Table of Contents

1 Carreguem les Dades

In [50]:

```
from IPython.display import Latex
%matplotlib inline
%run ../starter.py
```

Dades carregades directament: error experimental fet al mesurar de 0.017 graus, i angle zero de 117.6+/-0.02:

In [51]:

```
base_error = 1./60. #1 minut
zero_angle = uc.ufloat(117.6,base_error)
```

Carreguem les dades preses al laboratori:

In [52]:

```
data_Hg = pd.read_csv("IN/CSV/hg.csv", s
```

Dades preses per al Mercuri:

	Theo	M1_deg	M1_min	M2_deg	M2_min	M3_
0	5790.0	43	35.0	43	30	43
1	5769.6	43	28.0	43	29	43
2	5460.7	42	43.0	42	45	42
3	4916.0	41	4.0	41	8	41
4	4358.3	38	22.5	38	22	38
5	4078.2	36	23.0	36	14	36
6	4046.6	36	0.0	36	0	36

In [53]:

```
data_Cd = pd.read_csv("IN/CSV/cd.csv", s
```

Dades preses per al Cadmi

	Theo	M1_deg	M1_min	M2_deg	M2_min	M3_
0	6438.5	44	40	44	40	44
1	5155.1	41	57	41	57	41
2	5085.8	41	40	41	40	41
3	4799.9	40	40	40	40	40
4	4678.2	40	7	40	5	40
5	4413.0	38	40	38	45	38

La columna `'Theo'` de les dades mesurades són les λ teòriques, i a partir de les mesures `'Mi_deg'` i `'Mi_min'` construim les mesures del laboratori.

In [54]:
```
hg_lambda = np.array(data_Hg['Theo'])
```

In [55]:
```
cd_lambda = np.array(data_Cd['Theo'])
```

Creem les mesures amb incertesa a partir de les mesures preses:

In [56]:
```
hg_meas = np.array([data_Hg['M'+str(j)+'_deg'.format(j)]
                    +data_Hg['M'+str(j)+'_min']/60. for j in range(1,3+1)])

cd_meas = np.array([data_Cd['M'+str(j)+'_deg'.format(j)]
                    +data_Cd['M'+str(j)+'_min']/60. for j in range(1,3+1)])
```

In [57]:
```
hg_delta = unp.uarray(np.mean(hg_meas, a
                      (np.std(hg_meas, a
                       +base_error**2)**
hg_delta = zero_angle - hg_delta
```

In [58]:
```
cd_delta = unp.uarray(np.mean(cd_meas, a
                            (np.std(cd_meas, a
                             +base_error**2)**
cd_delta = zero_angle - cd_delta
```

Angles obtinguts per al Mercuri

	Lambda	delta
0	5790.0	74.07+/-0.05
1	5769.6	74.133+/-0.027
2	5460.7	74.84+/-0.04
3	4916.0	76.50+/-0.04
4	4358.3	79.236+/-0.026
5	4078.2	81.31+/-0.07
6	4046.6	81.600+/-0.024

Angles obtinguts per al Cadmi

	Lambda	delta
0	6438.5	72.933+/-0.024
1	5155.1	75.644+/-0.025
2	5085.8	75.917+/-0.033
3	4799.9	76.933+/-0.024
4	4678.2	77.506+/-0.028
5	4413.0	78.91+/-0.05

2 Ajust contra les funcions del guió

Definim les funcions $\delta(\lambda)$ i $\lambda(\delta)$ tal i com ens les dona el guió (aproximació de hartman):

$$\delta(\lambda) = \delta_0 + \frac{c}{\lambda - \lambda_0}$$

$$\lambda(\delta) = \lambda_0 + \frac{c}{\delta - \delta_0}$$

In [59]:

```
delta_hartman = lambda lam, delta_0, c, lam_0 : delta_0 + c/(lam-lam_0)
lam_hartman = lambda delta, delta_0, c, lam_0 : lam_0 + c/(delta-delta_0)
```

Ara, l'objectiu seria utilitzar el mercuri com a calibratge per els paràmetres de la funció de hartman. Així podríem trobar les longituds d'ona del Cadmi, i després comprovar la qualitat dels resultats obtinguts. La forma natural de fer-ho seria mitjançant una regressió de mínims quadrats, caculant els estimadors associats a les desviacions estàndard de cada paràmetre. Tot i això, el guió ens recomana un procediment diferent. Provarem tots dos procediments.

2.1 Mètode de mínims quadrats

Ara podem utilitzar aquestes dades per a calibrar utilitzant del Mercuri. n (fer-ho mitjançant $\delta(\lambda)$ ens ha semblat més adequat que mitjançant $\lambda(delta)$ ja que ens trobem en una situació molt més propera a la que s'utilitza en els teoremes que suporten les hipòtesis de regressions lineals)

In [60]:

```
hartman_params_hg = scp.optimize.error_curve_fit(delta_hartman,
                                                 hg_lambda,
                                                 hg_delta,
                                                 p0=[54,110000,0] )

hartman_params = scp.optimize.error_curve_fit(delta_hartman,
                                              np.concatenate([hg_lambda, cd_lamb
                                              np.concatenate([hg_delta, cd_delta
                                             p0=unp.nominal_values(hartman_param
```

In [61]:

```
r = [Latex('${:L}$'.format(x)) for x in hartman_params_hg]
```

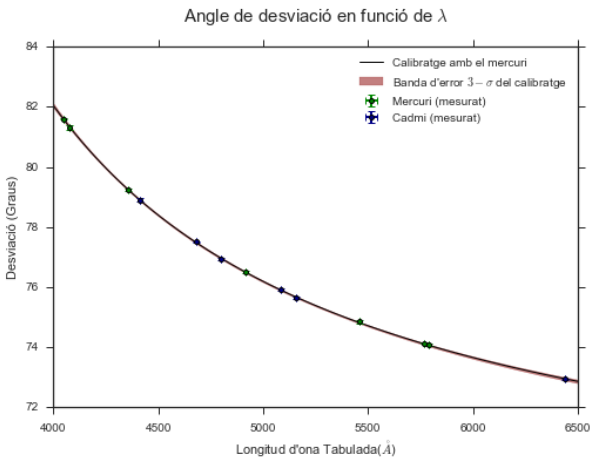
De les dades obtingudes per al mercuri, obtenim un valor de:

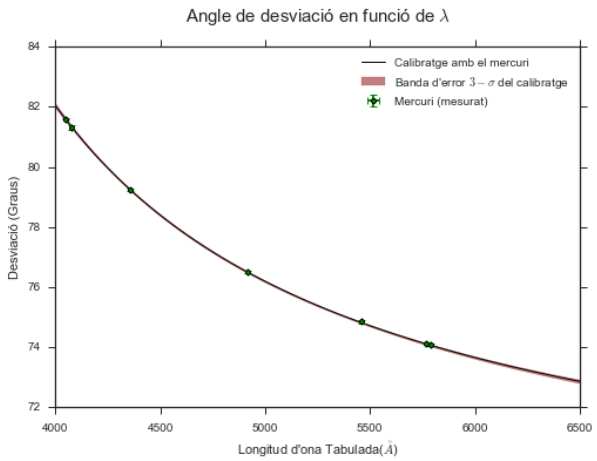
- δ_0 de 67.34 ± 0.22
- c de $(2.23 \pm 0.10) \times 10^4$
- λ_0 de $(2.48 \pm 0.05) \times 10^3$

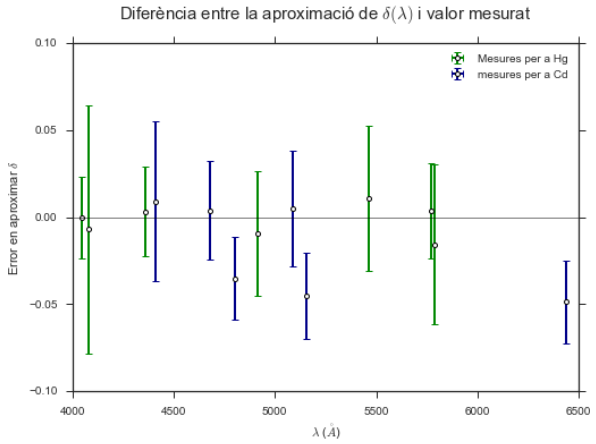
Punt important: Aquestes dades les hem obtingut només amb el mercuri, sense utilitzar el cadmi, ja que volem utilitzar el mercuri per calibrar i mesurar el cadmi.

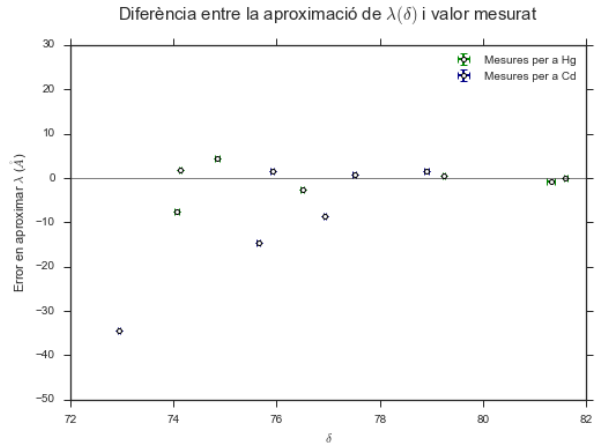
Ara podem representar les dades obtingudes, així com la regressió obtinguda:

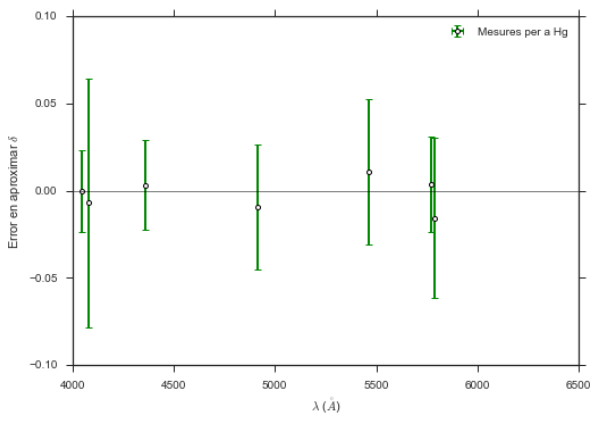
Com que en el dibuix es veuen els punts extremadament propers a la corba, no podem apreciar correctament la precisió en el dibuix. Per aquest motiu, hem pensat que podia ser una bona idea representar les diferències entre els punts i la corba. Amb aquest objectiu representarem dues funcions: $\lambda_i - \lambda(\delta_i)$ i δ_i - . Així podrem observar verdadermanet els errors comesos:

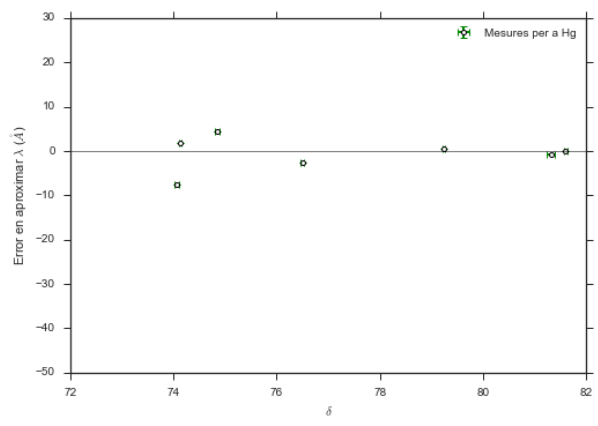












Una pegunta que podria resultar interessant fer-se mirant aquests gràfics és: per què en un cas es veuen unes barres d'error importants i en l'altre no hi ha barres d'error visibles?

La resposta és que coneixem l'error en δ, calculat per nosaltres, i de l'ordre de 0.05, mentre l'error en λ, tot i que no és conegut, degut a que les dades donades tenen un nivell de significació de $0.1\mathring{A}$, podem assumir que és extremadament petit. Com que el que estem calculant són diferències, l'error en l'eix y es veu molt més amplificat que el de l'eix x, ja que la escala és molt més reduida. Això és suficient per a mostrar l'error de forma no negligible en el les barres d'error en el primer gràfic, però no en el segon, on son molt més petites.

2.2 Mètode del guió:

El guió ens proposa utilitzar les següents formules per a calcular els paràmetres de hartman només utilitzant tres punts. Ens recomanen agafar la ralla groga, verda i blava:

In [68]:

```
l_y , l_g, l_b = hg_lambda[(0,2,4),]
d_y , d_g, d_b = hg_delta[(0,2,4),]
print l_y, l_g, l_b
```

```
5790.0 5460.7 4358.3
```

A partir d'aquestes ens indica, per a calcular els valors dels paràmetres, que calculem abans tres valors intermedis:

In [69]:

```
A = (d_y-d_g)/(l_g-l_y)
B = (d_y-d_b)/(l_b-l_y)
K = (d_g-d_b)/(A-B) #K es C/(d_y-del0)
```

I a partir d'aquests paràmetres podem calcular:

In [70]:

```
lam0 = l_y - K
del0 = d_y - (l_g-lam0)*A
c = K*(d_y-del0)
hartman_params_interp = np.array((del0,c,lam0))
```

In [71]:

```
r = [Latex('${:L}$'.format(x)) for x in hartman_params_hg]
r1 = [Latex('${:L}$'.format(x)) for x in (del0, c,lam0)]
r2 = ["{:0.10}".format(x.n) for x in ( del0, c, lam0)]
```

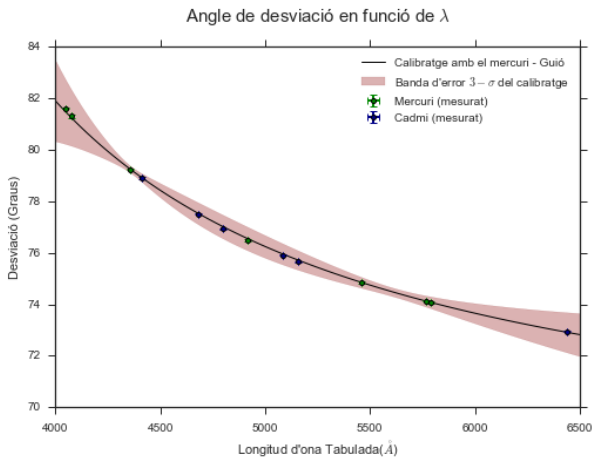
Els resultats obtinguts per als valors són per tant:

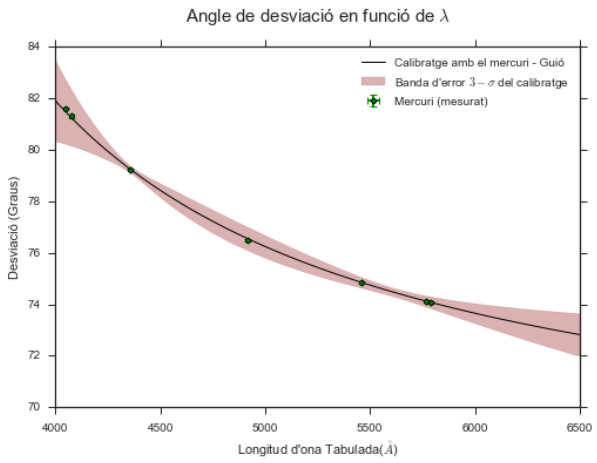
Paràmetre	Mínims Quadrats	Mètode Guió	Mètode Guió [10 xifres]
δ_0 [°]	67.34 ± 0.22	**KeyError**: 0	**KeyError**: 0
C [Å]	$(2.23 \pm 0.10) \times 10^4$	**KeyError**: 1	**KeyError**: 1
λ_0 [Å]	$(2.48 \pm 0.05) \times 10^3$	**KeyError**: 2	**KeyError**: 2

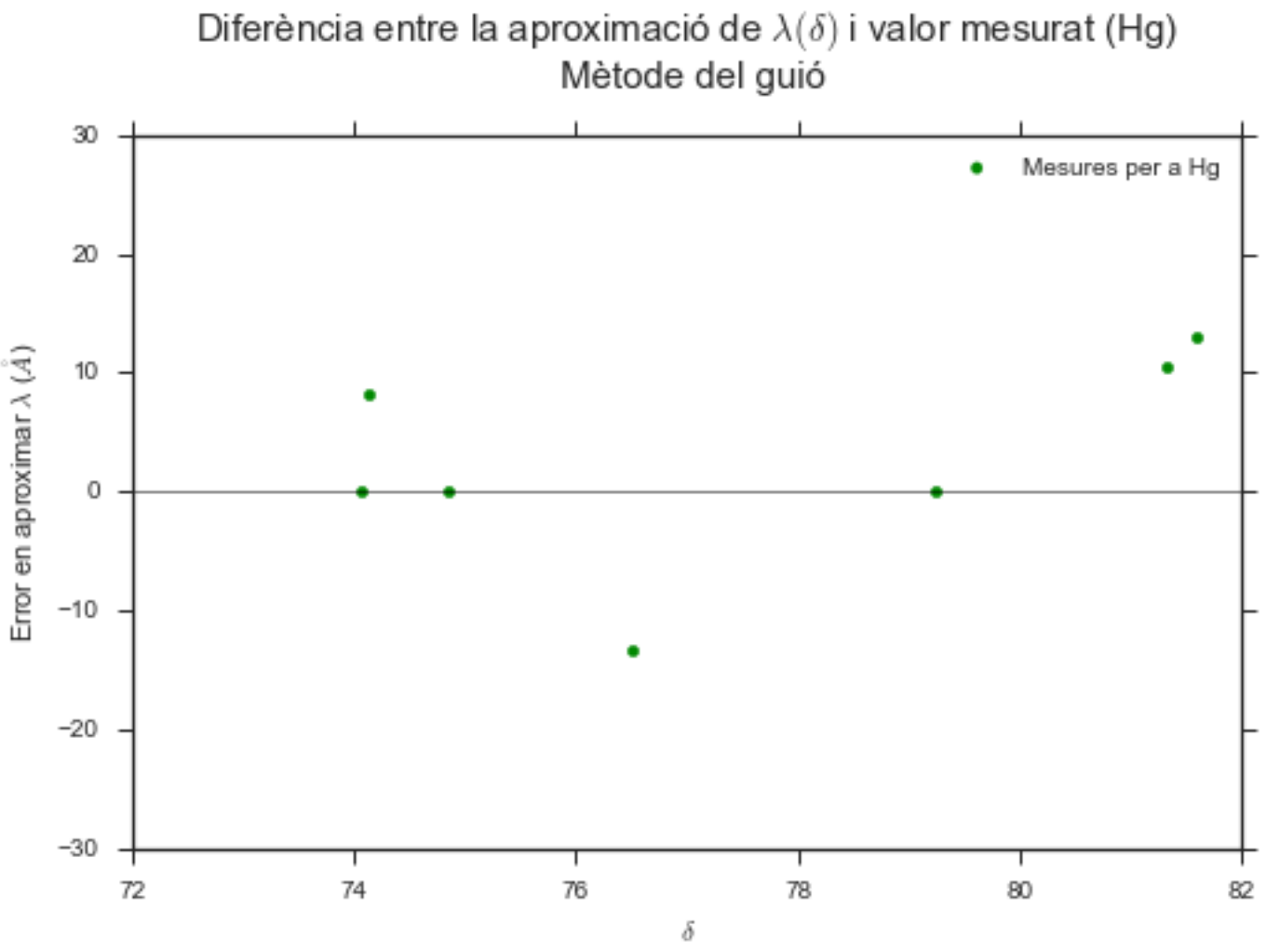
Com es pot observar, els valors són considerablement diferents, anem a representar per tant la nova interpolació, per a veure també si te sentit.

El primer que observem és que la banda d'error a $3-\sigma$ és molt major, cosa que té sentit: si utilitzem menys dades, augmentem considerablement l'error. Aquest és el fenòmen de què ens parlava el guió. El segon fenòmen a observar és que la corba també aproxima sorprenentment bé els punts. Fins i tot tenint un gran error associat, és un bon estimador. Per tant, anem a seguir el mètode del guió per a recalcular les incerteses.

El guió comenta que hauriem d'estimar una la incertesa en l'estimació a partir d'observar les diferències entre λ i la interpolació de λ. Per a fer-ho representarem aquesta diferència en funció de δ. A partir del gràfic que ens mostri les diferències, decidirem el valor oportú per a l'error:







Notem que en el gràfic no hi ha barres d'error per que l'objectiu és utilitzar el propi gràfic per a trobar els errors!

A partir d'aquí, i de forma totalment arbitrària, associem un error en la aproximació de $10\mathring{A}$ a la mesura de la longitud d'ona. Notem que, com esperaríem, l'error en els tres punts que hem utilitzat per a interpolar és zero. A més, a aquest mètode, recordem que li haurem de sumar les incerteses associades a l'error de δ

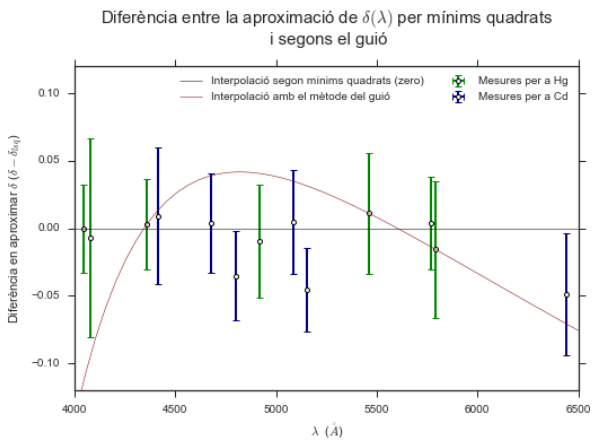
In [75]:

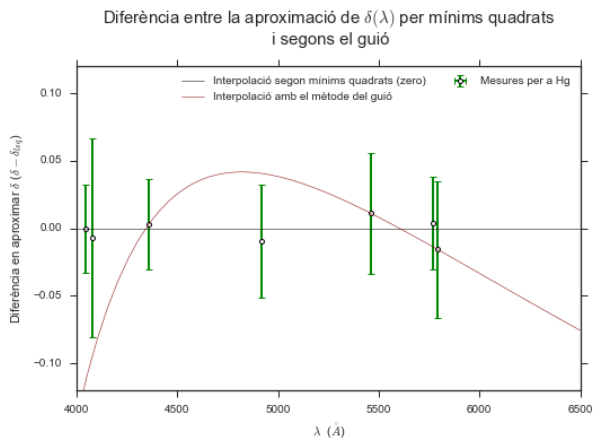
```
guio_interp_error = 10
```

Ara que ja hem trobat una forma d'associar una incertesa a la interpolació, ens interessaria comparar com de bons són els dos estimadors que hem trobat fins ara, per a decidir amb quin ens quedem.

2.3 Comparació dels dos mètodes

Agafem i a tots els valors experimentals, i els restem els valors obtinguts amb el mètode de mínims quadrats, tal i com hem fet abans. Al calibratge segons el guió li fem el mateix. La motivació de fer això és que en els gràfics observats fins ara els errors són massa petits, i no podem observar les diferències a ull nu. En canvi, si estudiem les diferències entre les corbes, podrem veure realment quines aproximen millor.





Com podem observar, la corba negra (zero - la associada a la corba de mínims quadrats que acabem de restar) aproxima millor les dades que no la corba vermella. La corba vermella, recordem, és la associada a les dades del guió. Per tant, el mètode dels mínims quadrats ens dona una millor aproximació i és amb el que ens quedarem. És el que s'hauria d'esperar, per que el mètode dels mínims quadrats utilitza al màxim la informació disponible en les 6 mesures preses.

Com a punt final de la comparació, tant per al mercuri com per al cadmi, podem calcular els resultats de les interpolacions, amb els seus errors, i comparar-los amb els valors tabulats. Recordem que la λ via la interpolacio la hem de fer amb els valors nominals (sense incertesa) obtinguts abans

In [78]:

```
hg_leastsq_lam = lam_hartman(hg_delta, *hartman_params_hg)
hg_guio_lam = lam_hartman(hg_delta, *unp.nominal_values(hartman_params_interp))
hg_guio_lam = unp.uarray(unp.nominal_values(hg_guio_lam),
                         unp.std_devs(hg_guio_lam)+guio_interp_error)

cd_leastsq_lam = lam_hartman(cd_delta, *hartman_params_hg)
cd_guio_lam = lam_hartman(cd_delta, *unp.nominal_values(hartman_params_interp))
cd_guio_lam = unp.uarray(unp.nominal_values(cd_guio_lam),
                         unp.std_devs(cd_guio_lam)+guio_interp_error)
```

In [79]:

```
r1 = pd.DataFrame({"Nominal":hg_lambda,"Mínims Quadrats":hg_leastsq_lam, "Guió":
r2 = pd.DataFrame({"Nominal":cd_lambda,"Mínims Quadrats":cd_leastsq_lam, "Guió":
```

Interpolació vs Nominal: Mercuri

KeyError: "['Nominal' 'M\xc3\xadnims Quadrats' 'Gui\xc3\xb3'] not in index"

Interpolació vs Nominal: Cadmi

	Nominal	Mínims Quadrats	Guió
0	6438.5	6473+/-33	6424+/-26
1	5155.1	5170+/-10	5179+/-18
2	5085.8	5084+/-12	5095+/-20

3	4799.9	4808+/-8	4819+/-16
4	4678.2	4677+/-8	4686+/-16
5	4413.0	4411+/-8	4414+/-18

3 Càlcul de l'Index de refracció

Ara podem calcular ja la famosa gràfica $n(\lambda)$, tant a partir dels punts calculats, com a partir de la interpolació feta. La equació que relaciona els dos termes és:

$$n(\delta) = \frac{\sin\left(\frac{\alpha+\delta}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} \tag{1}$$

Passem per tant a calcular-ho:

In [80]:

```
def n(delta, alpha): return unp.sin((delta+alpha)/2.)/unp.sin(alpha/2.)
```

In [81]:

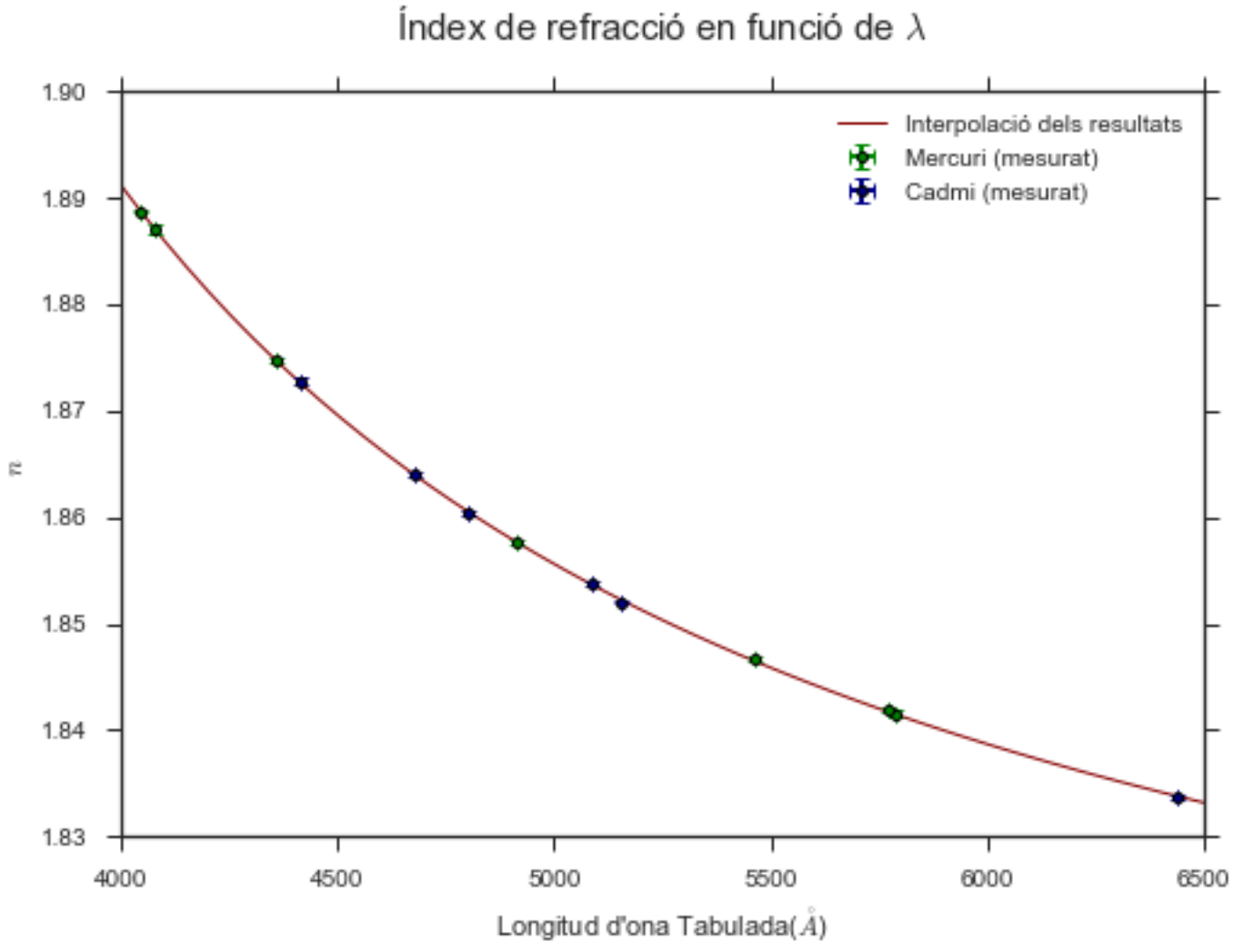
```
lamb = np.linspace(4000,6500)
interp_delta = delta_hartman(lamb, *unp.nominal_values(hartman_params))

interp_n = n(interp_delta/360*2*np.pi, np.pi/3.)
hg_n = n(hg_delta/360*2*np.pi, np.pi/3.)
cd_n = n(cd_delta/360*2*np.pi, np.pi/3.)
```

n en funció de la longitud d'ona per al Mercuri:

	Lambda	**n**
0	5790.0	1.84150+/-0.00031
1	5769.6	1.84192+/-0.00019
2	5460.7	1.84672+/-0.00028
3	4916.0	1.85762+/-0.00023
4	4358.3	1.87478+/-0.00016
5	4078.2	1.8871+/-0.0004
6	4046.6	1.88875+/-0.00014

n en funció de la longitud d'ona per al Cadmi:

	Lambda	**n**
0	6438.5	1.83366+/-0.00016
1	5155.1	1.85203+/-0.00016
2	5085.8	1.85382+/-0.00022
3	4799.9	1.86041+/-0.00015
4	4678.2	1.86405+/-0.00018
5	4413.0	1.87277+/-0.00028



3.1 Càlcul del nombre d'Abbe

Ara podem passar a calcular el paràmetre d'Abbe, que en teoria ens hauria de dir coses interessants sobre quin tipus de vidre tenim. La equació del paràmetre d'Abbe és:

$$\nu_{abbe} = \frac{n_{groc} - 1}{n_{blau} - n_{vermell}} \tag{2}$$

In [83]:

```
def nu_abbe (ny, nb, nr): return (ny-1)/(nb-nr)
```

ens cal, però una definició precisa de quines són les longituds d'ona adequades per al "vermell", "blau" i "verd" comentats al guió. Una petita cerca a la literatura [1] ens mostra que de fet la equació mereix ser escrita com:

$$\nu_{abbe} = \frac{n_D - 1}{n_F - n_C} \tag{3}$$

on D, F, i C indiquen frequències conegudes, concretament:

De fet, no són les úniques definicions trobades en la literatura [1]. Entre d'altres, podem trobar el subíndex D canviat pels subíndexs d o e, variant lleugerament els valors de la frequència associada.

Els valors de les longituds d'ona per als subíndexs utilitzats són:

- λ_D : 5893 $\AA$
- λ_F : 4861.3 $\AA$
- λ_C : 6562.7 $\AA$

In [84]:

```
LD = 5893
LC = 6562.7
LF = 4861.3
############
nD = n(delta_hartman(LD, *hartman_params)*2*np.pi/360, np.pi/3)
nC = n(delta_hartman(LC, *hartman_params)*2*np.pi/360, np.pi/3)
nF = n(delta_hartman(LF, *hartman_params)*2*np.pi/360, np.pi/3)
```

Els valors dels índexs de refracció a les longituds demanades són per tant:

- n_D: $1.84003 \pm 0.00009\ \mathring{A}$
- n_F: $1.85896 \pm 0.00007\ \mathring{A}$
- n_C: $1.83255 \pm 0.00015\ \mathring{A}$

A partir d'aquí podem calcular el valor de l'índex d'Abbe, i obtenim:

In [86]:

```
nu_abbe(nD,nF,nC)
```

Out[86]:

```
31.80660749479406+/-0.214857910050373
```

Codi per a generar les taules:

In [87]:

```
f = lambda x:'${:L}$'.format(x)
r1 = pd.DataFrame({"lambda"       :       hg_lambda,
                   "delta"        : map(f,hg_delta),
                   "interpminsq"  : map(f,hg_leastsq_lam),
                   "interpguio"   : map(f,hg_guio_lam),
                   "n"            : map(f,hg_n),
                  })
r1.to_csv("OUT/TBL/hg.csv")

f = lambda x:'${:L}$'.format(x)
r1 = pd.DataFrame({"lambda"       :       cd_lambda,
                   "delta"        : map(f,cd_delta),
                   "interpminsq"  : map(f,cd_leastsq_lam),
                   "interpguio"   : map(f,cd_guio_lam),
                   "n"            : map(f,cd_n),
                  })
r1.to_csv("OUT/TBL/cd.csv")
```

4 Referències:

[1] http://glassproperties.com/abbe_number/ (http://glassproperties.com/abbe_number/)