NASA/TM—2019-220233

Space Communications and Navigation (SCaN) Tool Capabilities Updates for Antenna Pattern and Jitter

Alay M. Shah and Bryan W. Welch
Glenn Research Center, Cleveland, Ohio

October 2019

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Technical Report Server—Registered (NTRS Reg) and NASA Technical Report Server—Public (NTRS) thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers, but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., "quick-release" reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 757-864-6500

- Telephone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Program
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM—2019-220233

Space Communications and Navigation (SCaN) Tool Capabilities Updates for Antenna Pattern and Jitter

Alay M. Shah and Bryan W. Welch
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

October 2019

Acknowledgments

Development of the tools herein was a collaborative effort with Bryan Welch of NASA's Space Communications and Navigation (SCaN) office. The reflector tool was originally adapted by Bryan Welch from "Antenna Engineering Using Physical Optics: Practical CAD Techniques and Software" by Leo Diaz and Thomas Milligan. The effort was supported by the Lewis' Educational and Research Collaborative Internship Project (LeRCIP) and the SCaN Intern Project (SIP) at NASA Glenn Research Center.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA STI Program
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
703-605-6000

This report is available in electronic form at http://www.sti.nasa.gov/ and http://ntrs.nasa.gov/

Space Communications and Navigation (SCaN) Tool Capabilities Updates for Antenna Pattern and Jitter

Alay M. Shah* and Bryan W. Welch
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Summary

The ability to accurately simulate the dynamic misalignment of pointing across a communications link is critical to determining the link performance between or across spacecraft and ground stations. The communications system antenna pattern and dynamic misalignment are key components to understanding the pointing loss induced on the communications link. Currently, projects performing these analyses use a variety of simulation tools and packages. NASA's Space Communications and Navigation (SCaN) office aims to provide a unified tool set for these functions. However, the analysis tool to calculate antenna patterns is currently static, two-dimensional, and approximated based on limited input. Steps were taken to change the inputs used by the tool to create a more accurate approximation of the antenna's far-field pattern as well as create a three-dimensional pattern. An additional tool was developed to calculate time-dynamic pointing loss from the antenna pattern data, generated or observed, given dynamic pointing errors.

Symbols

β	binormal-off-normal angle; angle between binormal vector and normal vector of target
$\mathbf{b}$	binormal vector
c	speed of light
D	diameter of reflector
D_{lam}	size of aperture in units of number of wavelengths
f	frequency
F	focal length
F/D	ratio of focal length to reflector diameter
$\mathbf{n}$	normal vector
$\hat{n}$	normal unit vector
T	antenna feed taper
ϖ	velocity-off-normal angle; angle between velocity vector and normal vector of target
$\mathbf{v}$	velocity vector
ϕ	angle with respect to the positive x-axis in the x,y-plane
θ_r	θ vectors (array of θ angles)
θ	angle between the x,y-plane and vector being described
ϕ_r	ϕ vectors (array of ϕ angles)
λ	wavelength

*Spring Intern in Lewis' Educational and Research Collaborative Internship Project (LeRCIP).

1.0 Introduction

Link analysis requires knowledge of an antenna's far-field response. This phenomenon, also known as the Fraunhofer diffraction pattern, contains the angular variation in signal response measured by gain for a respective antenna (Refs. 1 and 2). The pattern can vary depending on the type of reflector, including a reflector's shape, size, diameter, focal length, and other characteristics. A plot of example data of a far-field pattern is pictured in Figure 1. A previous tool was developed by NASA's Space Communications and Navigation (SCaN) office, which approximated the antenna feed-induced fields based on a cosine function in a limited angular tradespace.

Jitter is a time-dynamic misalignment of the source and target (Refs. 1 to 3). The misalignment leads to an alteration in the signal. A beam smeared by jitter may have an irregular or spread out shape, with a higher edge response and lower peak (Ref. 3). This is illustrated in Figure 2. It causes pointing loss, a degradation in signal strength resulting from the source straying away from the target or viceversa. Pointing loss affects message transmission, and SCaN requires the capability to analyze communications links with detailed knowledge of the impact of pointing error to obtain an accurate link analysis. For a dynamic link analysis, pointing error data is available as a function of time, and the calculated pointing loss must be computed across the same pattern using an efficient search process. The pointing loss tool was developed in MATLAB® (The Mathworks, Inc.) for easy integration with existing tools. The end goal of SCaN's analysis capabilities is to be able to use a communications system node's position, velocity, attitude state, and pointing direction in space to determine a more accurate link performance simulation.

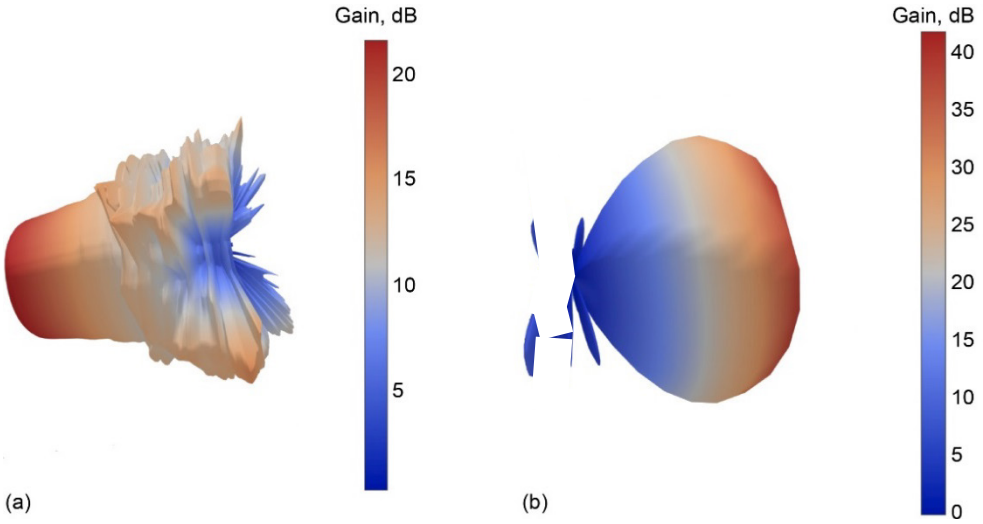


Figure 1.—Antenna far-field patterns. (a) Measured. (b) Idealized.

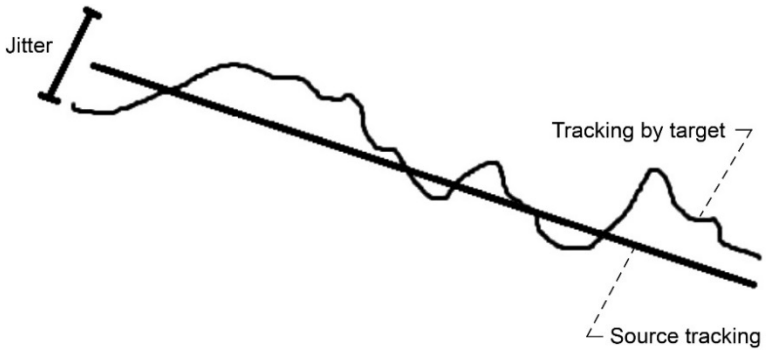


Figure 2.—Time-dynamic misalignment of antenna tracking adapted from Reference 3.

2.0 Methods

The changes to the parabolic reflector calculation process are described in this section. Additionally, a short introduction to the new pointing loss calculation process is given, which will be further described in Section 3.0.

2.1 Updating Parabolic Reflector Approximation

Originally, the parabolic reflector tool used the inputs of the ratio of focal length to reflector diameter F/D, the aperture size in units of number of wavelengths D_{lam}, and the antenna feed taper T, then utilized a combination of hard-coded variables and a number of limited inputs to calculate the approximate far-field pattern. Then, it plotted a graph of the pattern and its characteristics, such as directivity, peak gain, and side lobe information. The original inputs and outputs are shown in Figure 3. The parabolic reflector code was originally developed by Diaz and Milligan (Ref. 4) and was updated to use a more efficient and detailed grid for calculating antenna patterns. It also contained different inputs, some of which were hard-coded for quicker development.

The code used inefficient calculation methods in subroutines, such as "For loops." The code was converted into a vectorized format in an effort to increase efficiency, although due to the large quantity of data processed, the vectorized code's efficiency improvement was marginal or worse than the For loop. This is discussed further in the results. The iterating and vectorized codes are shown in Figure 4.

The output was converted into a matrix that could be read by other dynamic link analysis tools, such as the pointing loss tool developed and discussed in this report.

2.2 Pointing Loss Calculation Tool

A new tool was created to perform dynamic analysis of pointing loss given a time-dynamic pointing error in a $n\times2$ matrix and a far-field pattern matrix, measured or approximated, containing angular coordinates and gain values. The output can be written to a .mat file or saved into memory for use by another program for easy integration into other dynamic analysis tools.

3.0 Results

Results of the runtime improvements made to the parabolic reflector calculation process are given in this section. Additionally, some of the pointing loss calculation process equations to convert the angles are provided, followed by a workflow diagram of the pointing loss calculation process.

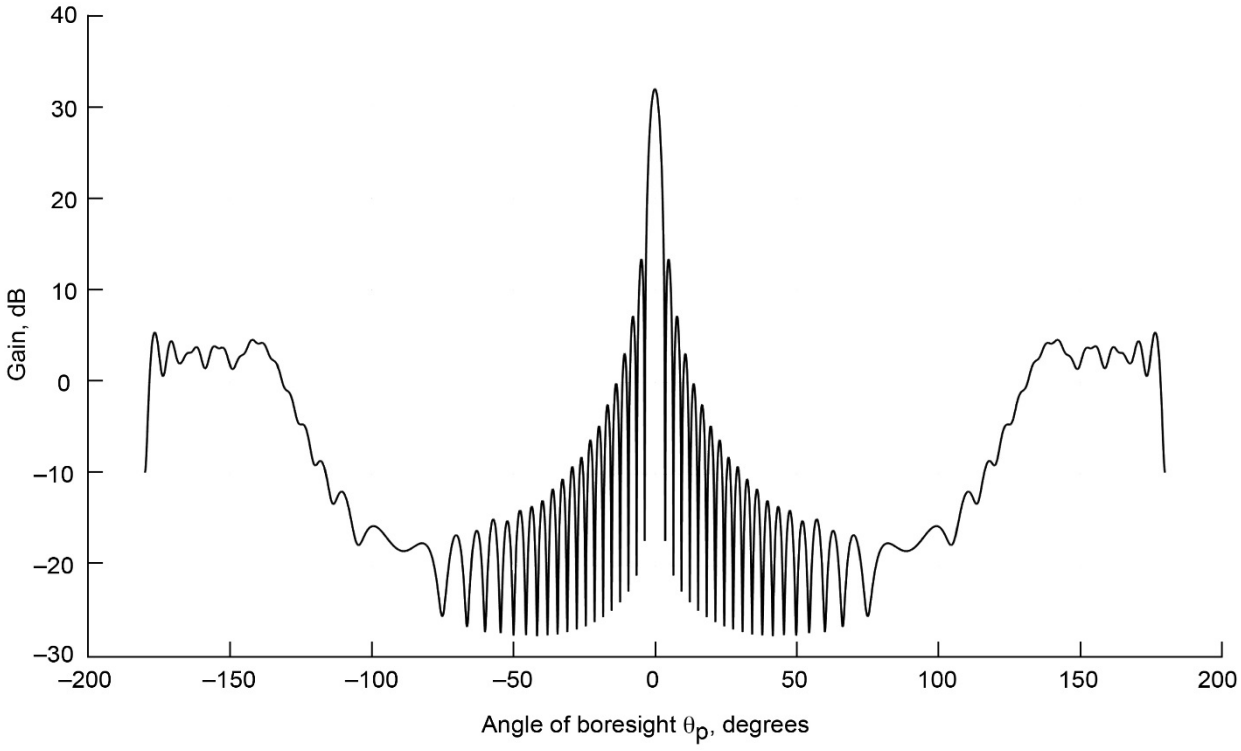
3.1 Optimizing Parabolic Reflector Simulation

One of the first steps was to optimize the subroutines to generate a grid and approximate the pattern induced by the antenna feed, a process that can take anywhere from 15 s to 10 to 20 min depending on the size of the angular tradespace. Screenshots of a comparison between the two versions of baseline antenna pattern analysis code are shown in Figure 4. A majority of the calculations remain the same, with element-wise operators used in lieu of iterating over a list and completing the calculations one by one. MATLAB® is a programming language that is more efficient with matrix computations. However, when the For loop code was converted into a vectorized version, the runtime was longer.

```
function [Directivity,PeakGain,efficiency,HP_angle,First_Null_angle,...
    First_Side_Lobe_angle,First_Side_Lobe_Level_delta]=originalParabolicPO(D_lam,...
    f_over_D,feed_taper)

%Defines constants
EO = 8.8542e-12;
UO = 4*pi*1e-7;
ZO = sqrt(UO/EO);
C = (sqrt(EO*UO))^(-1);
%Model electrical parameters
FREQ = 11.803e9;
LAM = C/FREQ;
W = 2*pi*FREQ;
KO = W*sqrt(UO*EO);
%Reflector physical parameters
D = D_lam*LAM;
F = f_over_D*D;
xa0=0;                          % aperture center x location
ya0=0;                          % aperture center y location
zv0=-F;                        % paraboloid vertex z location
% max allowed wavelength sampling spacing
dmax=1/8;
```

(a)



(b)

Figure 3.—Original inputs and outputs of code. (a) Original code header. (b) Pattern generated by original code.

```
function [E_co_p_f, E_x_p_f] = calcCoXPolPatterns(theta, phi, rr , KO, Hinc_x, Hinc_y, Hinc_z, normal_x
%For loop which will calculate the radiation pattern as a function of the angle off broadside (theta_p)
if ~exist('phi', 'var'), phi = 0; end
for iii = 1:length(phi)
    %Radiation pattern cut
    phi_p_deg = phi(iii);
    phi_p = pi/180*phi_p_deg;
    %Determines repeated cos and sin functions of phi_p
    cpp = cos(phi_p);
    spp = sin(phi_p);
    cpp2 = cpp^2;
    spp2 = spp^2;

    Js_x = 2.*(normal_y.*Hinc_z-normal_z.*Hinc_y);
    Js_y = 2.*(normal_z.*Hinc_x-normal_x.*Hinc_z);
    Js_z = 2.*(normal_x.*Hinc_y-normal_y.*Hinc_x);

    rr_1_cpp = rr(:,1).*cpp;
    rr_2_spp = rr(:,2).*spp;
    rr_12_cppspp_sum = -rr_1_cpp + rr_2_spp;

    Js_x_dSr = Js_x.*dSr;
    Js_y_dSr = Js_y.*dSr;
    Js_z_dSr = Js_z.*dSr;

    for ii = 1:length(theta)
        theta_p = pi/180*theta(ii);
        %Determines repeated cos and sin functions of theta_p
        stp = sin(theta_p);
        ctp = cos(theta_p);
        stp2 = stp^2;
        ctp2 = ctp^2;

        %Defines exponential term of the integrand
        %%%NOTE:  This term (templ) below changes for non-parabolic
        % templ = r_s.*(sts.*cps.*stp.*cpp - sts.*sps.*stp.*spp - cts.*ctp);
        templ = rr_12_cppspp_sum.*stp + rr(:,3).*ctp;
        expo = exp(1i*KO*templ);
        %Defines the integrand to be numerically integrated
        Fx = sum(Js_x_dSr.*expo);
        Fy = sum(Js_y_dSr.*expo);
        Fz = sum(Js_z_dSr.*expo);
```

(a)

```
]function [E_co_p_vect, E_x_p_vect] = calculateCoXPolPatternsV(theta, phi,rr, KO, Hinc_x, Hinc_y, Hinc_z, normal_x, n
 vectorPreallocationLen = length(phi) * length(theta);

 finalThetaPhiPolMat = zeros(vectorPreallocationLen, 4);

]for ind = 1:length(phi)
     phi_p_deg = phi(ind);
     theta_p_vect = pi/180 .* theta;

     finalThetaPhiPolMat((length(theta)*(ind-1) + 1):(length(theta)*ind),1) = (ones(length(theta),1) .* phi_p_deg)';
     finalThetaPhiPolMat((length(theta)*(ind-1) + 1):(length(theta)*ind),2) = (theta_p_vect)';

     %Radiation pattern meshcut
     phi_p = pi/180*phi_p_deg;

     %Determines repeated cos and sin functions of phi_p
     cpp = cos(phi_p);
     spp = sin(phi_p);
     cpp2 = cpp.^2;
     spp2 = spp.^2;

     Js_x = 2.*(normal_y.*Hinc_z-normal_z.*Hinc_y);
     Js_y = 2.*(normal_z.*Hinc_x-normal_x.*Hinc_z);
     Js_z = 2.*(normal_x.*Hinc_y-normal_y.*Hinc_x);

     rr_1_cpp = rr(:,1).*cpp;
     rr_2_spp = rr(:,2).*spp;
     rr_12_cppspp_sum = -rr_1_cpp + rr_2_spp;

     Js_x_dSr = Js_x.*dSr;
     Js_y_dSr = Js_y.*dSr;
     Js_z_dSr = Js_z.*dSr;

     % Determines repeated cos and sin functions of theta_p
     stp_vect = sin(theta_p_vect);
     ctp_vect = cos(theta_p_vect);
     %   Square sin & cos functions
     stp2_vect = stp_vect.^2;
     ctp2_vect = ctp_vect.^2;
     % Define exponential term of the integrand
     templ_vect = rr_12_cppspp_sum.*stp_vect + rr(:,3).*ctp_vect;
     %   ***NB: term templ changes for non-parabolic***
```

(b)

Figure 4.—Iterating versus vectorized subroutines. (a) Original subroutine with For loop. (b) Vectorized pattern calculation.

System performance was examined. It was found that rather than maximizing usage of the central processing unit (CPU), like during the iterating code, random access memory (RAM) and disk activity increased to 100 percent. The components of both the iterating and vectorized codes were timed and plotted. An attempt was made to mitigate the intense load on system resources by piecing a large vector into smaller chunks and running the vectorized code in a For loop format, but this yielded only a marginal improvement. The hybrid-vectorized code only had an improved calculation time over the For loop at a size of 7,200, with the improvement of 2 s less out of 15 s of runtime. The results of the relative percentages of time (runtimes for each case compared concurrently, where all cases are compared relative to the worst case for that calculation) used to calculate the patterns for 7,200 points and a dataset exhibiting a longer calculation time for vectorized code (3,600 points) are plotted in Figure 5. This hybrid-vectorized approach may be used as a framework for developing parallelization and improving calculation times.

Because of this marginal improvement, the code is configured to use For loop code by default unless otherwise specified by the user. Both the vectorized and iterating calculation of the vector fields are shown in Figure 4. Figure 5 demonstrates that while the vectorized code was faster at performing the calculations than the For loop, when performing an analysis that required multiple vectors to be pulled in and out of memory, calculation time increased drastically for the vectorized code. The vectorized and iterating codes were then put into two separate functions, with the For loop being used as the default unless switched by the user. Unless running the code several times, the performance difference would be negligible, however.

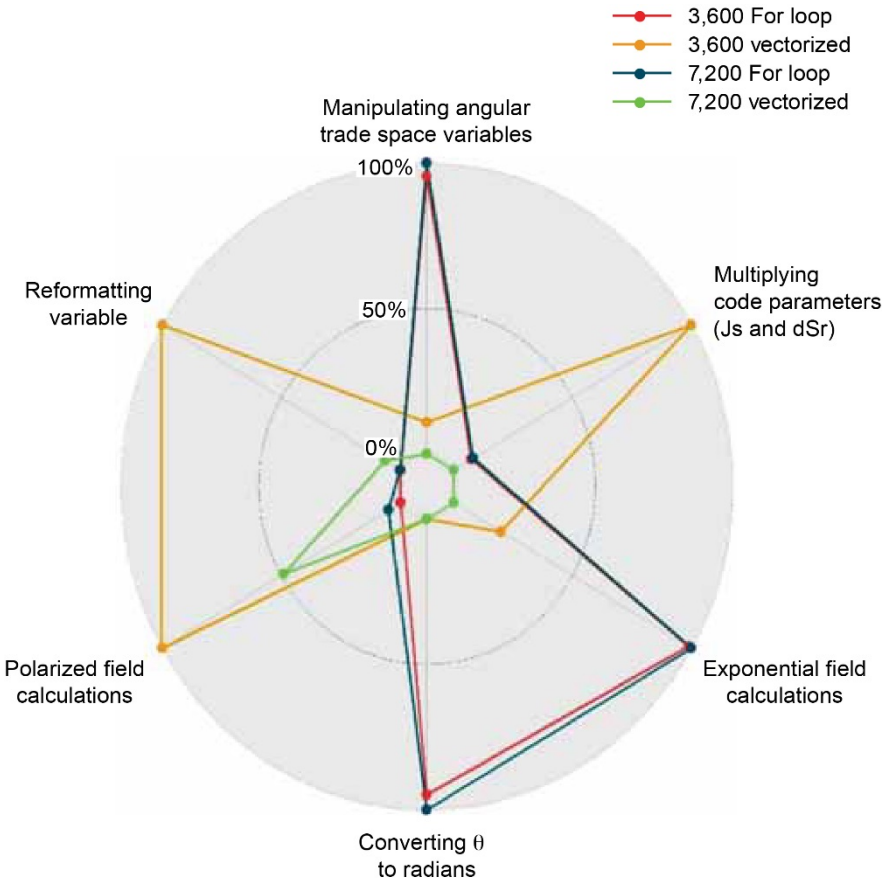


Figure 5.—Relative percent time taken to calculate components of each code. θ = angle between the x,y-plane and vector being described.

3.2 Adjusting Parabolic Reflector Inputs and Outputs

Previously, the parabolic reflector function was only capable of performing analysis in a limited two-dimensional angular tradespace that the user manually defined in the code. Capability was added to specify the angle with respect to the positive x-axis in the x,y-plane (ϕ) and the angle between the x,y-plane and the vector being described (θ) manually as a vector or single number. This allows for a higher fidelity dynamic link analysis. Further modification includes a change from using a hard-coded wavelength and frequency to the user specifying frequency *f* in GHz. Wavelength λ is then calculated after *f* is converted from GHz using the speed of light *c*:

$$\lambda = \frac{c}{f} \tag{1}$$

The tool then converts its output into a θ-ϕ-gain matrix, which can be saved by the user or input directly into the pointing loss tool. This ensures a seamless transition with both existing ways to measure data as well as future tools in the dynamic analysis toolkit.

3.2.1 Future Work

Possible next steps for the parabolic reflector code would be adding additional antenna configurations, such as various feed and reflector shapes, Cassegrain support, and different antenna types. The old reflector simulation was made more compatible with these future changes by compartmentalizing more of the code into functions.

3.3 Developing Pointing Loss Calculation Tool

The pointing loss tool, calculatePointingLoss, takes as inputs the antenna pattern in matrix form, the string pattern type (what order the variables (ϕ vectors (ϕ_r), θ vectors (θ_r), and gain) are in), the vector pointing error, and the pointing error type (a string identifying what type of error was input).

After the ϕ_r, θ_r, and gain vectors are parsed from the matrix based on the information in the pattern type string, the gain is normalized. Potential error formats include a (θ, ϕ) vector and a vector containing the angles between the binormal and normal vectors and the velocity and normal vectors (β and ϖ, respectively). The β and ϖ angles need to be converted as follows into θ and ϕ so that the pointing loss can be interpolated and the pointing error can be calculated. First, the binormal (**b**), velocity (**v**), and normal (**n**) vectors are determined:

$$\mathbf{b} = \sin\beta \tag{2}$$

$$\mathbf{v} = \sin\varpi \tag{3}$$

$$\mathbf{n} = \cos\varpi \tag{4}$$

Then the normal vector is turned into a normal unit vector ($\hat{n}$):

$$\hat{n} = \frac{\mathbf{n}}{\sqrt{\varpi^2 + \beta^2 + \mathbf{n}^2}} \tag{5}$$

Finally, the θ and ϕ are calculated:

$$\theta = \cos \mathbf{v} \cdot \mathbf{b} \tag{6}$$

$$\phi = \text{arc} \ \ \tan \frac{\mathbf{v}}{\mathbf{b}} \tag{7}$$

Once θ and ϕ are calculated, the tool selects whether to use a two- or three-dimensional interpolation depending on the antenna pattern's size and the pointing error type. Pointing loss is then calculated and output into a matrix. The inputs also go through error and logical checks to prevent user error from crashing the tool. A structure was not used because the calculatePointingLoss tool only has the one input and so would likely not benefit from this. A different version of the tool was also developed to take a structure as an input for better integration with other SCaN analysis tools.

3.3.1 Future Work

It is important to note that for antenna patterns that are measured, such as the one found in Figure 1(a), MATLAB® functions' (interp1 and interp2) use of spline interpolation may result in a value that is not completely representative of the surrounding points because of data points straying from the majority of the measured pattern. This may be improved in the future by adding a wider interpolation process or smoothing the antenna pattern.

3.3.2 Verification Testing and Documentation

Pursuant to the NASA Standard for Modeling and Simulation, 7009a (Ref. 5), documentation of the code with comments and activity diagrams was completed. An overview of the calculatePointingLoss tool in the format of an activity diagram is shown in Figure 6. Unit Verification Testing was also completed and the results entered into Table 1.

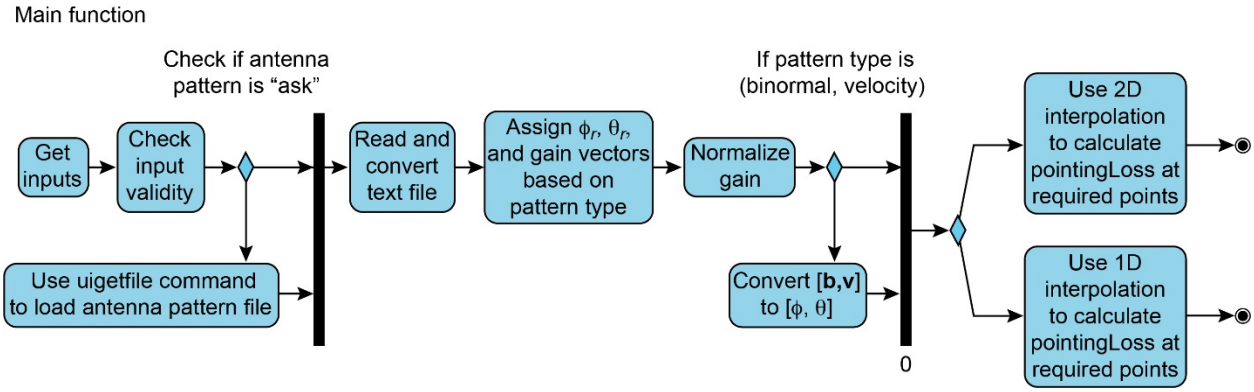


Figure 6.—Activity diagram of "calculatePointingLoss" tool. **b** = binormal vector. **v** = velocity vector. ϕ = angle with respect to the positive x-axis in the x,y-plane. θ_r = θ vectors (array of θ angles). θ = angle between the x,y-plane and vector being described. ϕ_r = ϕ vectors (array of ϕ angles).

TABLE 1.—UNIT VERIFICATION TESTING

Test cases	Test scenarios[a]	Successful
1, 2, 3	Does the θ_r, gain, and ϕ_r (if needed) work in different orders?	Yes
4, 5	Do all pointing errors work?	Yes
6, 7, 8, 9	Pointing error is greater than 180°	Yes
10, 11, 12, 13	Pointing error is negative	Yes
14, 15	Do (θ,ϕ) and (**b**,**v**) pointing errors work with two-dimensional patterns?	Yes

[a]ϕ = angle with respect to the positive x-axis in the x,y-plane. θ_r = θ vectors (array of θ angles). θ = angle between the x,y-plane and vector being described. ϕ_r = ϕ vectors (array of ϕ angles). **b** = binormal vector. **v** = velocity vector.

4.0 Conclusion

The parabolic reflector simulation and calculatePointingLoss tool are two important pieces crucial to a unified toolkit for determining link performance dynamically and providing analysis for its stakeholders, especially with the larger angular tradespace and more specific inputs. The current limitations of the reflector code are such that higher-fidelity far-field pattern approximation is required to reliably predict link performance and reasonably account for the different designs used by spacecraft or ground stations. Optical link analysis is also a capability that the toolset will be able to adapt to. Validation should also be completed on the tool to further improve credibility.

References

1. Balanis, Constantine A.: Antenna Theory: Analysis and Design. Second ed., John Wiley & Sons, New York, NY, 1997.
2. Balanis, C.A.: Antenna Theory: A Review. Proc. IEEE, vol. 80, no. 1, 1992, pp. 7–23.
3. Masson, C.: Submillimeter Array Technical Memorandum #29. Harvard-Smithsonian, 1990.
4. Diaz, Leo; and Milligan, Thomas: Antenna Engineering Using Physical Optics: Practical CAD Techniques and Software. First ed., Artech House, Inc., Norwood, MA, 1996.
5. Standard for Models and Simulations. NASA–STD–7009A, 2016.