IMPLEMENTING RYTOVPROP IN MATLAB

Michael Oliker
SAIC

ABSTRACT

RytovProp.m is a single file of Matlab Code which implements David Fried's new RytovProp method. The result is easy to use, flexible, and remarkably fast. Results are compared with two different propagation codes.

1. WHAT IS RYTOVPROP?

RytovProp was invented by David Fried, who contributed a paper in this conference which goes into the theoretical details of this concept. It predicts Strehls at a single point at altitude from a source aperture on the ground. It uses Rytov theory to predict covariance of phases, log amplitudes and prior time tilts. It is valid for Rytov (or log amplitude variance) < 0.2; it becomes unreliable as Rytov rise towards saturation.

RytovProp is implemented in Matlab as a single function and is designed to be fast, easy to use and very flexible.

2. THE ESSENCIAL CONCEPT

For someone using RytovProp, the main thing to understand is that CM, the covariance matrix, holds the covariance of every pair of: phases and log amplitudes defined at each point in the aperture and prior time tilt estimates; the diagonal of the matrix holds the variances of each of these. The CM matrix is calculated from Rytov theory. As Fig. 1 shows in detail, the square root of CM, i.e. Csr where $Csr \bullet Csr^t = CM$ is calculated using the singular value decomposition. Given a vector of Gaussian random numbers, "g", a vector for one realization of phases, log-amplitudes, and prior time tilt estimates, "p", is produced by matrix multiplication:

$$p = Csr \bullet g$$

$<g \bullet g^t> = \text{Identity Matrix} = I$
$<p \bullet p^t> = CM$
$CM = Csr \bullet Csr^t$
$= Csr \bullet I \bullet Csrt$
$= Csr \bullet <g \bullet g^t> \bullet Csrt$
$= <(Csr \bullet g) \bullet (gt \bullet Csr^t)>$
$= <(Csr \bullet g) \bullet (Csr \bullet g)^t>$
$= <p \bullet p^t>$
So:
$p = Csr \bullet g$

Fig. 1. How Phase and Log Variance Estimates Are Calculated

3. USING RYTOVPROP IS EASY

RytovProp.m is a single file of Matlab code which is called with one argument and which returns a vector of independent Strehl values. This vector can then be used to generate, for example, a Cumulative Probability Density Function plot. The RytovProp package includes code to generate such plots. RytovProp runs much faster than traditional wave simulations (discussed in detail later).

The one input argument is a structure which holds all the parameters which describe the optics, the turbulence, and the simulation. A structure is a Matlab data type which has many fields, each of which has a name, and can point to any of Matlab's other data types. Examples of this are below in "Using RytovProp."

This approach makes it easier to have readable code since the parameters all have names and can be set one parameter per line. It also makes it easy to duplicate a set of parameters, and to store results and parameters together, thus documenting what parameters values were used to generate the results.

It also makes it easier for the developers to add features, which usually involves adding optional parameters. Some examples of optional features are specifying a beam profile of intensity or phase (e.g. a tophat or Gaussian profile, or a diverging or focused beam, and an arbitrary aperture shape).

Beyond the structure, RytovProp can take optional input and output argument pairs. This allows internal variables in the code to be output. For example, the Csr matrix can be output along with the Strehl values. A further feature allows us to input such internal variables as part of the input structure. This overrides them from being calculated. Thus we could, for example, get Csr out and add it to the input structure, which will then skip recalculating it, and go straight to generating more independent Strehls, by reusing this input Csr matrix.

4. OUTLINE OF PROCESSING

Once RytovProp is called, there are a number of internal steps in processing. Understanding these might help the user tune it for his application.

First the code checks to see that sufficient inputs have been provided. Then it forms the grid of sample points for which the calculations will be performed. If the divergent beam or other phase variation at launch is specified, the various functions needed to support that are calculated next.

Next the sample times and weighting functions are calculated to support the tracker bandwidths specified. Most of the time a single prior time will be sufficient and is set at one radian in the past at the tracker bandwidth. So if 10 Hz is requested, the single prior tilt estimate will be calculated at $1/(2\pi\ 10\ \text{Hz}) = 0.159$ seconds in the past, and it will be weighted 100% in applying a tilt correction.

In the rare (and time consuming) case that detailed tracker behavior is being modeled, or more than one bandwidth (not counting zero or infinite Hz, i.e. no tracker or perfect tracking) is specified, then a series of times are calculated, with a geometrically increasing time between samples, and these are weighted to match the requested bandwidths.

The C_n^2 profile is then integrated into the specified number of layers. The layers are spaced so that they contribute equal amounts of expected optical path difference. The wind speed in each layer is also calculated, and then wind directions are added.

Next, the correlation matrix is calculated. This is one of the most time consuming steps. The correlation of each element of phase, log-amplitude and prior full aperture tilt with each other element is calculated. For example, if there are 100 phases and 100 log-amplitudes and 5 prior tilts being calculated, then a matrix 205×205 is formed holding all the cross correlations. The matrix is built up from subarrays: all phases with other phases, all phases with log-amplitudes, all log-amplitudes with other log-amplitudes, tilts with tilts, phases with tilts, log-amplitudes with tilts. This matrix is called CM, and can be requested as an output, as described below.

Next the CM matrix is factored using singular value decomposition, and recombined to form the "square root" of the matrix, i.e. Csr such that $\text{Csr}\cdot\text{Csr}^t = \text{CM}$. Csr can also be requested as an output.

Finally, realizations are calculated by multiplying Csr by a series of vectors of Gaussian random numbers. The output of the multiplication is a vector of phases, log-amplitudes, and tilts. These are combined to form phasors, with tracker tilt correction applied, and the phasors are averaged and the resulting amplitude squared to form a Strehl ratio estimate. As many of these as requested are formed, and output in a large vector. If a launch phase is requested, or an aperture mask, or a large divergence, then these other calculations are done, as appropriate.

Once the vector of Strehls is returned, there are utilities provided with RytovProp to aid in analysis. CDFplot.m will form a CDFplot with the X-axis in dB of Strehl (i.e. $10\cdot\log_{10}(\text{Strehl})$) and the Y-axis showing probability with an axis scaled so that a log-normal distribution will show up as a straight line. writehdf5complex.m and readhdf5complex.m are provided to save a structure of data out in HDF5 format. Finally Print2LaTex.m is provided to allow a LaTex document to be built up showing the results of many RytovProp runs.

5. USING RYTOVPROP

The first step is **describing the path**, as shown in Fig. 2. Notice that each line starts with "d." ; d is the structure of parameters we are building so that we can call RytovProp. We describe the zenith angle to the target, the distance to

the target, the slew rate, and point ahead angle. The "h" and "z" fields hold a vector of heights and positions along the path where the turbulence will be described below.

```
% PATH
d.psi=pi/4;              % zenith angle in radians
d.Z=3e5*sec(d.psi);      % distance to target along path in meters
d.theta_dot=[7e3*cos(d.psi)/d.Z 0];       % slew rate in radians/sec
d.theta_PA=[2*(7e3*cos(d.psi))/3e8 0];    % point ahead angle
                              in radians, normally parallel to theta_dot
dh=1;                    % fine steps in height in meters
d.h=1:dh:3e4;            % vector of heights in atmosphere
d.z=d.h*sec(d.psi);      % distance along propagation direction
```

Fig. 2. Describe the Path

Next we **describe the light source** (see Fig. 3). This includes the light's wavelength, lambda, in meters, the telescope diameter (also in meters), and the various tracker bandwidths we would like to simulate. To accomplish this, the system will estimate not only phase and log amplitude but also the tilt at various times in the past.

```
% SOURCE AND LIGHT AND TRACKER
d.lambda=0.5e-6;           % wavelength in meters
d.D=0.1;                          % telescope diameter in meters
d.bws=[0 3 10 30 100 inf];         % tracker bandwidth(s) in Hz

% BEAM PROFILE: current options: 'tophat', 'gaussian' (needs eRad)
d.profile.name='gaussian';        % profile
d.profile.eRad=0.9*d.D/2;         % distance from center to
            % where beam amplitude is 1/e of peak, intensity is 1/e^2
```

Fig. 3. Describe the light source

Inf designates infinite bandwidth, i.e. perfect tracking, and 0 designates no track correction at all. If only 0 and inf are requested, no prior time estimates are needed. The number of prior times we must calculate has a large impact in processing time.

```
% TURBULENCE
d.vel=Bufton(d.h);                   % wind speed model
d.CNsq=0.1*TurbulenceModels(1,d.h); % turbulence strength model
                                %(#1 is HV57, see function for others)
d.anglerange=[0 pi/2];               % range of wind velocity angles
                                     %     to add to final layers
```

Fig. 4. Describe the turbulence

It takes only a few lines to **describe the turbulence**. We call a wind speed model, to get wind speed at each altitude specified in `d.h`. We call TurbulenceModels.m to fill in C_n^2 model, in this case Hufnagel-Valley 5/7. This function has a number of models available; the "1" argument specifies which model is desired. Here, in Fig. 4, we've chosen to reduce the turbulence strength by multiplying the C_n^2's by 0.1. Finally we specify a range of angles to be added to the wind model, giving direction to the speeds.

```
% SIMULATION
d.Realizations=1e5;     % number of Strehl estimates
                              % to calculate for each bandwidth

d.K=20;                 % number of atmosphere layers

d.minCount=100;         % minimum number of points in aperture
d.maxSpace=0.01;        % maximum distance between adjacent
                        % points in meters.  Can be calc'd from r0;
                        % D.Fried suggests 0.2074*r0
d.NumTimeSteps=1;       % number of time steps to use in forming
                  % estimates of prior tilts for tracker simulation.
```

Fig. 5. Set the Simulation Parameters

These parameters **control how the simulation runs**. Realizations determines how many Strehl values to calculate. I n this case we've requested 100,000; enough to generate a pretty smooth CDF plot without taking too long. A million realizations is not unworkable.

The next parameter, K, sets how many atmospheric layers to use in calculating the correlation matrix (CM). This is normally much smaller than the length of vector `d.h`. The program integrates the C_n^2data appropriately to form this number of layers.

minCount sets the minimum number of points to use in defining the aperture. Here we have specified (see Fig. 5) that we must use at least 100 points; when the code ran, it used 112, the best fit of a circular aperture on a square grid with at least 100 points.

maxSpace similarly sets the maximum point spacing allowed. This can also control the number of points in the aperture; the system will vary the point spacing to meet both criteria.

```
% CALL FOR STREHL CALCULATION

d.SR = RytovProp( d );

% OR

[d.SR, d.Csr, d.X, d.Y] = RytovProp( d, 'Csr', 'X', 'Y' );
```

Fig. 6. Call the function

Calling the function is a single line of code. Two variations are shown in Fig. 6. The first line takes only the structure, "d" as an argument, and returns only the vector of Strehls, "SR," which is stored in the structure "d" thus documenting how they were calculated.

The second variation also requests that three internal variables be returned. The three are specified as strings in the function call, and are returned as matrices, and are here stored in the structure "d" along with "SR." The "Csr" matrix is the matrix which turns random numbers into atmospheric realizations. "X" and "Y" hold the coordinates of the points used in the aperture, so we can see where they are and how numerous.

There is support, as well, for a phase or intensity profile at the ground aperture, for example a Gaussian beam profile or a diverging beam. These options require additional input parameters and are documented with the code.

6. PARAMETERS VS. PERFORMANCE

The three key variables controlling runtimes are:

- Points in the Aperture
  - This is affected by maxSpace and minCount as well as Diameter, and Aperture Shape
  - CM and Csr grow as the square of this value, plus the number of prior time estimates
  - It is displayed on the left in Fig. 7. Note that the last point is half an hour; still not very long.

- Number of Prior Time Estimates Required (center plot of Fig. 7)
  - For no tracking or perfect tracking cases, no prior time tilt estimates are required
  - For a single tracker bandwidth, a single prior time estimate is generally sufficient, at $1/(2\pi \text{bandwidth})$ seconds prior.
  - For detailed tracker modeling, many prior times can be generated.
  - It takes considerably longer to generate CM with prior time tilt estimates.
  - We got good agreement with Barry Foucault's ACS simulation using 25 prior times, with log time steps.
  - All bandwidths were handled by weighting this set.
- Number of Realizations Requested (right plot of Fig. 7) has the least impact. Even 3 million points took around 5 minutes.

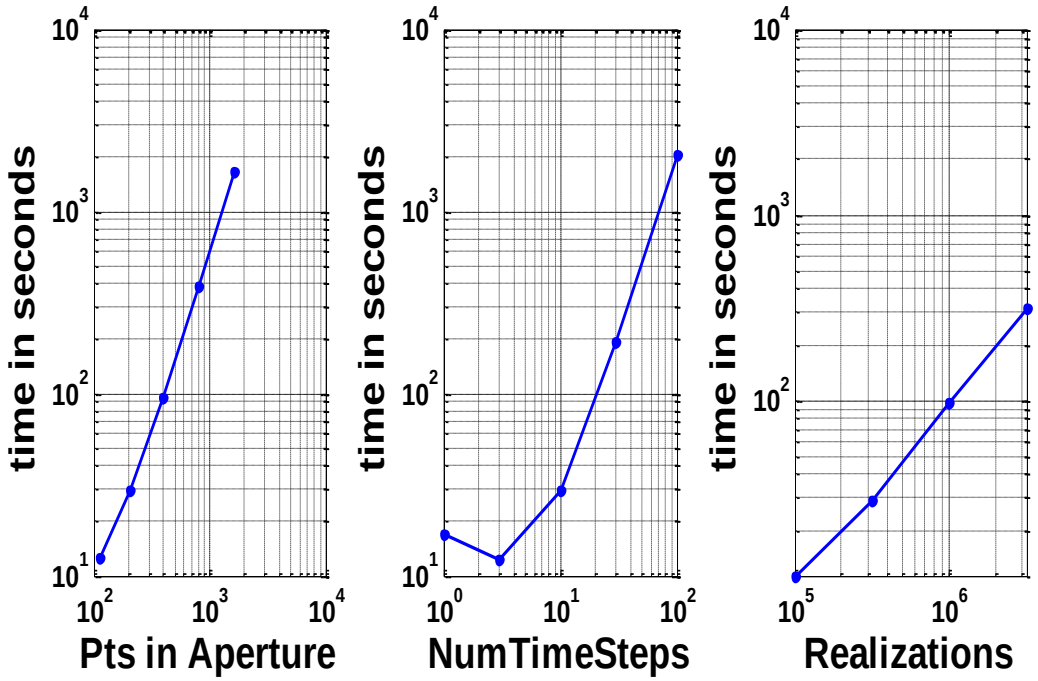

Fig. 7. Parameters vs. Runtime

Fig. 8 shows the effect of repeating the same case repeatedly. Case 39 is one of the simulations the Barry Foucault had done which was reproduced with RytovProp. Notice that the lower end of the plots is a bit ragged. This would be lessened if more points were used. For example, if 100,000 points are used (as is the case in the figure) then only 10 points show up at or below 0.0001 probability. These are random, so the standard deviation of the value 10 will be the square root of 10 or around 32%. If a million points were used for this case, then 100 points would fall below 0.0001 probability, having a standard deviation of only 10%, smoothing the curves and making them more reproducible.

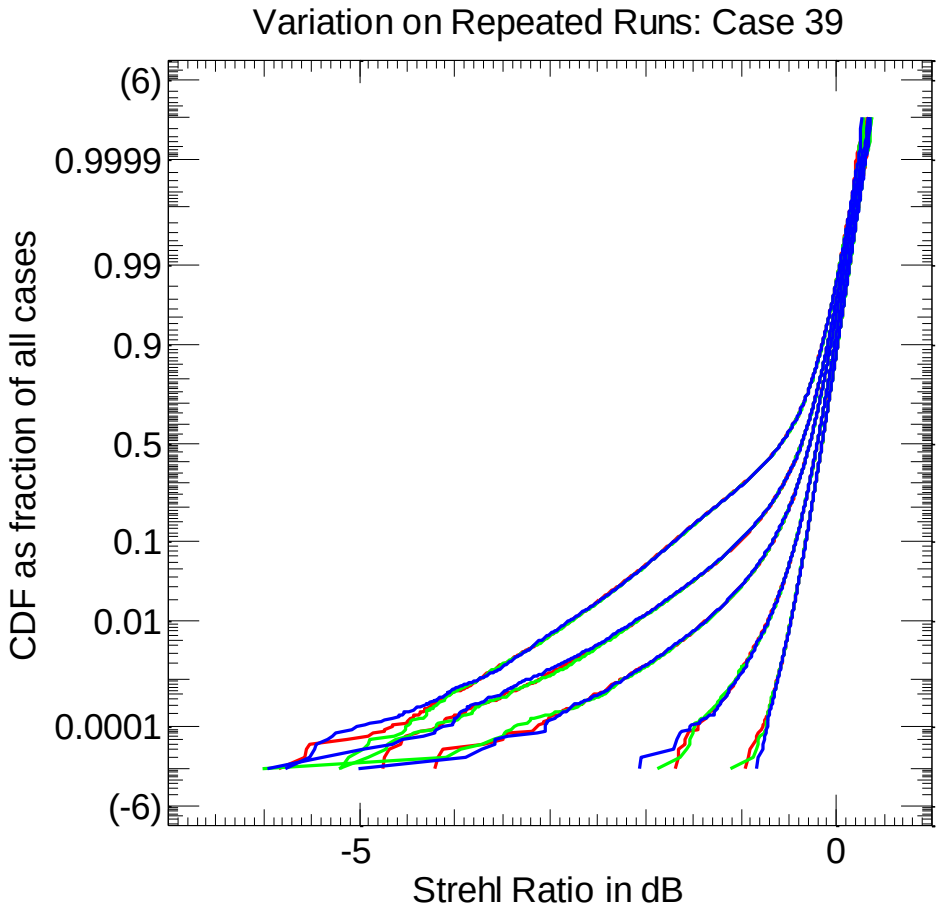

Fig. 8. Reproducibility at 100,000 realizations. More realizations would appear smoother

7. APPLYING RYTOVPROP AND COMPARING RESULTS TO WAVE SIMULATIONS

RytovProp was compared to two wave simulation results, a published study by Parenti et al [1] and an in house series run by Barry Foucault. The process of running cases was quite straightforward. I generated the structure holding the parameters for each case, ran RytovProp with that input, and plotted and stored the results. Except for initializing the structure, which took a page of code, the loop to generate the results was on the order of a dozen lines.

Fig. 9 is taken from Parenti [1] with RytovProp results overlaid. These cases have diameters from 2mm to 2 meters, and are HV5/7 atmospheres with Geosynchronous orbits and 1.55 micron light. The RytovProp runs used 100,000 realizations. The blue and green curves are from RytovProp and show remarkable agreement with Parenti's black circles and triangles. The curves end at around 2 meters diameter where the computer being used had insufficient memory to continue. Fig. 10 shows the run times for these 31 cases run (four cases run are not shown, being to the left of the y-axis). The first 26 took only ~15 seconds per case. These were consistent, because they all used the specified minimum number of points in the aperture. For the larger apertures, the number of points starts to climb, and the run times rise rapidly, with the last point taking more than half an hour. The first 25 cases took 6 minutes total, the last 6 cases took 60 minutes.[1]

[1] Parenti, R; Sasiela, R. J..; Andrews, L.C..; Phillips, R L., *Modeling the PDF for the irradiance of an uplink beam in the presence of beam wander*, Atmospheric Propagation III. Edited by Young, C. Y.; Gilbreath, G. C. Proceedings of the SPIE, Volume 6215, pp. 621508 (2006).

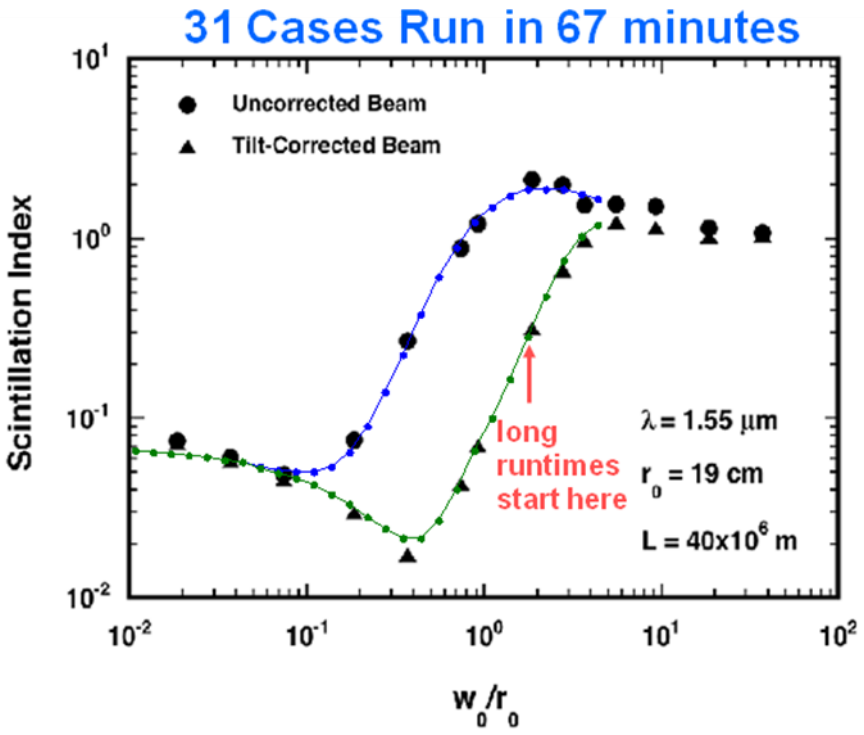

Fig. 9. Parenti et al simulations (black) and RytovProp (blue and green)

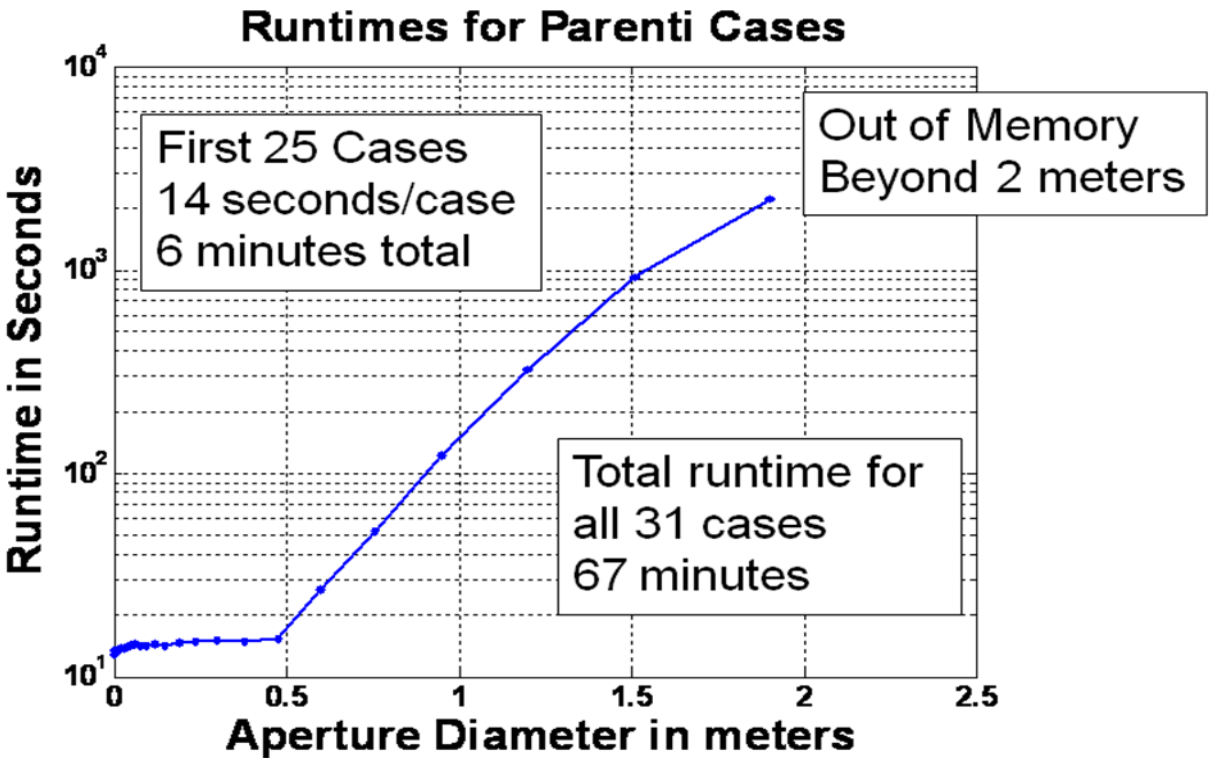

Fig. 10. Runtimes vs. Aperture Diameter to run Parenti Cases in RytovProp

Barry Foucault ran a series of cases using the ACS wavefront propagation simulation code. He generated 66 cases with 0.5 and 1.5 micron wavelengths, three turbulence strengths, 10 and 25 cm apertures, 45 and 90 degree

elevations and 400 km, 1500 km and Geosynchronous orbits. The agreement was quite good in all these cases (discussed in more detail in Dr. Fried's talk); Figs. 11 and 12 show a couple examples of this agreement with dashed curves being RytovProp (here labeled KL from an earlier name for this method) and the solid curves being the ACS wave simulation results. Again there were 100,000 realizations in RytovProp. The ACS runs had fewer points and these were correlated in time. The discrepancies are largely the result of having few independent degrees of freedom in the ACS data.

The times for the ACS runs were not recorded, but run over several months on a 24 node cluster, while the RytovProp cases were run overnight on a single PC with 4-10 times more points; thousands of times faster.

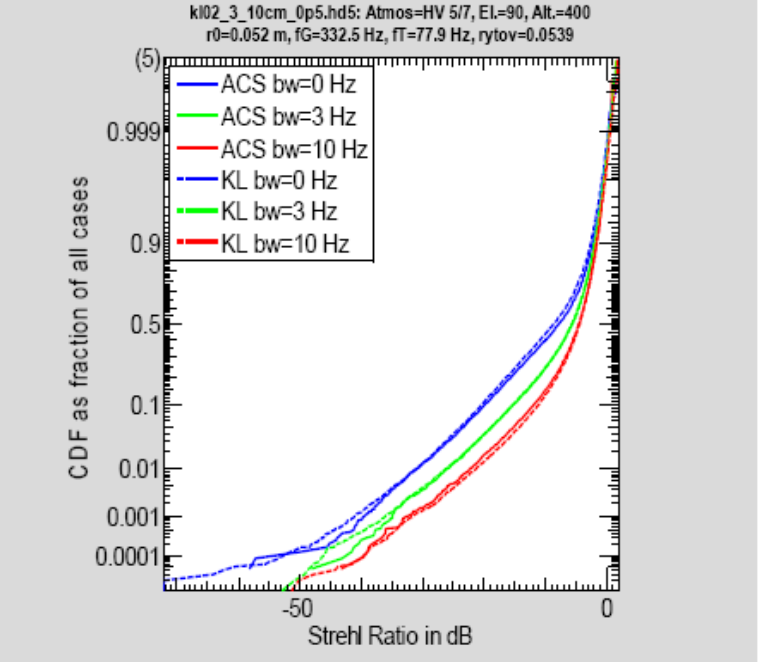

Fig. 11 ACS/RytovProp Comparison at low altitude, small diameter and 5.2 cm r0

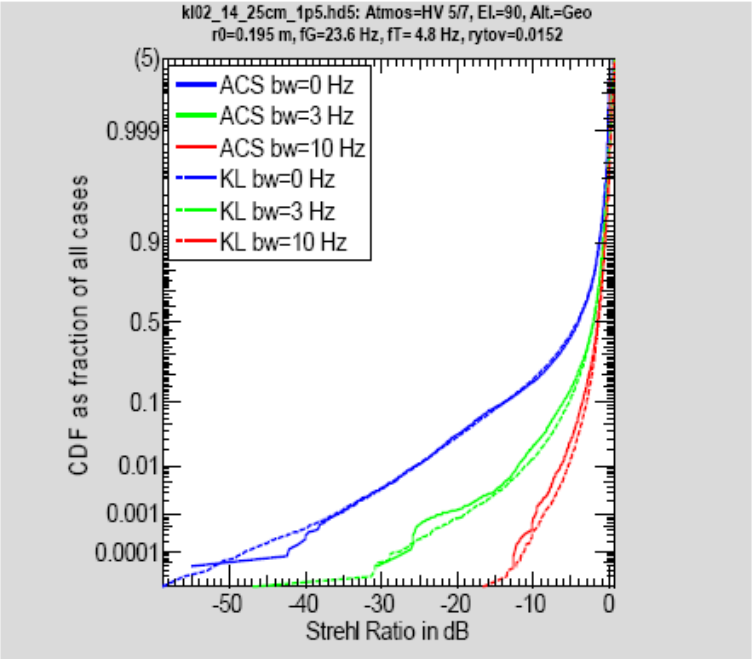

Fig. 12. ACS/RytovProp Comparison at high altitude, large diameter and 19.5 cm r0

8. SUMMARY

- RytovProp is Easy to Use and Flexible
- RytovProp Runs Fast
- RytovProp Agrees Well with Wave Simulation, when Rytov Theory Holds

9. DISTRIBUTION

The RytovProp software package currently has limited distribution to U.S. Government agencies and their contractors. Make requests to Air Force Research Laboratory/RDS, 3550 Aberdeen Ave SE, Kirtland AFB, NM 87117-5776

10. ACKNOWLEDGEMENTS

This work was performed under USAF/AFRL Contract No. F29601-03-C-0148. It has been a great honor and distinct pleasure to work with Dr. David Fried on this project, especially in the creative environment created by Dr. Earl Spillar in the Air Force Research Laboratory's Starfire Optical Range on Kirtland Air Force Base. Dr. Barry Foucault's partnership in comparing our results was a key to our success.