Ray Trace Notes

Charles Rino

September 2010

1 Introduction

A MATLAB function that performs a direct numerical integration to the ray optics equation described in book Section 1.3.2 has been implemented. The code, which was written by Dennis Hancock *http:\\www.dennishancock.com*, is embodied in the MATLAB function `Launch_Rays`. The function launches a prescribed bundle of rays from a point on the z axis in a rectangular coordinate reference system. Each ray is identified by its launch-point polar angles in the reference xyz coordinate system. The supporting functions `Index_Of_Refraction` and `Gradient_Of_Index_Of_Refraction` calculate the refractive index and its gradient at any point in the propagation space. Users should recognize that ray tracing in a real propagation environment is a very slow process without prior information about the propagation environment. For example, in a vacuum rays are straight lines and ray tracing is unnecessary. If the geometry is constrained, e. g. the refractive index variation is spherically symmetric, the ray tracing operation can be greatly simplified. `Launch_Rays` is completely general in that a fixed step size for ray propagation. Each ray propagates until a stopping criterion is satisfied. An example-specific utility, e.g. `RunRayTrace,` defines the ray family, calls the appropriate `Launch_Rays function`, and stores the results in an output `*.mat` file. The output of the `Launch_Rays` function is an array structure (`Rays`)with the following fields:

```
% .launch =[Height,theta,phi]
% .dn     = refractive index                      (1xNumber_of_segments)
% .ds     = path length increment                 (1xNumber_of_segments)
% .xyz    = position in tcs coordinate system     (3xNumber_of_setments)
% .tau    = ray tangent vector                    (3xNumber_of_setments)
% .above_surface = 0 if ray penetrated surface else 1
```

Examples have been constructed to demonstrate the code and to reproduce examples in the book *The Theory of Scintillation with Applications in Remote Sensing* by Charles L. Rino, John Wiley & Sons IEEE Press, 2010.

2 Example Description

Execution of the specific examples and the graphic outputs that will be generated are described below. First transfer the MATLAB active directory to

\RayTrace_Examples

and execute SetPath4RayTrace to place ray trace codes and utilities on the MATLAB path. The SetPath4RayTrace script will prompt the user to identify the directory \GPS_CoordinateXforms and place it on the MATLAB path. The directory can be downloaded from MATLAB Central

http://www.mathworks.com/matlabcentral/fileexchange/28813
−gps − coordinate − transformations

To examples are presented. The first example uses a PropCode1 beam propagation example output to define a ray bundle for comparison of full diffraction and ray trace calculations as discussed in Chapter 2.2.3 of the book cited above. The PropCode1 utilities can be downloaded from MATLAB Central

http://www.mathworks.com/matlabcentral/fileexchange/28800
−2 − d − propagation − simulation

Instructions for execution of the code are in the directory.

Because the second example is not discussed in the book, the development is presented in more detail in this document. It illustrates some generally known but interesting aspects of electromagnetic wave propagation in the earth's atmosphere.

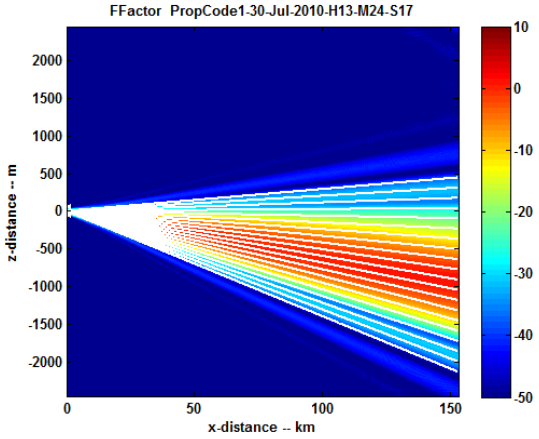
2.1 Raytrace PropCode1 Overlays

In the folder

· · · \RayTrace_Examples\RayTraceOverlay

run the script SetupPropCodeFigRefraction to initiate code execution. The run is initiated with a GUI that allows selection of a PropCode1 beam refraction example, which must be run prior to execution of this example. Keyboard input allows selection of the number of rays (hit enter for default). The beam extent and height is used to define a cone of rays that can be compared to the full diffraction PropCode1 output. The ray trace output is stored in a *.mat file whose name contains the run date and time.. Executing the script MakeFig28 will generate Figure 2.1. MakeFig28 first executes the subroutine DisplayPropCode1Output with appropriate inputs to plot the beam intensity profile. Note that the first GUI *.mat selection is from the folder

· · · \PropCode1_Examples\Refraction.



`MakeFig28` then executes the script `RunRayTracePropCode1Overlay`. The second GUI `*.mat` selection is from the current folder

$$\cdots\backslash\texttt{RayTrace\_Examples}\backslash\texttt{PropCode1Overlay}.$$

The example shows the ray bundle overlaid on the color coded intensity display of the beam propagation. The rays do not comprehend the intensity variation across the beam; however, the ray density is a measure of the normal field intensity.

2.2 Earth Atmosphere

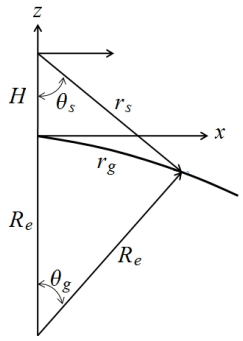
To illustrate the effects of refraction in the Earth's atmosphere the subroutine `Launch_Rays_Earth` was adapted to terminate the ray trace when the maximum number of increments has been reached or the ray penetrates a sphere at the mean radius of the earth. The propagation environment is the CRPL standard atmosphere model described in the Chapter 3 of the book cited. In directory `\AtmosphereScintillation` the MATLAB script `DisplayRFN` will display the reference standard atmosphere refractive index profile.

The spherical earth model is simple enough that all the straight-line ray geometry can be calculated analytically. Figure 2.2 shows the geometry for a source at a height H above the surface. The angle $\theta_s = \pi/2 + \theta_d$ where θ_d is the negative depression angle that defines the downward ray direction. With the notation

$$c_t = 1 + H/R_e, \tag{1}$$

the following formulas are readily derived:

$$r_s/R_e = c_t \cos\theta_s - \sqrt{c_t^2 \cos^2\theta_s - (c_t^2 - 1)} \tag{2}$$



$$\cos\theta_e = \frac{c_t^2 + 1 - (r_s/R_e)^2}{2c_t} \tag{3}$$

$$x = R_e \sin\theta_e \tag{4}$$

$$z = R_e (\cos\theta_e - 1) \tag{5}$$

The range to the visible horizon is

$$r_h = \sqrt{2R_eH + H^2}. \tag{6}$$

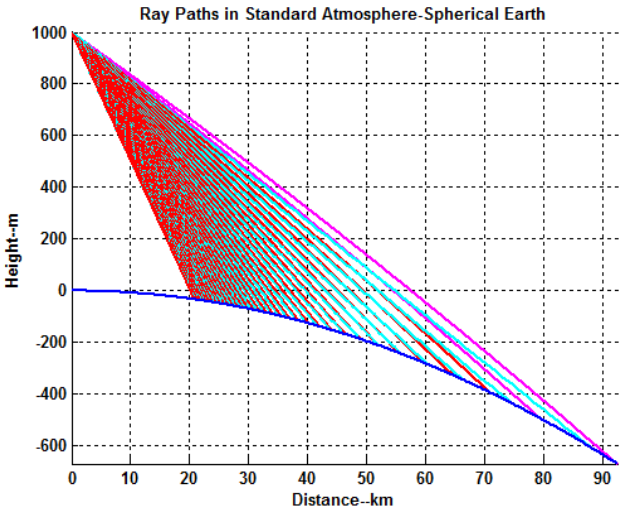
The maximum great circle angle is

$$\max\theta_s = \cos^{-1}\left(\frac{(r_h/R_e)^2 + c_t^2 - 1}{2\,(r_h/R_e)\,c_t}\right). \tag{7}$$

Executing the script

`...\RayTrace_Examples\EarthAtmosphere\RunRayTrace_Earth`

will generate the file `RayTraceEarth_Data.mat` file containing a set of rays launched at a 1 km height into the standard atmosphere. Executing the script `ProcessRayData_Earth` will generate the figures shown below. Figure 2.2 shows a comparison of the computed ray paths in the CRPL standard atmosphere (magenta and red) and the stright-line rays computed from the formulas above. The magenta ray has no complementary geometric ray because at that launch angle the straight-line ray does not intercept the earth's surface. The tangent ray



defines the visible horizon. The atmosphere extends the visible horizon from source at a fixed height. The practical ramifications are clear. For example, one is trying to aim laser at a point on the ground, the appropriate launch angle must be determined. Ray tracing provides a practical way to calculate an appropriate set of pointing directions; moreover, propagation anomalies such as ducting can be investigated as well.

An airborne radar presents a different challenge. A radar can measure the round-trip delay to a reflection point very accurately, but with comparatively coarse angular resolution. The time delay for a signal traversing the ray path is determined by the path integral

$$\tau_s = \frac{1}{c_0} \int n ds, \tag{8}$$

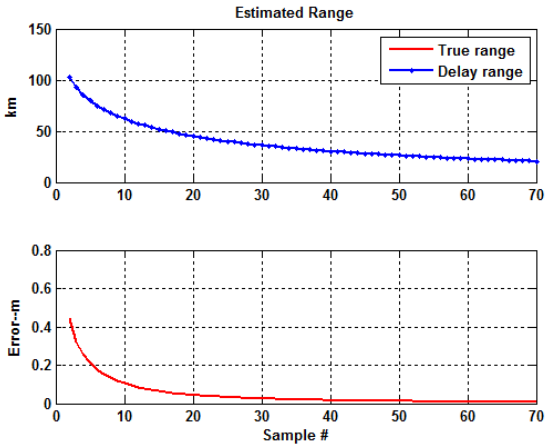
where c_0 is the vacuum velocity of light. The distance along the path, which defines the true radar range is

$$r_s = \int ds. \tag{9}$$

However, the desired information is usually the surface intercept point, which would require a ray trace to match the measured time delay. What is more commonly done is to use the approximation

$$r_s \simeq \overline{c} \tau_s, \tag{10}$$

where $\overline{c} = 2.99708$ is the mean velocity of light over the vertical distance H. Using the spherical earth model, the ground range, which effectively locates the



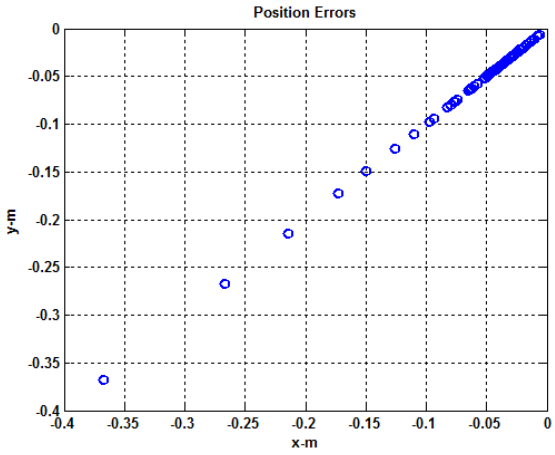
reflection point on the surface can be computed as

$$\theta_g \simeq \cos^{-1}\left(c_t^2 + 1 - (\tau_s \bar{c}/Re)^2\right)/\left(2c_t\right)\right)$$
$$r_g \simeq Re\theta_g. \tag{11}$$

The grazing angle, which is often important as well, can be computed as

$$r_N^2 \simeq 1 + c_t^2 - 2c_t \cos(r_g/R_e) \tag{12}$$
$$GZA \simeq -\sin^{-1}((1 - c_t^2 + r_N^2)/\left(2r_N\right)) \tag{13}$$

Figure 2.2 shows a comparison of the true range and the estimated range using the mean velocity. The errors are less than a meter. Figure 2.2 shows a comparison of the true surface intercept points in the TCS coordinate system in which the ray trajectories are reported and the estimated intercept points using the spherical-earth linear ray model. The results are generally acceptable considering that one rarely has precise refractive index information. In practice, the analysis would be performed using the GPS reference ellipsoid as a the surface model or actual terrain elevation data.


Position Errors
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
y-m
-0.4
-0.35
-0.3
-0.25
-0.2
-0.15
-0.1
-0.05
0
x-m