

Stefan Nellen

Bitácora

20111120

## Mediendo la constante de Planck

7:31

Reunión con prof. y ayudante.

- > Discutimos experimento
- > Tenemos material para hacerlo en casa.

- LEDs

- 2 multímetros

(IsoTronic, Radioshack)

- Resistencias de carbón

- 9V (22x44)

10:14

-> En el taller

Encuentramos artículo brasileño muy completo.

-> Sugiere usar buffer o convertidor de corriente a voltaje.

V<sub>LED</sub> vs I<sub>LED</sub>, (páng. mut. time exp.)



-> Fuente?

LM742

Fuente opamp  
2 pilas AA en serie, 2 veces.  
12:22

Está demandado bajó.

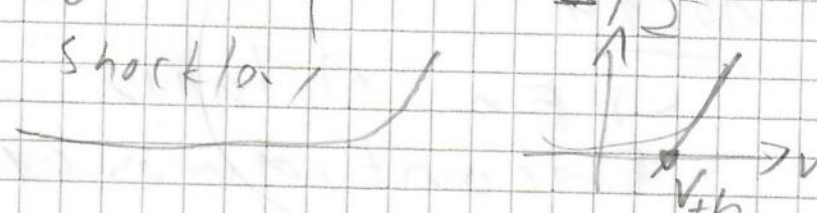
Fuente opamp  $\rightarrow$  (ampas vieja)  
 $\pm 12V$ .

Discontinuo senso de bufer.

$\rightarrow$  Preguntar a profesor y mónica.

Teoría:

Queremos  $I = I_0 e^{-\frac{eV_{th}}{nkT}} \left( e^{\frac{eV}{nkT}} - 1 \right)$   
Shockley



S:

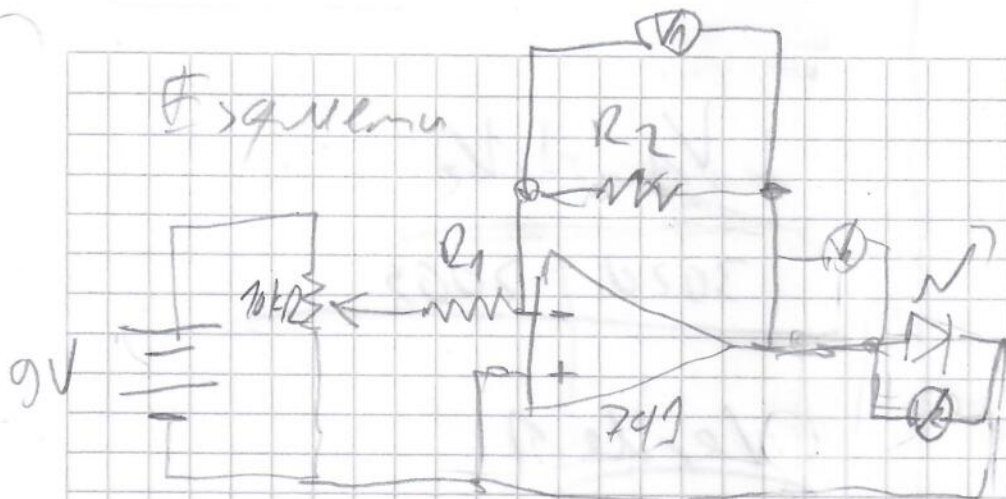
Ajustar recta suponiendo que  
se prende con  $V > V_{th}$ .  
 $eV_{th} = E_{emitida} = h\nu \Rightarrow V_{th} = \frac{h\nu}{e}$

Experimento

$\Rightarrow$  midiendo  $V_{th}$  varias veces de ajuste lineal  
a  $m(V - V_{th}) \rightarrow$  De estas med. graficar  $\frac{1}{m}$  de ajuste.  
(lineal)  $\frac{1}{m}$  de ajuste.



25/11/20



$$R_1 = 102.0 \pm 0.05 \Omega$$

$$R_2 = 102.7 \pm 0.05 \Omega$$

Proj 2

$V_1$ (mV)	$V_2$ (mV)	$V_1$	$V_2$
4.4	10.0	29.2	15.80
9.6	336.6	37.1	16.02
9.3	228.8	49.6	16.16
10.5	544	64.5	16.42
22.9	794	71.2	16.48
13.0	895	78.3	16.53
15.0	1255	81.5	16.56
18.5	1306	91.3	16.63
22.2	1545	99.7	16.68

$V_1$	$V_2$
117.7	1675
129.5	1684
141.2	1656
160.8	1698
188	1709
233	1722
287	1776
229	1735
324	1746
396	1761
471	1775
631	1803
515	1783
777	1845
834	1834
920	1843
1165	1873
1430	1904

Se satwika

$V_1$	$V_2$
3020	2069

Verde 1

$V_1$ [mV]	$V_2$ [mV]
4.3	-10.0
8.8	160.7
9.3	277.0
10.5	460
11.5	650
13.7	927
14.2	1130
15.2	1375
16.2	1508
19.1	1652
24.3	1737
23.5	175
27.4	1768
32.7	1783

Scribe



$V_1$	$V_2$	$V_1$	$V_2$
45.0	1800	678	2026
50.8	1876	916	2026
58.2	1823	2.89	2.024
63.6	1830	se saturó	
65.2	1838		
70.7	1841		
88.9	1851		
95.2	1855		
110.5	1865		
122.3	1874		
137.7	1879		
157.6	1886		
172.5	1894		
195	1903		
214	1911		
259	1926		
365	1950		
462	1981		
513	2007		

29.17, 20

Hasta un certo.

Se voló el LED amarillo

curios el error x quiza

se tenga que volver a medir

todo

Amarillo

$V_1$ [mV]	$V_2$ [mV]
55.6	53.5
59.6	57.5
62.0	60.7
78.2	76.7
80.8	78.8
85.5	83.6

# Rojo 2

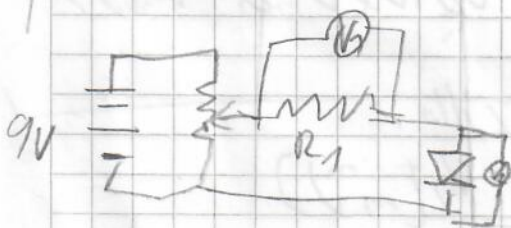
$V_1$	$V_2$	$V_1$ (mV)	$V_2$ (mV)
92.0	84.1	3.1	1489
96.1	94.4	1.5	1462
99.5	97.8	0.3	1397
108.4	106.0	14.3	1552
115.8	114.6	18.3	1568

15:35 /

Error extraño  
en opamp

Se quemaron  
dos diodos

Nuevo circuito:



$R_1 = 700.2$

(El error

fue que  
la salida del  
opamp es  
negativa.

No queremos

operación inversa de (Scribe LEP) -

25.6	1552
41.8	1602
50.4	1616
57.4	1621
63.5	1627
68.8	1631
76.0	1637
83.7	1647
96.3	1698
104.0	1653
121.3	1661
147.2	1670
155.3	1675
174.1	1681



		Verde 2			
$V_1$	$V_2$	$V_1 [mV]$	$V_2 [mV]$	$V_1$	$V_2$
232	1707	0.2	1590	459	1979
270	1712		1702	730	2044
342	1728	2.5		1129	2116
438	1748	8.7	1751 ?	1378	2153
541	1767	17.7	1775	834	2052
676	1789	26.6	1796	555	2009
843	1812	36.2	1811	1167	2123
945	1825	46.2	1827	se saturou	
1202	1855	51.6	1828	1430	2160
1400	1880	63.5	1838	1370	2152
1690	1912	72.1	1845	se saturou	
2020	1944	81.9	1852	5290	2612
se saturou		96.7	1862	4860	2740
5290	2272	105.4	1866		
5790	2319	113.2	1870		
		140.7	1883		
		164.5	1893		
		205	1910		
		260	1928		
		326	1946		

Todo son mv

Amarillo 2

$V_1$ [mV]	$V_2$ [mV]	$V_1$	$V_2$	$V_1$	$V_2$	$V_1$	$V_2$
0.5	1536	396	1697	36.2	2541	1179	3080
1.2	1574	550	1910	42.2	2554	Se soft u/v	
4.4	1626	666	1932	52.2	2573	4560	3543
8.1	1653	807	1955	63.2	2592	4700	3493
12.6	1673	945	1977	72.2	2602	ER	
20.0	1694	1184	2016	77.2	2614		
24.3	1703	1412	2050	93.5	2637	$V_1$	$V_2$
34.3	1710	1670	2080	110.2	2658	0.5	879
40.4	1728	4890	2510	122.2	2672	3.2	892
49.9	1740	5540	2587	137.4	2688	10.8	551
60.2	1750	A2 u12		161.7	2712	17.3	976
70.2	1758	$V_1$	$V_2$	214	2754	22.8	989
84.0	1767	0.7	2228	283	2804	37.6	1004
91.6	1773	6.5	22120	410	2972	51.8	1026
121.2	1790	2.0	2330	450	2990	66.1	1035
136.5	1798	3.4	2335	528	2927	77.8	1043
157.3	1807	9.0	2447	626	2959	89.3	1052
193.6	1817	16.7	2479	716	2984	100.7	1057
206	1824	22.7	2504	824	3015	118.2	1063
267	1845	37.6	2528	987	3053	147.7	1073

Scribe



$\sigma \sim 10\%$

1 10

$V_1$	$V_2$	$V_1$	$V_2$	Verde 1		$V_1$	$V_2$
193.4	1083	30.0	2692	1808	198.4		
272	1088	40.6	2666	1845	435		
250	1097	50.8	2683				
333	1104	62.2	2700				
434	1113	69.6	2705				
502	1126	75.7	2720				
706	1133	91.4	2733				
828	1139	103.7	2743				
1206	1159	121.8	2758				
1408	1161	143.4	2775				
		168.5	2788				

Demasiado sensible

! I have more mediantes observados cuando se enciende.

Saturación

6130 1243

6730 1247

3V 92.20  
Nuevo diseño

Verde 2

$V_1$   $V_2$

3.4 2489

9.7 2550

15.0 2599

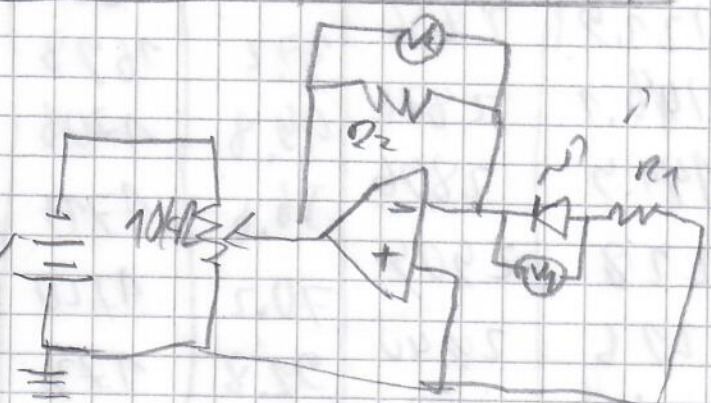
23.6 2621

$R_2 = 800.2 \Omega$

$R_1 = 102.0 \Omega$

(Experimento  
mucho otros)

pero no  
suficiente  
medida



Idea: Invertir circuito



o siempre min. escala

$R = 100 \Omega$

Regresando

42412



$V_2$ (mV)	$V_1$ (mV)	$V_1$	$V_2$	$V_1$	$V_2$	
0.4		687	3095	1699	1753	18:56
19.1						
2510	1.7	759	3052	217	1764	Se satura
2670	9.3	1105	3123	322	1781	de mas
2720	20.9	80	3123	413	1793	rápido
		3040	3355	537	1807	con el
$V_1$	$V_2$	4560	3526	635	1820	circuito 2.
32.7	2756			916	1840	Med: con
43.7	2777	Rojó 2		1191	1855	directamente
		$V_1$	$V_2$	4487	1870	cuando se
50.8	2789	0.5	1561			dispara
60.8	2802	9.5	1631			o se enciende
80.3	2823	13.9	1664	5960	2039	el diodo.
100.0	2847	25.4	1685			
121.5	2856	37.2	1698			
148.7	2872	49.8	1708			
191.5	2896	56.6	1713			
278	2909	70.2	1720			
296	2940	92.8	1730			
387	2970	115.3	1738			
540	3012	140.8	1746			

El riesgo  
es medir  
el voltaje  
de saturación  
en vez del  
voltaje  
umbral.



19:08

mediciones directas

	Ven (mV)
Rojos 1	$2030 \pm 50$
Rojos 2	$1960 \pm 50$
Verdes 1	$2420 \pm 50$
Verdes 2	$2430 \pm 50$
Amarillo	$2160 \pm 50$
Azul 1	$2720 \pm 50$
Azul 2	$2850 \pm 50$
IR	$1150 \pm 50$

3. 12. 20

Andrés dice que tomar la freq.  
como todo el espectro nada bueno  
resultados.

20:17

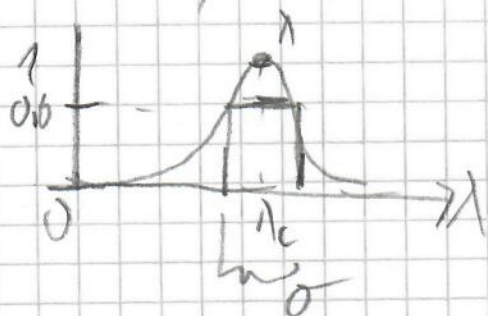
→ Encontré tabla con frecuencias  
de LEN y código de color.

→ Tiene curvas para espectro de longitudes de onda.

20:22

Coinciden con repetidos colores.  
para  $\delta$ .

Hay curvas para cada  $\lambda$  de la fuente



aproximar a  $\exp\left(-\left(\frac{\lambda - \lambda_c}{\sigma}\right)^2\right)$   
(no está normalizada)

Así si  $\lambda - \lambda_c = 0 \Rightarrow \exp\left(-\frac{0^2}{\sigma^2}\right) = \frac{1}{\sigma}$

El error es el ancho de la curva a  $\frac{1}{\sigma} = 0.607$

Con esto se obtiene

LED	$\lambda$ (nm)	LED	$\lambda$ (nm)
Roj01	$657 \pm 15$	Azul1	$470 \pm 27$
Roj02	$657 \pm 15$	Azul2	$430 \pm 35$
Verde1	$525 \pm 27$	Infrarojo	
Verde2	$525 \pm 27$		
Amarillo	$583 \pm 16$		



# Bitácora Práctica 3

Diodos: R, V, Amarillo, Cyan, Azul, Infrarojo  
 dos dos dos "Azules distintos"

Ajuste lineal:  $y = m(x - x_0)$  ← El ajuste a  $x_0$  será  $V_{s, sat}$

$$(\sigma_y)^2 = \left(\frac{\partial y}{\partial m}\right)^2 \sigma_m^2 + \left(\frac{\partial y}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial y}{\partial x_0}\right)^2 \sigma_{x_0}^2 = (x - x_0)^2 \sigma_m^2 + m^2 (\sigma_x^2 + \sigma_{x_0}^2)$$

$$y = I, \quad x = V_1$$

$$I = V_2 / R \Rightarrow \sigma_I^2 = \frac{V_2^2}{R^4} \sigma_R^2 + \frac{\sigma_{V_2}^2}{R^2}$$

Para ajuste a  $h$ :

$$E = h\nu, \quad V = E/q, \quad \nu = c/\lambda \Rightarrow V = \frac{h\nu}{q}$$

Para generalizar el ajuste tomamos

$$q, c \text{ se toman como exactas: } q = 1.602176634 \times 10^{-19} \text{ C}$$

$$c = 299792458 \text{ m/s}$$

$$V = \frac{h\nu}{q} + V_0$$

$$\Rightarrow \sigma_V^2 = \frac{V^2}{q^2} \sigma_h^2 + \frac{h^2}{q^2} \sigma_\nu^2, \quad \sigma_V = \frac{1}{q} \sqrt{(V\sigma_h)^2 + (h\sigma_\nu)^2}$$

$$\sigma_V^2 = \frac{c^2}{\lambda^4} \sigma_\lambda^2, \quad \sigma_V = \frac{c}{\lambda^2} \sigma_\lambda = V \frac{\sigma_\lambda}{\lambda}$$

$$\sigma_V = \sqrt{\left(\frac{V\sigma_h}{q}\right)^2 + \left(\frac{h\sigma_\nu}{q}\right)^2 + \sigma_{V_0}^2}$$

Espectro visible:

Propuesta:

Rojos: 625-700 nm	$\Rightarrow \lambda = 662.5 \pm 37.5 \text{ nm}$
Amarillo: 565-590 nm	$\lambda = 577.5 \pm 12.5 \text{ nm}$
Verde: 500-565 nm	$\lambda = 532.5 \pm 32.5 \text{ nm}$
Cyan: 485-500 nm	$\lambda = 492.5 \pm 7.5 \text{ nm}$
Azul: 450-485 nm	$\lambda = 467.5 \pm 17.5 \text{ nm}$

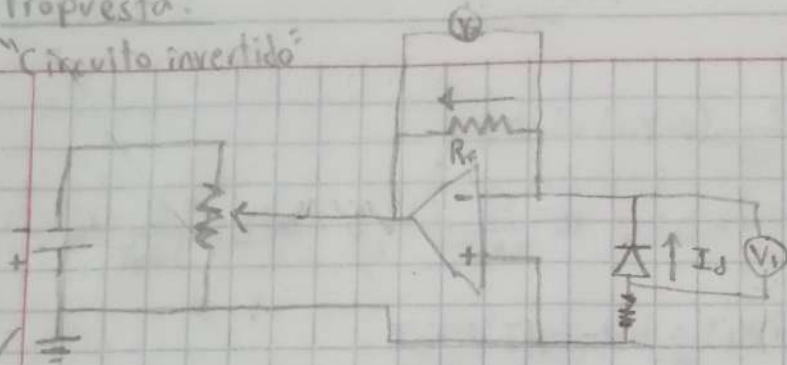
Rango máximo en el que se puede poner la longitud de onda

Nueva propuesta: Gráfica de espectro de diodos

Propuesta:

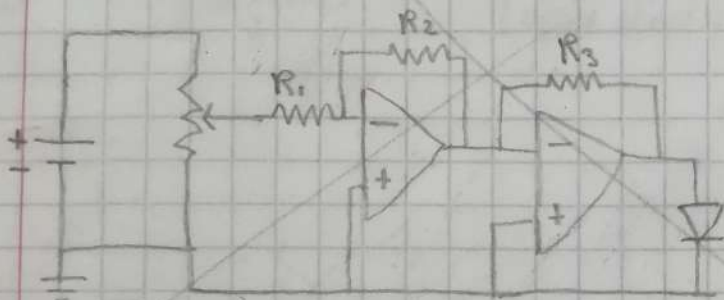
$$V_2 = -I_D R_f$$

"Circuito invertido"



No estamos seguros  
de las polarizaciones  
Si están bien  
las polaridades ✓

Propuesta:



Assaziogramen

Arbeits B

Avances bitáron  
gramático  
Vi 11.

$$R_1 = 100.2 \Omega$$

Experimento 1: "Muy sensible"

Experimento 2:

	$V_{sat} [V]$ ← Cuando enciende
Rojos 1	2.03
Verde 1	2.42 ±1
Amarillo 1	2.16
Azul 1	2.72
IR	1.19
Rojos 2	1.96
Verde 2	3.18 (?!)
Azul 2	2.89

	$V_{sat} [V]$
Rojos 1	1.79
Verde 1	2.01
Amarillo 1	1.91
Azul 1	2.75
IR	1.12
Rojos 2	1.82
Verde 2	2.43
Azul 2	2.75

$$C = \lambda V$$

Lado P, Lado N

erzeugen = generar

Gesch erhöhen



$$I = I_0 e^{e(V-V_{th})/nK_B T}, \quad \ln(I) = \frac{e}{nK_B T} (V - V_{th}) + \ln(I_0)$$

$$\text{ie } \ln(I) = mV + b, \quad m = \frac{e}{nK_B T}, \quad b = \ln(I_0) - \frac{e}{nK_B T} V_{th}$$

de ahí:  $Y(V) = -be/m = eV_{th} - \ln(I_0) nK_B T = h\nu - \ln(I_0) nK_B T$

$\uparrow E = eV_{th} = h\nu \uparrow$  ordenada cambia en cada diodo!  
(inviabile)

Depurado  $R_{01a}$  y  $V_{e1}$  (circuito viejo)

Ent.  $R_{01b}, R_{02} \rightarrow R_{01}, R_{02}$

$V_{e2}, V_{e3} \rightarrow V_{e1}, V_{e2}$

Proc.: determino

Carátula Presentación:

Univ. Escuela

Fac. Título. Nombre completo

Solo 1 par de gráficas  $I_d - V_d$

UNAM  $\rightarrow$  Grande

Fac.  $\rightarrow$  Grande

Infrarojo  $\rightarrow$  IR

$$M = h/e$$

$I_d(V_d)$ , lo relevante es  $V_{th}$

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import os #Para adquirir el path global y guardar figuras
```

```
In [39]: h_real = 6.62607015e-34 #J*s
e = 1.602176634e-19 #C
c = 299792458 #m/s
#R = 102.7 #Ohm En Los primeros dos experimentos
R = 100.2 #Ohm
sR = 0.05
M_real = h_real/e

def y(x,m,x0): #Modelo Lineal general para ajustar
    return m*(x-x0)

def sy(x,sx,m,sm,x0,sx0):
    return np.sqrt(m**2*(sx**2+sx0**2)+sm**2*(x-x0)**2)

#En deshuso
#def V(nu,M,V0): #Modelo Lineal para ajustar h, q exacta. M = h/q
#    return M*nu+V0
#
#def sV(nu,snu,M,sM,V0,sV0):
#    return np.sqrt((nu*sM)**2+(M*snu)**2+sV0**2)

def V(nu,M): #Modelo Lineal para ajustar h, q exacta. M = h/q
    return M*nu

def sV(nu,snu,M,sM):
    return np.sqrt((nu*sM)**2+(M*snu)**2)
```

```
In [3]: #Lambdas directo del espectro visible
lamb_rojo, slamb_rojo = (625+700)/2, (700-625)/2
lamb_verde, slamb_verde = (500+565)/2, (565-500)/2
lamb_amarillo, slamb_amarillo = (565+590)/2, (590-565)/2
#lamb_cyan, slamb_cyan = (485+500)/2, (500-485)/2
lamb_azul, slamb_azul = (450+485)/2, (485-450)/2
lamb_IR, slamb_IR = 940, 940-760
```

```
In [3]: #Lambdas y desviaciones estándar estimadas de La gráfica técnica
lamb_IR, slamb_IR = 945, 22 #Infrared (1)
lamb_rojo, slamb_rojo = 657, 15 #Se tomó el denominador rojo, pero hay otros to
nos de rojo posibles
lamb_verde, slamb_verde = 525, 27 #Ultra Green (A)
lamb_amarillo, slamb_amarillo = 583, 18 #Yellow
lamb_azul, slamb_azul = 430, 35 #Ultra Blue
lamb_cyan, slamb_cyan = 470, 27 #Pure Blue
```



```
In [77]: #Con cyan
L = ["Ro1", "Ro2", "Ve1", "Ve2", "Am1", "Azul1", "Azul2", "IR"]
Nombre = ["rojo 1", "rojo 2", "verde 1", "verde 2", "amarillo", "azul 1", "azul 2", "infrarrojo"]
Lamb = np.array([lamb_rojo, lamb_rojo, lamb_verde, lamb_verde, lamb_amarillo, lamb_cyan, lamb_azul, lamb_IR])*1e-9 #m
sLamb = np.array([slamb_rojo, slamb_rojo, slamb_verde, slamb_verde, slamb_amarillo, slamb_cyan, slamb_azul, slamb_IR])*1e-9 #m
V1th = [0,0,0,0,0,0,0,0]
sV1th = [0,0,0,0,0,0,0,0]
```

## Método 1

```
In [5]: def Read(i):
    Datos = np.loadtxt("Datos"+L[i]+".txt", skiprows=1).T
    Datos = Datos[:, Datos[0].argsort()] #Ordena datos

    LOW = [25,16,19,10,20,23,10,21]#indice menor
    low = LOW[i] #indice menor
    ind = np.arange(low, len(Datos[0])) #Datos a ajustar desde low hasta el máximo
    ind_extra = np.arange(0, low)
    d_ajustados = len(Datos[0]) - low

    V1 = Datos[1][ind]
    V2 = Datos[0][ind]
    sV1 = Datos[3][ind]
    sV2 = Datos[2][ind]
    Ic = V2/R
    sIc = np.sqrt(V2**2*sR**2/(R**4)+(sV2/R)**2)

    V1_extra = Datos[1][ind_extra]
    V2_extra = Datos[0][ind_extra]
    sV1_extra = Datos[3][ind_extra]
    sV2_extra = Datos[2][ind_extra]
    Ic_extra = V2_extra/R
    sIc_extra = np.sqrt(V2_extra**2*sR**2/(R**4)+(sV2_extra/R)**2)

    return V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados
```

```
In [66]: #Rojo1
i = 0
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popl
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))

m_fit = 0.08545421947741375 +- 0.0011276096106633958
x0_fit = 1713.9598265831926 +- 1.5476839387967183
```

```
In [96]: #Rojo2
i = 1
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popl
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))

m_fit = 0.16156529172551726 +- 0.011506967589816156
x0_fit = 1780.8445484790097 +- 4.464394427173474
```

```
In [99]: #Verde1
i = 2
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popl
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))

m_fit = 0.04932791908859819 +- 0.0007583532266485239
x0_fit = 1888.3127232716035 +- 2.507666460481306
```

```
In [102]: #Verde2
i = 3
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popl
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))

m_fit = 0.013446659584589954 +- 0.0006951073478863978
x0_fit = 2666.230174816681 +- 4.659784290695643
```



```
In [85]: #Amarillo1
i = 4
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popl
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))

m_fit = 0.060954826949696575 +- 0.0009134918552428922
x0_fit = 1822.0942188119 +- 1.974902156291858
```

```
In [87]: #Azul1
i = 5
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popl
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))

m_fit = 0.039416306341173726 +- 0.0014936905064095037
x0_fit = 2801.8917644478333 +- 8.075792756863747
```

```
In [89]: #Azul2
i = 6
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popl
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))

m_fit = 0.04324573968167225 +- 0.0026229092360033133
x0_fit = 2878.9018180118096 +- 8.698030044131542
```

```
In [91]: #IR
i = 7
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popl
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))

m_fit = 0.5937737750075394 +- 0.08962577526595315
x0_fit = 1132.8137087298692 +- 4.432573118138958
```

```

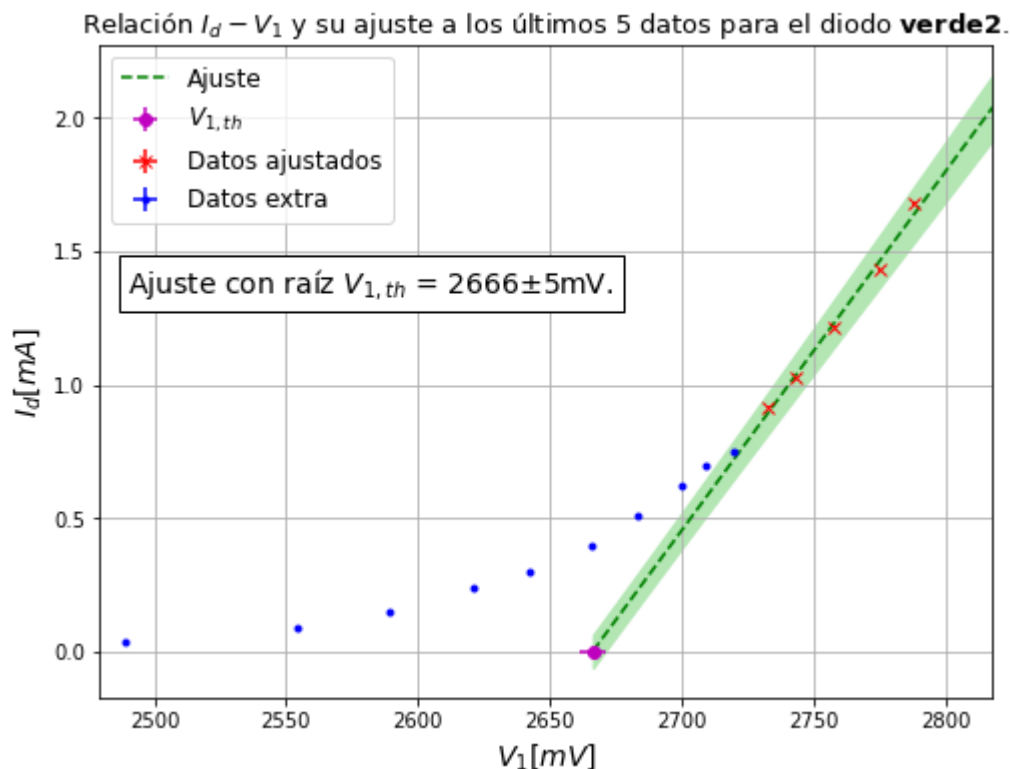
In [106]: #Gráfica individual
V_cont = np.linspace(x0_fit,V1.max()+30,100)

fig = plt.figure(figsize=(8,6))
plt.errorbar(x0_fit,0,0,sx0_fit,"mo",label=r"$V_{1,th}$")
plt.errorbar(V1,Ic,sIc,sV1,"rx",label="Datos ajustados")
plt.errorbar(V1_extra,Ic_extra,sIc_extra,sV1_extra,"b.",label="Datos extra") #
Datos extra que fueron medidos, pero no ajustados
y_fit = y(V_cont,m_fit,x0_fit)
plt.plot(V_cont,y_fit,"g--",label="Ajuste")
sy_fit = sy(V_cont,0,m_fit,sm_fit,x0_fit,sx0_fit)
plt.fill_between(V_cont,y_fit+sy_fit,y_fit-sy_fit,color=(0.7,0.9,0.7))
plt.title(r"Relación $I_d - V_1$ y su ajuste a los últimos "+str(d_ajustados)+
" datos para el diodo "+ r"$\bf{"+Nombre[i]+ "}$.",fontsize=13)
plt.xlabel(r"$V_1[mV]$",fontsize=14)
plt.ylabel(r"$I_d[mA]$",fontsize=14)
plt.xlim(V1_extra.min()-10,V1.max()+30)
POSX = [1400,1560,1600,2490,1550,2050,2755,830] #DatosR1a, DatosR1b , DatosR
2...
POSY = [14,13,10,1.35,12,8,8,60] #DatosR1a, DatosR1b, DatosR2...
posx , posy = POSX[i], POSY[i]
props=dict(boxstyle='square', facecolor='white', alpha=1)
plt.text(posx, posy, r"Ajuste con raíz $V_{1,th}$ = "+"{: .0f}".format(x0_fit)+
"$\pm$"+"{: .0f}".format(sx0_fit)+"mV.",
        fontsize=14, bbox=props)
plt.legend(fontsize=12)
plt.grid()
plt.show()

#Para guardar la gráfica
fig.savefig("C:\Andrés\IX Semestre\Lab. Contemporánea I\Práctica 3\Plots\ "[:-1]+str(i)+".png",bbox_inches='tight') #El[:-1] es para quitar el espacio, nec
esario para que se comente bien por el \

```





```
In [52]: #Recopilación de voltajes de Vth
for i in range(0,8):
    V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados =
    Read(i)
    popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
    m_fit, x0_fit = popt
    sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])

    V1th[i], sV1th[i] = x0_fit*1e-3, sx0_fit*1e-3 #Convertimos a V
    nu = c/Lamb #Hz
    snu = nu*sLamb/Lamb #Hz

print(V1th)
print(sV1th)
```

```
[1.71395983 1.78084455 1.88831272 2.66623017 1.82209422 2.80189176
 2.87890182 1.13281371]
[0.00154768 0.00446439 0.00250767 0.00465978 0.0019749 0.00807579
 0.00869803 0.00443257]
```

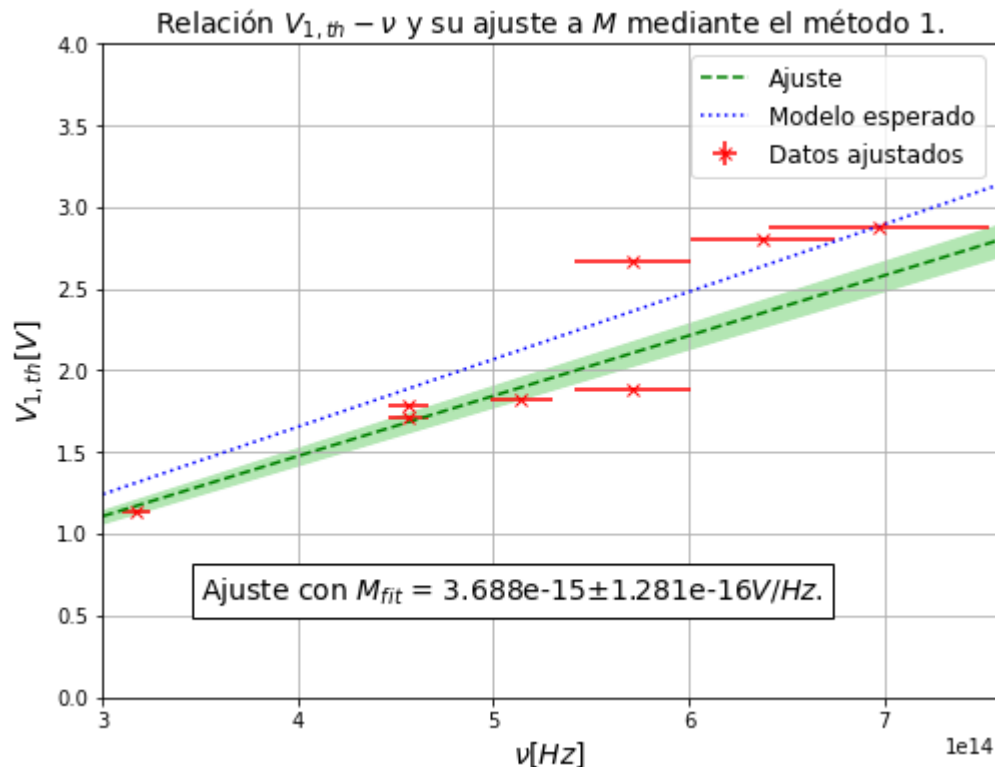
```
In [53]: #Ajuste a h
popt, pcov = curve_fit(V, xdata = nu, ydata = V1th, sigma = sV1th)
M_fit = popt[0]
sM_fit = np.sqrt(pcov[0][0])
h_fit, sh_fit = M_fit*e, sM_fit*e
print("M_fit = "+str(M_fit)+" +- "+str(sM_fit))
print("h_fit = "+str(h_fit)+" +- "+str(sh_fit))
print("h_real = "+str(h_real))
print("Diferencia porcentual a la real: "+"{:.3f}".format(100*np.abs(h_fit-h_r
eal)/h_real)+"%")

M_fit = 3.687859400758826e-15 +- 1.2813465379821185e-16
h_fit = 5.908602161373032e-34 +- 2.0529434832117436e-35
h_real = 6.62607015e-34
Diferencia porcentual a la real: 10.828%
```



```
In [54]: numin, numax = 3e14, nu.max()+0.6e14
nu_cont = np.linspace(numin,numax,100)

plt.figure(figsize=(8,6))
plt.errorbar(nu,V1th,sV1th,snu,"rx",label="Datos ajustados")
y_fit = V(nu_cont,M_fit)
plt.plot(nu_cont,y_fit,"g--",label="Ajuste")
sy_fit = sV(nu_cont,0,M_fit,sM_fit)
plt.fill_between(nu_cont,y_fit+sy_fit,y_fit-sy_fit,color=(0.7,0.9,0.7))
plt.plot(nu_cont,V(nu_cont,M_real),"b:",label="Modelo esperado")
plt.title(r"Relación  $V_{1,th}$  -  $\nu$  y su ajuste a  $M$  mediante el método 1.",
, fontsize=14)
plt.xlim(numin,numax)
plt.ylim(0,4)
plt.xlabel(r" $\nu$  [Hz]", fontsize=14)
plt.ylabel(r" $V_{1,th}$  [V]", fontsize=14)
posx , posy = 3.5e14, 0.6
props=dict(boxstyle='square', facecolor='white', alpha=1)
plt.text(posx, posy, r"Ajuste con  $M_{fit} =$  "+"{: .3e}".format(M_fit)+" $\pm$ "
+"{: .3e}".format(sM_fit)+r" $V/Hz$ .",
        fontsize=14, bbox=props)
plt.legend(fontsize=12)
plt.grid()
plt.show()
```



## Método 2

```
In [55]: #Con cyan
V1th = np.array([2.03,1.96,2.42,2.43,2.16,2.72,2.89,1.19]) #V
sV1th = np.array([0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05]) #V
Lamb = np.array([lamb_rojo,lamb_rojo,lamb_verde,lamb_verde,lamb_amarillo,lamb_
cyan,lamb_azul,lamb_IR])*1e-9 #m
sLamb = np.array([slamb_rojo,slamb_rojo,slamb_verde,slamb_verde,slamb_amarillo
,slamb_cyan,slamb_azul,slamb_IR])*1e-9 #m
nu = c/Lamb #Hz
snu = nu*sLamb/Lamb #Hz
```

```
In [65]: print(snu)
```

```
[1.04179249e+13 1.04179249e+13 2.93674245e+13 2.93674245e+13
 1.58765486e+13 3.66428084e+13 5.67481667e+13 7.38549769e+12]
```

```
In [57]: #Ajuste a h
popt, pcov = curve_fit(V, xdata = nu, ydata = V1th, sigma = sV1th)
M_fit = popt[0]
sM_fit = np.sqrt(pcov[0][0])
h_fit, sh_fit = M_fit*e, sM_fit*e
print("M_fit = "+str(M_fit)+" +- "+str(sM_fit))
print("h_fit = "+str(h_fit)+" +- "+str(sh_fit))
print("h_real = "+str(h_real))
print("Diferencia porcentual a la real: "+"{:.3f}".format(100*np.abs(h_fit-h_r
eal)/h_real)+"%")
```

```
M_fit = 4.22431790349848e-15 +- 4.849292561101135e-17
h_fit = 6.7681034395731315e-34 +- 7.769423232826255e-36
h_real = 6.62607015e-34
Diferencia porcentual a la real: 2.144%
```

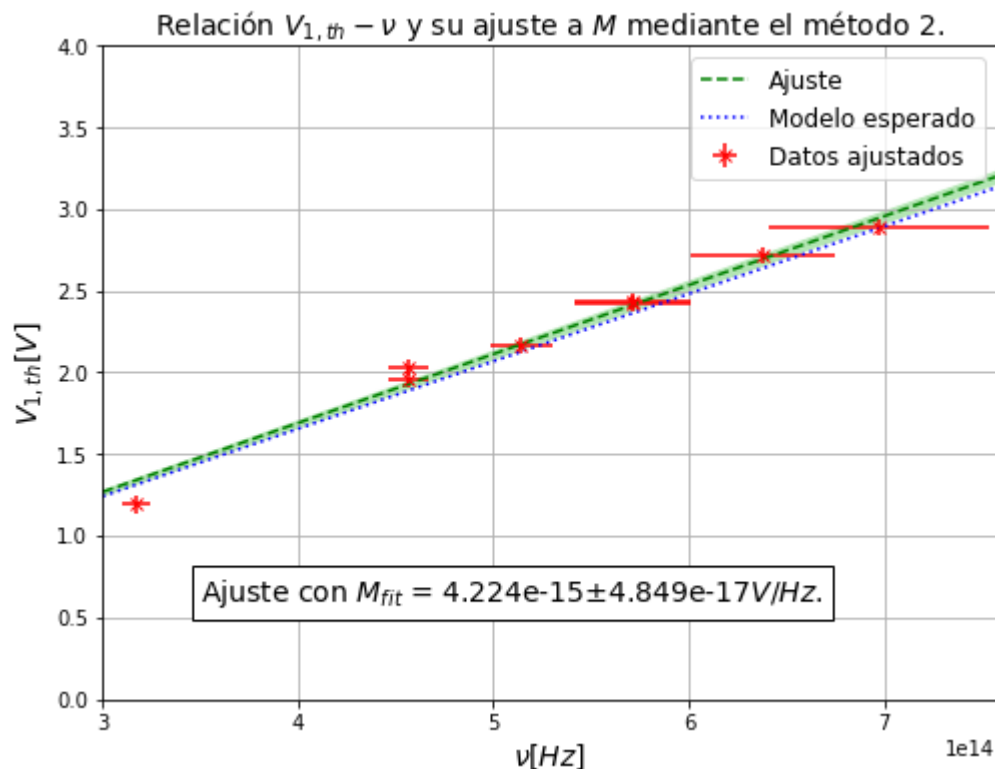


```

In [50]: numin, numax = 3e14, nu.max()+0.6e14
nu_cont = np.linspace(numin,numax,100)

plt.figure(figsize=(8,6))
plt.errorbar(nu,V1th,sV1th,snu,"rx",label="Datos ajustados")
y_fit = V(nu_cont,M_fit)
plt.plot(nu_cont,y_fit,"g--",label="Ajuste")
sy_fit = sV(nu_cont,0,M_fit,sM_fit)
plt.fill_between(nu_cont,y_fit+sy_fit,y_fit-sy_fit,color=(0.7,0.9,0.7))
plt.plot(nu_cont,V(nu_cont,M_real),"b:",label="Modelo esperado")
plt.title(r"Relación  $V_{1,th}$  -  $\nu$  y su ajuste a  $M$  mediante el método 2.")
, fontsize=14)
plt.xlim(numin,numax)
plt.ylim(0,4)
plt.xlabel(r" $\nu$ [Hz]", fontsize=14)
plt.ylabel(r" $V_{1,th}$ [V]", fontsize=14)
posx , posy = 3.5e14, 0.6
props=dict(boxstyle='square', facecolor='white', alpha=1)
plt.text(posx, posy, r"Ajuste con  $M_{fit} =$  "+"{: .3e}".format(M_fit)+" $\pm$ "
+"{: .3e}".format(sM_fit)+r"$V/Hz$.",
        fontsize=14, bbox=props)
plt.legend(fontsize=12)
plt.grid()
plt.show()

```



## Generación de tablas con corrientes

```
In [6]: def Write(i):  
        """  
        Escribe la tabla de datos con V1, V2 e Ic ordenados.  
        """  
        Datos = np.loadtxt("Datos"+L[i]+".txt", skiprows=1).T  
        Datos = Datos[:, Datos[0].argsort()] #Ordena datos  
  
        ind = np.arange(0, len(Datos[0])) #Indice total de datos  
        V1 = Datos[1][ind]  
        V2 = Datos[0][ind]  
        sV1 = Datos[3][ind]  
        sV2 = Datos[2][ind]  
        Ic = V2/R  
        sIc = np.sqrt(V2**2*sR**2/(R**4)+(sV2/R)**2)  
  
        DatosW = np.array([V1, sV1, V2, sV2, Ic, sIc]).T  
        np.savetxt("Datos_tabla_"+L[i]+".txt", DatosW, fmt="%0f\t%.1f\t%.1f\t%.2f\t  
%.2e\t%.2e") #Formato de cada columna, con tabulación
```

```
In [13]: Write(7)
```