RotazioniSubreflettore

September 3, 2014

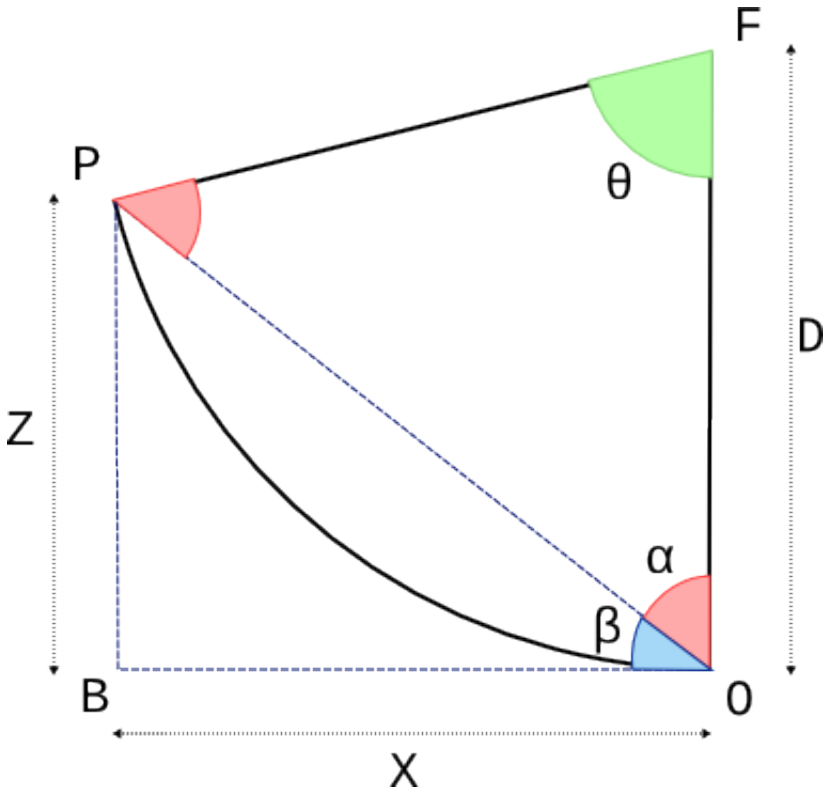
```
In [187]: %pylab inline
          from sympy import *
          from IPython.display import Image
          init_printing(use_latex="mathjax")
```

```
Populating the interactive namespace from numpy and matplotlib

WARNING: pylab import has clobbered these variables: ['prod', 'Circle', 'power', 'diag', 'sinh', 'trunc
'%matplotlib' prevents importing * from pylab and numpy
```

1 Rotazione del subreflettore intorno al fuoco

```
In [188]: Image("files/rot_sub.png")
```

Out[188]:



In [188]:

Sia dato il sistema descritto il figura in cui: * **O** rappresenta il centro del subriflettore in posizione 0 * **F** rappresenta il fulcro del parboloide * **D** è la distanza tra il subriflettore e il fuoco nella posizione 0 * $ $\theta = \widehat{OFP}$$, $angoloditiltdelsubriflettorenecessarioadinquadrareunfeedinvertex * |FO| = |FP| = \mathbf{D}$

Vogliamo trovare le compensazioni **Z** e **X** da applicare al movimento del subriflettore per mantenerlo centrato sull'asse di fuoco senza introdurre errori di puntamento.

Essendo il triangolo OFP isoscele abbiamo che:

$$\alpha = \frac{\pi - \theta}{2} = \frac{\pi}{2} - \frac{\theta}{2}$$

da cui possiamo ricavare l'ampiezza dell'angolo β:

$$\beta = \frac{\pi}{2} - \alpha = \frac{\pi}{2} - \frac{\pi}{2} + \frac{\theta}{2} = \frac{\theta}{2}$$

Possiamo quindi calcolare **X** e **Z** sapendo che:

$$|PO| = 2 * (D * \sin\frac{\theta}{2})$$

E sfruttando il fatto che PBO è rettangolo in B:

$$X = |PO| * \cos\beta = 2 * D * \sin\frac{\theta}{2} * \cos\frac{\theta}{2} = D * \sin\theta$$

$$Z = |PO| * \sin\beta = 2 * D * \sin\frac{\theta}{2} * \sin\frac{\theta}{2} = 2 * D * (\sin\frac{\theta}{2})^2$$

In [189]:
```
d, t, z, x = symbols("D theta z x")
z = 2 * d * (sin(t/2)**2)
x = d * sin(t)
```

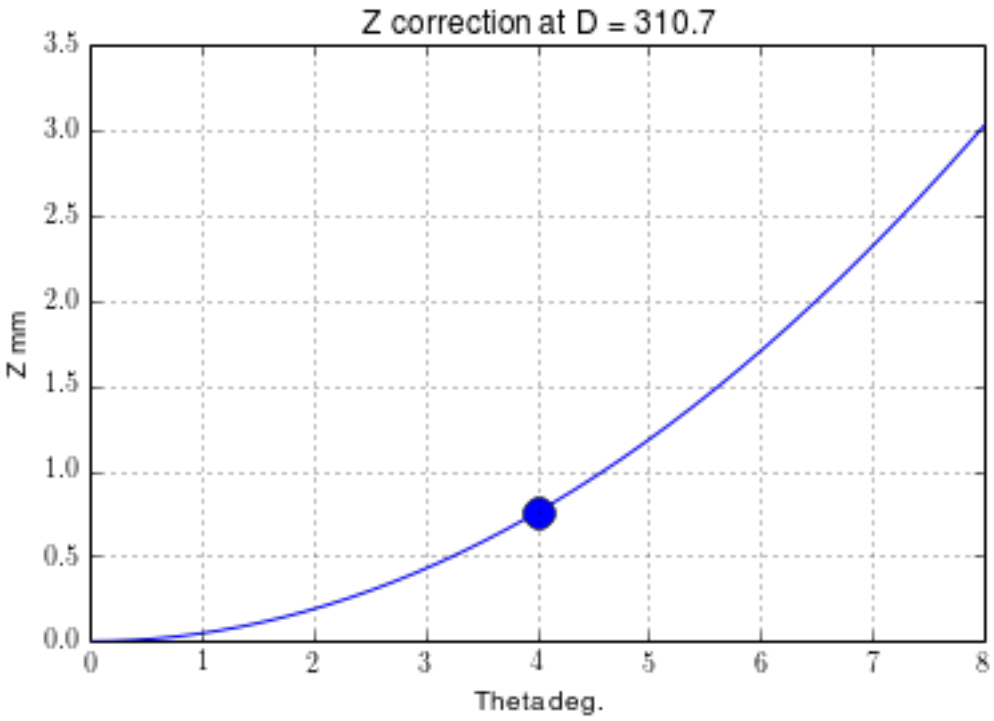
Sapendo che $ D = 310.7mm $ calcolato geometricamente per il subreflettore possiamo calcolare il variare di **X** e **Z** al variare dell'angolo in vertex

In [190]:
```
xaxis = linspace(0, radians(8), 1000)
xaxis_labels = linspace(0, 8, 1000)
z_at_fixed_d = 2 * 310.7 * np.sin(xaxis/2)**2
x_at_fixed_d = 310.7 * np.sin(xaxis)
#z_at_fixed_d = array([N(z.subs(D, 310.7).subs(t, _theta)) for _theta in xaxis])
#x_at_fixed_d = array([N(x.subs(D, 310.7).subs(t, _theta)) for _theta in xaxis])
```

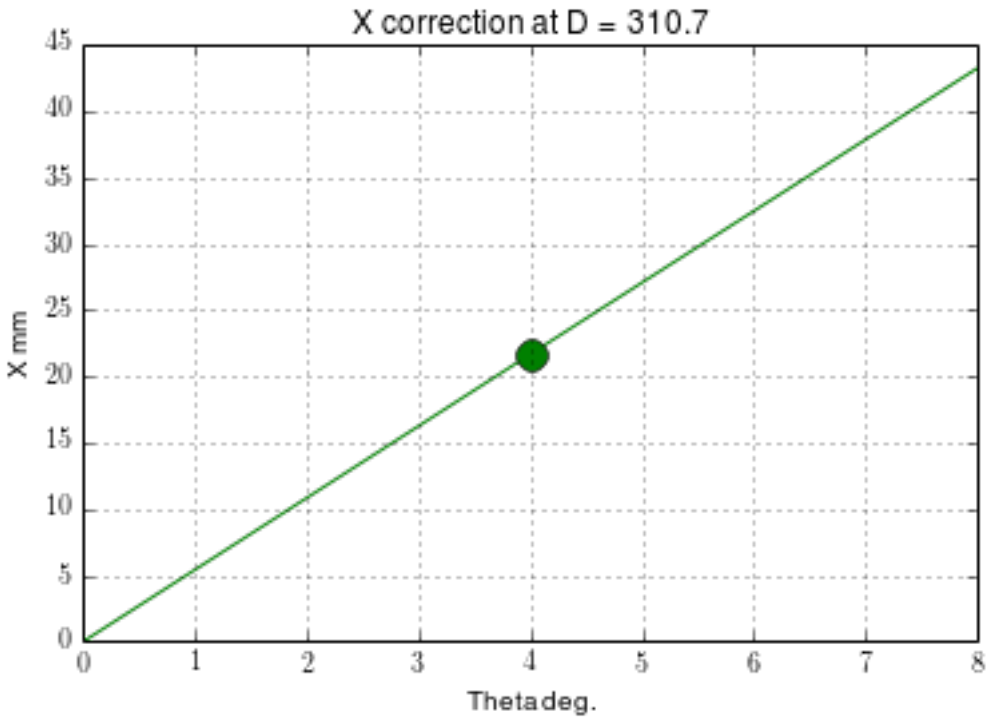
Dai plot seguenti vediamo come nella posizione 0 del sureflettore una rotazione di un angolo θ attorno al fuoco rappresenti una piccola correzione in Z e una più consistente correzione in X

In [191]:
```
pylab.title("Z correction at D = 310.7")
pylab.xlabel("Theta deg.")
pylab.ylabel("Z mm")
pylab.grid(True)
pylab.plot(xaxis_labels, z_at_fixed_d, 'b-',
           xaxis_labels[500], z_at_fixed_d[500], 'bo',
           markersize = 12)
```

Out[191]:
```
[<matplotlib.lines.Line2D at 0xee374d0>,
 <matplotlib.lines.Line2D at 0xee37750>]
```

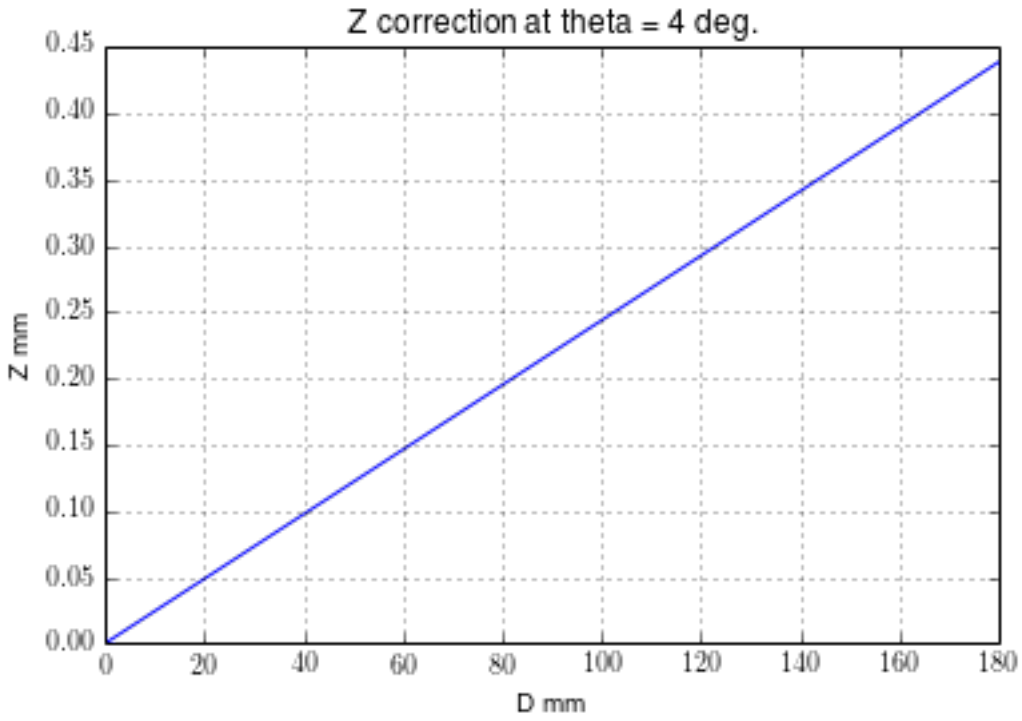



```
In [192]: pylab.title("X correction at D = 310.7")
          pylab.xlabel("Theta deg.")
          pylab.ylabel("X mm")
          pylab.grid(True)
          pylab.plot(xaxis_labels, x_at_fixed_d, 'g-',
                     xaxis_labels[500], x_at_fixed_d[500], 'go',
                     markersize = 12)

Out[192]: [<matplotlib.lines.Line2D at 0xf074ad0>,
           <matplotlib.lines.Line2D at 0xf074d50>]
```

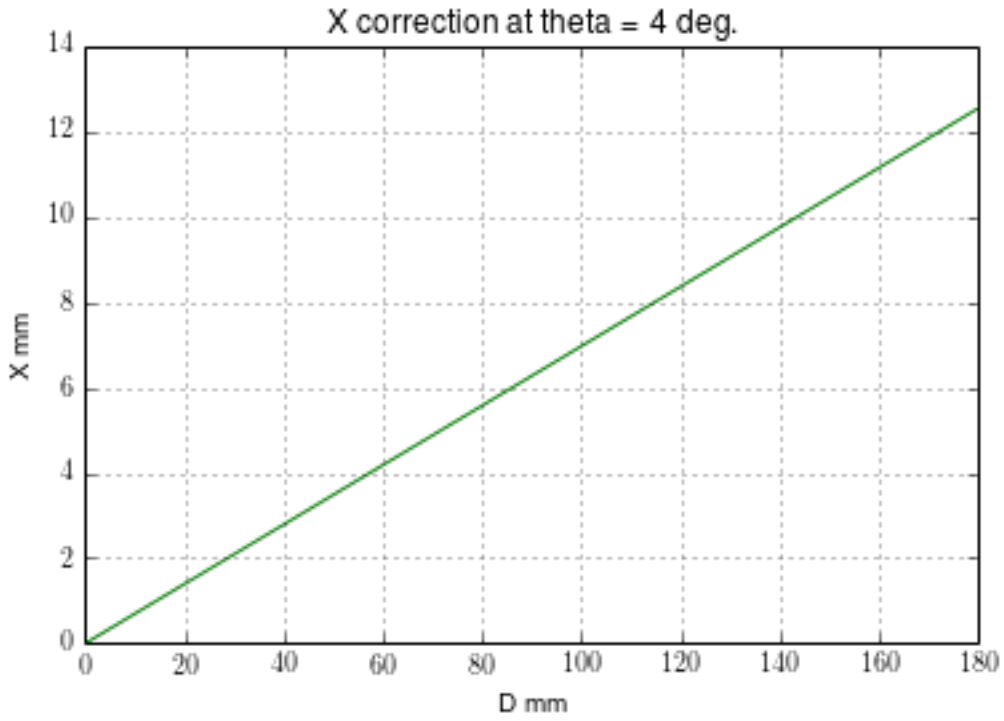


Presupponiamo ora di inquadrare un ricevitore con un angolo di $\theta = 4°$ e di voler effettuare un focheggiamento muovendo il subreflettore verso il ricevitore di una quantità che facciamo variare tra 0 e 18cm, calcolati come $ $3\lambda@5GHz$.*Ancheinquestocasopossiamocontrollareilvariaredellecorrezioniapplicatein***Z***e***X***alvariaredelladistanzadelsubr*

```
In [193]: xaxis = linspace(0, 180, 1000)
          z_at_fixed_t = array([N(z.subs(t, radians(4)).subs(D, _d)) for _d in xaxis])
          x_at_fixed_t = array([N(x.subs(t, radians(4)).subs(D, _d)) for _d in xaxis])

In [194]: pylab.title("Z correction at theta = 4 deg.")
          pylab.xlabel("D mm")
          pylab.ylabel("Z mm")
          pylab.grid(True)
          pylab.plot(xaxis, z_at_fixed_t, 'b-')

Out[194]: [<matplotlib.lines.Line2D at 0xf2de290>]
```



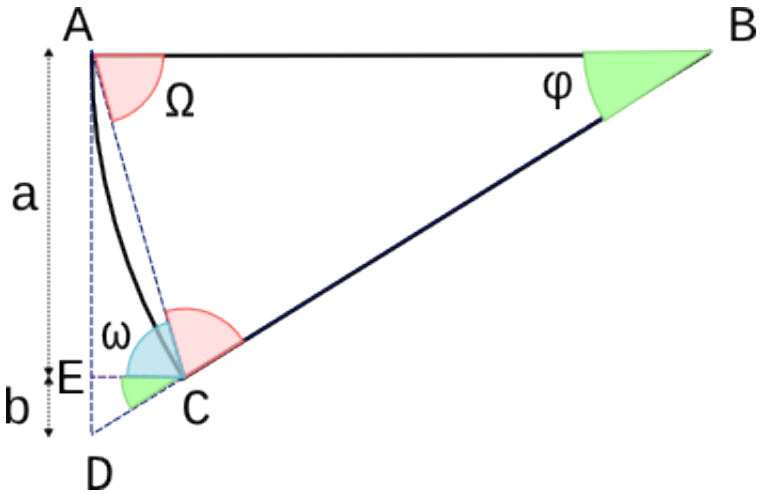

```
In [195]: pylab.title("X correction at theta = 4 deg.")
          pylab.xlabel("D mm")
          pylab.ylabel("X mm")
          pylab.grid(True)
          pylab.plot(xaxis, x_at_fixed_t, 'g-')

Out[195]: [<matplotlib.lines.Line2D at 0xf364cd0>]
```



2 Tilt del subreflettore

In [196]: `Image("files/tilt_sub2.png")`

Out[196]:



Sia dato il sistema in figura in cui: * AB è un lato del triangolo subreflettore (vista B in Fig. 3.1.2 pag. 90 "Matematica di sistema" 4 di 4) * $|AB| = 2068mm$ * φ è l'angolo di cui vgliamo ruotare attorno al suo asse il subreflettore * BC rappresenta il posizionamento del subreflettore una volta compiuta la rotazione

NOTA: *Per semplicità consideriamo il caso di rotazione sull'asse X applicando un angolo θ_y che si riflette sul solo movimento di due attuatori. In questo caso sto solo cercando di quantificare un possibile errore, in realtà B si dovrebbe muovere verso l'alto mentre A verso il basso, in questo caso direi che quello che ci interessa è la differenza tra i due.*

Ragionando sui triangoli e sfruttando il fatto che $|AB| = |BC|$ possiamo desumere che: * $\omega = \Omega$ perchè alterni-interni

$$\Omega = \omega = \frac{\pi - \varphi}{2} = \frac{\pi}{2} - \frac{\varphi}{2}$$

$$\sin\Omega = \sin\omega = \cos\frac{\varphi}{2}$$

$$\cos\Omega = \cos\omega = \sin\frac{\varphi}{2}$$

Ragionando a questo punto sui lati otteniamo che:

$$|AC| = 2 * |AB| * \sin(\frac{\varphi}{2})$$

$$|AE| = |AC| * \sin\omega = 2 * |AB| * \sin\frac{\varphi}{2} * \cos\frac{\varphi}{2} = 2 * |AB| * \frac{\sin\varphi}{2} = |AB| * \sin\varphi$$

$$|EC| = |AC| * \cos\omega = 2 * |AB| * \sin\frac{\varphi}{2} * \sin\frac{\varphi}{2} = 2 * |AB| * (\sin\frac{\varphi}{2})^2 = |AB| * (1 - \cos\varphi)$$

$$|ED| = |EC| * \tan\varphi = |AB| * \tan\varphi - |AB| * \cos\varphi * \frac{\sin\varphi}{\cos\varphi} = |AB| * (\tan\varphi - \sin\varphi)$$

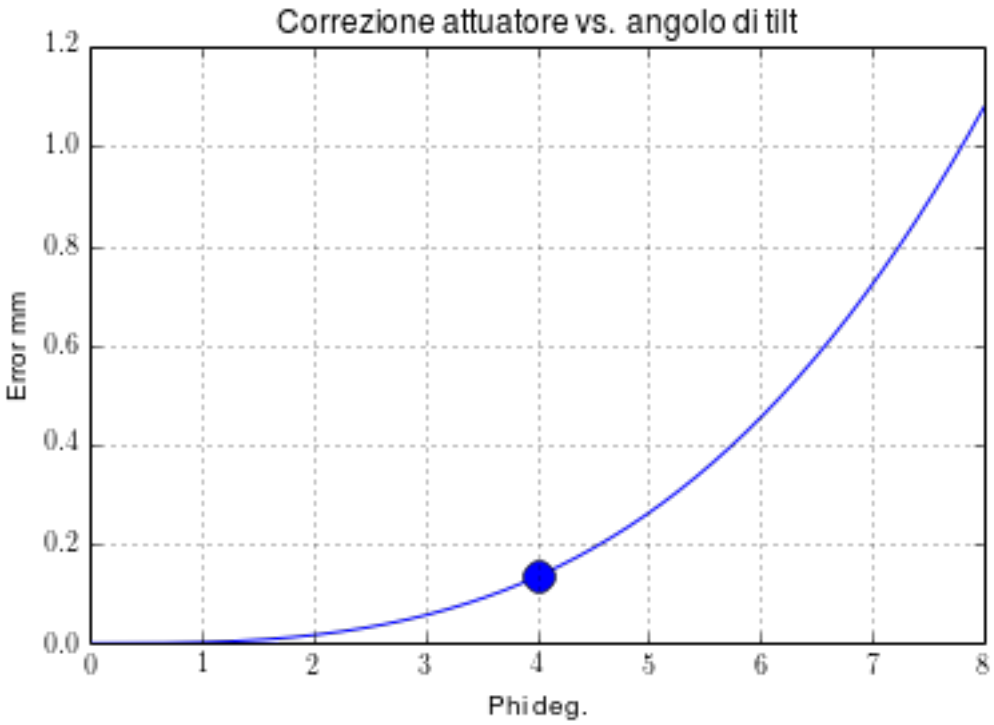
$$|AD| = |AB| * \tan\varphi$$

Que e ci interessa è che quindi quando noi vogliamo comandare un tilt di φ e diamo come posizione comandata $|AD|$ in realtà dovremmo dare un valore di $|AC|$, per cui:

$$|AD| - |AC| = |AB| * \tan\varphi - 2 * |AB| * \sin\frac{\varphi}{2} = |AB| * (\tan\varphi - 2 * \sin\frac{\varphi}{2})$$

Fissanso quindi come da ipotesi $|AB| = 2098mm$ calcoliamo l'errore al variare di φ:

```
In [199]: xaxis = linspace(0, radians(8), 1000)
          xaxis_labels = linspace(0, 8, 1000)
          error = (2098/2) * (np.tan(xaxis) - 2*np.sin(xaxis/2))

In [201]: pylab.title("Correzione attuatore vs. angolo di tilt")
          pylab.xlabel("Phi deg.")
          pylab.ylabel("Error mm")
          pylab.grid(True)
          pylab.plot(xaxis_labels, error, 'b-',
                     xaxis_labels[500], error[500], 'bo',
                     markersize = 12)

Out[201]: [<matplotlib.lines.Line2D at 0xfa16310>,
           <matplotlib.lines.Line2D at 0xfa16590>]
```


Correzione attuatore vs. angolo di tilt
Error mm
Phideg.
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
1
2
3
4
5
6
7
8

In []:

In []: