LittlePMC

March 11, 2016

1 Design of small Mode Cleaner for table top use (Caltech West Bridge labs)

2 Design Requirements

1. RF Noise Filtering
2. HOM Filtering
3. Transmission
4. Laser Wavelength: 1064 nm for now
5. Vibration Sensitivity
6. Scattering inside

2.0.1 Questions

1. How do these things vary with triangle aspect ratio?
2. Why do we want a triangle instead of a rectangle or zig-zag?
3. Can we use CVI mirrors or do we get a custom coating run? Custom coatings.

3 Definition of constants and mechanical parameters

```
In [14]: %matplotlib inline
         from numpy import *
         from matplotlib.pyplot import *
         import scipy.signal as sig
         import scipy.constants as const
         #from scipy.optimize import leastsq
         #from scipy.optimize import minimize
         from __future__ import division
         from IPython.display import display, Image, display_jpeg
         #clist = ['#0000dd', '#dd0000', '#00aa00', '#770077', '#774400']

         # Update the matplotlib configuration parameters:
         plt.rcParams.update({'font.size': 20, 'font.family': 'serif',
                              'figure.figsize': (10, 8),
                              'axes.grid': True,
                              'grid.color': '#888888'})
```

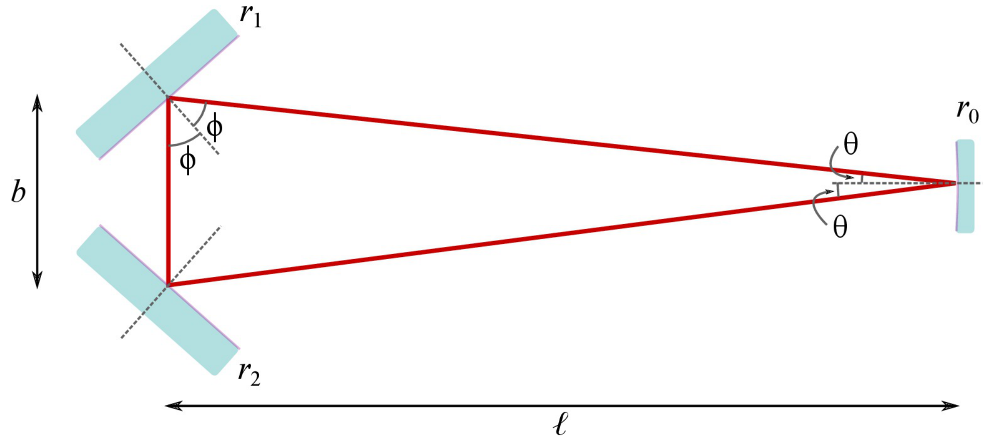
We define some constants, along with some basic optical and mechanical parameters for the system. The choices for some of these parameters are explained below.

```
In [15]: c = const.c  # m/s; speed of light
         lam = 1064e-9 # m; wavelength
```

```
fPDH = 30e6 # Hz; PDH modulation frequency
L = 0.240 # m; cavity round-trip length
b = 1.0 * 0.0254 # m; base of triangle
R = 1 # m; ROC for mirror 0
r0 = np.sqrt(1-200e-6) # mirror 0 (back mirror)
r1 = np.sqrt(0.99) # mirror 1 (input mirror)
r2 = r1 # mirror 2 (output mirror)
```

In [38]:
```
pmcdiag = Image('Figures/pmcdiag.png')
pmcdiag
```

Out[38]:



4 Optical design

Having set the above parameters, we can compute some other important quantities for the cavity.

In [17]:
```
l = sqrt((L - b)**2 / 4 - (b/2)**2) # m; height of triangle
theta = arctan(b / 2 / l) # rad; angle of incidence on the back optic
phi = arctan(l / (b / 2)) / 2 # rad; angle of incidence on front optics
fsr = c / L # Hz; free spectral range for a ring cavity
r = r0 * r1 * r2
t = sqrt(1 - r**2)
finesse = pi * np.sqrt(r) / (1 - r)
hwhm = fsr / (2 * finesse) # Hz; hwhm frequency (i.e., cavity pole)
g = (1 - L / R) # cavity g factor
tms = fsr * np.arccos(g) / (2 * pi) # Hz; transverse mode spacing for a ring cavity
w0 = (lam / (2 * pi))**0.5 * (L * (2 * R - L))**0.25 # m; waist size
w = (lam * R / pi)**0.5 / (2 * R / L - 1)**0.25
zR = pi * w0**2 / lam
print('Distance between front mirrors: {0:.3f} cm = {1:.3f} inches'.format(b * 100, b / 0.0254)
print('Transverse distance from front mirrors to back mirror: {0:.3f} cm = {1:.3f} inches'.for
print('Angle of incidence on back mirror: {0:.2f} degrees'.format(theta * 180 / pi))
```

```
print('Angle of incidence on front mirrors: {0:.2f} degrees'.format(phi * 180 / pi))
print('FSR: {0:.0f} MHz'.format(fsr / 1e6))
print('HWHM (cavity pole) frequency: {0:.1f} MHz'.format(hwhm / 1e6))
print('Finesse: {0:.0f}'.format(finesse))
print('g-factor: {0:.2f}'.format(g))
print('Transverse mode spacing: {0:.0f} MHz'.format(tms / 1e6))
print('Waist size: {0:.0f} um'.format(w0 * 1e6))
print('Spot size on back mirror: {0:.0f} um'.format(w * 1e6))
print('Rayleigh range: {0:.0f} cm'.format(zR * 100))
print('Quotient FSR / TMS = {0:.2f}'.format(fsr / tms))
```

```
Distance between front mirrors: 2.540 cm = 1.000 inches
Transverse distance from front mirrors to back mirror: 10.655 cm = 4.195 inches
Angle of incidence on back mirror: 6.80 degrees
Angle of incidence on front mirrors: 41.60 degrees
FSR: 1249 MHz
HWHM (cavity pole) frequency: 2.0 MHz
Finesse: 310
g-factor: 0.76
Transverse mode spacing: 141 MHz
Waist size: 332 um
Spot size on back mirror: 354 um
Rayleigh range: 32 cm
Quotient FSR / TMS = 8.88
```

```
In [18]: s1 = 0.25 * 0.0254 # m; thickness of substrate of mirror 1
         n1 = 1.45 # index of refraction of silica substrate
```

4.1 Choice of round-trip length

We search for an L that is on the order of a few tens of centimeters. This is long enough that we can stick with a triangle geometry in which the curved optic is at nearly normal incidence. It is also short enough that our spacer's longitudinal mechanical resonance happens above 10 kHz.

4.1.1 Basic formulas: FSR, TMS, and g factor

Since this is a ring cavity, the resonance condition for the transverse mode TEM_{mn} is

$$k_{mn}L - 2(m+n+1)\arctan\left(\frac{L}{2z_\mathrm{R}}\right) = 2\pi(q+1) + \beta\pi,$$

where q is the axial mode number, and $\beta = 0$ if the mode is spatially even and 1 if the mode is spatially odd (cf. Kogelnik and Li eq. 49 for the usual two-mirror FP cavity).

With $k_{mn} = 2\pi f_{mn}/c$ and $f_\mathrm{FSR} = c/L$, we have

$$\frac{f_{mn}}{f_\mathrm{FSR}} = (q+1) + \frac{m+n+1}{\pi}\arctan\left(\frac{L}{2z_\mathrm{R}}\right) = (q+1) + \frac{m+n+1}{2\pi}\arccos g + \frac{\beta}{2},$$

with $g = 1 - L/R$ (cf. K&L eq. 56) and $2\arctan(L/2z_\mathrm{R}) = \arccos g$. It is convenient to define the transverse mode spacing

$$f_\mathrm{TMS} = \frac{f_\mathrm{FSR}}{2\pi}\arccos g.$$

For two modes of the same transverse order, we define the detuning $\Delta f_{mn} = f_{mn} - f_{00}$.

A full analysis would also consider the additional splitting due to polarization: one would introduce a term $\gamma\pi$ into the phase equation above, with $\gamma = 0$ for p-polarization and $\gamma = 1$ for s-polarization.

4.1.2 Horizontal/vertical mode splitting

For our ring cavity, the beam hits the back (curved) optic at an angle $\theta = \pi/2 - \phi$. Using the result of Massey and Siegman, we find that the effective horizontal and vertical radii of curvature are

$$R^{(\mathrm{H})} = R/\cos\theta \qquad \text{and} \qquad R^{(\mathrm{V})} = R\cos\theta.$$

(In the notation of Massey and Siegman, H is the transverse coordinate and V is the sagittal coordinate.) From these radii we define the horizontal and vertical g factors $g^{(\mathrm{H,V})}$ and TMSs $f_{\mathrm{TMS}}^{(\mathrm{H,V})}$. For a given mode order p, we say that the maximal spread in frequency due to this splitting is

$$f_{0p} - f_{p0} = f_{\mathrm{FSR}} \frac{p}{2\pi} \left[\arccos g^{(\mathrm{V})} - \arccos g^{(\mathrm{H})} \right].$$

4.1.3 Numerical analysis of HOM spacing as a function of L

We let L vary, thereby letting the horizontal and vertical TMSs vary as well.

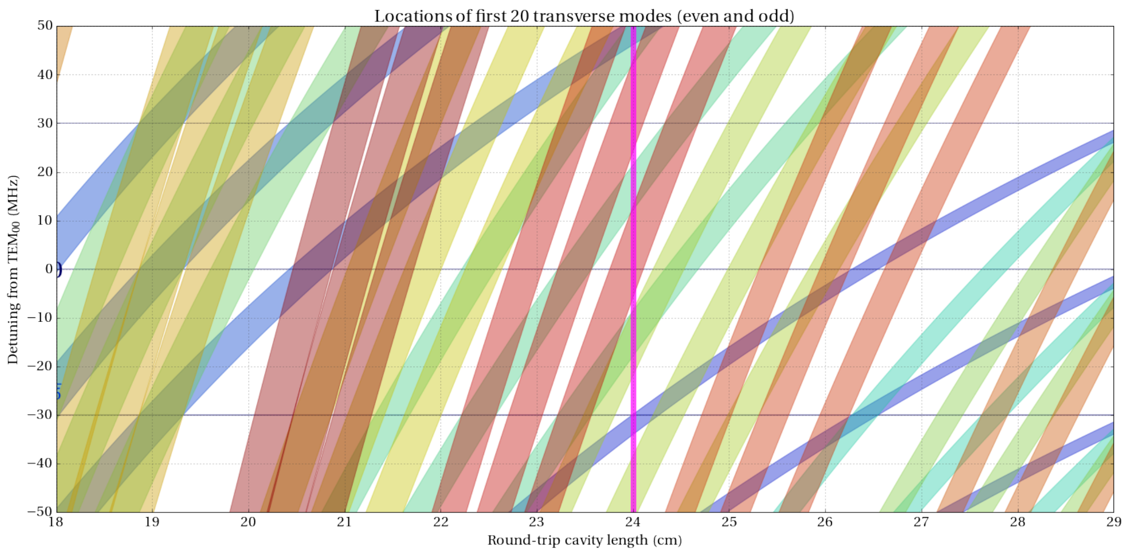
```
In [19]: # Define a range of possible round-trip lengths
         LArr = arange(18e-2, 30e-2, 0.1e-3)
         lArr = sqrt((LArr - b)**2 / 4 - (b/2)**2) # m; height of triangle
         thetaArr = arctan(b / 2 / lArr)
         # Now compute the horizontal and vertical TMSs
         RHArr = R / cos(thetaArr)
         RVArr = R * cos(thetaArr)
         gHArr = 1 - LArr / RHArr
         gVArr = 1 - LArr / RVArr
         fsrArr = c / LArr
         tmsHArr = fsrArr * arccos(gHArr) / (2 * pi)
         tmsVArr = fsrArr * arccos(gVArr) / (2 * pi)
```

Now we compute the detunings of the higher-order modes (modulo the cavity FSR) as a function of L.

```
In [20]: modeOrders = arange(20)
         modeMat = tile(modeOrders, (len(LArr), 1))
         fsrMat = tile(reshape(fsrArr, (len(fsrArr), 1)), (1, len(modeOrders)))
         tmsHMat = tile(reshape(tmsHArr, (len(tmsHArr), 1)), (1, len(modeOrders)))
         tmsVMat = tile(reshape(tmsVArr, (len(tmsVArr), 1)), (1, len(modeOrders)))
         LMat = tile(reshape(LArr, (len(LArr), 1)), (1, len(LArr)))
         homHMatEven = modeMat * tmsHMat
         homHMatOdd = modeMat[:,1:] * tmsHMat[:,1:] + fsrMat[:,1:]/2
         homHMatEven = mod(homHMatEven, fsrMat)
         homHMatOdd = mod(homHMatOdd, fsrMat[:,1:])
         homVMatEven = modeMat * tmsVMat
         homVMatOdd = modeMat[:,1:] * tmsVMat[:,1:] + fsrMat[:,1:]/2
         homVMatEven = mod(homVMatEven, fsrMat)
         homVMatOdd = mod(homVMatOdd, fsrMat[:,1:])
         # arrange the HOMs to be in the interval [-fsr/2, fsr/2) instead of [0, fsr)
         homHMatEven = homHMatEven - fsrMat * (homHMatEven >= fsrMat / 2)
         homHMatOdd = homHMatOdd - fsrMat[:,1:] * (homHMatOdd >= fsrMat[:,1:] / 2)
         homVMatEven = homVMatEven - fsrMat * (homVMatEven >= fsrMat / 2)
         homVMatOdd = homVMatOdd - fsrMat[:,1:] * (homVMatOdd >= fsrMat[:,1:] / 2)
```

We now plot these detunings, along with the detunings of the sidebands. The bands indicate the frequency spread $f_{0p} - f_{p0}$ for each mode order p, as defined above. In the plot, bars over the number indicate a spatially odd mode, and the absence of bars a spatially even mode. The pink bar indicates the chosen value of L.

```
In [21]: h = figure(figsize=(20, 10))
         ylimlow = -150e6 # Hz
         ylimhigh = 150e6 # Hz
         for ind, thisMode in enumerate(modeOrders):
             thisColor = array(cm.jet(int(ind * cm.jet.N / len(modeOrders)))) * array([0.8, 0.8, 0.8, 1]
             thisMarker = '$'+str(thisMode)+'$'
             thisMaskedHomHEven = ma.masked_outside(homHMatEven[:,ind], ylimlow, ylimhigh)
             thisMaskedHomVEven = ma.masked_outside(homVMatEven[:,ind], ylimlow, ylimhigh)
             fill_between(LArr * 1e2, thisMaskedHomHEven / 1e6, thisMaskedHomVEven / 1e6, color=thisCol
             fill_between(LArr * 1e2, (thisMaskedHomHEven - fPDH)/ 1e6, (thisMaskedHomVEven - fPDH)/ 1e
             fill_between(LArr * 1e2, (thisMaskedHomHEven + fPDH)/ 1e6, (thisMaskedHomVEven + fPDH)/ 1e
             plot(LArr * 1e2, (thisMaskedHomHEven + thisMaskedHomVEven) / 2e6, ' ',
                  marker=thisMarker, ms=20, mew=0, markevery=ceil(80.0 / (0.5 * thisMode+1)), c=thisCol
             if ind < len(modeOrders) - 1:
                 thatMarker = '$\overline{'+str(thisMode+1)+'}$'
                 thisMaskedHomHOdd = ma.masked_outside(homHMatOdd[:,ind], ylimlow, ylimhigh)
                 thisMaskedHomVOdd = ma.masked_outside(homVMatOdd[:,ind], ylimlow, ylimhigh)
                 fill_between(LArr * 1e2, thisMaskedHomHOdd / 1e6, thisMaskedHomVOdd / 1e6, color=thisC
                 fill_between(LArr * 1e2, (thisMaskedHomHOdd - fPDH)/ 1e6, (thisMaskedHomVOdd - fPDH)/ 
                 fill_between(LArr * 1e2, (thisMaskedHomHOdd + fPDH)/ 1e6, (thisMaskedHomVOdd + fPDH)/ 
                 plot(LArr * 1e2, (thisMaskedHomHOdd + thisMaskedHomVOdd) / 2e6, ' ',
                      marker=thatMarker, ms=20, mew=0, markevery=ceil(80.0 / (0.5 * thisMode+1)), c=thi
         plot(array([L, L]) * 1e2, array([ylimlow, ylimhigh]), color='#ff00ff', lw=7, alpha=0.7, zorder
         xlabel('Round-trip cavity length (cm)')
         ylabel('Detuning from TEM$_{00}$ (MHz)')
         xticks(arange(-1 + ceil(LArr[0] * 1e2), floor(LArr[-1] * 1e2) + 1))
         yticks(arange(-50, 60, 10))
         ax = gca()
         ax.set_xticks(arange(LArr[0], LArr[-1], 0.1), minor=True)
         ylim(-50, 50)
         xlim(ceil(LArr[0] * 1e2), floor(LArr[-1] * 1e2))
         grid(color='#444444')
         #legend(loc=2, bbox_to_anchor=(1, 1), ncol=2)
         title('Locations of first ' + str(len(modeOrders)) + r' transverse modes (even and odd)')
         tight_layout()
         savefig('homVersusLength.pdf')
```



4.2 Choice of finesse and cavity pole

We aim for a cavity pole of a few megahertz. Given an FSR of roughly 1 GHz (set by the choice of L, described above), this sets the finesse at 300 or so. This finesse is easily achievable using two front mirrors with 99% power reflectivity, and a back mirror with a significantly higher reflectivity (99.9% or higher).

4.2.1 Basic formulas

Given mirrors with field reflectivities r_0, r_1, and r_2, the cavity finesse is

$$\mathcal{F} = \frac{\pi\sqrt{r_0 r_1 r_2}}{1 - r_0 r_1 r_2}$$

and the cavity pole is

$$f_{\mathrm{HWHM}} = f_{\mathrm{FSR}}/\mathcal{F}.$$

4.2.2 Optical transfer function

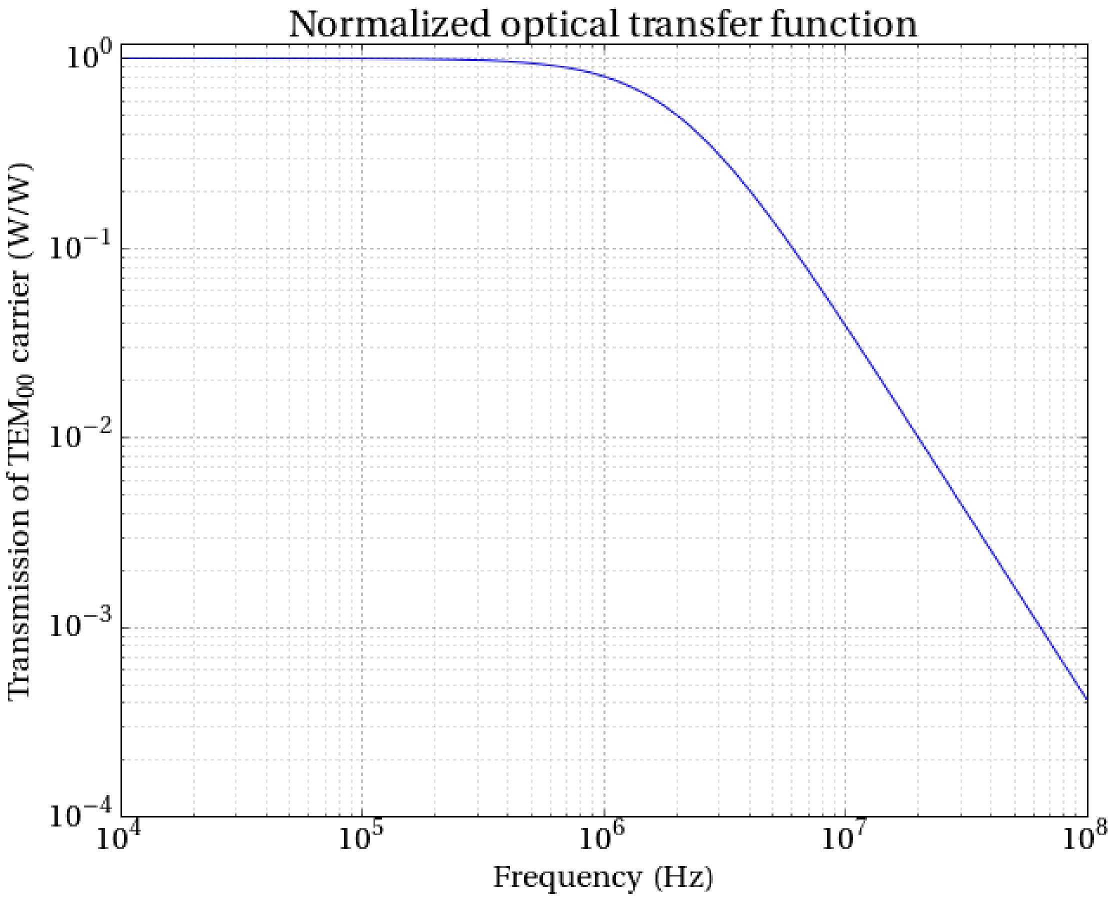
The (normalized) field transfer function $H(f)$ gives the transmission of audio- or radio-frequency field sidebands through the cavity:

$$H(f) = \frac{(1 - r)\exp[-2\pi \mathrm{i} f/f_{\mathrm{FSR}}]}{1 - r\exp[-2\pi \mathrm{i} f/f_{\mathrm{FSR}}]}$$

Here we plot the power transmission $|H(f)|^2$.

In [22]:
```
# We define the field transmission function, as given above
def FPTrans(f, f0, fsr, r):
    return (1 - r) * exp(-2j * pi * (f - f0) / fsr) / (1 - r * exp(-2j * pi * (f - f0) / fsr))
```

In [23]:
```
ff = logspace(4, 8, 200)
loglog(ff, abs(FPTrans(ff, 0, fsr, r))**2)
xlabel('Frequency (Hz)')
ylabel('Transmission of TEM$_{00}$ carrier (W/W)')
title('Normalized optical transfer function')
ylim(ylim()[0], ylim()[1] * 1.2)
grid('on', color='#888888', which='both')
grid(color='k', which='major')
```



4.2.3 HOM suppression

Having chosen a length L, we want to see quantitatively how suppressed the HOMs are in transmission.

Given a mode at frequency f_{mn}, the normalized field transmittance function of the ring cavity is as given above:

$$H(f_{mn} - f_{00}) = \frac{(1-r)\exp[-2\pi \mathrm{i}(f_{mn} - f_{00})/f_{\mathrm{FSR}}]}{1 - r\exp[-2\pi \mathrm{i}(f_{mn} - f_{00})/f_{\mathrm{FSR}}]}.$$

4.2.4 Numerical analysis of HOM suppression

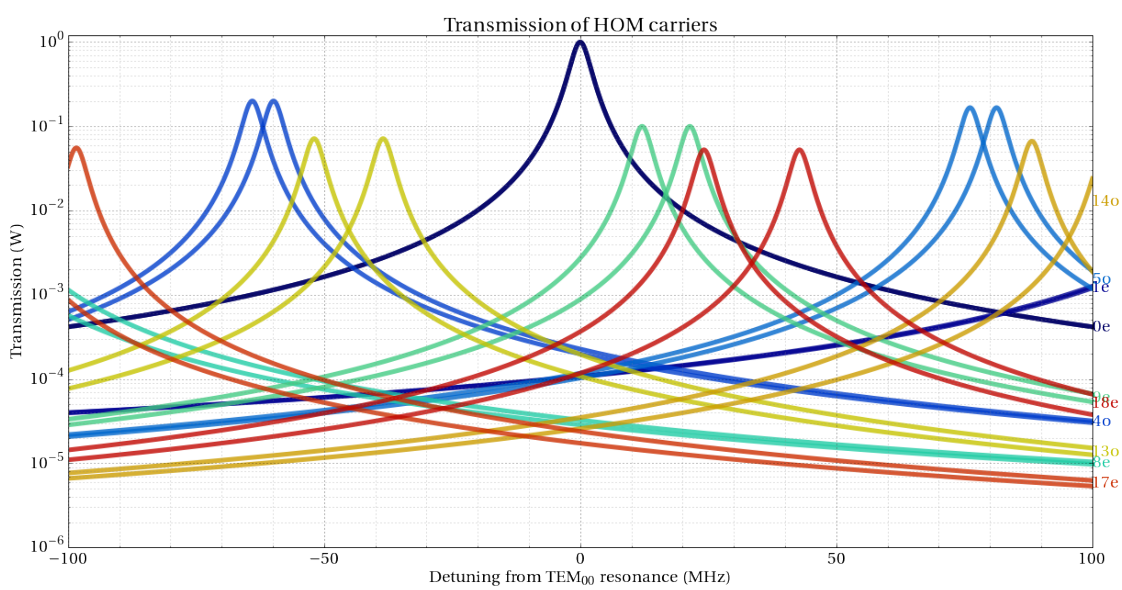
Now we numerically compute the normalized transmittance of the HOMs.

In [24]:
```
# Using the previous numerical work in computing the HOM detunings as a function of L,
# we extract the detuning frequencies for our particular value of L
LArg = argwhere(abs(LArr-L) < 0.01e-3)
homHDetsEven = squeeze(homHMatEven[LArg,:])
homHDetsOdd = squeeze(homHMatOdd[LArg,:])
homVDetsEven = squeeze(homVMatEven[LArg,:])
homVDetsOdd = squeeze(homVMatOdd[LArg,:])
```

Now we plot the transmittance of the HOMs. For ease of interpretation, we have normalized each HOM by its mode order.

```
In [25]: h = figure(figsize=(20,10))
         ff = arange(-100e6, 100e6, 0.01e6)
         for ind, homHOffsEven in enumerate(homHDetsEven):
             homVOffsEven = homVDetsEven[ind]
             thisColor = array(cm.jet(int(ind * cm.jet.N / len(modeOrders)))) * array([0.8, 0.8, 0.8, 1]
             thisHTrans = abs(FPTrans(ff, homHOffsEven, fsr, r))**2 / (1 + ind)
             thisVTrans = abs(FPTrans(ff, homVOffsEven, fsr, r))**2 / (1 + ind)
             if any(thisHTrans > 5e-4):
                 #fill_betweenx(thisHTrans, ff / 1e6, (ff + homHOffsEven) / 1e6, where = homHOffsEven +
                 plot(ff / 1e6, thisHTrans, c=thisColor, lw=5, alpha=0.8)
                 plot(ff / 1e6, thisVTrans, c=thisColor, lw=5, alpha=0.8)
                 text(ff[-1] / 1e6, (thisHTrans[-1] + thisVTrans[-1]) / 2, str(ind) + 'e', color=thisCol
             if ind > 0:
                 homHOffsOdd = homHDetsOdd[ind-1]
                 homVOffsOdd = homVDetsOdd[ind-1]
                 thatHTrans = 1 / (1 + ((ff - homHOffsOdd)/ hwhm)**2) / (1 + ind)
                 thatVTrans = 1 / (1 + ((ff - homVOffsOdd)/ hwhm)**2) / (1 + ind)
                 if any(thatHTrans > 5e-4):
                     plot(ff / 1e6, thatHTrans, c=thisColor, lw=5, alpha=0.8)
                     plot(ff / 1e6, thatVTrans, c=thisColor, lw=5, alpha=0.8)
                     text(ff[-1] / 1e6, (thatHTrans[-1] + thatVTrans[-1]) / 2, str(ind) + 'o', color=thi
         #plot(ff / 1e6, abs(FPTrans(ff, 0, fsr, r))**2)
         #xticks(arange(-100, 110, 10))
         semilogy()
         xlabel('Detuning from TEM$_{00}$ resonance (MHz)')
         ylabel('Transmission (W)')
         ylims = ylim()
         ylim(ylims[0], ylims[1] * 1.2)
         ax = gca()
         ax.set_xticks(arange(-100, 110, 10), minor=True)
         grid(color='#888888', which='both')
         grid(color='k', which='major')
         title('Transmission of HOM carriers')

Out[25]: <matplotlib.text.Text at 0x10e932bd0>
```



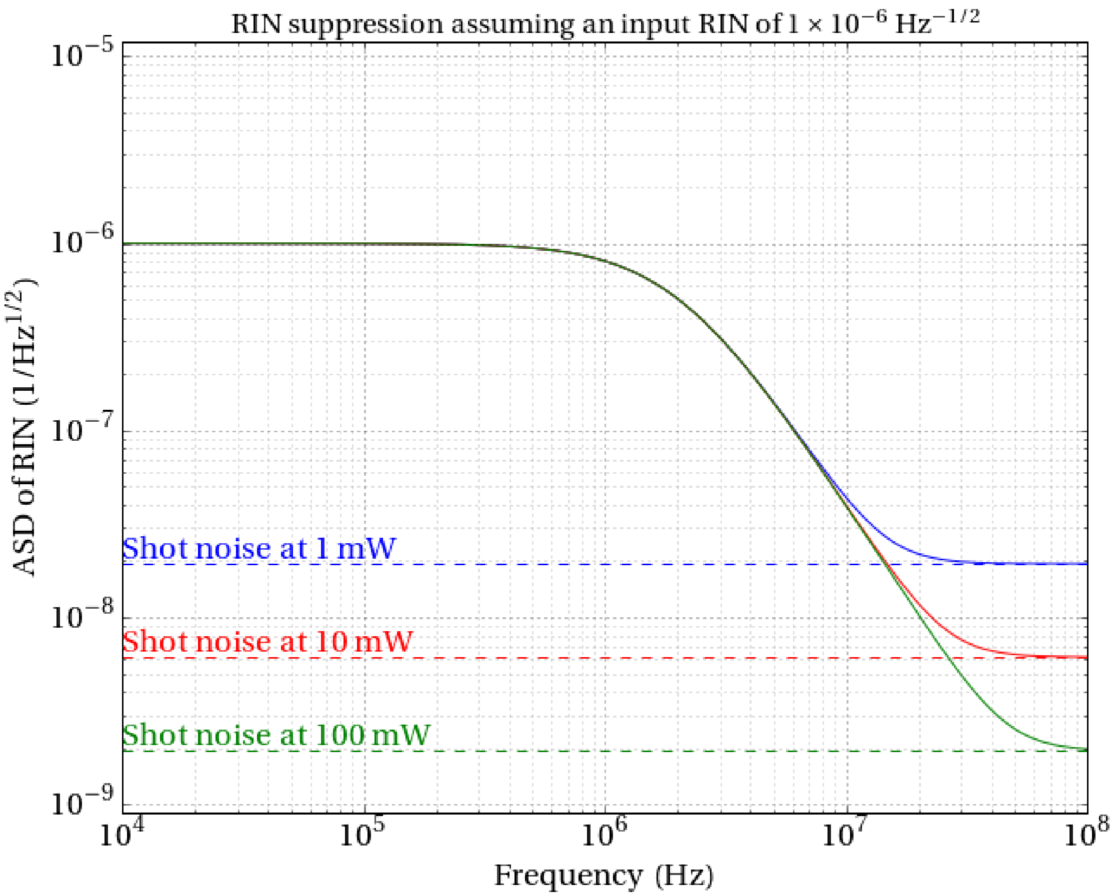
4.2.5 RIN/RAM suppression

Using the field transmittance function, we can get an idea of how well the cavity can suppress laser RIN. We assume a flat input RIN spectrum $\sqrt{S_{\mathrm{RIN}}(f)} = A_0$.

For an input power of 10 mW with a RIN of $A_0 = 1 \times 10^{-6}\,\mathrm{Hz}^{-1/2}$, we achieve shot-noise limited light at about 30 MHz.

```
In [26]: ff = logspace(4, 8, 200)
         A0 = 1e-6 # 1/rtHz
         P0arr = array([1e-3, 10e-3, 100e-3]) # W
         loglog(ff, A0 * ones(shape(ff)), label='Input RIN')
         clf()
         const.Planck * c / lam
         clist = ['b', 'r', 'g', 'm']
         for ind, P0 in enumerate(P0arr):
             ShotNoiseRIN = sqrt(2 * const.Planck * c / lam / P0) # 1/rtHz
             thisRIN = sqrt(A0**2 * abs(FPTrans(ff, 0, fsr, r))**4 + ShotNoiseRIN**2)
             loglog(ff, thisRIN, clist[ind])
             loglog(ff, ShotNoiseRIN * ones(shape(ff)), clist[ind] + '--')
             text(ff[0], 1.1 * ShotNoiseRIN, 'Shot noise at {0:.0f} mW'.format(P0 / 1e-3),
                  color=clist[ind])
         grid('on', which='both')
         grid(color='k', which='major')
         xlabel('Frequency (Hz)')
         ylabel('ASD of RIN (1/Hz$^{1/2}$)')
         title(r'RIN suppression assuming an input RIN of $1\times 10^{-6}$ Hz$^{-1/2}$', fontsize=18)
         ylims = ylim()
         ylim(0.9 * ylims[0], 1.2 * ylims[1])

Out[26]: (9.000000000000001e-10, 1.2e-05)
```



4.3 Scattering

The total integrated scatter is approximately

$$\frac{P_{\text{scat}}}{P_0} = \left(\frac{4\pi\sigma}{\lambda}\right)^2,$$

where σ is the rms surface roughness.

In [27]:
```
sigma = 3e-10 # m; rms surface roughness
scatfrac = (4 * pi * sigma / lam)**2 # fraction of power scattered, according to above formula
print('Scattering loss per mirror: {0:.2g} ppm'.format(scatfrac / 1e-6))
Pinc = 100e-3 # W
print('Power scattered assuming {0:.0f} mW incident: {1:.2g} mW'.format(Pinc * 1e3, Pinc * 1e3
```

```
Scattering loss per mirror: 13 ppm
Power scattered assuming 100 mW incident: 1.2 mW
```

4.4 Choice of polarization

We have chosen p polarization, with little justification so far.

5 Tolerances

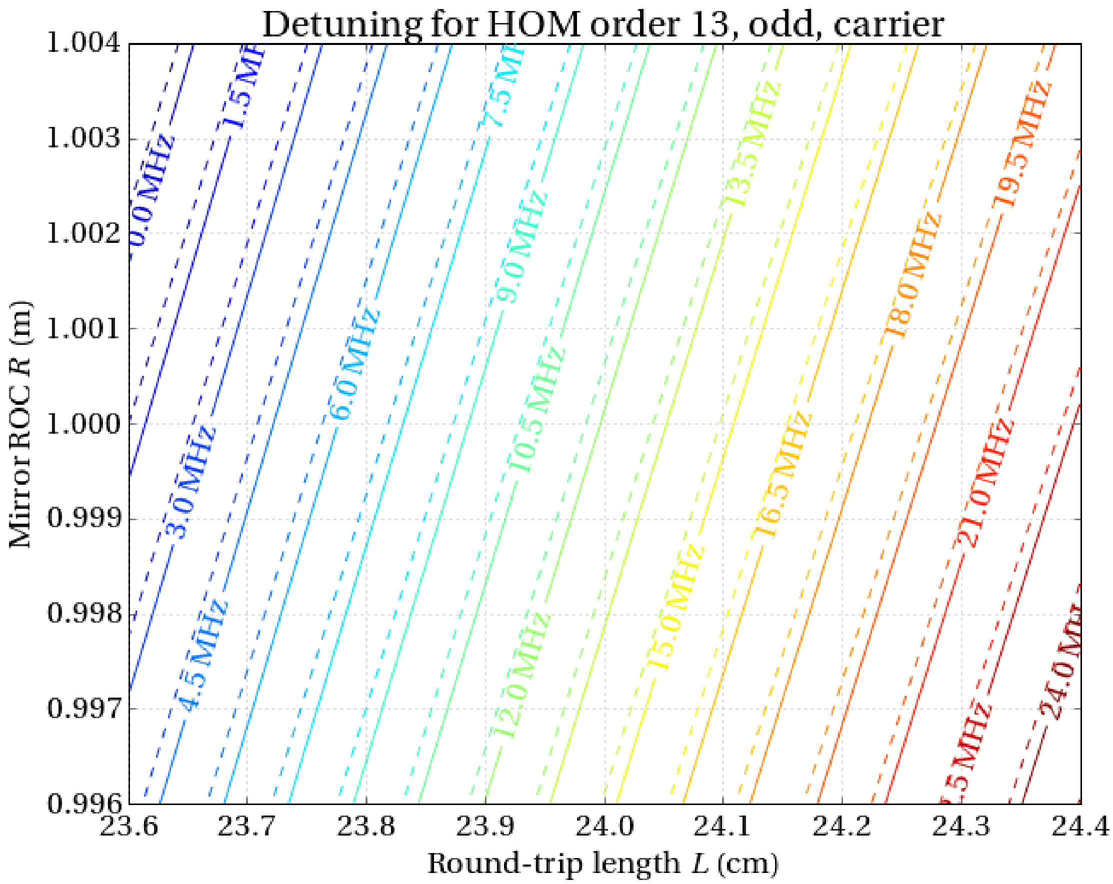
5.1 On R and L

5.1.1 Constraint from HOM filtering

We want the tolerances on δR and δL to be small enough that the corresponding uncertainties $\delta g^{(\mathrm{H,V})}$ do not allow the HOMs considered above (carrier or sidebands) to move into the bandwidth of the TEM_{00} carrier.

Of the few nearly-resonant modes, we examine the one of highest order, since its detuning depends most sensitively on changes in the g-factors.

```
In [28]: LArr2 = arange(L - 4e-3, L + 4e-3, 1e-5)
         RArr = arange(R - 4e-3, R + 4e-3, 1e-5)
         Lmesh, Rmesh = meshgrid(LArr2, RArr)
         fsrMesh = c / Lmesh
         p = 9
         beta = 0
         sb = 0 * fPDH
         detHMesh = fsrMesh * (p * arccos(1 - Lmesh / (Rmesh / cos(theta))) / (2 * pi) + beta / 2)
         detVMesh = fsrMesh * (p * arccos(1 - Lmesh / (Rmesh * cos(theta))) / (2 * pi) + beta / 2)
         detHMesh = mod(detHMesh, fsrMesh) + sb
         detVMesh = mod(detVMesh, fsrMesh) + sb
         detHMesh -= fsrMesh * (detHMesh >= fsrMesh / 2)
         detVMesh -= fsrMesh * (detVMesh >= fsrMesh / 2)
         hfiltering = contour(Lmesh * 100, Rmesh, detHMesh / 1e6, 20)
         ax = gca()
         ax.contour(Lmesh * 100, Rmesh, detVMesh / 1e6, 20, linestyles='dashed')
         def filtfunc(x):
             return '{0:.1f} MHz'.format(x)
         clabel(hfiltering, fmt=filtfunc)
         xlabel(r'Round-trip length $L$ (cm)')
         ylabel(r'Mirror ROC $R$ (m)')
         title(r'Detuning for HOM order 13, odd, carrier')
         ax = gca()
         ax.ticklabel_format(useOffset=False)
```

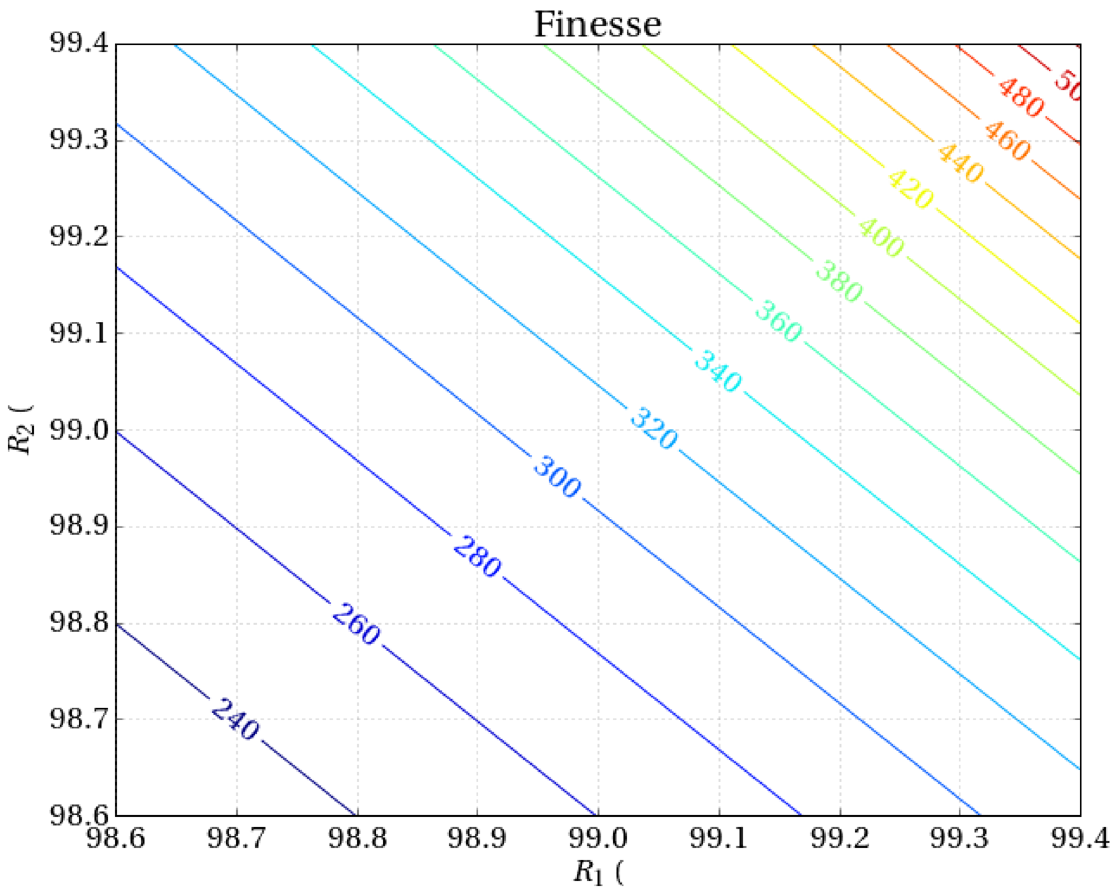


5.2 On reflectivities

5.2.1 Constraint from finesse

We want the reflectivity tolerances $\delta R_{1,2}$ to be small enough that they do not significantly affect the finesse. Note that δR_0 is less critical, since $T_0 \ll T_{1,2}$, and hence the finesse is determined primarily by the two front mirrors.

```
In [29]: r1arr = sqrt(arange(0.986, 0.994, 0.5e-3))
         r2arr = sqrt(arange(0.986, 0.994, 0.5e-3))
         r1mesh, r2mesh = meshgrid(r1arr, r2arr)
         finesseGrid = pi * sqrt(r1mesh * r2mesh) / (1 - r1mesh * r2mesh)
         hfinesse = contour(r1mesh**2 * 100, r2mesh**2 * 100, finesseGrid, 15)
         clabel(hfinesse, fmt='%.0f')
         xlabel(r'$R_1$ (%)')
         ylabel(r'$R_2$ (%)')
         title('Finesse')
         dFdr1 = r / r1 * finesse * (1 / (2 * r) + 1 / (1 - r))
         deltaF = 0.1 * finesse
         deltar1 = deltaF / dFdr1
         deltaR1 = 2 * r1 * deltar1
         #print('To maintain finesse within 10 % of nominal, front mirror reflectivities should each be
```



5.2.2 Constraint from impedance matching

We want to know how well matched the front mirrors' reflectivities have to be in order to get a reasonable impedance match. The plot below shows the reflected power fraction as the reflectivities R_1 and R_2 are varied.

Intracavity field:

$$\frac{E_2}{E_0} = \frac{t_1}{1 + r_0 r_1 r_2 e^{i\omega_0 L/c}}$$

Reflected field:

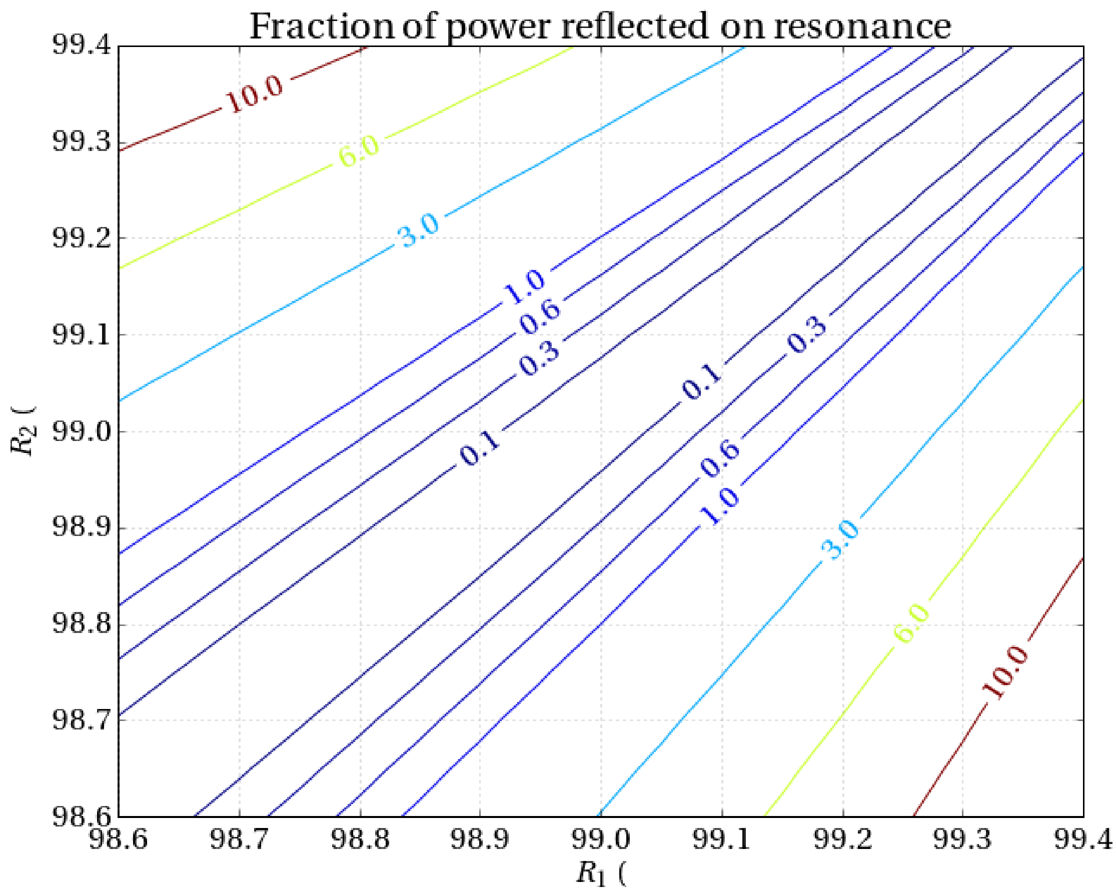
$$\frac{E_1}{E_0} = r_1 + \frac{r_0 t_1^2 r_2 e^{i\omega_0 L/c}}{1 + r_0 r_1 r_2 e^{i\omega_0 L/c}}$$

Resonance occurs for $e^{i\omega_0 L/c} = -1$.

```
In [30]: reflFrac = r1 - r0 * (1 - r1**2) * r2 / (1 - r0 * r1 * r2)
         reflFracGrid = r1mesh - r0 * (1 - r1mesh**2) * r2mesh / (1 - r0 * r1mesh * r2mesh)
         hcoupling = contour(r1mesh**2 * 100, r2mesh**2 * 100, reflFracGrid**2, array([0.1, 0.3, 0.6, 1
         def myfun(x):
             return '{0:.1f} %'.format(x * 100)
         clabel(hcoupling, fmt=myfun)
         matplotlib.patches.Rectangle((r1**2 * 100, r2**2 * 100), 0.1, 0.1)
         xlabel(r'$R_1$ (%)')
         ylabel(r'$R_2$ (%)')
```

```
        title('Fraction of power reflected on resonance')
        print('Power fraction reflected on resonance: {0:.2g} %'.format(reflFrac**2 * 100))
```

```
Power fraction reflected on resonance: 0.0097 %
```



5.3 On substrate imperfections

5.3.1 Homogeneity of index of refraction

The phase ϕ accumulated by a wave travelling through a substrate with thickness s and index of refraction n is

$$\phi = k_n s = \frac{2\pi n}{\lambda} s.$$

Given a small spatial variation Δs of the surface figure and a small spatial variation Δn of the index of refraction, the induced spatial phase error $\Delta\phi$ is given by

$$\left(\frac{\Delta\phi}{\phi}\right)^2 = \left(\frac{\Delta n}{n}\right)^2 + \left(\frac{\Delta s}{s}\right)^2.$$

If we want the phase error to be dominated by the surface figure variation Δs, we should choose a substrate with homogeneity such that $(\Delta n/n)^2 \ll (\Delta s/s)^2$.

```
In [31]: ds1 = lam / 10 # surface figure lambda / 10
         print('Fractional surface figure error: {0:.2g} ppm'.format(ds1 / s1 * 1e6))
```

```
Fractional surface figure error: 17 ppm
```

6 Mechanical design

6.1 Vibration frequency

For an elastic bar of length ℓ with both ends free, Landau and Lifshitz (vol. 7, sec. 25, problem 2) give frequency f_0 of the first longitudinal mode:

$$f_0 = \frac{1}{2\ell}\sqrt{\frac{E}{\rho}}$$

We can use this to get an approximate result for the longitudinal mode of the PMC spacer, but should really compare to the COMSOL model...

In [32]:
```
E = 193e9                              # Pa; Young modulus of steel
rho = 8027                             # kg/m^3; density of steel
# E = 69e9                             # Pa; Young modulus of aluminum
# rho = 2700                           # kg/m^3; density of aluminum
f0 = 1 / (2 * l) * sqrt(E / rho)
print('Approximate frequency of first longitudinal mode: {0:.0f} kHz'.format(f0 / 1e3))
```

```
Approximate frequency of first longitudinal mode: 23 kHz
```

6.2 Tolerancing on spacer length and opening angle

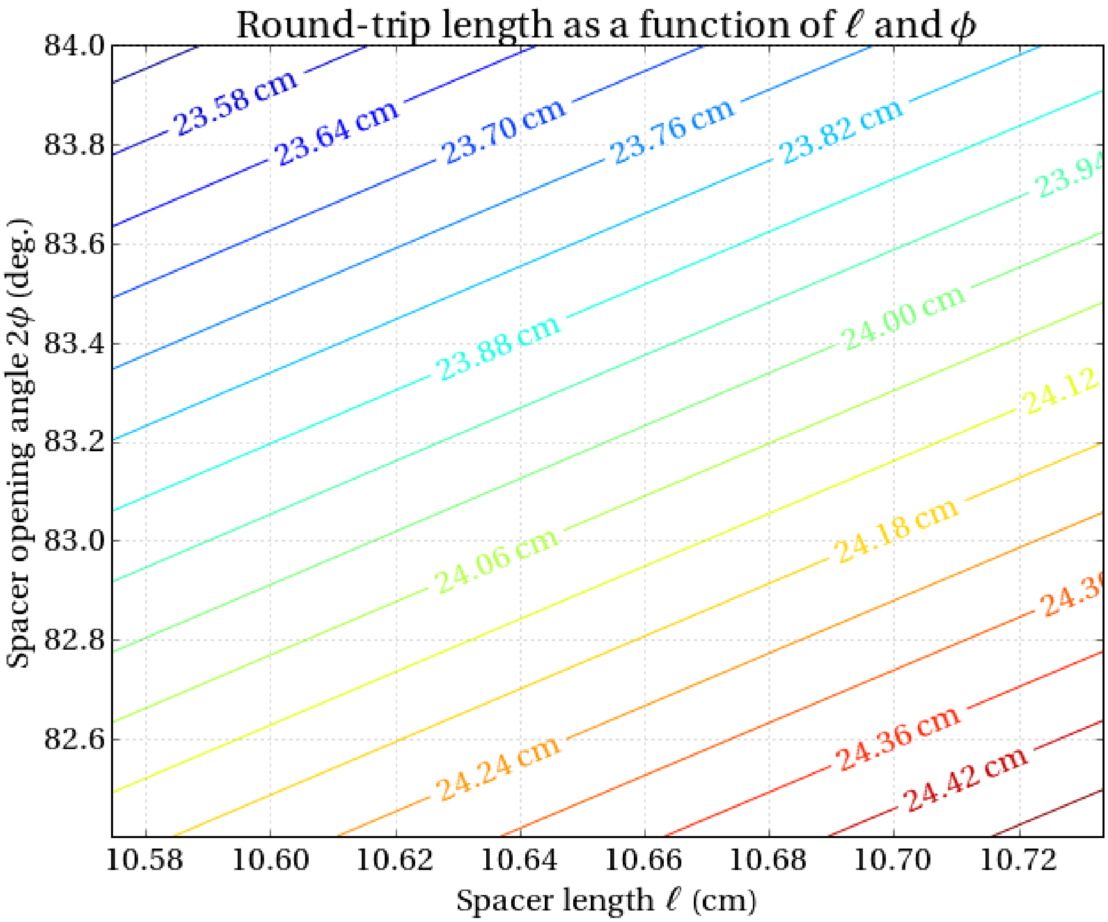
With $b = 2\ell \cot 2\phi$, we have

$$L = b + 2\ell \csc 2\phi = 2\ell(\cot 2\phi + \csc 2\phi) = 2\ell \frac{1 + \cos 2\phi}{\sin 2\phi}.$$

Previously, we have established a tolerance on L.

In [33]:
```
lArr2 = arange(l - 0.8e-3, l + 0.8e-3, 1e-5)
phiArr = arange(phi - 0.4 * pi / 180, phi + 0.4 * pi / 180, 0.01 * pi / 180)
lMesh, phiMesh = meshgrid(lArr2, phiArr)
hrt = contour(lMesh * 100, 2 * phiMesh * 180 / pi, 2 * lMesh * (1 + cos(2 * phiMesh)) / sin(2
xlabel(r'Spacer length $\ell$ (cm)')
ylabel(r'Spacer opening angle $2\phi$ (deg.)')
title(r'Round-trip length as a function of $\ell$ and $\phi$')
clabel(hrt, fmt='%.2f cm')
```

Out[33]:
```
<a list of 15 text.Text objects>
```




```
In [34]: # From the above plot, we choose a tolerance on the spacer length and opening angle
         lTol = 0.02e-2 # m; tolerance on spacer length
         phiTol = 0.1 * pi / 180 # rad;
         print('Tolerance on spacer length: {0:.2f} um = {1:.0f} mils'.format(lTol * 1e6, lTol * 1e6 /
         print('Tolerance on spacer opening angle: {0:.2f} deg'.format(2 * phiTol * 180 / pi))
```

```
Tolerance on spacer length: 200.00 um = 8 mils
Tolerance on spacer opening angle: 0.20 deg
```

6.3 Thermal considerations

Given a linear coefficient of thermal expansion α, the change in round-trip length due to temperature fluctuations is

$$\Delta L \simeq 2\ell\alpha\Delta T.$$

```
In [35]: alpha = 11.5e-6 # 1/K; coefficient of thermal expansion for steel
         dLdK = 2 * l * alpha
         print('Coupling of temperature to round-trip length: {0:.1f} um/K'.format(dLdK * 1e6))
```

```
Coupling of temperature to round-trip length: 2.5 um/K
```

6.4 Mounting

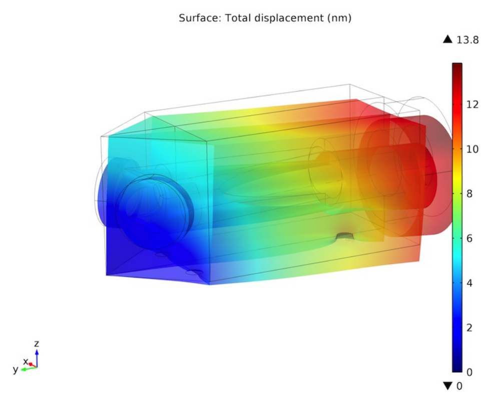
The PMC spacer should rest on three supports, and we should have kinematic contact that fully constrains the six degrees of freedom of the PMC.

6.4.1 Position of supports

The following figures show the gravity-induced deformation of the PMC for two possible configurations of the support points. Evidently, the displacement is more uniform in the case of a single support near the front mirrors and two supports near the endcap. Is this what we want?

In [37]:
```
pmc_supp_back = Image('Figures/pmc_two_back_sm.png')
pmc_supp_front = Image('Figures/pmc_two_front_sm.png')
pmc_supp_back
pmc_supp_front
```

Out[37]:



6.4.2 Kinematicity of supports

For kinematic contact, we can use a modified Kelvin clamp.

In an ideal Kelvin clamp, the first support has a single point contact (e.g., a sphere touching a plane), the second support has two point contacts (e.g., a sphere touching a V-groove), and the third support has three point contacts (e.g., a sphere touching a trihedral socket). A trihedral socket is a pain to machine, so we can instead use a conical socket. This forms a modified Kelvin clamp. A sphere touching a conical socket produces a line contact, not three point contacts, but it should still approximately constrain three of the six degrees of freedom.

- Sapphire
- Press fit?

In [ ]:

7 Other

7.1 Vendor options

7.1.1 Optics

REO, ATF, MLD, G&H, Coastline, Laseroptik

7.1.2 PZT

Noliac NAC2124 seems to be what we want:

- Ring actuator
- OD = 15 mm, ID = 9 mm, good to <0.5 mm
- Thickness: 2 ± 0.05 mm = 79 ± 2 mil
- Free stroke: 3 μm ($\Rightarrow$ 6 FSRs), good to 15%
- Capacitance: 510 nF, good to 15%
- Choose option A01 for wiring

Reverse voltage protection?

7.1.3 Mechanical

MillItNow

7.2 Thoughts

Astigmatism
Koji says no to half-inch input/output mirrors; half-inch back mirror OK (0.125" thickness)
Input/output mirrors should be 0.25" thick (not 0.375"), polished barrel
Hold input/output mirrors with screws and ring
PZT: Rana says use Noliac