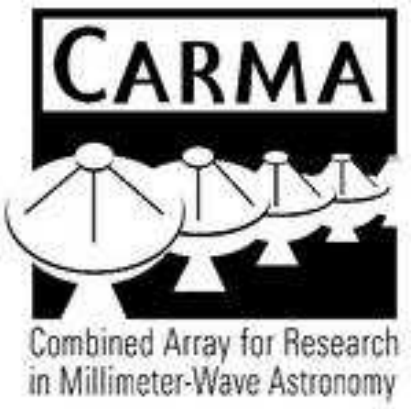

CARMA Memorandum Series #54

Circular Polarizers for the CARMA 1mm Receivers

Richard Plambeck, Greg Engargiola
U.C. Berkeley

April 13, 2010

ABSTRACT

We describe the design of waveguide circular polarizers for the CARMA 1mm receivers. A polarizer will be installed between the feedhorn and the orthomode transducer in each dewar, at a temperature of 4 K. It will convert incoming R and L circularly polarized signals to X and Y linearly polarized signals, which will be separated by the OMT.

The polarizer is a 2-section design, using half-wave and quarter-wave retarder sections rotated axially by 59.5° with respect to each other to achieve broadband performance. It is constructed in 0.047″ diameter circular waveguide; the retarders are sections of reduced height, or 'faceted,' circular waveguide. Network analyzer measurements of K-band scale models were used to verify the retarder design.

The polarizer will be fabricated by electroforming on an aluminum mandrel. Analysis shows that the diameter of the circular waveguide and the heights of the retarder sections must be controlled to ±0.0003″, and the angular offset of the two retarders to ±0.2°, in order to achieve a polarization leakage of < 0.05 across the 210-270 GHz frequency band. Tapered transitions are used to reduce reflections from the faceted waveguide sections; the predicted return loss is < -20 dB.

The CARMA receivers use Mylar beamsplitters to couple local oscillator power into the signal beam outside the dewar. The beamsplitters have unequal transmission coefficients for signals polarized parallel and perpendicular to the angle of incidence, which will slightly increase the polarization leakage.

Change Record

Revision	Date	Author	Sections/Pages Affected
	Remarks		
1.0	2010-Apr-13	Dick Plambeck	
	Initial version.		

1. Introduction

With an aperture synthesis array it is advantageous to measure *linear* polarization by cross-correlating right and left *circular* polarizations. This avoids the necessity of taking the difference of two large numbers (the total fluxes E_X^2 and E_Y^2 in the X and Y directions) in order to measure a small number (e.g., Stokes Q $= E_X^2 - E_Y^2$). Instead, with circularly polarized receivers, Stokes Q and U are derived from sums and differences of the RL and LR cross-correlations, which are zero for an unpolarized input signal.

The dual polarization 1mm receivers for CARMA use a waveguide orthomode transducer (OMT) (Navarrini & Plambeck 2006; Navarrini, Bolatto, & Plambeck 2006) to split the incoming radiation into two orthogonal linear polarizations. To observe circular polarizations, one must install a polarizer in front of the OMT to transform incoming R and L circular polarized signals into X and Y linear polarizations. The polarizer may be a room temperature waveplate outside the dewar, or a waveguide device at 4 Kelvin installed between the feedhorn and the orthomode transducer. We prefer the waveguide polarizer because it is more compact and has lower loss.

The simplest circular polarizer consists of a quarter wave retarder with its principal axes oriented at 45° to the linearly polarized receiver. Unfortunately such a single section polarizer has a fractional bandwidth much narrower than the 210–270 GHz tuning range of the 1mm receivers. One can achieve broader bandwidths by cascading several retarder sections that are rotated axially with respect to each other. This was first shown by Pancharatnam (1955) for birefringent wave plates. Pancharatnam visualized the polarization states on a Poincaré sphere, and used spherical trigonometry to compute the optimum angles for the retarder axes; Fig. 1 illustrates the principle. Stacked quartz waveplates of this design were used for the SCUBA polarimeter (Greaves et al. 2003).

Kovac and Carlstrom were the first to build broadband waveguide polarizers. For the DASI experiment (Leitch et al. 2002; Kovac 2004) they constructed 2-section polarizers in 0.315″ diameter circular waveguide, with dielectric vanes as retarder elements, obtaining good polarization purity across the 26-36 GHz frequency band. We considered trying to miniaturize this design for the CARMA polarizers. We planned to broach slots into the walls of 0.047″ diameter circular waveguide and insert tiny vanes of quartz or other dielectric into these slots. Electromagnetic simulations showed, however, that the slots themselves produced significant phase shifts, which ultimately led us to abandon the dielectric vanes altogether. Because of the small dimensions of the waveguide, we chose to design the retarders from simple truncated, or 'faceted' circular waveguide, an idea orignally suggested by Pyle (1964). The possibility of constructing multisection polarizers from faceted circular waveguide segments was discussed in an ATNF technical memo by Lilie (2001).

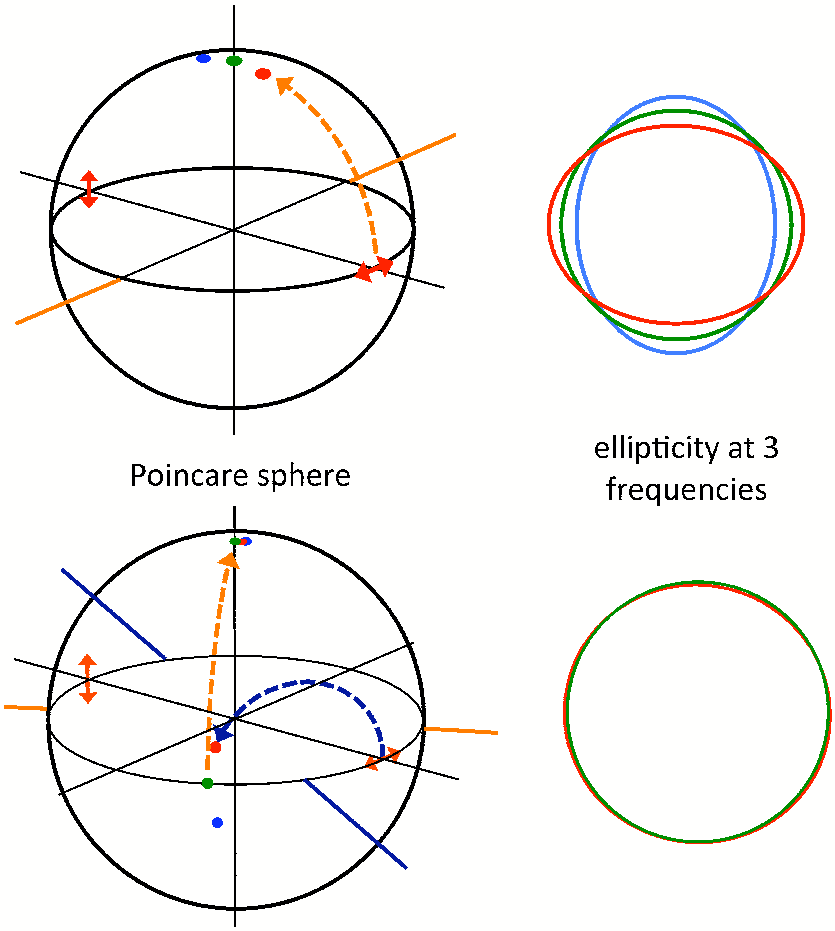

Fig. 1.— Action of single and 2-element circular polarizers visualized on a Poincaré sphere, adapted from Fig. 3.7 and 3.8 of Kovac (2004). The goal is to transform linearly polarized radiation, on the equator of the sphere, to pure circular polarization, at the pole. (*top*) Passing the incoming radiation through a single quarter wave plate with principal axes at 45° to the linear polarization direction corresponds to a rotation on the sphere that places the central frequency exactly at the pole, but spreads neighboring frequencies along an arc, leading to substantial ellipticity. (*bottom*) In a 2-stage polarizer the signal passes first through a half wave plate that rotates the center frequency back to the equator (a different linear polarization) and again spreads neighboring frequencies along an arc. The dispersion along this arc offsets the chromatic errors introduced by a subsequent rotation to the pole by a quarter wave plate, thus broadening the bandwidth.

2. Faceted circular waveguide retarder sections

The building blocks for the polarizer are quarter wave and half wave retarder sections fabricated from faceted circular waveguide (Fig. 2). Single mode signals propagating from port 1 to port 2

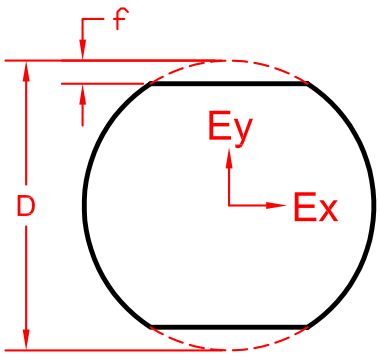

Fig. 2.— Cross section of faceted circular waveguide with diameter D and facet depth f. Signals polarized in the Y-direction (the 'slow' axis) undergo a greater phase shift as they propagate through a length of this guide.

through a length L of waveguide undergo a phase shift ϕ such that $E_2 = E_1 \exp(-j\phi)$, where

$$\phi = \frac{2\pi L}{\lambda_g}, \quad \text{with} \quad \lambda_{\rm g} = \frac{\lambda_0}{\sqrt{1-(\lambda_0/\lambda_{\rm c})^2}} = \frac{\rm c}{\sqrt{\nu^2-\nu_{\rm c}^2}}.$$

λ_c is the cutoff wavelength of the guide. The X- and Y-polarized signals in a faceted guide have different cutoff frequencies, so after propagating through length L they acquire a differential phase shift

$$\Delta\phi(\nu) = \phi(E_Y) - \phi(E_X) = \frac{2\pi L}{c}\left(\sqrt{\nu^2-\nu_{cY}^2} - \sqrt{\nu^2-\nu_{cX}^2}\right) \tag{1}$$

L is chosen to make $\Delta\phi(\nu_0) = 90°$ for a quarter wave retarder, 180° for a half wave retarder.

In the CARMA receivers the polarizers will be installed between 1mm feed horns with 0.050″ (6-m antennas) or 0.047″ (10-m antennas) diameter circular waveguide outputs, and OMTs with 0.044″ diameter circular waveguide inputs. To minimize reflection losses, we chose to fabricate the circular polarizer from 0.047″ diameter circular waveguide.

There are no exact analytic expressions for the cutoff frequencies ν_{cX} and ν_{cY} in faceted circular waveguide. Calculations based on various approximations have been published by Pyle and Angley (1964), Sinnott et al. (1969), Levy (1995,1997), Wang (2000), and Lin et al. (2001). We used both the analytic approximations of Wang (2000) and electromagnetic simulations with Ansoft HFSS to find the cutoff frequencies for 0.047″ diameter faceted guide. The `python` script used for the analytic solutions is given in Appendix A. Initially the HFSS calculations were disappointing – they failed to reproduce the cutoff frequency $\nu = c/(3.4126\,r)$ for plain circular waveguide. Finally we realized that HFSS was representing the circular guide as an inscribed N-sided polyhedron with a rather coarse value for N ($\sim$ 24), even when we specified very fine gridding. Because an inscribed polygon has a smaller effective radius than the circle that it approximates, the waveguide cutoff frequency was too high. One can get better accuracy by specifying an override value for the Number of Segments parameter in the Properties Window, after creating a cylindrical form in the 3D Modeler window. With N=192, HFSS returned a cutoff frequency of 147.189 GHz for 0.047″ diameter circular guide; the correct value is 147.176 GHz. The ratio of these two numbers, 1.000088, is almost precisely equal to $r/r_{\rm eff}$, the ratio of the true waveguide radius r to an effective

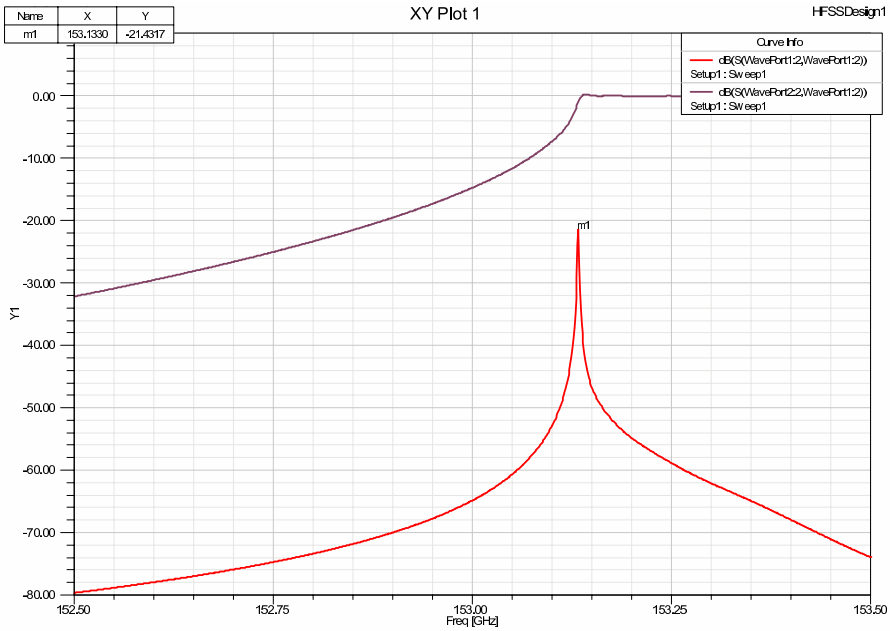

Fig. 3.— Example HFSS cutoff frequency calculation, in this case for an X-polarized signal propagating through a 0.5″ long section of 0.047″ diameter waveguide with 0.002″ deep facets. The wave is launched inside the faceted guide. The cutoff frequency is most easily measured by locating the cusp on the S11 curve.

radius $r_{\rm eff}$, defined such that:

$$\pi\, r_{\rm eff}^2 = \text{area of } N\text{−sided polygon inscribed in radius } r = \frac{1}{2} N r^2\, sin(\frac{2\pi}{N}).$$

A sample HFSS result is shown in Fig. 3. Table 1 lists the cutoff frequencies for facet depths up to 0.007″. The values derived from the analytic calculation and from the HFSS simulation are typically within 30 MHz of one another. The final column gives the lengths L_{90} of quarter wave retarders at 230 GHz, computed from equation (1) and the HFSS cutoff frequencies. For calculational convenience, we fit the normalized cutoff wavenumbers $k_c = 2\pi r \nu_c / c$ from the HFSS simulations with 5th order polynomials:

$$k_{cX} = 1.841184 + 0.301574\, x + 8.9118\, x^2 - 33.253\, x^3 + 93.2359\, x^4 - 94.615\, x^5 \quad (2)$$

$$k_{cY} = 1.841184 - 0.0862305\, x - 3.41638\, x^2 + 14.65\, x^3 - 32.7165\, x^4 + 31.7498\, x^5 \quad (3)$$

where $x = f/r$. Cutoff frequencies derived from the polynomial fits are plotted in Fig. 4.

What is the optimum facet depth f? On the one hand, reflection losses from the circular–faceted interfaces are smaller for shallow facets. On the other hand, manufacturing tolerances are tighter for shallow facets because a small error in f causes a larger *fractional* error in $\Delta\phi$. Based on the tolerance tests given in section 5, we ultimately chose $f = 0.006''$.

f (in)	ν_{cX} [Wang] (GHz)	ν_{cY} [Wang] (GHz)	ν_{cX} [HFSS] (GHz)	ν_{cY} [HFSS] (GHz)	L_{90}(230GHz) (in)
0.000	147.176	147.176	147.176	147.176	...
0.001	149.316	146.465	149.307	146.471	1.2392
0.002	153.134	145.193	153.119	145.203	0.4373
0.003	158.066	143.660	158.048	143.674	0.2359
0.004	164.011	142.004	163.985	142.030	0.1503
0.005	170.960	140.346	170.938	140.363	0.1042
0.006	179.016	138.706	178.985	138.732	0.0757
0.007	188.321	137.108	188.256	137.176	0.0562

Table 1: Cutoff frequencies ν_{cX}, ν_{cY} as a function of facet depth f for 0.047″ diameter waveguide, derived from the analytic approximations of Wang (2000) and from HFSS simulations. The HFSS cutoff frequencies listed here have been divided by 1.000088 to correct for $r/r_{\rm eff}$, as discussed in the text. The last column gives lengths of 90° retarder sections.

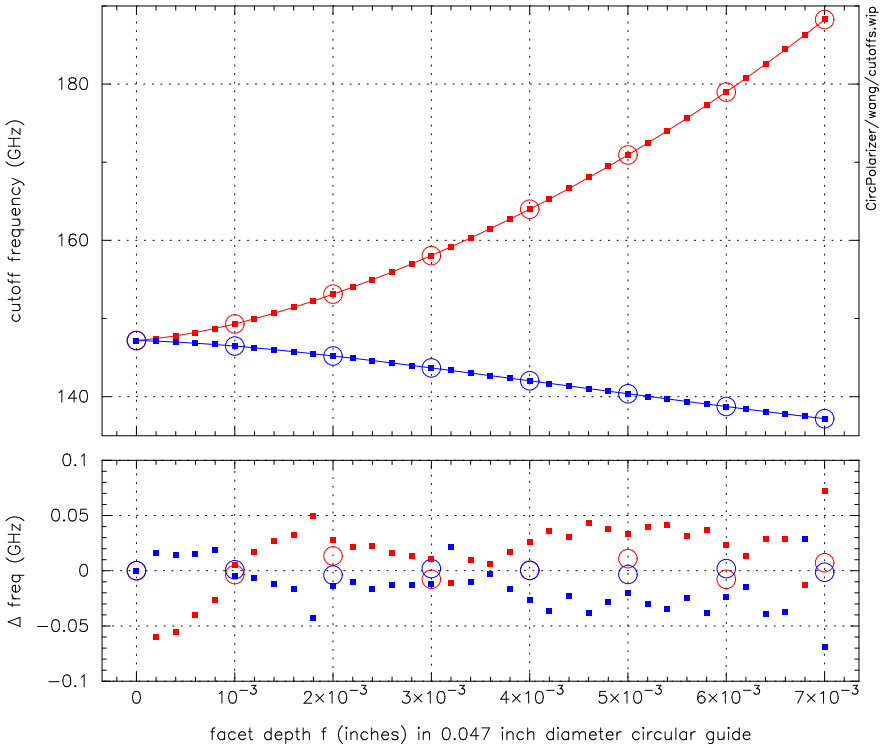

Fig. 4.— (*top*) Cutoff frequencies ν_{cX} and ν_{cY} for 0.047″ diameter faceted circular waveguide. Smooth curves show polynomial fits (eqn 2,3); small squares, analytic approximations of Wang (2000); open circles, HFSS results. (*bottom*) frequency differences from the polynomial curves.

3. Transitions

Thus far we have ignored the effect of reflections at the interfaces of the circular and faceted waveguide sections. The voltage reflection coefficient is

$$\Gamma = \frac{Z_{facet} - Z_{circ}}{Z_{facet} + Z_{circ}},$$

where Z, the wave impedance for TE modes in circular waveguide is (Pozar 1998, equation 3.129)

$$Z = \frac{\eta k}{\beta} = 377 \frac{k}{\sqrt{k^2 - k_c^2}} = 377 \frac{\nu}{\sqrt{\nu^2 - \nu_c^2}}.$$

We assume that this formula also can be applied to faceted circular guide if the cutoff frequencies dervied from equations (2) and (3) are used. At the junction of $0.047''$ diameter circular guide and faceted guide with a $0.006''$ facet depth, the expected return loss $-20\,log|\Gamma|$ is roughly -19.9 dB for X-polarized signals and -34.5 dB for Y-polarized signals. In a 2-section polarizer there are four such junctions; at unfavorable frequencies where the reflections add in-phase, the return loss could be as high as ~ -8 dB.

Stepped or tapered transitions may be used to minimize reflections. Fig. 5 shows a slice through a retarder with curved transitions that are convenient to produce if the flats on an electroform mandrel are machined with the side of a milling cutter. To compute the differential phase shift $\Delta\phi$ that occurs in the transitions, `pol.dphitaper2` (part of the `python` script given in Appendix B) approximates the transition as a series of small steps. With $R = 0.125''$, each curved transition produces 26° differential phase shift at 230 GHz. The lengths of quarter and half wave retarders must be adjusted to take into account the phase shifts through the transitions.

4. Verification of the retarder design with scale models

To verify the retarder design we constructed scale models in K-band (18–26.5 GHz) waveguide and measured the differential phase shifts through them with an 8722 network analyzer. The first scale

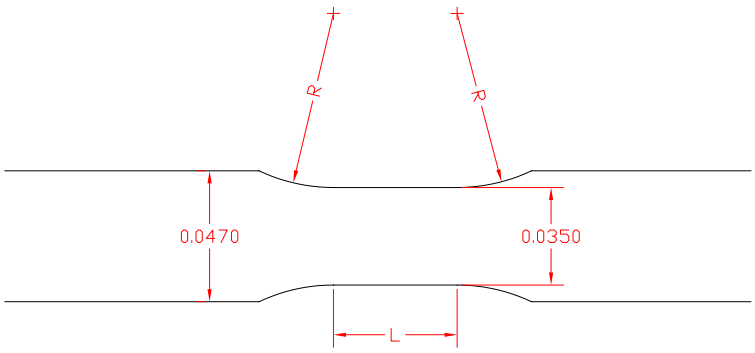

Fig. 5.— Cross section of a waveguide retarder with curved transition sections. Dimensions are in inches. The retarder is fabricated by electroforming on an aluminum mandrel; the $0.006''$ deep flats on the top and bottom of the mandrel are machined by side cutting with an end mill of radius R.

model was a section of straight faceted guide; the second used cylindrical inserts simulating back to back curved transitions. Dimensions of the models are given in Fig. 6. The waveguide diameter, 0.455″, was chosen to match K-band rectangular–circular waveguide transitions that we had in hand; thus the models are scaled up by a factor of 9.68085 relative to the mm retarders. The two heights tested in model (a) correspond to facet depths $f = 0.0038''$, $0.0060''$ in 1mm waveguide. Coax to waveguide adapters (Agilent K281C) and rectangular–circular waveguide transitions were attached to each end of the model as shown in Fig. 7. The network analyzer 'through' calibration was done with polarization parallel one axis, then the retarder was rotated by 90° and the phase of the orthogonal polarization was measured relative to this calibration.

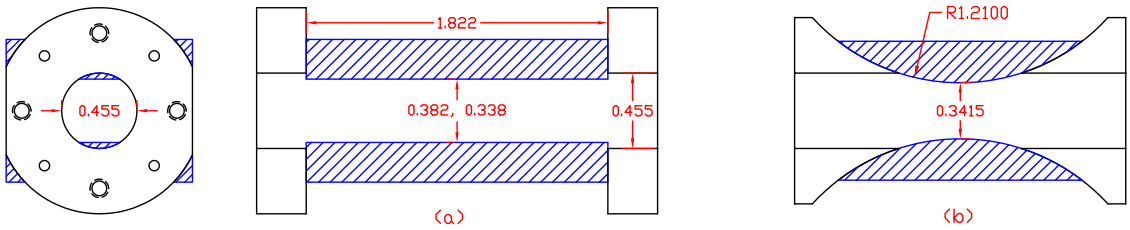

Fig. 6.— K-band scale models of (a) straight faceted waveguide retarder, (b) back to back curved transitions from circular to faceted guide. Dimensions are in inches. For model (a), one set of measurements was made with 0.382″ waveguide height, then the part was remachined and remeasured with 0.338″ height.

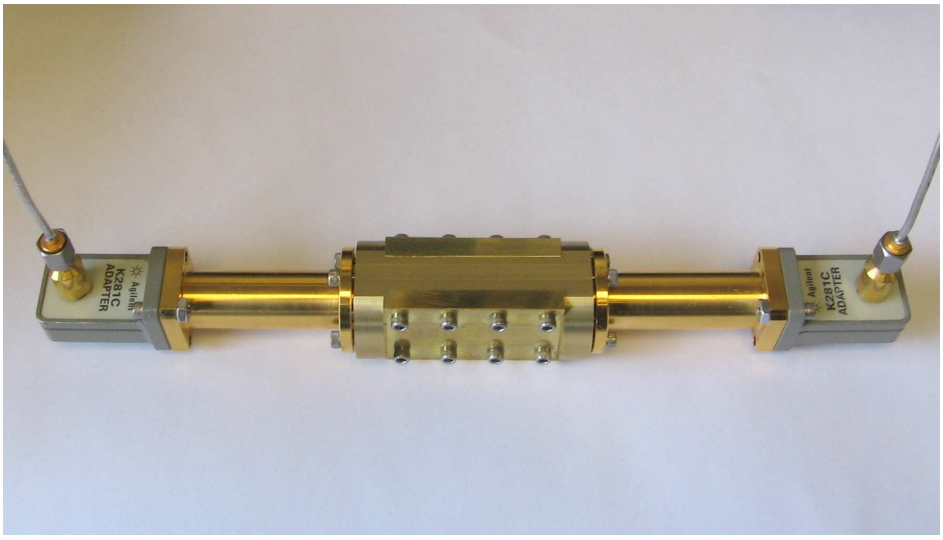

Fig. 7.— Photo of K-band scale model (a) under test.

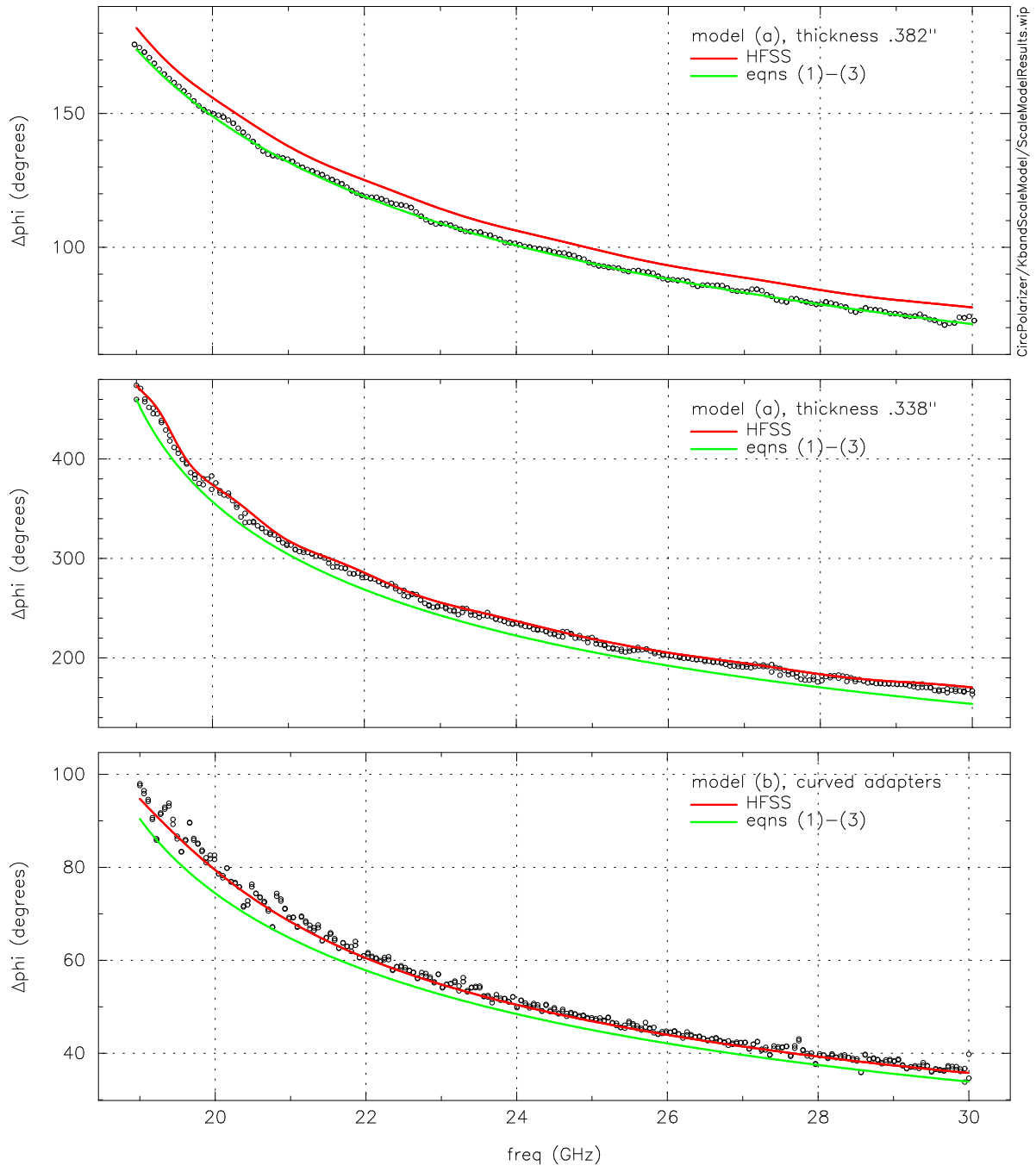

Fig. 8.— Network analyzer measurements of the differential phase shift $\Delta\phi$ through the K-band scale models in Fig. 6 are shown as open circles. Red curves show the predictions from HFSS simulations; green curves, from `pol.py` using equations (1)-(3). For both the network analyzer data and the HFSS simulations, the phase shifts are computed as $\Delta\phi = -\text{phase}(S_{21}^{Y}) + \text{phase}(S_{21}^{X})$ – phases are inverted to maintain consistency with our sign convention.

Fig. 8 compares the phase shifts measured on the scale models with the predictions from HFSS and from formulas (1)-(3). The phase shifts predicted by HFSS are 4–6% larger than the ones predicted by formulas (1)-(3) because HFSS correctly accounts for reactances at the interfaces of the circular and faceted guides. For model (a) we confirmed that the HFSS simulation exactly reproduces the analytic calculation if signals are launched and received inside the faceted guide, with no transition to circular waveguide at either end. The discrepancy between the HFSS results and the measurements for the $0.382''$ thickness in model (a) could be explained if the faceted guide were actually $0.384''$ thick; unfortunately the part is no longer available to check this dimension.

5. Leakage calculation

The performance of the polarizer may be characterized by its *leakage* D. Following Thompson, Moran, & Swenson (2001), the leakages D_r and D_l are defined by

$$v'_r = v_r + D_r\, v_l, \qquad v'_l = v_l + D_l\, v_r.$$

Here v'_r and v'_l are the measured signal voltages at the two outputs of the OMT, and v_r and v_l are the signals that would be observed with an ideally polarized feed. The magnitudes of the leakages (which are complex numbers) should be as small as possible, but they need not be zero. As long as the leakages are stable, they can be calibrated by observing astronomical sources.

To evaluate the leakage for various polarizer designs, we represent the electric field in the polarizer as a two component column vector

$$\mathbf{p} = \begin{bmatrix} E_x \\ E_y \end{bmatrix}.$$

E_x and E_y are complex numbers representing the E-field amplitudes and phases. Only the relative amplitudes and phases are significant for the leakage calculation. Basis vectors for linearly and circularly polarized waves (traveling in the $+z$ direction, out of the plane of the paper in Fig. 2) are

$$\mathbf{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{l} = \begin{bmatrix} 1/\sqrt{2} \\ j/\sqrt{2} \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} 1/\sqrt{2} \\ -j/\sqrt{2} \end{bmatrix}$$

To follow the signals through one or more polarizer sections, we multiply $\mathbf{p}$ by a series of 2×2 *Jones matrices*. There are just two operations: $\mathbf{J}_{rot}(\theta)$ computes the polarization vector in a new coordinate system rotated by angle θ, while $\mathbf{J}_{delay}(\Delta\phi(\nu))$ retards the phase of the Y-component of the polarization vector by $\Delta\phi(\nu)$.

$$\mathbf{J}_{rot}(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \quad \mathbf{J}_{delay}(\Delta\phi(\nu)) = \begin{bmatrix} 1 & 0 \\ 0 & e^{-j\Delta\phi(\nu)} \end{bmatrix}$$

It is convenient to deal with the case where the receiver transmits signals toward the antenna. In this case we assume that a purely linearly polarized wave ($\mathbf{x}$ or $\mathbf{y}$) enters the circular polarizer

from the OMT. We calculate the polarization vector $\mathbf{p}'(\nu) = \mathbf{J}_n\,\mathbf{J}_{n-1}\cdots\mathbf{J}_1\,\mathbf{p}$ that emerges from the polarizer. Ideally $\mathbf{p}'(\nu)$ is a pure circular polarization. The leakage D is the dot product of $\mathbf{p}'$ with the complex conjugate of the other circular basis vector **r*** or **l***. D is a complex number. For circular polarization the phase of D depends on the angle at which it is evaluated, but the magnitude is independent of this angle. Our goal is to minimize the magnitude $|D(\nu)|$ over a wide frequency range.

As an example, compute the polarization that emerges when an X-polarized input is incident on a perfect quarter wave retarder with its fast axis oriented at 45°:

$$\begin{aligned}
\mathbf{p}' &= \mathbf{J}_{rot}(-45^\circ)\;\mathbf{J}_{delay}(90^\circ)\;\mathbf{J}_{rot}(45^\circ)\;\mathbf{x} \\
&= \begin{bmatrix} \cos(-\frac{\pi}{4}) & \sin(-\frac{\pi}{4}) \\ -\sin(-\frac{\pi}{4}) & \cos(-\frac{\pi}{4}) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & e^{-j\pi/2} \end{bmatrix} \begin{bmatrix} \cos(\frac{\pi}{4}) & \sin(\frac{\pi}{4}) \\ -\sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{2}(1-j) \\ \frac{1}{2}(1+j) \end{bmatrix} = e^{-j\pi/4} \begin{bmatrix} 1/\sqrt{2} \\ j/\sqrt{2} \end{bmatrix} = e^{+j\pi/4}\;\mathbf{l}
\end{aligned}$$

The output $\mathbf{p}'$ will be recognized as pure left circular polarization: E_X and E_Y have equal amplitude, and E_X lags E_Y by 90°. Thus the leakage $D_l = \mathbf{p}' \cdot \mathbf{r}^* = 0$. The same retarder converts a Y-polarized signal to pure right circular polarization. Or, we can reverse the direction of the signals, injecting **l** or **r** into the horn end of the polarizer and computing the X and Y leakages at the OMT end; the magnitudes of the leakages are just the same.

6. Multisection polarizer designs

Table 2 summarizes a few of the possible designs for multisection circular polarizers derived by Kovac (2004) and by Pancharatnam (1955). Each retarder section is described by its phase shift (always 90° or 180°) at the center frequency, and by the angle of its fast axis (the axis with smaller phase shift) relative to the input reference plane. Polarizer A is a simple quarter wave retarder. Polarizers B and D (Kovac 2004) are 'maximally flat' in the sense that the derivatives of the leakage are zero at the center frequency. Designs C and E achieve greater bandwidths by rotating the retarders slightly to make the leakages (but not the derivatives) zero at 2 or 3 frequencies within the band. Fig. 9 shows the leakage vs. frequency for the polarizers in Table 2, as computed using script `pol.py`, listed in Appendix B. The detailed frequency response depends on the delay vs. frequency of the retarder elements, thus on the particular choice of f/r, the ratio of facet depth to waveguide radius.

Note that multisection polarizers are directional, in the sense that the 'horn' and 'OMT' ends are distinct. Injecting a linearly polarized signal into the 'horn' end does not, in general, produce a circularly polarized signal at the 'OMT' end. This is because rotations on the Poincaré sphere are not commutative.

	$\Delta\phi_1$	θ_1	$\Delta\phi_2$	θ_2	$\Delta\phi_3$	θ_3
A	90°	45°				
B	180°	15°	90°	75°		
C	180°	15°	90°	74.5°		
D	180°	6.05°	180°	34.68°	90°	102.27°
E	180°	6.50°	180°	34.57°	90°	101.14°

Table 2: Retarder phase shifts and angles for 1-, 2-, and 3-section polarizer designs. A is a simple quarter wave retarder; B and D are maximally flat designs from Kovac(2004); C is the design adopted for the CARMA polarizers; E was a 50% bandwidth design from Pancharatnam (1955).

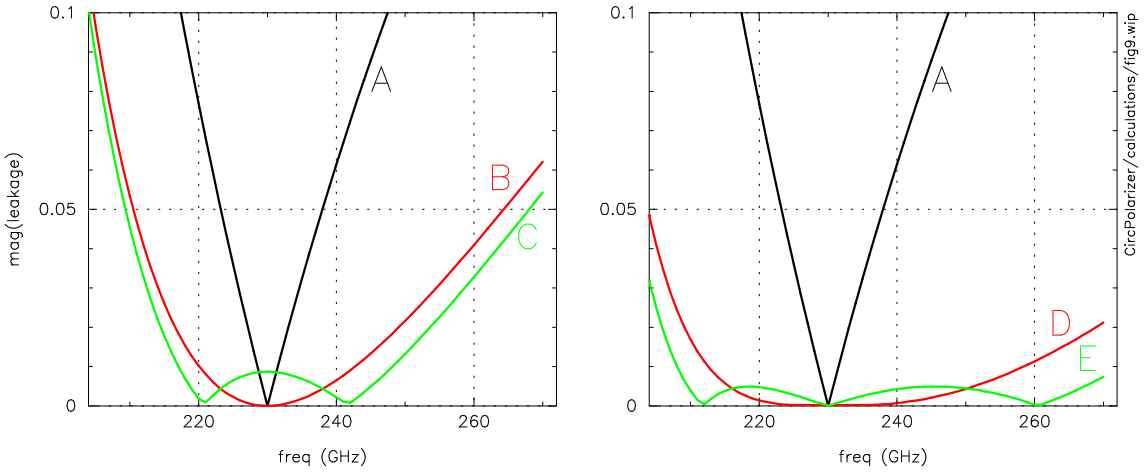

Fig. 9.— Leakages computed for the ideal polarizers in Table 2, for retarder elements constructed of faceted circular waveguide (diameter $d = 0.047''$, facet depth $f = 0.006''$). The retarder lengths were chosen to achieve phase shifts $\Delta\phi = 90°, 180°$ at 230 GHz. Curve C is the design adopted for the CARMA polarizers.

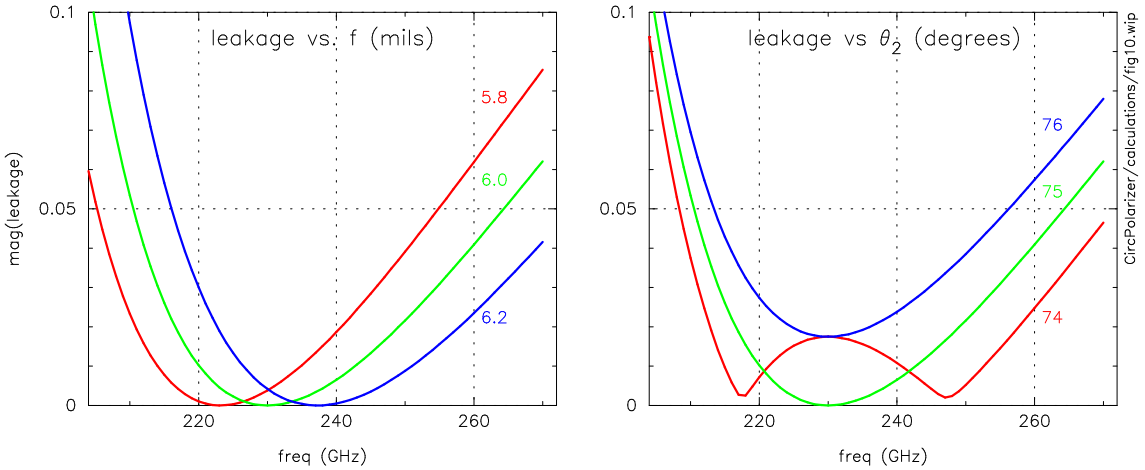

Fig. 10.— Leakages computed for polarizer design B, for $\pm 0.0002''$ facet depths and for $\pm 1°$ rotations.

Fig. 10 shows the leakage response of polarizer design B if the facet depths vary by $\pm 0.0002''$ or if the angle between the halfwave and quarter wave retarders varies by $\pm 1^\circ$. Changes in the facet depth shift the center frequency of the polarizer, while angular variations affect the shape of the leakage response. From Fig. 10 one concludes that it is advantageous to shift θ_2 to an angle slightly less than 75° to broaden the frequency response; this is the genesis of design C, which was chosen for the CARMA receivers.

7. Tolerance analysis

It is the exquisite cancellation of chromatic errors by a series of retarder sections that leads to excellent broadband performance in a multisection polarizer. Tiny variations in the dimensions of the retarder elements quickly spoil these cancellations, however. What are the allowable manufacturing tolerances that will give acceptable polarizer performance?

The Jones matrix formalism makes it easy to investigate the effect of fabrication tolerances on the polarizer performance. We consider errors in the waveguide diameter, facet depths, retarder section lengths, and retarder section angles. An error in the waveguide diameter or facet depth that is uniform for all the polarizer sections simply shifts the center frequency of the polarizer. More likely, however, the facet depth will differ randomly from one section to another. In order to simulate such effects, we assume that dimensional errors have truncated Gaussian distributions, with probability

$$P(\delta) = \begin{cases} A\, exp(-\delta^2/2\sigma^2) & \text{for } \delta \le \sigma \\ 0 & \text{for } \delta > \sigma \end{cases} \tag{4}$$

For each choice of machining tolerances, we computed leakage vs frequency for 200 instances of each polarizer design. We assumed that the error in the circular waveguide radius was uniform along each polarizer, but varied the facet depths, lengths, and rotations of the sections independently, with the following choices for σ:

$$\sigma(r), \sigma(f) = 0.00005'', 0.00010'', 0.00015'' \tag{5}$$

$$\sigma(L) = 0.001'' \tag{6}$$

$$\sigma(\theta) = 0.2^\circ \tag{7}$$

For each retarder, we assume that the facet depths are precisely equal on the top and bottom of the waveguide so that the facets remain symmetrical about the midplane of the waveguide.

Fig. 11 and 12 display the results of some of these tolerance tests; the red curve with error bars on each plot displays the mean and rms of the 200 tests. Fig. 11 demonstrates that the mean leakage and its rms scatter are smaller for deeper facets. This is simply because the *fractional* error in the facet depth, and hence in the phase shift, is smaller for deep facets. The benefits of deeper facets are partially offset by the greater reflection losses at the ends of the retarder sections, which led

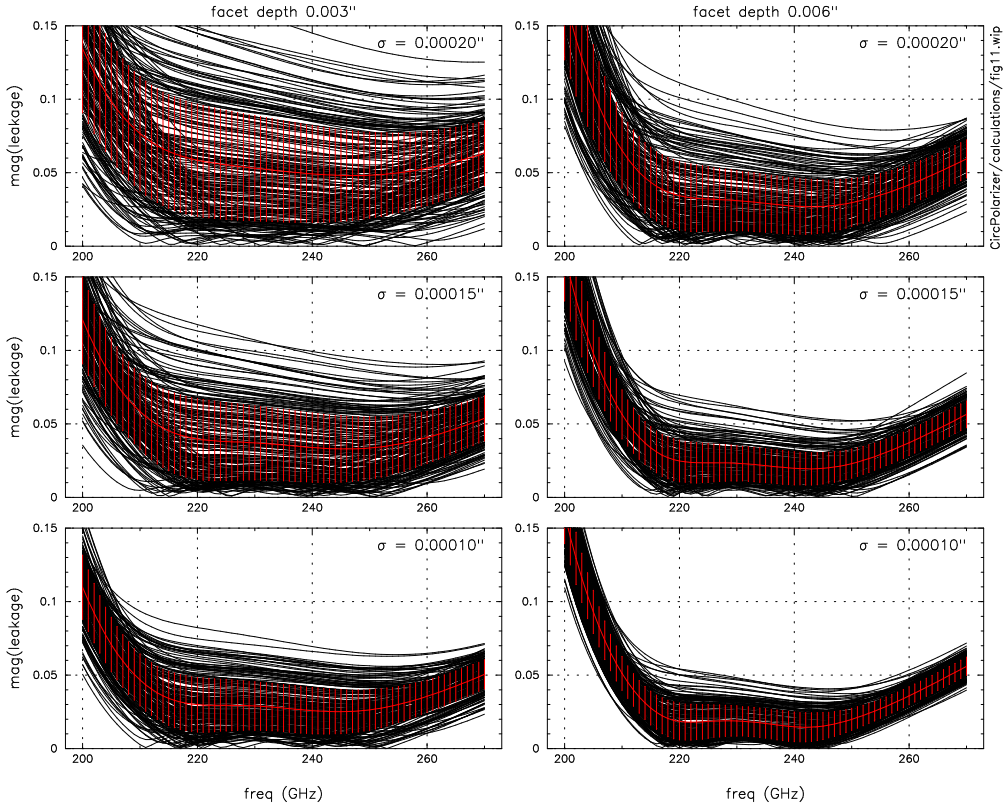

Fig. 11.— Leakages computed for 200 polarizers of design C, with truncated Gaussian dimensional errors given by equations (4)-(7). The nominal facet depth is 0.003″ for the simulations in the left hand panels, 0.006″ for the simulations on the right. $\sigma(r, f)$ is indicated in each panel. $\sigma(L) = \pm 0.001''$ and $\sigma(\theta) = \pm 0.2^\circ$ in all cases. Red curves with error bars on each plot show the mean and rms of the 200 trials.

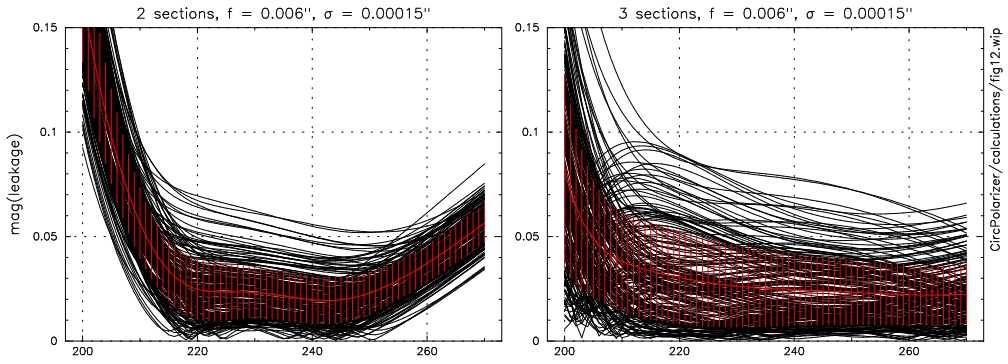

Fig. 12.— Leakages computed for 200 polarizers of 2-section design C and a 3-section design E, with truncated Gaussian dimensional errors given by equations (4)-(7). The mean facet depth is 0.006″ and $\sigma(r, f) = 0.00015''$ in both cases. When manufacturing tolerances are taken into account, the typical performance of a 3-section polarizer is little better than that of a 2-section polarizer.

us to limit the facet depth to $f = 0.006''$. Fig. 12 demonstrates that it is not worth the trouble to build a 3-section polarizer, at least not in the 1mm band. Its performance is little different than that of a 2-section polarizer unless one demands impossibly tight manufacturing tolerances.

8. Final design

Based on the tolerance analysis of section 7, we decided to use a 2-section polarizer (design C in Table 2) with $0.006''$ deep facets on the retarder sections. The curved transition in Fig. 5, with $R = 0.125''$, was adopted for the matching sections. HFSS simulations were used to determine the phase shifts through these matching sections, hence the correct total lengths of the 90° and 180° retarders. The final dimensions and tolerances are shown in Fig. 13; a rendering of the mandrel that will be used to electroform the polarizer is shown in Fig. 14.

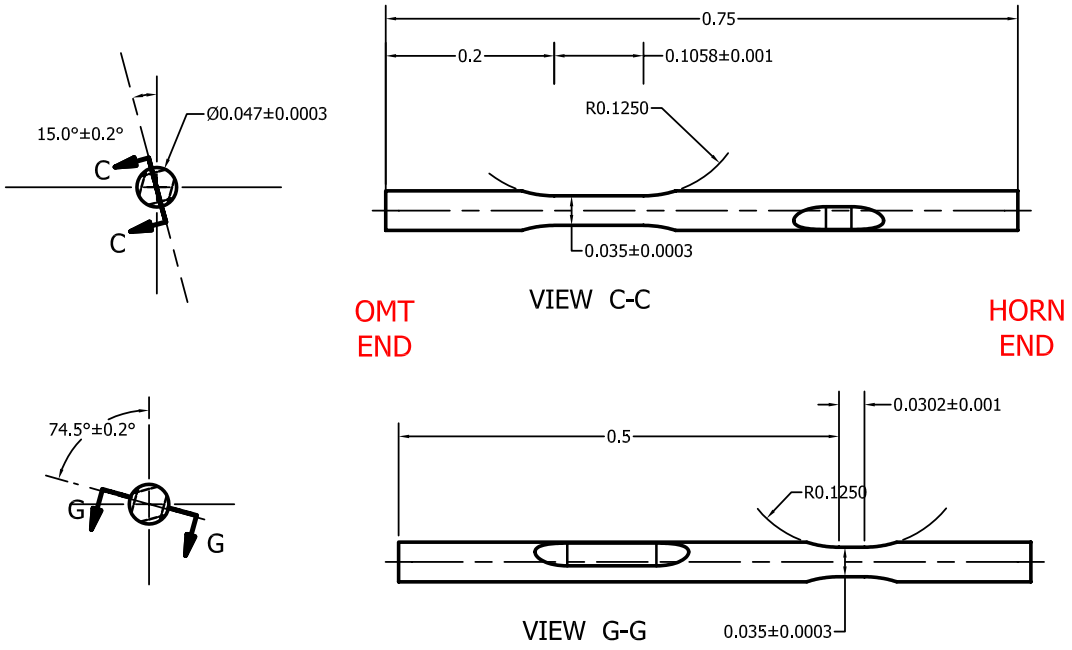

Fig. 13.— Dimensions and tolerances for the polarizer.

HFSS was used to check the final 2-stage design. Fig. 15 show that if a linearly polarized signal is incident on the OMT end of the polarizer, then the X- and Y-polarized signals at the horn end have almost equal amplitudes and differ in phase by approximately 90°, as expected for circular polarization. S11 and S22 (not shown) are below -20 dB across the band for both X- and Y-polarizations, not including the effect of the small step from the $0.047''$ polarizer waveguide to the $0.044''$ OMT waveguide. The leakage computed from the HFSS results is shown in Fig. 16. The E-field amplitudes in the polarizer, driven by both X- and Y-polarized signals at the OMT end, are shown in Fig. 17.

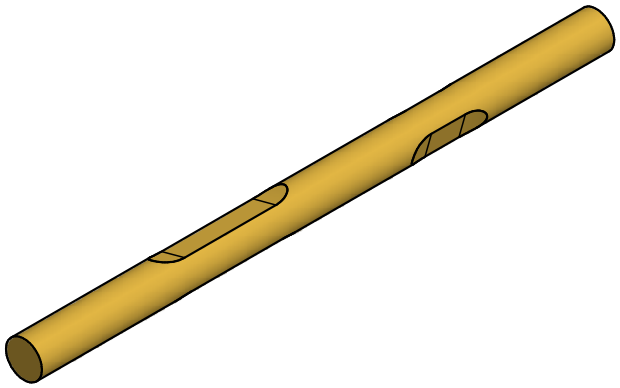

Fig. 14.— 3-D rendering of the final polarizer design. This can be thought of as an image of the aluminum mandrel that will be used for electroforming.

9. Leakage contributions from other optical elements

The polarization leakages computed thus far do not include the effects of optical elements between the feedhorn and the sky. There are at least 2 such elements that can degrade the polarization purity – the antireflection grooves on IR filters in the 6-m dewars, and the Mylar beamsplitters used on both the 6-m and 10-m receivers to inject the local oscillator into the beam.

Grooved IR filters. The 6-m dewars use $0.3''$ thick Teflon windows in the 50K radiation shields as infrared filters. Both the front and back surfaces of the windows are grooved to reduce reflections; the groove depth is $0.010''$ for the 1mm windows. The refractive index of the matching layers differ for signals polarized parallel and perpendicular to the grooves:

$$n_{\parallel} = \sqrt{\frac{\epsilon+1}{2}}, \qquad n_{\perp} = \sqrt{\frac{2\epsilon}{\epsilon+1}}.$$

For Teflon ($\epsilon = 2.08$) the matching layer has refractive index 1.24 for the electric field component $E_{\parallel}$ aligned with the grooves, and 1.16 for $E_{\perp}$. Unfortunately the grooves are oriented in the same direction on the front and back surfaces of the window, leading to a phase difference at 230 GHz of

$$\Delta\phi \;=\; \frac{2\pi}{\lambda}\,(2d)\,(n_{\parallel} - n_{\perp}) \;\sim\; 11^{\circ},$$

which causes a leakage of ~ 0.1, a very serious degradation in performance. Fortunately this can be avoided by regrooving one side of the Teflon filters in the perpendicular direction, or by replacing the Teflon windows with foam IR filters.

The lenses that serve as windows on the 6-m dewars also are antireflection coated with a series of concentric grooves. These grooves are expected to have much less effect on the leakage because the path delays are equal for the 2 polarizations when averaged over the lens. Another set of lenses at the feedhorn apertures – at an image of the primary mirror – were antireflection-coated by drilling a grid of holes into the surfaces (Plambeck 2000) to avoid polarization-sensitive delays.

Beamsplitters. The polarization purity also will be degraded by the beamsplitters that couple local oscillator power to the SIS mixers. The beamsplitter transmission differs for electric fields parallel

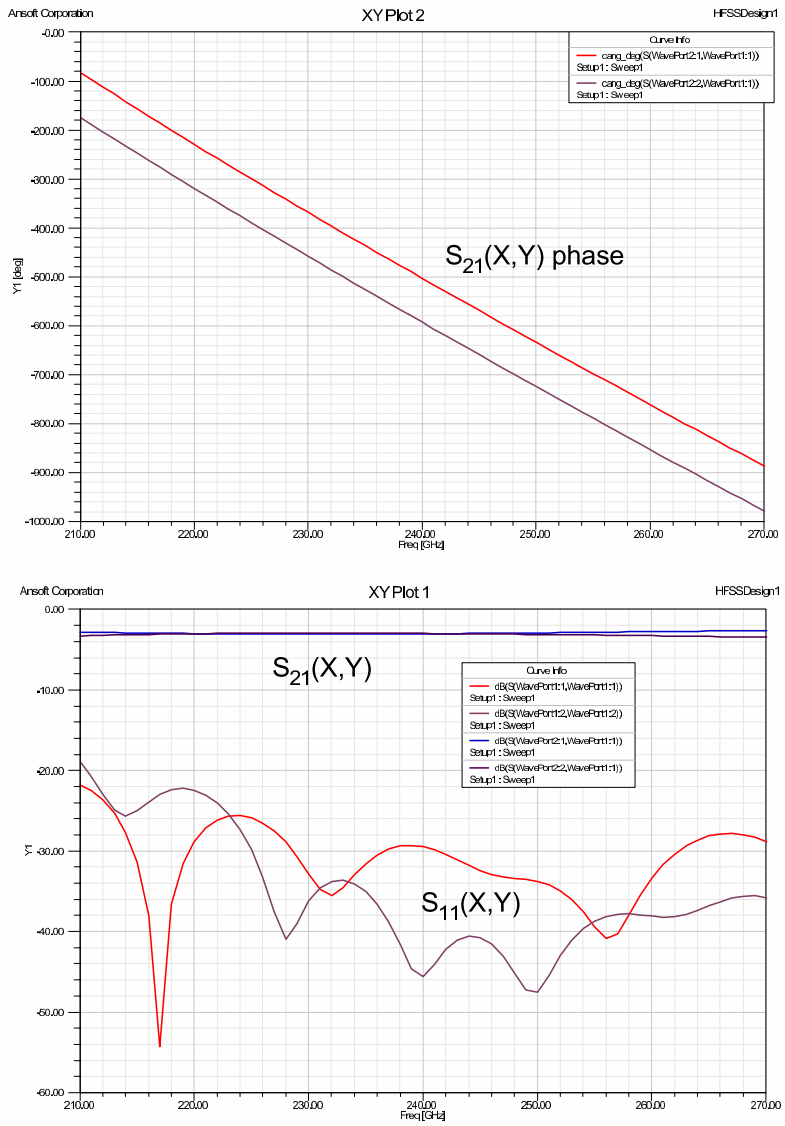

Fig. 15.— HFSS simulation results for the final 2 stage design in Fig. 13.

and perpendicular to the plane of incidence (at Brewster's angle $E_{||}$ is transmitted with no loss at all), hence a perfectly circularly polarized signal becomes elliptically polarized after passing through the beamsplitter. The effect is worse on the 10-m antennas, where the beamsplitters are mounted at a 45° angle of incidence; it is somewhat less troublesome on the 6-m antennas, where the angle of incidence is 35°. For the current SIS mixers (1 or 2 junction devices, 1 polarization), the 6-m receivers use a 0.0005″ thick Mylar beamsplitter in its more reflective orientation (LO polarized perpendicular to the plane of incidence), while the 10-m receivers use a 0.001″ thick splitter in its less reflective orientation. The beamsplitter reflectivities are plotted in Fig. 18. The new dual

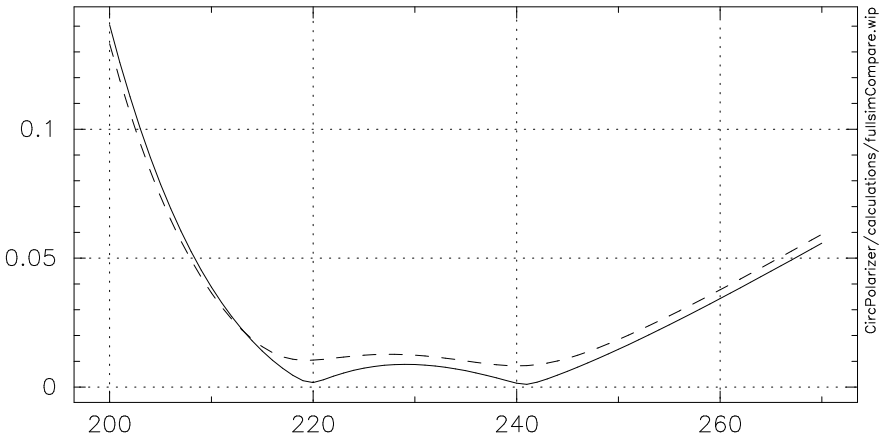

Fig. 16.— (*solid curve*) Polarization leakage through the polarizer in Fig. 13 computed from the HFSS simulation results in Fig. 15. (*dashed*) Leakage computed using the analytic model in `pol.py` for these same dimensions; the analytic model does not include phase shifts due to reactance at the transitions, hence the retarders are not exactly the correct lengths.

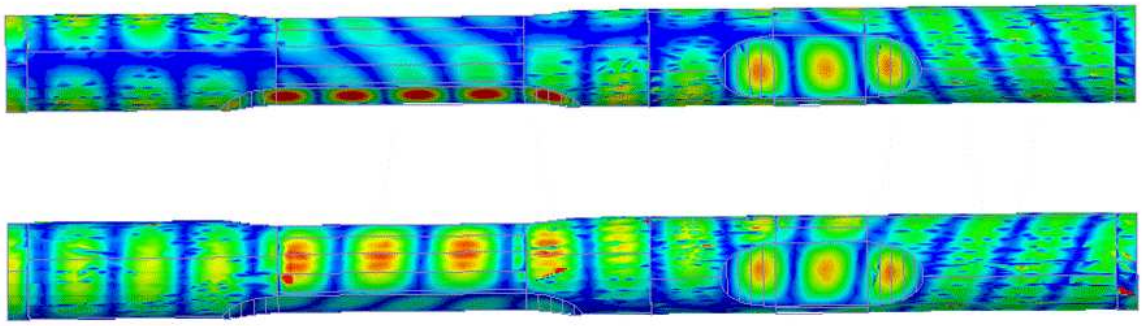

Fig. 17.— HFSS simulation showing E-field amplitudes in the polarizer. A Y-polarized signal incident from the left is converted to L (*top*), while an X-polarized signal is converted to R (*bottom*).

polarization system should require approximately 30 times more LO power because there are two mixers, each with a series array of 4 SIS junctions. The LO polarization will be flipped by 90 degrees on the 10-m telescopes when the new system is installed, and it will probably be necessary to use $0.001''$ or even $0.0015''$ thick beamsplitters.

We now evaluate the polarization leakage including the effects of the beamsplitters. From Born & Wolf (1959; sections 1.5.2 and 7.6.1) the amplitude of a signal transmitted through the beamsplitter is given by

$$A^{(t)} = \frac{tt'}{1 - r'^2 e^{i\delta}} A^{(i)} \tag{8}$$

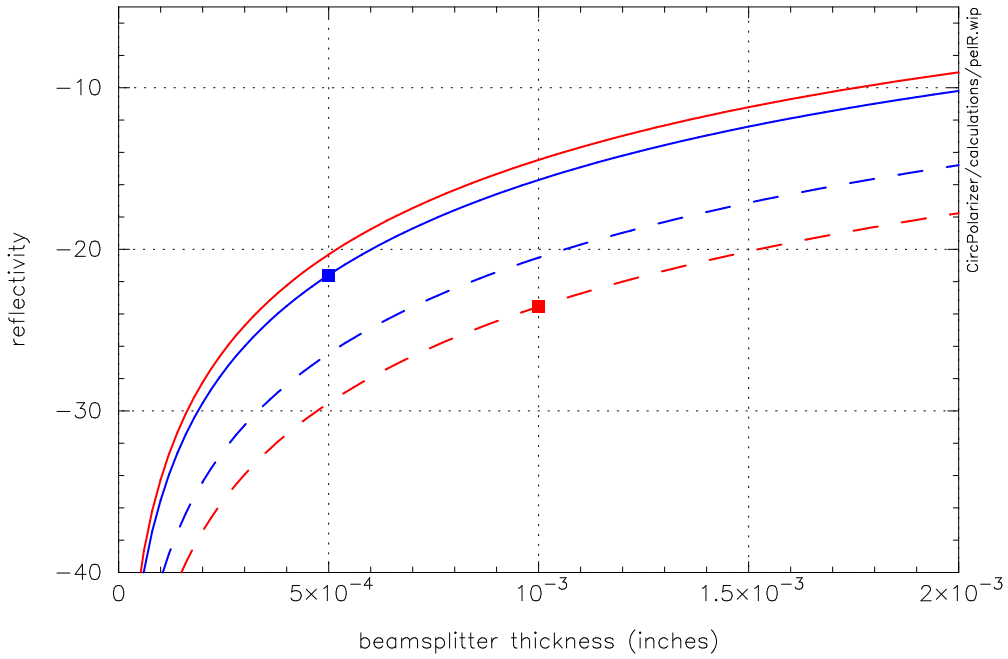

Fig. 18.— Reflectivity of the Mylar beamsplitters for signals polarized parallel (*dashed*) and perpendicular (*solid*) to the plane of incidence. The angle of incidence is 45° (red) on the 10-m telescopes, 35° (blue) on the 6-m telescopes. Solid squares indicate the beamsplitter thicknesses and orientations currently used for local oscillator injection.

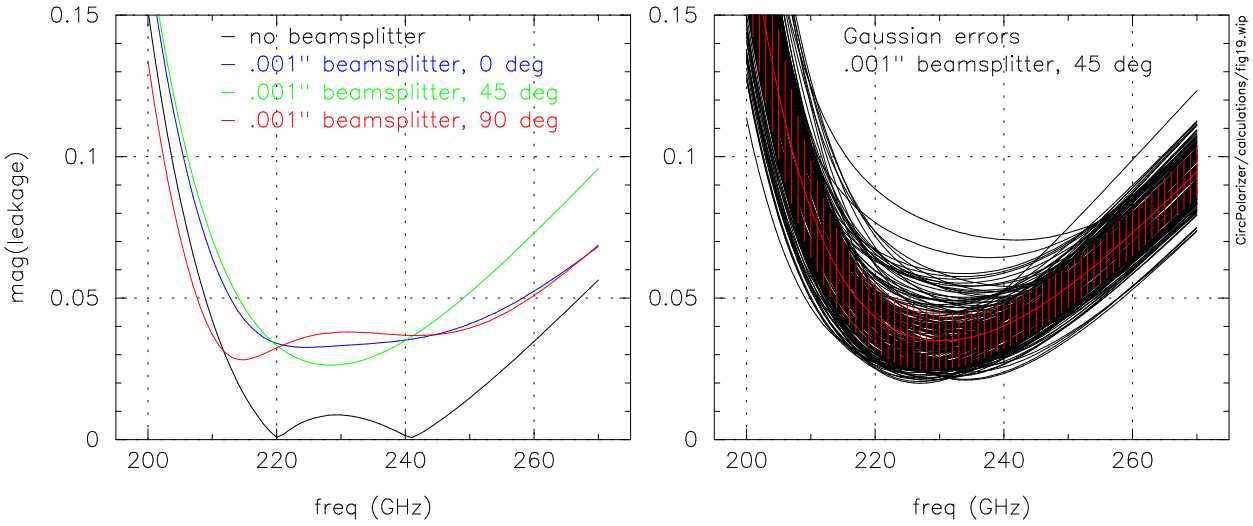

Fig. 19.— Theoretical effect on leakage for 0.0015″ thick mylar splitters at a 45° angle of incidence. The blue curve shows the *plane of incidence* parallel to the X-axis, while the red curve shows it parallel to the Y-axis. The right hand panel shows Gaussian error analysis of Fig. 12 with the beamsplitter.

where δ is the phase shift incurred by a signal that makes one round trip through the beamsplitter,

$$\delta = \frac{4\pi}{\lambda_0}\, n'\, t\, cos\theta', \tag{9}$$

and where the transmission and reflection coefficients at the air–beamsplitter interface are t and r going from air into the beamsplitter ($n_1 = 1, n_2 = 1.5$), and t' and r' from the beamsplitter into air ($n_1 = 1.5, n_2 = 1.0$). These coefficients differ for signals polarized parallel and perpendicular to the plane of incidence:

$$t_\parallel = \frac{2\, n_1\, cos\theta_i}{n_2\, cos\theta_i + n_1\, cos\theta_t} \qquad t_\perp = \frac{2\, n_1\, cos\theta_i}{n_1\, cos\theta_i + n_2\, cos\theta_t}$$

$$r_\parallel = \frac{n_2\, cos\theta_i - n_1\, cos\theta_t}{n_2\, cos\theta_i + n_1\, cos\theta_t} \qquad r_\perp = \frac{n_1\, cos\theta_i - n_2\, cos\theta_t}{n_1\, cos\theta_i + n_2\, cos\theta_t}$$

Since the beamsplitter attenuates the parallel and perpendicular components unequally, the orientation of beamsplitter plane of incidence relative to the polarizer axes matters, as demonstrated in the left hand panel of Fig. 19. Note that in rare cases the beamsplitter can even reduce the polarization leakage. The right hand panel in Fig. 19 shows the results of a Gaussian error analysis like that in section 7, but including the effect of a beamsplitter as well as machining tolerances for the polarizer. In this simulation the beamsplitter is $0.0010''$ thick, at angle of incidence 45°, with the plane of incidence at 45° to the principal axes of the OMT. From 218–250 GHz the average leakage is still ≤ 0.05. The beamsplitters will be eliminated in the future when CARMA switches to sideband separating mixers, for which the local oscillator is injected via waveguide directional couplers.

REFERENCES

Born, M. & Wolf, E. 1959, Principles of Optics (New York, Pergamon Press).

Greaves, J.S., et al. 2003, "A submillimetre imaging polarimeter at the James Clerk Maxwell Telescope,", MNRAS, 340, 353.

Kovac, J.M. 2004, "Detection of polarization in the cosmic microwave background using DASI," Ph.D. Thesis, University of Chicago.

Leitch, E.M. et al. 2002, "Measurement of Polarization with the Degree Angular Scale Interferometer," Nature, 420, 763.

Levy, R. 1995, "The Relationship Betweeen Dual Mode Cavity Cross-Coupling and Waveguide Polarizers," IEEE Transactions on Microwave Theory and Techniques, MTT-43, 2614.

Levy, R. 1997, "Correction to The Relationship Between Dual Mode Cavity Cross-Coupling and Waveguide Polarizers," IEEE Transactions on Microwave Theory and Techniques, MTT-45, 704.

Lilie, P. 2001, "Wide-Band Circular Polarizers Made with Quarter-Wave and Half-Wave Plates," ATNF Technical Document 39.3/106 (www.atnf.csiro.au/observers/memos/AT39.3_106.pdf).

Lin, S.-L., Li, L.-W., Yeo, T.-S., and Leong, M.-S. 2001, "Cutoff Wavenumbers in Truncated Waveguides," IEEE Microwave and Wireless Components Letters, 11, 214.

Navarrini, A., Bolatto, A., and Plambeck, R.L. 2006, "Preliminary test results of the turnstile junction waveguide orthomode transducer for the 1mm band," CARMA memo 32.

Navarrini, A., and Plambeck, R.L. 2006, "A Turnstile Junction Waveguide Orthomode Transducer,", IEEE Transactions on Microwave Theory and Techniques, MTT-54, 272.

Pancharatnam, S. 1955, "Achromatic Combinations of Birefringent Plates," Proceedings of the Indian Academy of Sciences, 41, 130 (www.ias.ac.in/j_archive/proca/41/vol41contents.html).

Plambeck, R. 2000, "Measurements of the 1mm Receiver Optics," BIMA memo 79 (http://carma.astro.umd.edu/memos).

Pyle, J.R. and Angley, R.J. 1964, "Cutoff Wavelengths of Waveguides with Unusual Cross Sections," IEEE Transactions on Microwave Theory and Techniques, MTT-12, 556.

Pyle, J.R. 1964, "Circular Polarizers of Fixed Bandwidth," IEEE Transactions on Microwave Theory and Techniques, MTT-12, 557.

Sinnott, D.H., Cambrell, G.K., Carson, C.T., and Green, H.E. 1969, "The Finite Difference Solution of Microwave Circuit Problems," IEEE Transactions on Microwave Theory and Techniques, MTT-17, 464.

Thompson, A.R., Moran, J.M., & Swenson, G.W., Jr. 2001, Interferometry and Synthesis in Radio Astronomy, 2nd edition (New York, Wiley).

Wang, C.Y. 2000, "Frequencies of a Truncated Circular Waveguide – Method of Internal Matching," IEEE Transactions on Microwave Theory and Techniques, 48, 1763.

10. Appendix A

Python script to compute cutoff frequencies for faceted circular waveguide using the method of Wang (2000).

```
# ------------------------------------------------------------------------------------------------------------ #
#  wang.py - compute cutoff wavenumbers and frequencies for faceted circular waveguide using the
#   method of Wang (2000, IEEE MTT, 48, 1763-65)
# ------------------------------------------------------------------------------------------------------------ #
from scipy.special import *
import math

# --- convert wavenumber to frequency (GHz) for waveguide with radius r_inches --- #
def fGHz ( k, r_inches ) :
  c = 2.99792458e10
  rcm = 2.54 * r_inches
  return 1.e-9 * c * k / (2 * math.pi * rcm)

# --- eqn (4) --- #
def rho ( theta, a ) :
  return sqrt( 1. + 4.*a*a - 4.*a*cos(theta) )

# --- eqn (5) --- #
def phi ( theta, a ) :
  return math.atan2( sin(theta), (2.*a - cos(theta)) )

# --- compute cofactors of Cn from eqn (18) for pol = 'Y', or for equivalent eqn for pol = 'X' --- #
def element ( pol, n, a, theta, k ) :
  fact = float(math.factorial(2.*n - 1))
  if (cos(theta) > a ) :
    if (pol == 'X') :
      term = cos( (2.*n-1.)*theta ) * jv( 2.*n-1, k ) \
           - cos( (2.*n-1.)*phi(theta,a) ) * jv( 2.*n-1, k*rho(theta,a) )
    else:
      term = sin( (2.*n-1.)*theta ) * jv( 2.*n-1, k ) \
           - sin( (2.*n-1.)*phi(theta,a) ) * jv( 2.*n-1, k*rho(theta,a) )
  else :
    term0 = (k * jv( 2.*n-2., k ) - (2*n - 1) * jv( 2*n-1., k ) )
    if (pol == 'X') :
      term = cos( (2.*n-1.)*theta ) * term0
    else :
      term = sin( (2.*n-1.)*theta ) * term0
  return fact*term

# --- compute determinant; determinant should be zero for nontrivial Cn --- #
def detarray ( pol, N, a, k ) :
  ma = zeros( (N,N) )
  for j in range(0,N) :
    thetaj = (j + 0.5) * math.pi/ (2. * N)
    for i in range(0,N) :
      n = i+1
      ma[j,i] = element( pol, n, a, thetaj, k )
  return linalg.det(ma)

# ------------------------------------------------------------------------------------------------------------ #
# solve for k by stepping through range, finding zero-crossing of determinant
```

```
#  pol = 'X' (E-field parallel to facets, higher cutoff) or 'Y'
#  N = number of terms in Bessel function expansion
#  a = 1-f/R, where R is the radius, f is the facet depth
#  k0 = initial estimate for k
#  kstep = step size to begin with
# ------------------------------------------------------------------------------------------------------------ #
def ksolve ( pol, N, a, k0, kstep ) :
  k = k0
  lastdet = detarray( pol, N, a, k0 )
  while (kstep > .0000005) and (k < 4) :
    k = k + kstep
    det = detarray( pol, N, a, k )
    if ((lastdet > 0.) and (det > 0.)) or ((lastdet < 0.) and (det < 0.)) :
      lastdet = det
    else :                  # zero crossing somewhere between k-delta and k
      k = k - kstep
      kstep = kstep/2.    # reduce step size by factor of 2
    print k, kstep, lastdet, det
  if (k >= 4) : print "ERROR"
  return k

# ------------------------------------------------------------------------------------------------------------ #
#  ksolve answer oscillates vs N, so use average solution for N=20,60,1
# ------------------------------------------------------------------------------------------------------------ #
def kiterate ( pol, a ) :
  k0 = 1.5
  kstep = 0.5
  N = 20
  k20 = ksolve ( pol, N, a, k0, kstep )
  sum = k20
  n = 1
  for N in range (21, 60, 1) :
    k = ksolve ( pol, N, a, (k20-.01), 0.001 )
    sum = sum + k
    n = n + 1
  return sum/float(n)

# --- find cutoff frequencies vs facet depth for our waveguide --- #
def solvall ( ):
  for f in arange(0.0014,0.0082,0.0002):
    a = (.0235 - f)/.0235
    kX = kiterate( 'X', a )
    kY = kiterate( 'Y', a )
    ofile = open("wang.dat", "a")
    ofile.write("%8.5f %8.5f %9.6f %9.6f %11.5f %11.5f\n" % (a, f, kX, kY, fGHz(kX,.0235), fGHz(kY,.0235)) )
    ofile.close()
```

11. Appendix B

All calculations discussed in this memo were done with the following Python script. Subroutines `pol.fig1()`, `pol.fig2()`, etc. were used to generate data for the corresponding figures.

```
# ------------------------------------------------------------------------------------------------------------ #
# pol.py

from Numeric import *
import math
import cmath
import random

# ------------------------------------------------------------------------------------------------------------ #
# define basis vectors in reference coordinate system (aligned with OMT axes)
# each vector consists of 2 complex numbers [ (Re_x,Im_x), (Re_y,Im_y) ]
# R and L propagate in +Z direction according to right hand rule
# note: x.conjugate() gives complex conjugate of a number
# ------------------------------------------------------------------------------------------------------------ #
X = array( [1,0] )
Y = array( [0,1] )
R = array( [1./sqrt(2.), -1j*(1./sqrt(2.))] )
L = array( [1./sqrt(2.), +1j*(1./sqrt(2.))] )
clight = 29.9792458 # speed of light, cm/nanosec

# ------------------------------------------------------------------------------------------------------------ #
# returns new basis vector in a coordinate system rotated by thetaDegrees
# ------------------------------------------------------------------------------------------------------------ #
def Jrot ( vec, thetaDegrees ) :
  rad = math.pi * thetaDegrees/180.
  rotmat = array( [[cos(rad),sin(rad)],[-sin(rad),cos(rad)]] )
  return dot(rotmat,vec)

# ------------------------------------------------------------------------------------------------------------ #
# returns basis vector after passing through polarizer section
# component 2 (Y-axis) is advanced by delayDegrees relative to component 1 (X-axis)
# ------------------------------------------------------------------------------------------------------------ #
def Jdelay ( vec, delayDegrees ) :
  rad = math.pi * delayDegrees/180.
  rotmat = array( [ [1, 0], [0, cmath.exp(-1.j * rad)] ] )
  return dot( rotmat,vec )

# ------------------------------------------------------------------------------------------------------------ #
# returns basis vector after transmission through a beamsplitter
# tpel is thickness in inches, angI is angle of incidence in degrees
# X axis is assumed parallel to the plane of incidence; light polarized parallel to the plane of incidence
#   is less strongly reflected; at Brewster's angle none will be reflected
# Y axis is perp to plane of incidence; it is more strongly reflected
# ------------------------------------------------------------------------------------------------------------ #
def Jbsplit ( vec, tpel=.001, angI=45., fGHz=230. ) :
   [tpar,tperp,Rpar,Rperp] = pellicle( tpel=tpel, angI=angI, fGHz=fGHz)
   rotmat = array( [ [tpar, 0.], [0., tperp] ] )
   return dot( rotmat,vec )

# ------------------------------------------------------------------------------------------------------------ #
# returns cutoff freqs [fcX, fcY], in GHz, for faceted circular waveguide that is squeezed along Y-direction
```

```
# r is waveguide radius in inches, x is the normalized facet depth f/r
# based on 5th order polynomial fit to analytic results from Appendix A, or HFSS simulations
# use source='HFSS' for HFSS (default), source='Wang' for analytic fits
# ------------------------------------------------------------------------------------------------------------ #
def cutoff( r, x, source='HFSS' ) :
  if ( source == 'Wang') :
    kX = 1.841184 + 0.309847*x + 8.60483*pow(x,2.) - 29.3916*pow(x,3.) + 75.4104*pow(x,4.) - 67.4144*pow(x,5.)
    kY = 1.841184 - 0.0900839*x - 3.30519*pow(x,2.) + 13.3106*pow(x,3.) - 26.8144*pow(x,4.) + 22.970*pow(x,5.)
  else :
    kX = 1.841184 + 0.301574*x + 8.9118*pow(x,2.) - 33.253*pow(x,3.) + 93.2359*pow(x,4.) - 94.615*pow(x,5.)
    kY = 1.841184 - 0.0862305*x - 3.41638*pow(x,2.) + 14.65*pow(x,3.) - 32.7615*pow(x,4.) + 31.7498*pow(x,5.)
  fcX = clight * kX / (2. * math.pi * r * 2.54)
  fcY = clight * kY / (2. * math.pi * r * 2.54)
  return [fcX, fcY]

# ------------------------------------------------------------------------------------------------------------ #
# compute length in inches of faceted guide to achieve differential phase shift dphi0, in degrees
# r is waveguide radius in inches, f is facet depth in inches, fGHz is freq in GHz
# fcX and fcY are override cutoff freqs in GHz; if either is zero, cutoff freqs will be taken from
#   subroutine cutoff, using 5th order polynomial fit to the HFSS-derived cutoffs
# ------------------------------------------------------------------------------------------------------------ #
def length( r=0.0235, f=0.006, dphi0=90., fGHz=230., fcX=0., fcY=0. ) :
  if (f == 0.) : return 0.
  if ( (fcX == 0.) or (fcY == 0.) ) :
    [fcX,fcY] = cutoff( r, f/r, source='HFSS' )
  length = clight * dphi0 / (2.54 * 360.* (sqrt(fGHz*fGHz-fcY*fcY) - sqrt(fGHz*fGHz-fcX*fcX)))
  return length

# --- L90 retarder lengths in table 1 are from the actual HFSS freqs, not the polynomial fit --- #
def table1 () :
  print "0.001 %8.4f" % length( r=0.0235, fcX=149.307, fcY=146.471, fGHz=230.)
  print "0.002 %8.4f" % length( r=0.0235, fcX=153.119, fcY=145.203, fGHz=230.)
  print "0.003 %8.4f" % length( r=0.0235, fcX=158.048, fcY=143.674, fGHz=230.)
  print "0.004 %8.4f" % length( r=0.0235, fcX=163.985, fcY=142.030, fGHz=230.)
  print "0.005 %8.4f" % length( r=0.0235, fcX=170.938, fcY=140.363, fGHz=230.)
  print "0.006 %8.4f" % length( r=0.0235, fcX=178.985, fcY=138.732, fGHz=230.)
  print "0.007 %8.4f" % length( r=0.0235, fcX=188.256, fcY=137.176, fGHz=230.)

# --- generate smooth curves of cutoff freq for FIG 4 --- #
def fig4() :
  r = .0235
  for f in arange( 0.000, 0.0072, 0.0002) :
    [fcX,fcY] = cutoff( r, f/r )
    print "%6.4f  %8.4f  %8.4f" % (f, fcX, fcY)

# ------------------------------------------------------------------------------------------------------------ #
# compute phase delay in degrees of Y-pol vs X-pol signals traveling through length L; dimensions in inches
# fcX and fcY are override cutoff freqs in GHz; if either is zero, use values from subroutine cutoff
# ------------------------------------------------------------------------------------------------------------ #
def dphi( r=0.0235, f=0.006, L=0.001, fGHz=230., fcX=0., fcY=0. ) :
  if ( (fcX == 0.) or (fcY == 0.) ) :
    [fcX,fcY] = cutoff( r, f/r, source='HFSS' )
  dphi = 360. * L * 2.54 * (sqrt(fGHz*fGHz-fcY*fcY) - sqrt(fGHz*fGHz-fcX*fcX)) / clight
  return dphi

# ------------------------------------------------------------------------------------------------------------ #
# compute [length, differential phase shift] through a radiused adapter from faceted circ to circ guide
# Rc is the cutter radius in inches; axis of cutter is perpendicular to axis of waveguide
```

```
# ------------------------------------------------------------------------------------------------------------ #
def dphitaper2 ( r=0.0235, f=0.006, Rc=0.125, fGHz=230., nsteps=1000 ) :
  df = f/nsteps                                                 # increment in facet depth
  f1 = f                                                        # facet depth at start of segment
  x1 = 0.                                                       # x-coordinate at start of segment
  totphase = 0.
  for n in range(nsteps) :
    f2 = f1 - df                                                # facet depth at end of segment
    x2 = sqrt(Rc*Rc - pow((f - f2 - Rc), 2.))                   # x-coordinate at end of segment
    dphase = dphi(r=r, f=(f1+f2)/2., L=(x2-x1), fGHz=fGHz )     # using avg facet depth
    totphase = totphase + dphase
    x1 = x2
    f1 = f2
  return [x2, totphase]           # return X length of transition in inches, and total diff phase through it

# ------------------------------------------------------------------------------------------------------------ #
# predictions for Kband scale models; gap = 0.382" for straight case 1, 0.338" for straight case 2
# ------------------------------------------------------------------------------------------------------------ #
def fig8ab () :
  ofile = open("KbandStraight.dat","w")
  for f in arange(19.,30.05,.05) :
    ofile.write("%6.2f %8.4f %8.4f\n" %
     (f, dphi( r=0.2275, f=.0365, L=1.822, fGHz=f ), dphi( r=0.2275, f=.0585, L=1.822, fGHz=f )) )
  ofile.close()

def fig8c () :
  ofile = open("KbandCurved.dat","w")
  for f in arange(19.,30.05,.05) :
    [x2, ph] = dphitaper2( r=0.2275, f=.05675, Rc=1.210, fGHz=f )    # min gap is 0.3415"
    ofile.write("%6.2f %8.4f \n" % (f, 2.*ph))
  ofile.close()

# ------------------------------------------------------------------------------------------------------------ #
# reflection coefficient from circular guide of radius r1 (inches) to guide of radius r2 (inches)
# ------------------------------------------------------------------------------------------------------------ #
def reflect( fGHz, r1, r2 ) :
  fc1 = clight/(3.4126 * r1 * 2.54)
  fc2 = clight/(3.4126 * r2 * 2.54)
  Z1 = fGHz * 377./sqrt(fGHz*fGHz - fc1*fc1)
  Z2 = fGHz * 377./sqrt(fGHz*fGHz - fc2*fc2)
  vR = (Z1 - Z2)/(Z1 + Z2)
  print "voltage, power reflection coefficients: %.3e %.3e" % (vR, vR*vR)

# ------------------------------------------------------------------------------------------------------------ #
# check algebra of example at end of section 5
# ------------------------------------------------------------------------------------------------------------ #
def example() :
  v1 = X
  v2 = Jrot(v1, 45)
  v3 = Jdelay(v2, 90)
  v4 = Jrot(v3, -45)
  print v4
  rad = math.pi/4
  print cmath.exp(+1.j * rad) * v4
  print "D_L ", dot(v4,L), abs(dot(v4,L)) # note: R* = L
  print "D_R ", dot(v4,R), abs(dot(v4,R)) # note: L* = R

# ------------------------------------------------------------------------------------------------------------ #
```

```
# writes out dimensions of a single polarizer to 1 line of output file;
# convert inches to mils for more compact format
# ------------------------------------------------------------------------------------------------------------ #
def dumpDimensions( dimfile, r, a1, f1, L1, a2, f2, L2, a3, f3, L3 ) :
  ofile = open(dimfile, "a")
  ofile.write(" %6.3f %7.2f %5.3f %6.2f" % (1000.*r, a1, 1000.*f1, 1000.*L1) )
  ofile.write(" %7.2f %5.3f %6.2f" % (a2, 1000.*f2, 1000.*L2) )
  ofile.write(" %7.2f %5.3f %6.2f\n" % (a3, 1000.*f3, 1000.*L3) )
  ofile.close()

# ------------------------------------------------------------------------------------------------------------ #
# writes out one polarizer dimension (e.g., facet depth of section 1) per line for multiple polarizers
# ------------------------------------------------------------------------------------------------------------ #
def dumplist ( ofile, x, label, fmt, mult ):
  ofile.write( label )
  ntrials = len(x)
  for n in range(ntrials) :
    ofile.write( fmt % (mult * x[n] ) )
  ofile.write("\n")

# ------------------------------------------------------------------------------------------------------------ #
# compute ampX, phsX, ampY, phsY, phsdif given a pol vector [ (re X, im X), (re Y, im Y) ]
# ------------------------------------------------------------------------------------------------------------ #
def ampPhs( vec ) :
  phsX = 180. * math.atan2(vec[0].imag,vec[0].real) / math.pi
  phsY = 180. * math.atan2(vec[1].imag,vec[1].real) / math.pi
  phsdif = phsX - phsY
  if (phsdif > 180.) :
    phsdif = phsdif - 360.
  if (phsdif < -180.) :
    phsdif = phsdif + 360.
  return [ abs(vec[0]), phsX, abs(vec[1]), phsY, phsdif ]

# ------------------------------------------------------------------------------------------------------------ #
# compute leakages vs freq for 1-section, 2-section, or 3-section polarizers
# read polarizer dimensions from 'dimfile', write leakages to 'leakfile'
# note: one polarizer per ROW on input, one per COLUMN on output
# dimfile should have 10 columns (as in dumpDimensions); angles in degrees, lengths in mils
# optional: apfile lists amps and phase difference of X and Y components for comparison with HFSS
# optional: evaluate leakage after beamsplitter  apel,tpel,aIpel,aeval
#    apel = angle of plane of incidence, relative to X=0 axis
#    tpel = beamsplitter ("pellicle") thickness in inches
#    aIpel = angle of incidence to the beamsplitter, degrees (0 = beamsplitter normal to axis)
# optional: evaluate leakage at angle aeval (shouldn't affect magnitude of the leakage)
# optional: compute correlation efficiency < V1 V0* > for all polarizers vs the first one
# ------------------------------------------------------------------------------------------------------------ #
def computeLeakage( dimfile, leakfile, apfile=None, apel=0., tpel=0., aIpel=0., aeval=0., efficfile=None ) :
  r  = []   # empty list of circular waveguide radii
  a1 = []   # angle of section 1 relative to input Y-pol
  f1 = []   # facet depth of section 1
  L1 = []   # length of section 1
  a2 = []   # angle of section 2 relative to section 1
  f2 = []   # facet depth of section2
  L2 = []   # length of section 2
  a3 = []   # angle of section 3 relative to section 2
  f3 = []   # facet depth of section 3
  L3 = []   # length of section 3
```

```
# --- read the dimensions into internal lists --- #
infile = open(dimfile, "r")
ofile = open(leakfile, "w")
if (efficfile) : ofile3 = open( efficfile, "w" )
if (apfile) :
  ofile2 = open( apfile, "w")
  ofile2.write("# fGHz, abs(Ex), phs(Ex), abs(Ey), phs(Ey), phsdif, leakage, phsleak\n" )
ntrials = 0
for line in infile:
  if line.startswith("#"):
    ofile.write(line)
  else:
    a = line.split()                      # split line into string tokens
    r.append( float(a[0])/1000. )         # convert diameter from mils back to inches
    a1.append( float(a[1]) )
    f1.append( float(a[2])/1000. )     # convert facet depth from mils back to inches
    L1.append( float(a[3])/1000. )
    a2.append( float(a[4]) )
    f2.append( float(a[5])/1000. )     # convert facet depth from mils back to inches
    L2.append( float(a[6])/1000. )
    a3.append( float(a[7]) )
    f3.append( float(a[8])/1000. )     # convert facet depth from mils back to inches
    L3.append( float(a[9])/1000. )
    ntrials = ntrials + 1
infile.close()

# --- compute leakage array for each set of polarizer dimensions --- #
freq = arange(200.,271.)
leakage = zeros([len(freq),ntrials],Float)    # create array to hold leakages
effic = zeros([len(freq),ntrials],Float)      # create array to hold correlation efficiencies
for n in range(ntrials) :
  m = 0                                       # freq index
  for fGHz in freq :

    v1 = Y                                    # Y-pol incident on section 1
    v2 = Jrot( v1, a1[n] )
    afinal = -1.*a1[n]
    v3 = Jdelay( v2, dphi( r[n], f1[n], L1[n], fGHz ) )

    if (f2[n] > 0.) :                         # section 2, if present
      v2 = Jrot( v3, a2[n] )
      afinal = afinal - a2[n]
      v3 = Jdelay( v2, dphi( r[n], f2[n], L2[n], fGHz ) )

    if (f3[n] > 0.) :                         # section 3, if present
      v2 = Jrot( v3, a3[n] )
      afinal = afinal - a3[n]
      v3 = Jdelay( v2, dphi( r[n], f3[n], L3[n], fGHz ) )

    v1 = Jrot(v3, afinal)              # rotate back to original reference frame

    if (tpel > 0.) :                # optional beamsplitter section
      v2 = Jrot( v1, apel )                   # rotate X axis parallel to plane of plane of incidence
      v3 = Jbsplit(v2, tpel, aIpel, fGHz )    # apply amplitude and phase shifts
      v1 = Jrot( v3, -1.*apel ) # rotate back to original reference frame

    vout = Jrot( v1, aeval ) # evaluate leakage at angle aeval
```

```
    if (n == 0) : vsave = array( [vout[0].conjugate(), vout[1].conjugate() ] )
      # save polarization vector of 1st trial for optional efficiency calculations

    # --- assume RCP out (true for positive angles a1,a2,a3) --- #
    cleak = dot(vout, R)                    # single complex number; note R = L*
    leakage[m,n] = abs(cleak)               # magnitude of the leakage

    if (efficfile) : effic[m,n] = abs(dot(vsave,vout))
    if (apfile) :
      [ampX, phsX, ampY, phsY, phdif ] = ampPhs( vout )
      ofile2.write("%6.1f  %8.4f %8.4f %8.4f %8.4f %8.4f\n" % (fGHz, ampX, phsX, ampY, phsY, leakage[m,n]))
    m = m + 1

if (apfile) : ofile2.close()

# --- write dimensions in columns to output file --- #
dumplist ( ofile, r,  "# r  :", " %6.3f", 1000.)
dumplist ( ofile, a1, "# a1 :", " %6.2f", 1.)
dumplist ( ofile, f1, "# f1 :", " %6.3f", 1000.)
dumplist ( ofile, L1, "# L1 :", " %6.2f", 1000.)
dumplist ( ofile, a2, "# a2 :", " %6.2f", 1.)
dumplist ( ofile, f2, "# f2 :", " %6.3f", 1000.)
dumplist ( ofile, L2, "# L2 :", " %6.2f", 1000.)
dumplist ( ofile, a3, "# a3 :", " %6.2f", 1.)
dumplist ( ofile, f3, "# f3 :", " %6.3f", 1000.)
dumplist ( ofile, L3, "# L3 :", " %6.2f", 1000.)
if (tpel == 0.) :
  ofile.write("# evaluated without any beamsplitter\n")
else :
  ofile.write("# beamsplitter at angle apel = %.2f\n" % apel)
  ofile.write("# beamsplitter thickness tpel = %.4f\n" % tpel)
  ofile.write("# beamsplitter angIncidence aIpel = %.2f\n" % aIpel)
ofile.write("# leakages evaluated after rotation by aeval= %.2f\n" % aeval)

# --- dump leakages at each freq, for all polarizers; also compute avg and rms --- #
m = 0
for fGHz in freq :
  ofile.write("%6.1f" % fGHz)
  if (efficfile) : ofile3.write("%6.1f" % fGHz)
  sum = 0.
  for n in range(ntrials) :
    ofile.write(" %6.4f" % leakage[m,n])
    sum = sum + leakage[m,n]
    if (efficfile) : ofile3.write(" %6.4f" % effic[m,n] )
  avg = sum/ntrials
  var = 0.
  for n in range(ntrials) :
    dif = leakage[m,n] - avg
    var = var + dif*dif
  if (ntrials > 1) :
    var = var/(ntrials-1)
  ofile.write("   %6.4f %6.4f\n" % (avg,sqrt(var)))
  if (efficfile) : ofile3.write("\n")
  m = m + 1
ofile.close()
if (efficfile) : ofile3.close()

# ------------------------------------------------------------------------------------------------------ #
```

```
# leakage of idealized polarizers
# ------------------------------------------------------------------------------------------------------------ #
def fig9() :
  r = .0235
  f = 0.006
  L90 = length( r=r, f=f, dphi0=90., fGHz=230. )
  L180 = 2. * L90

  a1 = 45.
  dumpDimensions( 'dim9A.dat', r, a1, f, L90, 0., 0., 0., 0., 0., 0.)
  computeLeakage( 'dim9A.dat', 'leak9A.dat' )
     # simple 1-section polarizer

  a1 = 15.
  a2 = 60.
  dumpDimensions( 'dim9B.dat', r, a1, f, L180, a2, f, L90, 0., 0., 0.)
  computeLeakage( 'dim9B.dat', 'leak9B.dat' )
     # 2-section polarizer, maximally flat, from Kovac

  a1 = 15.
  a2 = 59.5
  dumpDimensions( 'dim9C.dat', r, a1, f, L180, a2, f, L90, 0., 0., 0.)
  computeLeakage( 'dim9C.dat', 'leak9C.dat' )
    # 2-section polarizer, CARMA design

  a1 = 6.05
  a2 = 28.63
  a3 = 67.59
  dumpDimensions( 'dim9D.dat', r, a1, f, L180, a2, f, L180, a3, f, L90 )
  computeLeakage( 'dim9D.dat', 'leak9D.dat' )
    # 3-section polarizer, maximally flat, from Kovac

  a1 = 6.50
  a2 = 28.07
  a3 = 66.57
  dumpDimensions( 'dim9E.dat', r, a1, f, L180, a2, f, L180, a3, f, L90 )
  computeLeakage( 'dim9E.dat', 'leak9E.dat' )
    # wideband 3-section polarizer derived by Pancharatnam

def fig10() :
  r = .0235
  f = 0.006
  L90 = length( r=r, f=f, dphi0=90., fGHz=230. )
  L180 = 2. * L90
  a1 = 15.
  a2 = 60.
    # nominal dimensions of 2-section polarizer, maximally flat, from Kovac

  dumpDimensions( 'dim10a.dat', r, a1, f-.0002, L180, a2, f-.0002, L90, 0., 0., 0. )
  dumpDimensions( 'dim10a.dat', r, a1, f,       L180, a2, f,       L90, 0., 0., 0. )
  dumpDimensions( 'dim10a.dat', r, a1, f+.0002, L180, a2, f+.0002, L90, 0., 0., 0. )
  computeLeakage( 'dim10a.dat', 'leak10a.dat' )
    # change facet depths for both sections

  dumpDimensions( 'dim10b.dat', r, a1, f, L180, a2-1., f, L90, 0., 0., 0.)
  dumpDimensions( 'dim10b.dat', r, a1, f, L180, a2,    f, L90, 0., 0., 0.)
  dumpDimensions( 'dim10b.dat', r, a1, f, L180, a2+1., f, L90, 0., 0., 0.)
  computeLeakage( 'dim10b.dat', 'leak10b.dat' )
```

```
    # change angle between section 1 and section 2

# ------------------------------------------------------------------------------------------------------------ #
# return dimension = (xtarg +/- random error)
# error is gaussian distributed with sigma=xtol, but cannot exceed xtol
# ------------------------------------------------------------------------------------------------------------ #
def dimension( xtarg, xtol ) :
  if (xtarg == 0.) : return 0.
  x = xtarg + 2.*xtol                  # enter loop with dummy value guaranteed to be unacceptable
  while (abs(x - xtarg) > xtol) :
    x = random.gauss( xtarg, xtol )
  return x

# ------------------------------------------------------------------------------------------------------------ #
# writes file of polarizer dimensions with truncated Gaussian deviations (one line per polarizer)
# ------------------------------------------------------------------------------------------------------------ #
def gaussdim( ntrials=200, dimfile='dims.txt', rtarg=0.0235, rtol=0.0001, ftarg=0.003, ftol=0.0001,
              a1targ=15.0, a1tol=0.2, L1targ=0.4718, Ltol=0.001, a2targ=59.5, atol=0.1, L2targ=0.2359,
              a3targ=0., L3targ=0. ) :
  ofile = open(dimfile, "w")

  # --- dump input parameters for future reference --- #
  ofile.write("# rtarg = %.6f, rtol = %.6f\n" % (rtarg,rtol) )
  ofile.write("# ftarg = %.6f, ftol = %.6f\n" % (ftarg,ftol) )
  ofile.write("# a1targ = %.2f, a1tol = %.2f, L1targ= %.6f, L1tol = %.6f \n" % (a1targ,a1tol,L1targ,Ltol) )
  ofile.write("# a2targ = %.2f, a2tol = %.2f, L2targ= %.6f, L2tol = %.6f \n" % (a2targ,atol,L2targ,Ltol) )
  ofile.write("# a3targ = %.2f, a3tol = %.2f, L3targ= %.6f, L3tol = %.6f \n" % (a3targ,atol,L3targ,Ltol) )

  # --- write one line per polarizer --- #
  for n in range(ntrials) :
    ofile.write(" %6.3f" % (1000.*dimension(rtarg,rtol)) )    # circ waveguide radius in mils
    ofile.write(" %7.2f" % dimension(a1targ,a1tol) )   # angle of section 1 relative to input ref axis
    ofile.write(" %5.3f" % (1000.*dimension(ftarg,ftol)) )   # facet depth of section 1, mils
    ofile.write(" %6.2f" % (1000.*dimension(L1targ,Ltol)) )   # length of section 1, mils
    ofile.write(" %7.2f" % dimension(a2targ,atol) )           # angle of section 2 relative to input section 1
    ofile.write(" %5.3f" % (1000.*dimension(ftarg,ftol)) )    # facet depth of section 2, mils
    ofile.write(" %6.2f" % (1000.*dimension(L2targ,Ltol)) )   # length of section 2, mils
    ofile.write(" %7.2f" % dimension(a3targ,atol) )           # angle of section 3 relative to input section 2
    ofile.write(" %5.3f" % (1000.*dimension(ftarg,ftol)) )    # facet depth of section 3, mils
    ofile.write(" %6.2f" % (1000.*dimension(L3targ,Ltol)) )   # length of section 3, mils
    ofile.write("\n")
  ofile.close()

# ------------------------------------------------------------------------------------------------------------ #
# leakages of polarizers with fabrication tolerances
# ------------------------------------------------------------------------------------------------------------ #
def fig11() :
  r = .0235
  f = 0.003
  L90 = length( r=r, f=f, dphi0=90., fGHz=230. )
  L180 = 2. * L90
  a1 = 15.
  a2 = 59.5
  gaussdim( ntrials=200, dimfile='dim11a.dat', rtarg=r, rtol=0.00010, ftarg=f, ftol=0.00010,
      a1targ=a1, a1tol=0.2, L1targ=L180, Ltol=0.001, a2targ=a2, atol=0.2, L2targ=L90 )
  gaussdim( ntrials=200, dimfile='dim11b.dat', rtarg=r, rtol=0.00015, ftarg=f, ftol=0.00015,
      a1targ=a1, a1tol=0.2, L1targ=L180, Ltol=0.001, a2targ=a2, atol=0.2, L2targ=L90 )
  gaussdim( ntrials=200, dimfile='dim11c.dat', rtarg=r, rtol=0.00020, ftarg=f, ftol=0.00020,
```

```
      a1targ=a1, a1tol=0.2, L1targ=L180, Ltol=0.001, a2targ=a2, atol=0.2, L2targ=L90 )
  computeLeakage( 'dim11a.dat', 'leak11a.dat' )
  computeLeakage( 'dim11b.dat', 'leak11b.dat' )
  computeLeakage( 'dim11c.dat', 'leak11c.dat' )

  f = 0.006
  L90 = length( r=r, f=f, dphi0=90., fGHz=230. )
  L180 = 2. * L90
  gaussdim( ntrials=200, dimfile='dim11d.dat', rtarg=r, rtol=0.00010, ftarg=f, ftol=0.00010,
      a1targ=a1, a1tol=0.2, L1targ=L180, Ltol=0.001, a2targ=a2, atol=0.2, L2targ=L90 )
  gaussdim( ntrials=200, dimfile='dim11e.dat', rtarg=r, rtol=0.00015, ftarg=f, ftol=0.00015,
      a1targ=a1, a1tol=0.2, L1targ=L180, Ltol=0.001, a2targ=a2, atol=0.2, L2targ=L90 )
  gaussdim( ntrials=200, dimfile='dim11f.dat', rtarg=r, rtol=0.00020, ftarg=f, ftol=0.00020,
      a1targ=a1, a1tol=0.2, L1targ=L180, Ltol=0.001, a2targ=a2, atol=0.2, L2targ=L90 )
  computeLeakage( 'dim11d.dat', 'leak11d.dat' )
  computeLeakage( 'dim11e.dat', 'leak11e.dat' )   # also used for fig 12a
  computeLeakage( 'dim11f.dat', 'leak11f.dat' )

def fig12b() :
  r = .0235
  f = 0.006
  L90 = length( r=r, f=f, dphi0=90., fGHz=230. )
  L180 = 2. * L90
  gaussdim( ntrials=200, dimfile='dim12b.dat', rtarg=r, rtol=0.00015, ftarg=f, ftol=0.00015,
      a1targ=6.50, a1tol=0.2, L1targ=L180, Ltol=0.001, a2targ=28.07, atol=0.2, L2targ=L180,
      a3targ=66.57, L3targ=L90)
  computeLeakage( 'dim12b.dat', 'leak12b.dat' )

# ------------------------------------------------------------------------------------------------------ #
# Fresnel formulae, transmission and reflection amplitude coeff, 1.5.2, eqn 20,21 (p. 40); for 1 surface
# ------------------------------------------------------------------------------------------------------ #
def fresnel( n1, thetaI, n2, thetaT ) :
  tpar = 2. * n1 * cos(thetaI) / (n2 * cos(thetaI) + n1 * cos(thetaT))
  tperp = 2. * n1 * cos(thetaI) / (n1 * cos(thetaI) + n2 * cos(thetaT))
  rpar = (n2 * cos(thetaI) - n1 * cos(thetaT)) / (n2 * cos(thetaI) + n1 * cos(thetaT))
  rperp = (n1 * cos(thetaI) - n2 * cos(thetaT)) / (n1 * cos(thetaI) + n2 * cos(thetaT))
  return [tpar, tperp, rpar, rperp]

# ------------------------------------------------------------------------------------------------------ #
# compute transmission through beamsplitter ("pellicle") of thickness tpel (inches)
# angI is the angle of incidence (degrees), nn is the refractive index of the beamsplitter material
# default nn=1.83 is appropriate for PETP = Mylar (Lamb 1996, Int. J. IR MM Waves, 17, pp. 1997-2034)
# returns  [tpar,tperp,Rpar,Rperp]
# tpar and tperp are the AMPLITUDE TRANSMISSION coefficients (complex numbers)
# Rpar and Rperp are the POWER REFLECTION coefficients (real numbers)
# equations referenced to Born & Wolf 1970, Principles of Optics, Fourth Ed. (Oxford: Pergamon Press).
# ------------------------------------------------------------------------------------------------------ #
def pellicle( tpel=.001, angI=45, nn=1.83, fGHz=230. ) :
  if (tpel == 0.00) : return [1.,1.,0.,0.]
  thetaI = math.pi * angI / 180.                        # convert to radians
  thetaT = math.asin((sin(thetaI))/nn)                        # Snell's law, eqn 8 (p. 38)
  [tpar,tperp,rpar,rperp] = fresnel( 1., thetaI, nn, thetaT )                       # entering dielectric
  [tprimepar,tprimeperp,rprimepar,rprimeperp] = fresnel( nn, thetaT, 1., thetaI )    # leaving dielectric
  delta = 4. * math.pi * 2.54 * tpel * nn * cos(thetaT) / (clight/fGHz)
    # phase shift of signal propagating once through the pellicle; eqn (1), p. 324
  result = []
  result.append(tpar*tprimepar/(1.-rpar*rpar*cmath.exp(1.j * delta)))
  result.append(tperp*tprimeperp/(1.-rperp*rperp*cmath.exp(1.j * delta)))
```

```
    # amplitude transmission coefficient, eqn (11), p. 325
  Fpar = 4.*rpar*rpar/pow((1. - rpar*rpar),2.)
  Fperp = 4.*rperp*rperp/pow((1.-rperp*rperp),2.)
  sinsq = pow( sin(delta/2.), 2. )
  result.append( Fpar * sinsq/(1. + Fpar * sinsq) )
  result.append( Fperp * sinsq/(1. + Fperp * sinsq) )
    # power reflection coefficients, eqn (15,16) p. 327
  return result

# --- beamsplitter reflectivity --- #
def fig18( outfile='pelR.dat' ) :
  ofile = open( outfile, "w" )
  ofile.write("# tpel  Rpar35  Rperp35  Rpar45  Rperp45\n")
  for tpel in arange(0.00002,.00202,.00002) :
    [tpar,tperp,Rpar,Rperp] = pellicle( tpel=tpel, angI=35, fGHz=230.)
    ofile.write("%7.5f %10.4f %10.4f" % (tpel, 10.*math.log10(Rpar), 10.*math.log10(Rperp)))
    [tpar,tperp,Rpar,Rperp] = pellicle( tpel=tpel, angI=45, fGHz=230.)
    ofile.write("  %10.4f %10.4f\n" % (10.*math.log10(Rpar), 10.*math.log10(Rperp)))

# --- leakages including degradation by beamsplitter --- #
def fig19() :
  r = .0235
  f = 0.006
  L90 = length( r=r, f=f, dphi0=90., fGHz=230. )
  L180 = 2. * L90
  a1 = 15.
  a2 = 59.5
  dumpDimensions( 'nominal.dat', r, a1, f, L180, a2, f, L90, 0., 0., 0. )
  computeLeakage( 'nominal.dat', 'bs-none.dat', apel=0., tpel=0.000, aIpel=45. )
  computeLeakage( 'nominal.dat', 'bs-0-45-1mil.dat', apel=0., tpel=0.001, aIpel=45. )
  computeLeakage( 'nominal.dat', 'bs-45-45-1mil.dat', apel=45., tpel=0.001, aIpel=45. )
  computeLeakage( 'nominal.dat', 'bs-90-45-1mil.dat', apel=90., tpel=0.001, aIpel=45. )
  computeLeakage( 'dim11e.dat', 'leak11e-0-45-1mil.dat', apel=0., tpel=0.001, aIpel=45. )
  computeLeakage( 'dim11e.dat', 'leak11e-45-45-1mil.dat', apel=45., tpel=0.001, aIpel=45. )
  computeLeakage( 'dim11e.dat', 'leak11e-90-45-1mil.dat', apel=90., tpel=0.001, aIpel=45. )

# ------------------------------------------------------------------------------------------------------------ #
# compute leakage from HFSS simulation results
# normally the HFSS calculation launches an X-polarized signal, returns S21(X,Y)
# each infile line: f_GHz, S21(X:X)amp_dB, S21(X:X)phs_degr, S21(X:Y)amp_dB, S21(X:Y)phs_degr
# ------------------------------------------------------------------------------------------------------------ #
def HFSSleak( infile, leakfile ) :
  infile = open( infile, "r" )
  ofile = open( leakfile, "w")
  for line in infile:
    if line.startswith("#"):
      ofile.write( line )
    else:
      a = line.split()                              # split line into string tokens
      fGHz =  float(a[0])                         # frequency
      ampX = math.sqrt(pow(10.,float(a[1])/10.))      # convert amp to VOLTAGE
      phsX = math.pi * float(a[2]) / 180.       # convert phs to RADIANS
      ampY = math.sqrt(pow(10.,float(a[3])/10.))
      phsY = math.pi * float(a[4]) / 180.
      vec = array( [ (ampX*math.cos(phsX) + 1j*ampX*math.sin(phsX)),
                     (ampY*math.cos(phsY) + 1j*ampY*math.sin(phsY)) ] )
      print vec, dot(vec,R), dot(vec,L)
      Rleak = abs(dot(vec, R))  # amplitude of single complex number
```

```
      Lleak = abs(dot(vec, L))
      ofile.write("%8.3f %8.3f %8.3f %8.3f %8.3f %10.5f %10.5f\n" %
        (fGHz, ampX, float(a[2]), ampY, float(a[4]), Rleak, Lleak))
  ofile.close()

# ------------------------------------------------------------------------------------------------------------ #
# analytic simulation of full polarizer including curved transitions
# ------------------------------------------------------------------------------------------------------------ #
def fig16( outfile='fullsim1.dat' ) :
  ofile = open( outfile, "w")
  for freq in arange(200.,271.,1.) :
    v1 = Y
    v2 = Jrot( v1, -15. )
    [dx, ph] = dphitaper2 ( r=0.0235, f=0.006, Rc=0.125, fGHz=freq, nsteps=1000 )
    phshift = 2.*ph + dphi( r=0.0235, f=0.006, L=0.1058, fGHz=freq )
    v3 = Jdelay( v2, phshift )
    v2 = Jrot( v3, -59.5 )
    phshift = 2.*ph + dphi( r=0.0235, f=0.006, L=0.0302, fGHz=freq )
    v3 = Jdelay( v2, phshift )
    v2 = Jrot( v3, 74.5 )
    vout = v2
    Rleak = abs(dot(vout, R))  # amplitude of single complex number
    Lleak = abs(dot(vout, L))
    ofile.write("%6.1f %8.5f %8.5f\n" % (freq, Rleak, Lleak ) )
  ofile.close()
```