that minimizes mass.

3. Given coverage duration and varying data rates determine the mass of the data storage components. Choose data rate that minimizes mass.

Some of the intermediate steps in our study will be:

- Choose a particular power system and determine relationship between power output and power mass.

- Given a particular data mass storage technology that is space-ready, determine the relationship between data storage capacity and its mass.

Data Mass Storage Design Trade-Off

Approach

The aim of this trade-off is to show the relationship between the data rate chosen (if given frequency, data rate can be calculated for the link design equation 13-4 in SMAD) and mass of the data storage needed to hold the data until it can be transmitted down to Earth. We will allow the data rate and coverage time to vary to calculate mass of the data storage and visualize the relationships.

Calculation of Quantity of Data Transmitted:

$$D = R(FT_{max} - T_{init})/M \quad (1)$$

It is assumed that all data collected is transmitted down to the ground station, therefore the data rate specifies how much data the satellite can collect and how much data mass storage is needed to accomplish that size. Equation (1) is from 13-2 in SMAD. As discussed in SMAD, F, the fractional reduction in viewing time, is assumed to be the average value of 80% for satellites in a circular LEO orbit. It is also assumed that Tinit = 120 seconds, a reasonable value for most satellites. In (1), M, the margin needed to account for missed passes due to ground station down time, is assumed to be a conservative value of 2.

Calculation of Mass of Data Storage:

$$m = Dw_f \quad (2)$$

The wf, weighting factor, for the relationship between data storage and mass, was estimated from the current technology in hard drives. Current estimates from Seagate Corporation for wf = 0.397 nanograms/bit assuming we choose a hard-drive disk technology.

Module Description

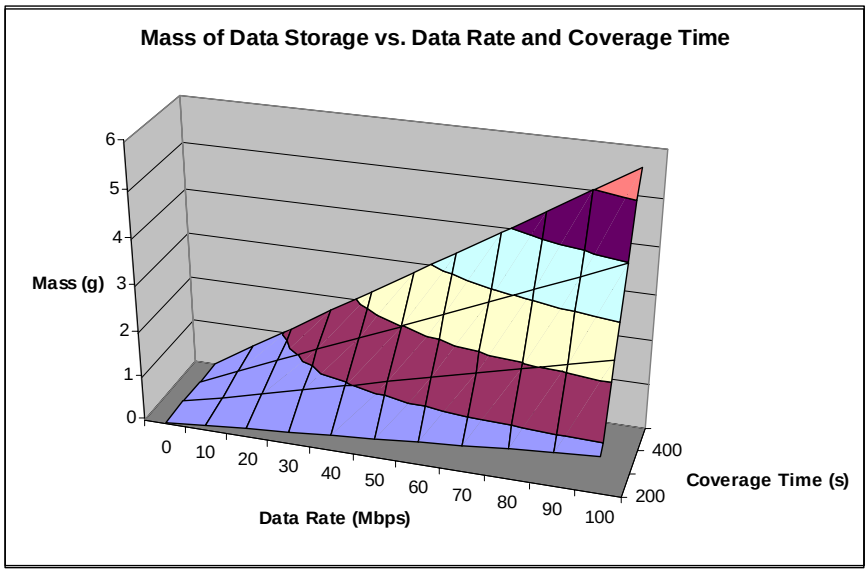
Inputs

The table and equation numbers in parenthesis refer to SMAD.

- *Data rate*: [Mbps] - data rate required
- *T_max* [seconds] – coverage time (amount of time satellite can transmit to ground station)
- *weight factor* [kg/bits] – constant related to data storage device

Outputs

The chart outputted from *datamassstorage_sizing*:



Results:

The mass increases as the data rate is increased. Thus a design that would incorporate this trade would want to use a small data rate to minimize mass. Since frequency and data rate are directly proportional, this means if there is a choice in choosing frequency, one would want to choose a smaller one to minimize mass.

Validation:

The masses calculated are very small. However, we made the assumption we could use the newest hard drives available. In most cases, this can not be used since these technologies have not been space tested and may not be reliable in a space environment. If a data storage technology such as bubble memory is used (which can be considered to be much heavier) on the order of w_f = 50 nanogram/bit, then data storage would have a mass of 2.1 kg for 100 Mbps and 1200 s coverage time. This is within the ranges of mass for C&DH weight for the Firesat example used in Table 11-29 in SMAD.

Transmitter Antenna Design Trade-Off

Approach

The aim of this trade-off is to choose the optimal transmitter antenna design (type and size), minimizing the transmitter power (and hence, the battery and solar array masses). We will allow the link frequency to vary, and compare transmitter powers for different antenna designs of a given mass.

The module first creates a search space, where each search vector represents the performances of the communication system (transmitter power, antenna mass and beamwidth) based on a given frequency and a given antenna type and dimensions. The various antenna designs considered by the module are taken from Table 13-14 in SMAD; they are the following:

- Parabolic reflector
- Helix
- Horn

- Biconical horn.

Note that whereas the dimensions of parabolic reflectors and horns (simple or biconical) are completely defined by the value of one parameter (the diameter), the dimensions of the helixes depend on two parameters: their length (usually referred as their "dimension") and their diameter, which can vary within a given range depending on the frequency.

The module then calculates the performances of each combination of frequency and antenna design, using the method described as follows.

Calculation Of The Bandwidth

The same table in SMAD (Table 13-14) gives the formulae that are used in the model to calculate the half-power bandwidth. They are not repeated here; please refer to SMAD.

Calculation Of The Transmitter Gain

The equations to derive the peak gain from transmitter antenna specifications are also given in Table 13-14 in SMAD. Provided this gain must be within the range of the maximal achievable gain presented in the same table, the actual peak gain will be the minimum between the previously calculated value and the value for the maximum gain corresponding to the given antenna design. To obtain the net transmit antenna gain, we must add to this peak gain (in dB) the losses (in dB) due to a pointing offset:

$$L_\theta = -12\left(e/\theta\right)^2$$

where θ is the antenna half-power beamwidth, and e is the pointing offset.
For biconical horns, whose beamwidth is noncircular, we have used the following equation, suggested by Eq. 13-20 in SMAD:

$$L_\theta = -12\left(e/\theta_x\theta_y\right)^2$$

where θ_x and θ_y are the two angles characterizing the beamwidth.

Calculation Of The Transmitter Power

To derive the transmitter power from the gain, we used Eq. 13-6 in SMAD, repeated here:

$$C = \frac{PL_lG_tL_aD_r^2\eta}{16S^2}$$

where C is the power received by the ground antenna, P is the transmitter power we are to calculate, L_l is the transmitter line loss (NOT in dB), G_t is the transmitting antenna gain (NOT in dB in this equation), L_a is transmission path loss (NOT in dB), D_r is the diameter of the receive antenna, η is the receive antenna efficiency, and S is the propagation path length. We can then solve this equation in P:

$$P = \frac{16S^2C}{L_lG_tL_aD_r^2\eta}$$

To avoid dividing two small numbers (Matlab produced many "Dividing by zero" warnings when testing this equation), and also because many of the variables in the equation are given in dB, the equation was converted to the following:

$$P = 10\log_{10}16 + 20\log_{10}S + 10\log_{10}C - L_l - G_t - L_a - 20\log_{10}D_r - 10\log_{10}\eta$$

where this time L_l, G_t and L_a are in dB, and so is P. P is then converted back into Watts.

Calculation of the mass

To determine the mass equation for each antenna design, we interpolated the data presented in Table 13-16 in SMAD. To do so, the mass m of an antenna has been assumed proportional to the antenna surface A, not taking into consideration extra mass due to more complex feed networks:

$$m \propto A$$

- Parabolic reflector:

$$m \propto A \propto D^2$$

where D is the diameter of the antenna. The data in Table 13-16 gives the following values for the proportionality factor k_p (in kg/m^2): k_p = 7.96 (fixed parabola), k_p = 4.94 and k_p = 6.24 (Intelsat-V parabola w/ feed array). The approximate mean value of k_p = 6 was used in the module to derive antenna mass from its diameter.

- Helix:

$$m \propto A \propto LD$$

where D is the diameter and L is the length of the helix. Data from Table 13-16 gives the approximate value of k_{he} = 11.

- Horn:

$$m \propto A \propto hD$$

where h is the horn "depth" and D is its diameter. We derived k_{ho} = 16 from Table 13-16.

- Biconical horn:

$$m \propto A \propto RD$$

where R is the "depth" of the horn and D its diameter. There is no data in Table 13-16 to derive the proportionality factor, so it was assumed twice larger than factor for a "single conical" horn: $k_b = 2\ k_{ho}$ = 32.

Module Description

Inputs

The table and equation numbers in parenthesis refer to SMAD.

- *f_min*, *f_max* and *f_step*: [GHz] link frequency domain
- *dim_min*, *dim_max* and *dim_step*: [m] antenna dimension domain. This "dimension" is either (see Table 13-14):
  - the dish diameter D for parabolic reflectors
  - the length L for helixes
  - the diameter D for horns
  - the diameter a for biconical horns.
- *received_power*: [W] power received by the ground antenna (written C in Eq. 13-6). The value for this input can be estimated through Eq. 13-8, using data from Table 13-13.
- *altitude*: [m] propagation path length (written S in Eq. 13-6)
- *line_loss*: [dB] transmitter line loss (written L_l in Eq. 13-6)
- *prop_loss*: [dB] propagation and polarization loss (written L_a in Eq. 13-6)

- *receive_diameter*: [m] receive antenna diameter (written D_r in Eq. 13-6)
- *receive_efficiency*: receive antenna efficiency
- *pointing_error*: [deg] transmit antenna pointing offset (written *e* in Eq. 13-21, and e_t in Table 13-13).

The following table is a sample set of test values for the inputs, taken from the "Telemetry and Data" column of Table 13-13 in SMAD. The receive antenna efficiency was taken equal to 0.7, as suggested page 553.

f_min	0.2
f_max	40
f_step	1
dim_min	.5
dim_max	7
dim_step	.1
received_power	6.46E-12
altitude	2,831,000
line_loss	-1
prop_loss	-.3
receive_diameter	5.3
receive_efficiency	0.7
pointing_error	27

TABLE ?. Sample Set Of Test Values For The Inputs

The value for the received power was derived from the following equation (Eq. 13-8 in SMAD):

$$C = (EIRP)L_s L_a G_r$$

where *C* is the power in dB, *EIRP* is the Effective Isentropic Radiated Power from the transmitter (in dB), L_s is the space loss (in dB), L_a is the propagation and polarization loss (in dB), and G_r is the receive antenna gain (in dB). The values for these last four parameters were taken from Table 13-13 in SMAD, in order to get a realistic value for the received power (assumed to be a constant given requirement).

Important note: The propagation and polarization loss was considered a given constant, which is a false assumption, since it strongly depends on the frequency and also on the altitude and elevation, as illustrated on Figures 13-10 and 13-11 in SMAD. But we have had no time to create a model for this dependence; inserting such a model into the module is an important improvement to be made as future work.

Please also note that the user might not want to choose a resolution too high for both the frequency and the dimension, because the search space might then exceed 100MB, which would also result in time-consuming calculations. This is due to the fact that the structure used to represent the search space is not optimal; further work on this module could include optimization of this structure, to fasten calculation and lower memory required.

Outputs

We would first like to emphasize the fact that, since the search space is constructed on a given dimension domain (with a given calculation step), this may result in wide discontinuities in the outputted graph, when the dimensions of the antenna is allowed to vary. Increasing the dimension resolution, i.e. choosing lower calculation steps for the dimension, can lower these discontinuities.

Two different charts can be outputted by the module, representing the relation between power and frequency, respectively for a given mass and beamwidth.

1) Choice of the most power-efficient design depending on the frequency, for a given mass

The module was given the inputs listed in Table ?. The function *PlotPowerRelativeToFrequencyForGivenMass* outputted the following chart.

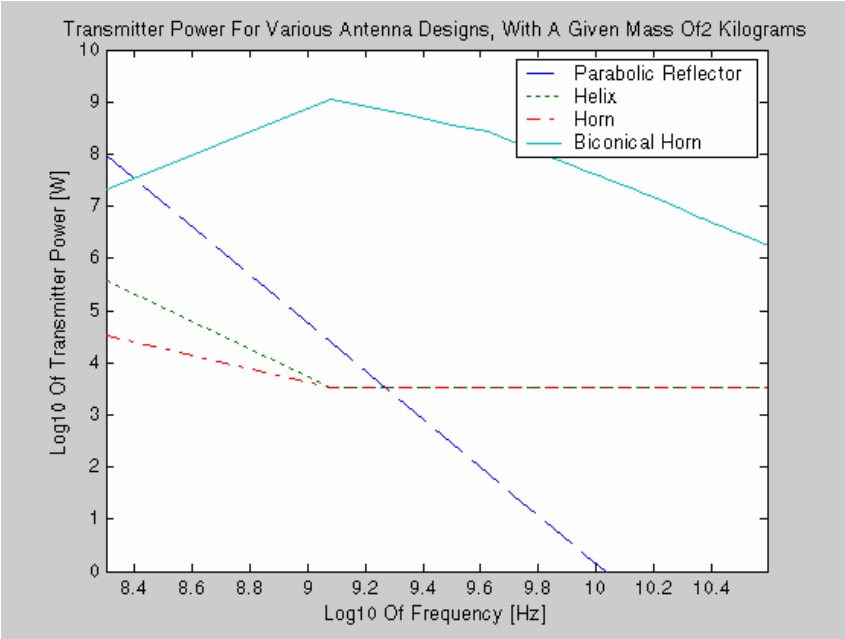


Figure ?. Relation Between Transmitter Power And Frequency (For A Given Mass)

It is difficult to validate this output from the module, since no mass figure is presented in SMAD for the FireSat example (to which the values for the inputs correspond). However, some important remarks can be made about this chart.

First note that on Figure ?, the performances of biconical horns are not very relevant: the maximum gain achievable with this design is too low (for this sample set of inputs), so that the power has to be very high to compensate for the low gain. This graph illustrates the fact that, for a given antenna mass of 2 kilograms, parabolic reflectors are the most efficient design for higher frequencies. For lower frequencies, horns should be preferred. For intermediate frequencies, horns and helixes have similar performances.
Note that this study does not take into account the beamwidth. This is relevant for satellites which are only required to transmit data to a single given ground antenna. For communications systems designed to have a given Earth coverage, this method is

not suitable to choose the antenna type, and the user should rather use the following second method.

2) Choice of the most power-efficient design depending on the frequency, for a given beamwidth:

The module was given the inputs listed in Table ?. The function *PlotPowerRelativeToFrequencyForGivenBeamwidth* outputted the following chart.

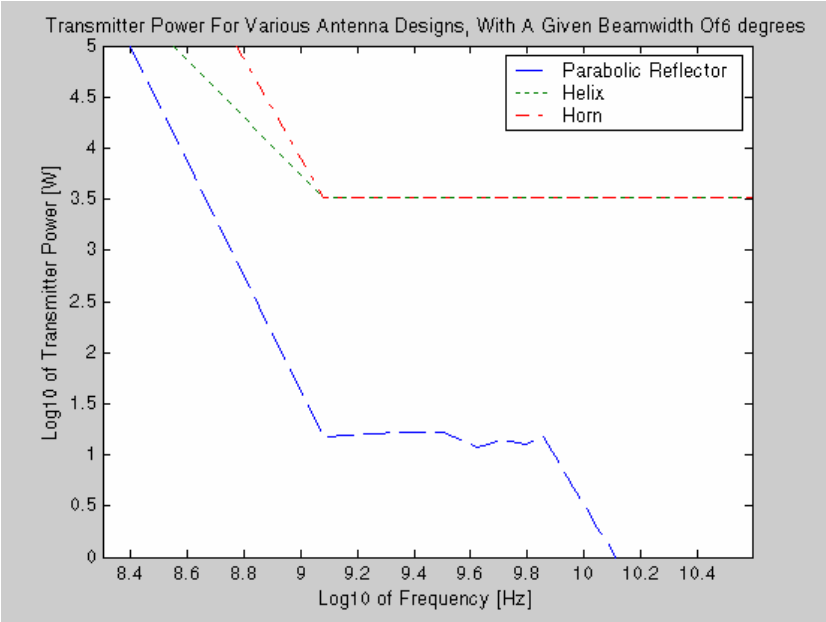


Figure ?. Relation Between Transmitter Power And Frequency (For A Given Beamwidth)

This figure must be compared with Figure 13-12 in SMAD for partial validation. One can see that a similar discontinuity appears for frequencies close to 1GHz: below this value, the dimension of the antenna is limited by the maximum dimension constraint, so the power has to increase as the frequency decreases to compensate the loss in antenna gain. The slope of the line for parabolic reflectors is the same as on Figure 13-12. In this domain of frequencies, the actual reachable beamwidth is higher than the required value (here, 6 degrees).
For frequencies higher than 1GHz, the beamwidth is constantly equal to the required value of 6 degrees, so that the peak gain is constant (equal to the same value as in SMAD), and so is the transmitter power, since the dependence of propagation loss in frequency was not taken into consideration (this is equivalent to the “free space” case in Table 13-12). The behavior of the parabolic reflector line on the chart for high frequencies is due to the fact that we also imposed a lower bound to the dimension domain in order to reduce calculation time.

Future work could be made to output a third graph showing the relation between antenna mass and frequency, for a given beamwidth. Antenna mass and power subsystem mass could also be added up to output a graph enabling the user to choose the antenna and power subsystem designs minimizing the combined mass of the two systems.

List Of The Functions

Details of the inputs and outputs of each function can be found in the extensive description of the function, in the code.

- *CreateBlankSearchSpace*: function creating a "blank" search space, given the frequency and antenna dimension domains
- *FillBlankSearchSpace*: function calling *CalculateGainFromFrequency* and *CalculateTransmitterPowerFromGain* to complete a given blank search space by filling in the missing values. Those values are:
  - dimension parameters for antenna designs depending on the frequency
  - transmitter power
- *CalculateGainFromFrequency*: function deriving peak transmitter gain from frequency, and calculating dimension values for antenna designs depending on frequency
- *CalculateTransmitterPowerFromGain*: function deriving transmitter power from transmitter peak gain and the specifications of the link and of the receive antenna
- *PlotPowerRelativeToFrequencyForGivenMass*: function plotting the transmitter power required by each antenna design, depending on the frequency and on a given antenna mass
- *PlotPowerRelativeToFrequencyForGivenBeamwidth*: function plotting the transmitter power required by each antenna design, depending on the frequency and on a given antenna beamwidth
- *Datamassstorage_sizing*: function calculates mass needed to store data collected in between transmission to ground station.

References –

Wertz, James R. and Wiley J. Larson (Ed.) Space Mission Analysis and Design. El Segundo: Microcosm Press, 1999.

Pisacane, Vincent L. and Robert C. Moore (Ed.) Fundamentals of Space Systems. New York: Oxford University Press, 1994.

www.seagate.com

Codes

```
function search_space = CreateBlankSearchSpace(f_min, f_max, f_step,
dim_min, dim_max, dim_step);

% Function creating a blank search space
% Inputs:
%     f_min       [GHz] frequency lower bound
%     f_max       [GHz] frequency upper bound
%     f_step      [GHz] frequency step
%     dim_min     [m] antenna dimension lower bound
%     dim_max     [m] antenna dimension upper bound
%     dim_step    [m] antenna dimension step
%           Note: The dimension is:
%                 - the diameter for parabolic reflectors
%                 - the length for helixes
%                 - the diameter for horns
%                 - the diameter for biconical horns.
% Output:
%     search_space      matrix where:
```

```
%                 - each row correspond to a given frequency
%                 - each column correspond to a given antenna design
%                 - each element e is of the following structure:
%                       e.frequency       [GHz] frequency
%                       e.power           [W] transmitter power
%                       e.design          structure of the type specs
%                       (see function CalculateGainFromFrequency)

global c    % speed of light IN METERS PER SECOND

f = f_min - f_step;
nf = 0;
while (f + f_step <= f_max)

      % this doesn't work: display(strcat('Processing... ',
int2str(floor((f-f_min)/(f_max-f_min))*100), ' %'))

      % For each frequency / lambda:
      nf = nf + 1;
      f = f + f_step;
      lambda = c / (f * 1000000000);

      % Building up the parabolic reflector space search:
      % display('Parabolic Reflector')
      d = dim_min - dim_step;
      j = 0;
      while (d + dim_step <= dim_max)

            %For each diameter:
            j = j + 1;
            d = d + dim_step;

            search_space(nf, j).frequency = f;
            search_space(nf, j).design.type = 'Parabolic Reflector';
            search_space(nf, j).design.dim = d;

      end

      % Building up the helix space search:
      % display('Helix')
      L = dim_min - dim_step;
      while (L + dim_step <= dim_max)

            %For each length:
            L = L + dim_step;

            d = 0.8 * lambda / pi;
            % display(1.2 * lambda / pi)
            while (d + dim_step <= 1.2 * lambda / pi)
                  % For each diameter in the range allowed by
                  % the frequency: (see TABLE 13-14)
                  j = j + 1;
                  d = d + dim_step;

                  search_space(nf, j).frequency = f;
                  search_space(nf, j).design.type = 'Helix';
                  search_space(nf, j).design.dim.diameter = d;
                  search_space(nf, j).design.dim.length = L;
            end
      end

      % Building up the horn space search:
      d = dim_min - dim_step;
      while (d + dim_step <= dim_max)

            %For each diameter:
            j = j + 1;
            d = d + dim_step;
```

```
            search_space(nf, j).frequency = f;
            search_space(nf, j).design.type = 'Horn';
            search_space(nf, j).design.dim.diameter = d;

      end

      % Building up the biconical horn space search:
      d = dim_min - dim_step;
      while (d + dim_step <= dim_max)

            %For each diameter:
            j = j + 1;
            d = d + dim_step;

            search_space(nf, j).frequency = f;
            search_space(nf, j).design.type = 'Biconical Horn';
            search_space(nf, j).design.dim.diameter = d;

      end

end


% ##########################################

function search_space = FillBlankSearchSpace(received_power,
altitude, line_loss, prop_loss, receive_diameter, receive_efficiency,
pointing_error, blank_space);

% Function filling in an input blank search space with the
appropriate values of power
% Input:
%     received_power    [W] received power
%     altitude          [m] propagation path length (IN METERS!)
%     line_loss         [dB] transmitter line loss
%                            (TABLE 13-13, written Ll)
%     prop_loss         [dB] propagation and polarization loss
%                            (TABLE 13-13, written La)
%     receive_diameter  [m] receive antenna diameter
%     receive_efficiency      receive antenna efficiency
%   pointing_error      [deg] transmit antenna pointing offset
%                            (TABLE 13-13, written et)
%     blank_space       object with the same structure as
%                       search_space in function
%                       CreateBlankSearchSpace
% Output:
%     search_space      object with the same structure as
%                       search_space in function
%                       CreateBlankSearchSpace

search_space = blank_space;
[number_of_rows, number_of_columns] = size(blank_space);

for row = 1 : number_of_rows

      display(strcat('Processing... ', int2str(floor(row /
number_of_rows*100)), ' %'))

      % For each frequency:
      for j = 1 : number_of_columns

            % For each antenna design:
            f = search_space(row, j).frequency;
            if ~isempty(f)
            % The number of antenna devices considered varies with f,
            % since the domain of values for the diameter of a helix
            % depends on f; the greater f, the lower lambda, so the
            % smaller domain for this diameter. So their might be
            % empty elements at the end of a row in search_space. So
```

```
            % regard first empty element in a row as the end of this
            % row.
            specs = search_space(row, j).design;
            [gain, new_specs] = CalculateGainFromFrequency(f,
pointing_error, search_space(row, j).design);
            search_space(row, j).design = new_specs;
            search_space(row, j).power =
CalculateTransmitterPowerFromGain(received_power, altitude,
line_loss, prop_loss, receive_diameter, receive_efficiency, gain);
            end
      end

end


% ########################################

function power = CalculateTransmitterPowerFromGain(received_power,
altitude, line_loss, prop_loss, receive_diameter, receive_efficiency,
gain);

% Function deriving transmitter power from transmitter gain
% using Eq. 13-6 in SMAD
% Inputs:
%     received_power    [W] received power
%     altitude          [m] altitude of the satellite (IN METERS!)
%     line_loss         [dB] transmitter line loss
%                            (TABLE 13-13, written Ll)
%     trans_loss        [dB] transmission path loss
%                            (TABLE 13-13, written La)
%     receive_diameter  [m] receive antenna diameter
%     receive_efficiency      receive antenna efficiency
%     gain              [dBi] transmitter peak gain
% Output:
%     power             [W] transmitter power

% first calculate power in dB-W
power = 10 * log10(16) + 20 * log10(altitude) + 10 *
log10(received_power) - line_loss - gain - prop_loss - 20 *
log10(receive_diameter) - 10 * log10(receive_efficiency);
% convert into Watts
power = 10^(power/10);


% ####################################

function [gain, new_specs] = CalculateGainFromFrequency(f,
pointing_error, specs);

global c    % speed of light IN METERS PER SECOND

% Function deriving the peak gain and other antenna dimensions
% from the frequency and the antenna specs
% using TABLE 13-14 in SMAD
% Inputs:
%     f                 [GHz] Frequency
%     pointing_error    [deg] transmit antenna pointing offset
%                            (TABLE 13-13, written et)
%     specs             object that has the following structure:
%           specs.type  type of the antenna
%                            ('Parabolic Reflector', 'Helix',
%                            'Horn' or 'Biconical Horn')
%           specs.dim   dimensions of the antenna:
%                            diameter of the 'Parabolic Reflector'
%                       OR   diameter, length and width of the
%                                  'Helix'
%                       OR   diameter and depth of the 'Horn'
%                       OR   diameter and semi_depth of the
%                                  'Biconical Horn'
```

```
%           specs.mass  antenna mass
%           specs.beam  beam shape:
%                             'Conical' for 'Parabolic Reflector',
%                                   'Helix' and 'Horn'
%                       OR    'Toroidal' for 'Biconical Horn'
%           specs.beamwidth   [deg]
%           specs.efficiency
%           specs.max_gain    [dB] maximum gain reachable
%                                   for this design (see TABLE 13-14)
% Outputs:
%     gain        [dBi] Transmitter Peak Gain
%     new_specs   it also returns mew_specs because some of the
%                 dimensions of the antennae depend on the frequency

new_specs = specs;
lambda = c / (f * 1000000000);

if strcmp(specs.type, 'Parabolic Reflector')
      new_specs.mass = 6 * specs.dim^2;
      % (the [very] approximate factor 6 was derived from the data in
TABLE 13-16)
      new_specs.beam = 'Conical';
      new_specs.max_gain = 65;
      new_specs.beamwidth = 21 / (f * specs.dim);
      new_specs.efficiency = 0.55;
      gain = 17.8 + 20 * log(specs.dim) + 20 *log(f) - 12 *
(pointing_error / new_specs.beamwidth)^2;
      % The maximum reachable gain is new_specs.max_gain:
      gain = min(gain, new_specs.max_gain);
elseif strcmp(specs.type, 'Helix')
      new_specs.beam = 'Conical';
      new_specs.max_gain = 20;
      lambda = c / (f * 1000000000);
      if (0.8 <= pi * specs.dim.diameter / lambda && pi *
specs.dim.diameter / lambda <= 1.2)
            % The diameter is within the range allowed
            % by the frequency
            new_specs.mass = 11 * specs.dim.diameter *
specs.dim.length;
            % (the [very] approximate factor 11 was derived
            % from the data in TABLE 13-16)
            new_specs.beamwidth = 52 / sqrt((pi *
specs.dim.diameter)^2 * specs.dim.length / lambda^3);
            new_specs.dim.width = 0.8 * lambda;
            new_specs.efficiency = 0.7;
            gain = 10.3 + 10 * log((pi * specs.dim.diameter)^2 *
specs.dim.length / lambda^3) - 12 * (pointing_error /
new_specs.beamwidth)^2;
            % The maximum reachable gain is new_specs.max_gain:
            gain = min(gain, new_specs.max_gain);
      else
            % The diameter and the frequency are inconsistent ;
            % return zero gain:
            gain = 0;
      end
elseif strcmp(specs.type, 'Horn')
      new_specs.beam = 'Conical';
      new_specs.max_gain = 20;
      new_specs.beamwidth = 225 / (pi * specs.dim.diameter / lambda);
      new_specs.dim.depth = specs.dim.diameter^2 / (3 * lambda);
      new_specs.mass = 16 * specs.dim.diameter * new_specs.dim.depth;
      % (the [very] approximate factor 16 was derived
      % from the data in TABLE 13-16)
      new_specs.efficiency = 0.52;
      gain = 20 * log(pi * specs.dim.diameter / lambda) - 2.8 - 12 *
(pointing_error / new_specs.beamwidth)^2;
      % The maximum reachable gain is new_specs.max_gain:
      gain = min(gain, new_specs.max_gain);
else
```

```
      new_specs.beam = 'Toroidal';
      new_specs.max_gain = 5;
      new_specs.dim.semi_depth = 2 * lambda;
      new_specs.mass = 32 * specs.dim.diameter *
new_specs.dim.semi_depth;
      % (the factor 32 was taken double to the factor
      % for a simple horn antenna)
      gain = 5 * log(specs.dim.diameter / lambda);
      if (gain > -1)
            new_specs.beamwidth = '40° * 360°';
            gain = gain  - 12 * (pointing_error / 40 * 360)^2;
            % The maximum reachable gain is new_specs.max_gain:
            gain = min(gain, new_specs.max_gain);
      else
            new_specs.beamwidth = '70° * 360°';
            gain = gain  - 12 * (pointing_error / 70 * 360)^2;
            % The maximum reachable gain is new_specs.max_gain:
            gain = min(gain, new_specs.max_gain);
      end
      new_specs.efficiency = 0.5;   % not in TABLE 13-14;
                                    % assumed worst-case efficiency
end


% ############################

function PlotPowerRelativeToFrequencyForGivenMass(search_space, m,
max_power_plot);

% Function creating a chart of transmitter power relative to
% frequency, for a given value of mass m
% Inputs:
%     search_space      object which structure is described
%                       in function CreateBlankSearchSpace
%     m                 [kg] mass of the antenna.
%     max_power_plot    [W] plotted power domain upper bound
% No outputs; the chart is directly exported to a figure
% named power_for_given_mass

[number_of_rows, number_of_columns] = size(search_space);

for row = 1 : number_of_rows

      % For each frequency (plotted on a log axis):
      f(row) = log10(search_space(row, 1).frequency * 1000000000);

      j = 1;

      % Get the parabolic reflector design whose mass is
      % about the closest to m:
      while ((search_space(row ,j).design.mass < m) &&
strcmp(search_space(row ,j).design.type, 'Parabolic Reflector'))
            j = j + 1;
      end
      if (search_space(row ,j).design.mass >= m)
            % Executed only if one design of mass equal or
            % greater than m was found
            power_parab(row) = log10(search_space(row, j).power);
      else
            % Otherwise, return Inf as the value for the power
            power_parab(row) = Inf;
      end

      % Skip the other parabolic reflector designs:
      while strcmp(search_space(row ,j).design.type, 'Parabolic
Reflector')
            j = j + 1;
      end
```

```
      % Get the helix design with smallest length whose mass
      % is about the closest to m: (we want to minimize the power,
      % which decreases when the gain increases, and the gain
      % increases with C2*L, which, for a given mass=C*L, is maximum
      % for L minimum)
      while (search_space(row ,j).design.mass < m &&
strcmp(search_space(row ,j).design.type, 'Helix'))
            j = j + 1;
      end
      if (search_space(row ,j).design.mass >= m)
            % Executed only if one design of mass equal or
            % greater than m was found
            power_helix(row) = log10(search_space(row, j).power);
      else
            % Otherwise, return Inf as the value for the power
            power_helix(row) = Inf;
      end

      % Skip the other helix designs:
      while strcmp(search_space(row ,j).design.type, 'Helix')
            j = j + 1;
      end


      % Get the horn design whose mass is about the closest to m:
      while (search_space(row ,j).design.mass < m &&
strcmp(search_space(row ,j).design.type, 'Horn'))
            j = j + 1;
      end
      if (search_space(row ,j).design.mass >= m)
            % Executed only if one design of mass equal or
            % greater than m was found
            power_horn(row) = log10(search_space(row, j).power);
      else
            % Otherwise, return Inf as the value for the power
            power_horn(row) = Inf;
      end

      % Skip the other horn designs:
      while strcmp(search_space(row ,j).design.type, 'Horn')
            j = j + 1;
      end


      % Get the biconical horn design whose mass is about
      % the closest to m:
      while (search_space(row ,j).design.mass < m && j <
number_of_columns)
            j = j + 1;
            if  isempty(search_space(row ,j).frequency)
                  search_space(row ,j).design.mass = 0;
                  break
            end
      end
      if (search_space(row ,j).design.mass >= m)
            % Executed only if one design of mass equal or
            % greater than m was found
            power_bic(row) = log10(search_space(row, j).power);
      else
            % Otherwise, return Inf as the value for the power
            power_bic(row) = Inf;
      end

end

% Create and export the plot:
h = plot(f,power_parab,f,power_helix,f,power_horn, f, power_bic);
axis([-Inf Inf 0 log10(max_power_plot)]);
```

```
set(h, {'LineStyle'}, {'--'; ':'; '-.'; '-'});
xlabel('Log10 Of Frequency [Hz]');
ylabel('Log10 Of Transmitter Power [W]');
legend(h, 'Parabolic Reflector', 'Helix', 'Horn', 'Biconical Horn');
title(strcat('Transmitter Power For Various Antenna Designs, With A
Given Mass Of ', int2str(m), ' Kilograms'));
print -depsc -tiff -r200 power_for_given_mass


% ################################

function PlotPowerRelativeToFrequencyForGivenBeamwidth(search_space,
theta, max_power_plot);

% Function creating a chart of transmitter power relative to
% frequency, for a given value of beamwidth theta
% Inputs:
%     search_space      object which structure is described
%                       in function CreateBlankSearchSpace
%     theta             [deg] half-power transmitting antenna
%                       beamwidth
%     max_power_plot    [W] plotted power domain upper bound
% No outputs; the chart is directly exported to a figure
% named power_for_given_beamwidth

[number_of_rows, number_of_columns] = size(search_space);

for row = 1 : number_of_rows

      % For each frequency (plot frequency on a log axis):
      f(row) = log10(search_space(row, 1).frequency * 1000000000);

      j = 1;

      % Get the parabolic reflector design whose beamwidth is
      % about the closest to theta:
      while ((search_space(row ,j).design.beamwidth > theta) &&
strcmp(search_space(row ,j).design.type, 'Parabolic Reflector'))
            j = j + 1;
      end
      power_parab(row) = log10(search_space(row, j).power);

      % Skip the other parabolic reflector designs:
      while strcmp(search_space(row ,j).design.type, 'Parabolic
Reflector')
            j = j + 1;
      end


      % Get the helix design with smallest length whose beamwidth is
      % about the closest to theta:
      while (search_space(row ,j).design.beamwidth > theta &&
strcmp(search_space(row ,j).design.type, 'Helix'))
            j = j + 1;
      end
      power_helix(row) = log10(search_space(row, j).power);

      % Skip the other helix designs:
      while strcmp(search_space(row ,j).design.type, 'Helix')
            j = j + 1;
      end


      % Get the horn design whose beamwidth is about the closest
      % to theta:
      while (search_space(row ,j).design.beamwidth > theta &&
strcmp(search_space(row ,j).design.type, 'Horn'))
            j = j + 1;
            if ~(strcmp(search_space(row ,j).design.type, 'Horn'))
```

```
                    % If no horn design was found and we got to
                    % the biconical horn designs (whose beamwidth is a
                    % STRING, not a number), then break.
                    break
              end
        end
        power_horn(row) = log10(search_space(row, j).power);


        % (The biconical horn is not considered because its beamwidth
        % is fixed)

end

% Create and export the plot:
h = plot(f,power_parab,f,power_helix,f,power_horn);
axis([-Inf Inf 0 log10(max_power_plot)]);
set(h, {'LineStyle'}, {'--'; ':'; '-.'});
xlabel('Log10 of Frequency [Hz]');
ylabel('Log10 of Transmitter Power [W]');
legend(h, 'Parabolic Reflector', 'Helix', 'Horn');
title(strcat('Transmitter Power For Various Antenna Designs, With A
Given Beamwidth Of ', int2str(theta), ' degrees'));
print -depsc -tiff -r200 power_for_given_beamwidth

##################################################################

function mass = datamassstorage_sizing(data_rate, t_max,
weight_factor)

 % data_rate [bps] - data rate required
 % t_max [seconds] - coverage time (amount of time satellite can
transmit
 % to ground station
 % weight_factor [kg/bytes] - constant related to data storage device


 % constants
 F = 0.8; %from SMAD this is an average value for satellites in a
circular LEO-Earth orbit
 t_init = 120; %seconds, estimate from SMAD
 M = 2; % estimate from SMAD, this is a conservative estimate unless
it is a dedicated ground
 %station with a specified value for the percentage of pass time that
will
 %be used for collecting data

 data_quantity = data_rate*(F*t_max - t_init)/M;

 mass = data_quantity*weight_factor;
```