Tarea2_Explicacion

September 2, 2021

1 Física Numérica

1.1 Tarea 2

1. **La fórmula cuádratica**

   Recuerde que la para la ecuación cuadrática:

   $$ax^2 + bx + c = 0$$

   tenemos las soluciones:

   $$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

   o también:

   $$x'_{1,2} = \frac{2c}{-b \pm \sqrt{b^2 - 4ac}} \tag{2}$$

   Una rápida inspección muestra que debemos tener cuidado con la cancelación sustractiva cuando $b^2 \gg 4ac$.

   (a) Escriba un programa que calcule las cuatro soluciones para valores arbitrarios de a, b y c.
   (b) Investigue cómo los errores en las respuestas calculadas se vuelven grandes debido a la cancelación sustractiva (Sugerencia: pruebe con $a = 1$, $b = 1$ y $c = 10^{-n}$, $n = 1, 2, 3, ...$

- *Solución*:

   (a) Primeramente asumiremos $a \neq 0$, ya que de forma contraria tendríamos una ecuación de la forma $bx + c = 0$ cuya solución es trivial. De esta manera, realizaremos dos funciones (una para cada tipo de solucón) cuyos parámetros serán los valores de a, b y c, y como resultado nos arrojarán el valor de las raíces dependiendo de la fórmula usada. Las funciones son:

[50]:
```
def raices_1( a:float , b: float, c: float ) -> list:
    """Esta función calcula las raíces de la ecuación cuadrática
    ax^2 + bx + c = 0 usando la fórmula 1 y regresa los valores de las
    raices"""


    x1 = (-b + (b**2 - 4*a*c)**(1/2) )/ (2*a)
    x2 = (-b - (b**2 - 4*a*c)**(1/2) )/ (2*a)
    return x1,x2

def raices_2( a , b, c ) -> list:
    """Esta función calcula las raíces de la ecuación cuadrática
    ax^2 + bx + c = 0 usando la fórmula 2 y regresa los valores de las raices
    en forma de lista"""


    x1 = (-2 * c)/( b + (b**2 - 4*a*c)**(1/2) )
    x2 = (-2 * c)/( b - (b**2 - 4*a*c)**(1/2) )
    return x1,x2
```

Veamos un ejemplo del uso de las funciones. Tomemos la ecuación:

$$x^2 + x + 2 = 0$$

Cuyas raíces son $x_{1,2} = \frac{-1 \pm \sqrt{7}i}{2}$. Usando el programa obtenemos:

[51]:
```
a = 1
b = 1
c = 2
r1,r2 = raices_1(a,b,c)
s1,s2 = raices_2(a,b,c)

print(f"Las raíces usando la fórmula (1) son: x_1 = {r1:<.8} y x_2 = {r2:<.8}")
print(f"Las raíces usando la fórmula (2) son: x'_1 = {s1:<.8} y x'_2 = {s2:<.8}")
```

```
Las raíces usando la fórmula (1) son: x_1 = (-0.5+1.3228757j) y x_2 =
(-0.5-1.3228757j)
Las raíces usando la fórmula (2) son: x'_1 = (-0.5+1.3228757j) y x'_2 =
(-0.5-1.3228757j)
```

Nota: python utiliza j como el número imaginario.

Podemos ver que nuestras funciones definidas obtienen una aproximación adecuada para la ecuación cuadrática que hemos ingresado.

(b) Para este inciso imprimamos en una tabla las raíces calculadas por las fórmulas (1) y (2) para cada valor de *a, b* y *c,* sugerido. Además compararemos los resultados para ver como difieren. Para ello nos apoyaremos de la libreria *PrettyTable* para que la tabla sea visualmente mejor.

[65]:
```
from prettytable import PrettyTable

tabla = PrettyTable() #Define la tabla
tabla.title = "Raíces de la ecuación x^2 + x + 10^(-n) = 0" # Agrega el título a la␣
 ↪tabla
tabla.field_names = ["n","x_1","x'_1","|x_1-x'_1|","x_2", "x'_2","|x_2-x'_2|"]# Le␣
 ↪damos nombre a los compos
n = 16 #valor del exponente máximo de c = 10^(-n)
a = 1
b = 1
for i in range(1,n+1):
    c = 10**(-i)
    r1,r2 = raices_1(a,b,c)
    s1,s2 = raices_2(a,b,c)

    tabla.add_row([i,
                   f"{r1:<.3e}",f"{s1:<.3e}",f"{abs(r1-s1):<.3e}",
                   f"{r2:<.3e}",f"{s2:<.3e}",f"{abs(r2-s2):<.3e}"])#Inserta un␣
 ↪elemento a la tabla

print(tabla)
```

```
+-------------------------------------------------------------------------------
---+
|                    Raíces de la ecuación x^2 + x + 10^(-n) = 0
|
+----+------------+------------+------------+------------+------------+---------
---+
| n  |    x_1     |    x'_1    | |x_1-x'_1| |    x_2     |    x'_2    |
```

```
|x_2-x'_2| |
+----+-----------+-----------+-----------+-----------+-----------+---------
---+
| 1  | -1.127e-01 | -1.127e-01 | 1.388e-17  | -8.873e-01 | -8.873e-01 |
1.110e-16  |
| 2  | -1.010e-02 | -1.010e-02 | 2.082e-17  | -9.899e-01 | -9.899e-01 |
1.998e-15  |
| 3  | -1.001e-03 | -1.001e-03 | 2.429e-17  | -9.990e-01 | -9.990e-01 |
2.420e-14  |
| 4  | -1.000e-04 | -1.000e-04 | 5.557e-18  | -9.999e-01 | -9.999e-01 |
5.562e-14  |
| 5  | -1.000e-05 | -1.000e-05 | 1.663e-17  | -1.000e+00 | -1.000e+00 |
1.662e-12  |
| 6  | -1.000e-06 | -1.000e-06 | 4.634e-18  | -1.000e+00 | -1.000e+00 |
4.634e-12  |
| 7  | -1.000e-07 | -1.000e-07 | 5.119e-18  | -1.000e+00 | -1.000e+00 |
5.119e-11  |
| 8  | -1.000e-08 | -1.000e-08 | 5.759e-18  | -1.000e+00 | -1.000e+00 |
5.759e-10  |
| 9  | -1.000e-09 | -1.000e-09 | 2.623e-17  | -1.000e+00 | -1.000e+00 |
2.623e-08  |
| 10 | -1.000e-10 | -1.000e-10 | 8.264e-18  | -1.000e+00 | -1.000e+00 |
8.264e-08  |
| 11 | -1.000e-11 | -1.000e-11 | 8.273e-19  | -1.000e+00 | -1.000e+00 |
8.273e-08  |
| 12 | -1.000e-12 | -1.000e-12 | 3.339e-17  | -1.000e+00 | -1.000e+00 |
3.339e-05  |
| 13 | -9.998e-14 | -1.000e-13 | 2.442e-17  | -1.000e+00 | -1.000e+00 |
2.442e-04  |
| 14 | -9.992e-15 | -1.000e-14 | 7.993e-18  | -1.000e+00 | -1.001e+00 |
7.999e-04  |
| 15 | -9.992e-16 | -1.000e-15 | 7.993e-19  | -1.000e+00 | -1.001e+00 |
7.999e-04  |
| 16 | -1.110e-16 | -1.000e-16 | 1.102e-17  | -1.000e+00 | -9.007e-01 |
9.928e-02  |
+----+-----------+-----------+-----------+-----------+-----------+---------
---+
```

Recordemos que para las columnas 4,5,7 y 8 se han usado las ecuaciones (1) y (2). Llamenos raíces positivas a x_1 y x_1' ya que para estas se toma el signo positivo. Mientras que para x_2 y x_2' nos referiremos a ellas como raíces negativas.

Primeramente notemos que para $n < 11$ obtenemos valores muy cercanos para las raíces obtenidas por ambos métodos. Sin embargo, a partir de $n = 12$ la diferencia empieza a ser notoria. Esto se de que a que para la raiz x_1 tenemos la fórmula:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Y para los valores que hemos tomado, a partir de $n = 12$ podemos hacer la suposición de que $b^2 \gg 4ac$. Por lo que $\sqrt{b^2 - 4ac} \approx |b|$. De esa forma tendríamos:

$$x_1 \approx \frac{-b + |b|}{2a}$$

Y como b>0, entonces el numerador sería la recta de dos números muy cercanos entre sí. Por lo que el numerador tendría el problema de que la computadora lo pueda interpretar como cero. Por otro lado, para x_1' tenemos la ecuación $x_1' = frac{-2cb + \sqrt{b^2 - 4ac}}$ y tomando $b^2 \gg 4ac$ obtenemos la aproxiamción:

$$x_1' \approx \frac{-2c}{b + |b|}$$

En este caso no se tendría ese problema, debido a que esta vez se suman los números y no se restan.

Ahora, si analizamos a x_2 y x_2' tenemos un caso parecido. Solo que esta vez las aproximaciones para la suposición de que $b^2 \gg 4ac$ reultan:

$$x_2 \approx \frac{-b-|b|}{2a} \quad ; \quad x_1' \approx \frac{-2c}{b-|b|}$$

En x_2 no hay tanto problema para los valores que consideramos. Sin embargo, para x_2' ocurre lo mismo que con x_1. Solo que esta vez este número cercano a cero está dividiendo, esto ocasionaria que nuetros resultados comiencen a diverger y nos den raices erróneas o simplemente el programa marque error al intentar dividir por un número que la computadora interpreta como cero.

Por estas razones, las diferencias se vuelven cada vez más grandes. Ocasionando que nuestros errores sean mayores.

2. **Funciones de Bessel esféricas**

   (a) Escriba un programa que utilice las fórmulas de recursión hacia arriba (*up*) y hacía abajo (*down*) para calcular $j_l(x)$ para los primeros 25 valores de l para $x = 0.1, 1, 10$.

$$j_{l+1}(x) = \frac{2l+1}{x} j_l(x) - j_{l-1}(x), \quad up \tag{3}$$

$$j_{l-1}(x) = \frac{2l+1}{x} j_l(x) - j_{l-+}(x), \quad down \tag{4}$$

   (b) Ajsute su programa para que al menor un método de "buenos" valores (error relativo de 10^{-10}.

   (c) Compare los metodos con las distintas fórmulas de recursión, imprimiendo una tabla con columnas: l, j_l^{up}, j_l^{down} y $\frac{|j_l^{up} - j_l^{down}|}{|j_l^{up}| + |j_l^{down}|}$.

- *Solución*:

   (a) Primeramente exportamos las librerías que requerimos:

[53]:
```
from collections import deque
from numpy import sin, cos
from scipy.special import spherical_jn
from prettytable import PrettyTable
```

De la librería *collections* se importó *deque* que es como una lista que permite controlar de mejor manera el insertar elementos al inicio o al final de la misma. De *numpy* las funciones sin y cos que se ocuparan para los valores de $j_0(x)$ y $j_1(x)$. De *scipy* se importaron las funciones de Bessel esféricas para comparar nuestro programa con estas. Finalmete de *prettytabble* solo la usaremos para hacer nuestras tablas visualmente mejor.

Para resolver esta parte se crearon 2 funciones. La primer función calcula el valor numérico de $j_l(x)$ para un l y x dados como parámentros, usando la relación de recurrencia *up*. Mientras que la segunda función tiene el mismo objetivo, pero en lugar de usar la relación *up*, utiliza la relación *down* apoyándose del algoritmo de Miller. Las funciones creadas fueron las siguientes:

[54]:
```
def bessel_esf_up(l: int, x: float) -> float:
    """Esta función utiliza una relación de recurrencia para calcular el valor
    numérico de la función de bessel esférica siguente a partir de las dos
    anteriores"""


    j = [sin(x)/x, sin(x)/x**2 - cos(x)/x]
```

```
    for i in range (1,l):
        j_sig = (2*i+1)/x * j[i] - j[i-1]
        j.append(j_sig)

    return j[l]

def bessel_esf_down(l: int, x = 1) -> float:
    """Esta función utiliza el algoritmo de Miller para obtener obtener
    los valores numéricos de las primeros l<50 funciones de Bessel esféricas"""

    n = 25
    j = deque([1,2])#En la posición 0 siempre tendremos al j_99 y en 1 al j_100


    for i in range(0,n):
        j_ant = (2*(n - i) + 1)/x * j[0] - j[1]
        j.appendleft(j_ant)

    normalizacion = (sin(x)/x)/j[0]
    j[l] = j[l]*normalizacion
    return j[l]
```

En la función *bessel_esf_up* lo que se hace es crear una lista, donde el elemento 0 es $j_0(x) = \frac{\sin x}{x}$ y el elemento 1 es $j_1(x) = \frac{\sin x}{x^2} - \frac{\cos x}{x}$. Luego de ello, con un ciclo *for* corremos un contador que va de 1 hasta el parámentro dado l, el cual se usará para calcular la función $j_l(x)$ usando la relación *up*. Todos los valores calculdos se irán guardando en la lista. De forma que al final de iterar el ciclo for, el elemento l de la lista será el valor del $j_l(x)$ buscado.

Por otro lado, en la función *bessel_esf_down* se crea una *deque* que es como una especie de lista. En esta especie de lista ingresamos los valores de $j_{25}(x) = 1$ y $j_{26}(x) = 2$ en las posiciones 0 y 1. Estos valores se tomaron arbitrariamente para aplicar el algoritmo de Miller. Después de ello, con un ciclo *for* que va de 1 a 24 usamos nuestra relación de recurrencia *down*, la cual se ha ajustado para los valores que toma la variable i. El $j_{l-1}(x)$ obtenido de esta relación se agrega al inicio de la lista, tomando la posición 0 y recorriendo el resto de los elementos a la siguiente posición. Por ejemplo, al calcular el $j_{24}(x)$ e insertarlo en la lista, se agregará en la posición 0 y el $j_{25}(x)$ se recorrerá a la posición 1. Luego, para el cálculo de $j_{23}(x)$ se requerirá de $j_{24}(x)$ y $j_{25}(x)$ cuyas posiciones serán j[0] y j[1] en la lista. Y así sucesivamente. Esa es la razón por la que al definir la variable *j_ant* se usan las posiciones 0 y 1 de la lista.

Una vez que termine de iterar ese ciclo for, en la lista tendremos los valorer numéricos de usar la relación *down* a partir de los valores $j_{25}(x) = 1$ y $j_{26}(x) = 2$. Sin embargo, estos valores aún no nos sirven. Debemos normalizarlos de la siguiente forma:

$$j_l^N(x) = j_l^C \times \frac{j_0(x)}{j_0^C(x)}$$

Donde $j_l^N(x)$ es el valor numérico normalizado (la aproximación deseada), j_l^C es el valor numérico que ya contiene la lista en la posición l, $j_0^C(x)$ es el valor numérico de la lista en la posición 0 y $j_0(x) = \frac{\sin x}{x}$ es la primer función esférica de Bessel que deberá calcularse para el x dado. Como queremos que esta función nos devuelva el valor numérico de la función $j_l(x)$ para un l y x dados, solo normalizamos el elemento en la posición l de la lista y lo regresamos como resultado de nuestra función.

Lo sigueinte que haremos, será crear dos funciones cuyo obejtivo será imprimir una tabla con los valores numéricos de las primeras 25 funciones de Bessel esféricas para un x dado y a su vez lo comparará con el valor numérico obtenido por las funciones de Bessel esféricas de la librería *scipy*. Las funciones son:

[60]:
```
def tabla_bessel_up(x: float):
    """Esta función imprime una tabla con los valores numéticos para las
    primeras 25 funciones de bessel esféricas calculadas por la relación de
    recurrencia up y también ponr la función de bessel integrada por la
    librería scipy. Además las compara y obtiene el error relativo."""

    tabla = PrettyTable() # Define la tabla
    tabla.title = (f"Bessel esférico método up: x = {x}") # Agrega título a la tabla
    tabla.field_names = ['j_l', 'Up', 'Scipy', 'Error relativo'] # Le da nombre a␣
 ↪las columnas

    for i in range(0, 26):
        error = abs(bessel_esf_up(i, x) - spherical_jn(i, x))/spherical_jn(i, x)
        tabla.add_row([f"j_{i}",
                    f"{bessel_esf_up(i, x):<.8e}",
                    f"{spherical_jn(i, x):<.8e}",
                    f"{error:<.8e}"])#Insertamos un registro. El número de␣
 ↪elementos debe coincidir con el número de columnas
    print(tabla) #Imprime la tabla

def tabla_bessel_down(x: float):
    """Esta función imprime una tabla con los valores numéricos para las
    primeras 25 funciones de bessel esféricas calculadas por el algoritmo de
    Miller."""

    tabla = PrettyTable() # Define la tabla
    tabla.title = (f"Bessel esférico método down: x = {x}") # Agrega título a la␣
 ↪tabla
    tabla.field_names = ['j_l', 'Down', 'Scipy', 'Error relativo']

    for i in range(0, 26):
        error = abs(bessel_esf_down(i, x) - spherical_jn(i, x))/spherical_jn(i, x)
        tabla.add_row([f"j_{i}",
                    f"{bessel_esf_down(i, x):<.8e}",
                    f"{spherical_jn(i, x):<.8e}",
                    f"{error:<.8e}"])#Insertamos un registro. El número de␣
 ↪elementos debe coincidir con el número de columnas
    print(tabla) #Imprime la tabla
```

Las funciones anteriores son análogas. En lo único que difieren es en que una llama a la función *bessel_esf_up* para mostrar las aproximaciones calculadas por la relación *up* y la otra usa la función *bessel_esf_down* para mostrar los valores numéricos obtenidos por el mecanismo de Miller. Además, ambas funciones utilizan el elemento *PrettyTable* para crear la tabla y que se vea mejor. Se ha puesto un comentario a lado de cada comando de la librería *prettytable* para explicar como editar la tabla e imprimirla.

Veamos el resultado de estas funcines para los valores que se piden: $ x=0.1,1,10$.

[61]:
```
tabla_bessel_up(0.1)
tabla_bessel_down(0.1)
tabla_bessel_up(1)
tabla_bessel_down(1)
tabla_bessel_up(10)
tabla_bessel_down(10)
```

```
+-----------------------------------------------------------+
|            Bessel esférico método up: x = 0.1             |
```

j_l	Up	Scipy	Error relativo
j_0	9.98334166e-01	9.98334166e-01	0.00000000e+00
j_1	3.33000119e-02	3.33000119e-02	4.95932780e-14
j_2	6.66190608e-04	6.66190608e-04	7.35572052e-11
j_3	9.51851727e-06	9.51851972e-06	2.57234097e-07
j_4	1.05600670e-07	1.05772015e-07	1.61994783e-03
j_5	-1.44569836e-08	9.61631023e-10	1.60338157e+01
j_6	-1.69586887e-06	7.39754109e-12	2.29248644e+05
j_7	-2.20448496e-04	4.93188748e-14	4.46986061e+09
j_8	-3.30655785e-02	2.90120010e-16	1.13972071e+14
j_9	-5.62092789e+00	1.52698569e-18	3.68106127e+18
j_10	-1.06794323e+03	7.27151100e-21	1.46866756e+23
j_11	-2.24262458e+05	3.16158151e-23	7.09336318e+27
j_12	-5.15792975e+07	1.26465134e-25	4.07853896e+32
j_13	-1.28946001e+10	4.68395367e-28	2.75293076e+37
j_14	-3.48149045e+12	1.61517440e-30	2.15548887e+42
j_15	-1.00961934e+15	5.21029094e-33	1.93774080e+47
j_16	-3.12978513e+17	1.57888971e-35	1.98226963e+52
j_17	-1.03281900e+20	4.51114830e-38	2.28948136e+57
j_18	-3.61483519e+22	1.21923772e-40	2.96483215e+62
j_19	-1.33747869e+25	3.12627012e-43	4.27819299e+67
j_20	-5.21613074e+27	7.62509231e-46	6.84074439e+72
j_21	-2.13860023e+30	1.77328645e-48	1.20600946e+78
j_22	-9.19592883e+32	3.94065518e-51	2.33360404e+83
j_23	-4.13814659e+35	8.38440913e-54	4.93552560e+88
j_24	-1.94491970e+38	1.71111075e-56	1.13664162e+94
j_25	-9.53006515e+40	3.35513153e-59	2.84044457e+99

Bessel esférico método down: x = 0.1

j_l	Down	Scipy	Error relativo
j_0	9.98334166e-01	9.98334166e-01	0.00000000e+00
j_1	3.33000119e-02	3.33000119e-02	6.25125353e-16
j_2	6.66190608e-04	6.66190608e-04	9.76479247e-16
j_3	9.51851972e-06	9.51851972e-06	1.77975772e-16
j_4	1.05772015e-07	1.05772015e-07	3.12816433e-15
j_5	9.61631023e-10	9.61631023e-10	4.30092516e-16
j_6	7.39754109e-12	7.39754109e-12	4.36790310e-16
j_7	4.93188748e-14	4.93188748e-14	1.91941339e-15
j_8	2.90120010e-16	2.90120010e-16	1.52948519e-15
j_9	1.52698569e-18	1.52698569e-18	2.39639894e-15
j_10	7.27151100e-21	7.27151100e-21	4.96611866e-15
j_11	3.16158151e-23	3.16158151e-23	7.43611606e-16
j_12	1.26465134e-25	1.26465134e-25	4.90166404e-15
j_13	4.68395367e-28	4.68395367e-28	7.27581464e-15
j_14	1.61517440e-30	1.61517440e-30	1.95206260e-15
j_15	5.21029094e-33	5.21029094e-33	8.92992130e-15
j_16	1.57888971e-35	1.57888971e-35	2.36993791e-15
j_17	4.51114830e-38	4.51114830e-38	9.25749849e-15
j_18	1.21923772e-40	1.21923772e-40	3.67946853e-15
j_19	3.12627012e-43	3.12627012e-43	8.91769022e-15
j_20	7.62509231e-46	7.62509231e-46	1.04055717e-14
j_21	1.77328645e-48	1.77328645e-48	1.33655435e-14
j_22	3.94065518e-51	3.94065518e-51	3.76506543e-15

j_23	8.38440913e-54	8.38440913e-54	6.67737338e-14
j_24	1.71111078e-56	1.71111075e-56	1.57396531e-08
j_25	3.36832830e-59	3.35513153e-59	3.93330950e-03

Bessel esférico método up: x = 1

j_l	Up	Scipy	Error relativo
j_0	8.41470985e-01	8.41470985e-01	0.00000000e+00
j_1	3.01168679e-01	3.01168679e-01	5.52957413e-16
j_2	6.20350520e-02	6.20350520e-02	2.34894253e-15
j_3	9.00658112e-03	9.00658112e-03	7.56942415e-14
j_4	1.01101581e-03	1.01101581e-03	4.58188899e-12
j_5	9.25611586e-05	9.25611586e-05	4.42941880e-10
j_6	7.15693586e-06	7.15693631e-06	6.23673410e-08
j_7	4.79007658e-07	4.79013420e-07	1.20281993e-05
j_8	2.81790093e-08	2.82649880e-08	3.04187902e-03
j_9	3.55007135e-11	1.49137650e-09	9.76196009e-01
j_10	-2.75044958e-08	7.11655264e-11	3.87486227e+02
j_11	-5.77629912e-07	3.09955185e-12	1.86360170e+05
j_12	-1.32579835e-05	1.24166260e-13	1.06776057e+08
j_13	-3.30871957e-04	4.60463768e-15	7.18562416e+10
j_14	-8.92028486e-03	1.58957599e-16	5.61173856e+13
j_15	-2.58357389e-01	5.13268612e-18	5.03357079e+16
j_16	-8.00015878e+00	1.55670827e-19	5.13915094e+19
j_17	-2.63746882e+02	4.45117750e-21	5.92532834e+22
j_18	-9.22314072e+03	1.20385574e-22	7.66133383e+25
j_19	-3.40992460e+05	3.08874236e-24	1.10398479e+29
j_20	-1.32894828e+07	7.53779572e-26	1.76304629e+32
j_21	-5.44527802e+08	1.75388258e-27	3.10469930e+35
j_22	-2.34014060e+10	3.89936131e-29	6.00134333e+38
j_23	-1.05251874e+12	8.30011892e-31	1.26807670e+42
j_24	-4.94449795e+13	1.69458017e-32	2.91783064e+45
j_25	-2.42175148e+15	3.32393637e-34	7.28579374e+48

Bessel esférico método down: x = 1

j_l	Down	Scipy	Error relativo
j_0	8.41470985e-01	8.41470985e-01	0.00000000e+00
j_1	3.01168679e-01	3.01168679e-01	3.68638276e-16
j_2	6.20350520e-02	6.20350520e-02	1.11854406e-16
j_3	9.00658112e-03	9.00658112e-03	0.00000000e+00
j_4	1.01101581e-03	1.01101581e-03	1.07238894e-15
j_5	9.25611586e-05	9.25611586e-05	1.02491900e-15
j_6	7.15693631e-06	7.15693631e-06	1.65691865e-15
j_7	4.79013420e-07	4.79013420e-07	1.10517904e-15
j_8	2.82649880e-08	2.82649880e-08	1.99003380e-15
j_9	1.49137650e-09	1.49137650e-09	4.15981785e-16
j_10	7.11655264e-11	7.11655264e-11	3.99552072e-15
j_11	3.09955185e-12	3.09955185e-12	3.12739495e-15
j_12	1.24166260e-13	1.24166260e-13	1.62643533e-15
j_13	4.60463768e-15	4.60463768e-15	5.48220093e-15
j_14	1.58957599e-16	1.58957599e-16	1.55084774e-15
j_15	5.13268612e-18	5.13268612e-18	2.70164496e-15
j_16	1.55670827e-19	1.55670827e-19	4.02083835e-15

j_17	4.45117750e-21	4.45117750e-21	5.07045417e-15
j_18	1.20385574e-22	1.20385574e-22	4.10105306e-15
j_19	3.08874236e-24	3.08874236e-24	3.56787917e-16
j_20	7.53779572e-26	7.53779572e-26	1.27925025e-14
j_21	1.75388258e-27	1.75388258e-27	4.90886783e-15
j_22	3.89936131e-29	3.89936131e-29	3.33764906e-12
j_23	8.30011897e-31	8.30011892e-31	7.03989579e-09
j_24	1.69460762e-32	1.69458017e-32	1.61986990e-05
j_25	3.45838291e-34	3.32393637e-34	4.04479883e-02

Bessel esférico método up: x = 10

j_l	Up	Scipy	Error relativo
j_0	-5.44021111e-02	-5.44021111e-02	-0.00000000e+00
j_1	7.84669418e-02	7.84669418e-02	0.00000000e+00
j_2	7.79421936e-02	7.79421936e-02	0.00000000e+00
j_3	-3.94958450e-02	-3.94958450e-02	-0.00000000e+00
j_4	-1.05589285e-01	-1.05589285e-01	-0.00000000e+00
j_5	-5.55345116e-02	-5.55345116e-02	-2.49894838e-16
j_6	4.45013223e-02	4.45013223e-02	3.11851133e-16
j_7	1.13386231e-01	1.13386231e-01	0.00000000e+00
j_8	1.25578024e-01	1.25578024e-01	2.21022555e-16
j_9	1.00096410e-01	1.00096410e-01	5.54576847e-16
j_10	6.46051545e-02	6.46051545e-02	3.65175805e-15
j_11	3.55744149e-02	3.55744149e-02	2.92579397e-15
j_12	1.72159997e-02	1.72159997e-02	6.04573711e-16
j_13	7.46558448e-03	7.46558448e-03	1.20828612e-14
j_14	2.94107834e-03	2.94107834e-03	8.49348951e-14
j_15	1.06354271e-03	1.06354271e-03	5.97179242e-13
j_16	3.55904074e-04	3.55904074e-04	4.82904640e-12
j_17	1.10940728e-04	1.10940728e-04	4.53984692e-11
j_18	3.23884744e-05	3.23884744e-05	4.91200361e-10
j_19	8.89662722e-06	8.89662727e-06	6.05033705e-09
j_20	2.30837177e-06	2.30837196e-06	8.40498801e-08
j_21	5.67697030e-07	5.67697772e-07	1.30641308e-06
j_22	1.32725463e-07	1.32728458e-07	2.25653727e-05
j_23	2.95675538e-08	2.95802899e-08	4.30562185e-04
j_24	6.24203967e-09	6.29890453e-09	9.02773771e-03
j_25	1.01844059e-09	1.28434224e-09	2.07033330e-01

Bessel esférico método down: x = 10

j_l	Down	Scipy	Error relativo
j_0	-5.44021111e-02	-5.44021111e-02	-0.00000000e+00
j_1	7.84669418e-02	7.84669418e-02	1.59175428e-15
j_2	7.79421936e-02	7.79421936e-02	3.56104625e-16
j_3	-3.94958450e-02	-3.94958450e-02	-2.98667357e-15
j_4	-1.05589285e-01	-1.05589285e-01	-1.05145425e-15
j_5	-5.55345116e-02	-5.55345116e-02	-4.99789677e-16
j_6	4.45013223e-02	4.45013223e-02	3.27443690e-15
j_7	1.13386231e-01	1.13386231e-01	1.34633337e-15
j_8	1.25578024e-01	1.25578024e-01	6.63067664e-16
j_9	1.00096410e-01	1.00096410e-01	2.77288424e-16
j_10	6.46051545e-02	6.46051545e-02	4.29618594e-15

```
| j_11 |  3.55744149e-02 |  3.55744149e-02 |  6.43674673e-15 |
| j_12 |  1.72159997e-02 |  1.72159997e-02 |  1.47112936e-14 |
| j_13 |  7.46558448e-03 |  7.46558448e-03 |  5.19330667e-14 |
| j_14 |  2.94107834e-03 |  2.94107834e-03 |  2.72204542e-13 |
| j_15 |  1.06354271e-03 |  1.06354271e-03 |  1.81865444e-12 |
| j_16 |  3.55904074e-04 |  3.55904074e-04 |  1.45980257e-11 |
| j_17 |  1.10940728e-04 |  1.10940728e-04 |  1.37107577e-10 |
| j_18 |  3.23884744e-05 |  3.23884744e-05 |  1.48331509e-09 |
| j_19 |  8.89662743e-06 |  8.89662727e-06 |  1.82704900e-08 |
| j_20 |  2.30837255e-06 |  2.30837196e-06 |  2.53809213e-07 |
| j_21 |  5.67700012e-07 |  5.67697772e-07 |  3.94503423e-06 |
| j_22 |  1.32737503e-07 |  1.32728458e-07 |  6.81416689e-05 |
| j_23 |  2.96187499e-08 |  2.95802899e-08 |  1.30018796e-03 |
| j_24 |  6.47062189e-09 |  6.29890453e-09 |  2.72614648e-02 |
| j_25 |  2.08729738e-09 |  1.28434224e-09 |  6.25187840e-01 |
+------+-----------------+-----------------+-----------------+
```

Como podemos ver, los valores calculados por el método *up* no son buenos. El error relativo es muy grande. A pesar de que para $x = 10$ fue pequeño, sigue siendo significativo. Por lo que usar la relación de recurrencia *up* a partir de j_0 y j_1 no es adecuado realizarlo numéricamente. Para valores pequeños de l, las aproximaciones empiezan a diverger y arrojan cosas complétamente erróneas.

Por otro lado, para las funciones calculadas por el mecanismo de Miller podemos observar que para valores pequeños de l tenemos errores relativos muy pequeños. Este error va aumentando conforme aumenta l, pero aún así para $l = 25$ tenemos un error no tan grande en comparación con el otro método. Por lo que, los valores calculados a través de este mecanismo si son adecuados. Para obtener aproximaciones más certeras y con ello errores sumamente pequeños, debemos comenzar a usar la relación de recurrencia *down* a partir de $j_l(x)$ y $j_l(x)$ con $l > 25$ si es que se quieren los primeros 25 valores de j_l. Ya que en nuestra función *bessel_esf_down* comenzamos a partir de $l = 25$ y eso ocasionó que tuvieramos errores más grandes para las $l's$ cercanas a 25. Así que, si se quiere tener errores más pequeños debemos modificar el valor de l con el que se comienza a utilizar la relación *down. Esto se realizará en el siguiente inciso.

(b) Para resolver este inciso, solo ajustaremos a la función *bessel_esf_down* ya que al usar el mecanismo de Miller presenta buanas aproximaciones. Para lograr que nuestros errores relativos sean menores a 10^{-10}, como ya se dijo anterioremnte, debemos comenzar a usar la relación de recurrencia *down* a partir de un l más grande. El problema es encontrar la mímina l que cumpla eso.

La forma en como se resolvió, fue usar un ciclo while dentro de la función ajustada. Este ciclo while evaluaría el error relativo del valor numérico de $j_{25}(x)$ y se repetiría si este error es mayor al solicitado. La razón de porque se evalua solo para $l = 25$ es que como solo nos interesan los primeros 25 valores de l, si j_{25} cumple, automaticamente todas las funciones anteriores lo harán ya que entre más pequeña sea l mayor es la convergencia al valor deseado. El código ajustado para calcular los valores numéricos de las primeras 25 funciones eféricas de Bessel y el código ajsutado para imprimir estos valores en forma de tabla son los siguientes:

[57]:
```python
def bessel_esf_down_ajuste(x: float) -> deque:
    """Esta función utiliza una el algoritmo de Miller para obtener obtener
    los valores numéricos de las primeros 25 funciones de Bessel esféricas
    con un error relativo menor a 10^(-10)"""

    n = 25  #Se inicia en 25 porque queremos los primeros 25 valores
    error = 1
    j = deque([1,2])

    while error > 10**(-10):
```

```
        for i in range(0,n):
            j_ant = (2*(n - i) + 1)/x * j[0] - j[1]
            j.appendleft(j_ant)

        normalizacion = (sin(x)/x)/j[0]
        for i in range(0,26):
            j[i] = j[i]*normalizacion

        error = abs( ( j[25] - spherical_jn(25, x) )/ spherical_jn(25, x) )
        print(f"l = {n}")
        n += 1
    return j

def tabla_bessel_down_ajuste(x: float):
    """Esta función imprime una tabla con los valores numéricos para las
    primeras 25 funciones de bessel esféricas calculadas por el algoritmo de
    Miller con un error relativo menor a 10^(-10)"""

    tabla = PrettyTable() # Define la tabla
    tabla.title = (f"Bessel esférico método down con ajuste: x = {x}") # Agrega un␣
 ↪título
    tabla.field_names = ['j_l', 'Down', 'Scipy', 'Error relativo'] # Agrega el␣
 ↪nombre de las columnas
    j = bessel_esf_down_ajuste(x)

    for i in range(0, 26):
        error = abs( (j[i] - spherical_jn(i, x)) / spherical_jn(i, x) )
        tabla.add_row([f"j_{i}",
                    f"{j[i]:<.10e}",
                    f"{spherical_jn(i, x):<.10e}",
                    f"{error:<.10e}"]) #Inserta un registro en forma de lista. El␣
 ↪número de elemntos debe coincidir con el número de columnas
    print(tabla)
```

La función *bessel_esf_down* lo que hace es crear una *deque* que es como una lista y les agrega los elementos arbitarios $j_l(x) = 1$ y $j_{l+1}(x) = 2$ para comenzar a usar el mecanismo de Miller. Luego, con ayuda de un contador n iniciado en 25 y con ayudar de la variable *error* se inicia un cliclo while. Lo primero es usar un ciclo for para usar la relación de recurrencia *down* y calcular los valores de $j_i(x)$ donde $i = 0, 1, ..., n-1$. Así calculamos todas las funciones de Bessel esféricas anteriores a partir de los valores iniciales $j_l(x) = 1$ y $j_{l+1}(x) = 2$ y las agregamos a la lista. De forma que en la posición 0 esté $j_0(x)$, en la posición 1 esté $j_1(x)$ y así sucesivamente hasta $j_{25}(x)$ ya que solo nos interesan estos valores. Luego de ello, se procede a realizar la normalización de $j_0(x), j_1(x), ..., j_{25}(x)$. Enseguida, se calcula el error relativo para $j_{25}(x)$ y se reescribe este valor en la variable *error*. Finalmente, nuestro contador n se aumenta en 1. Al término de este proceso, nuevamente se evalua la condición *error > 10**(-10)* en el ciclo while. Si se cumple, entonces no hemos obtenido la aproximación deseada. Por lo que se repite todo el proceso del while pero ahora para la n que ha sido aumentada en 1. Así sucesivamente hasta que se rompa el while y obtengamos la aproximación deseada. Cuando eso suceda, la función regresará la *deque* que se usó en la función, cuyos primeros 26 elementos (del 0 al 25) tendrán nuestras aproximaciones deseadas.

Por útlimo, la función *tabla_bessel_down_ajuste* usa la librería *prettytable* para imprimir la tabla. Esta tabla contendrá los valores obtenidos por la función anterior y así nuestros errores relativos serán los solicitados. Veamos las tablas de $x = 0.1, 1, 10$.

[58]:
```
tabla_bessel_down_ajuste(0.1)
tabla_bessel_down_ajuste(1)
tabla_bessel_down_ajuste(10)
```

```
l = 25
l = 26
l = 27
```

Bessel esférico método down con ajuste: x = 0.1			
j_l	Down	Scipy	Error relativo
j_0	9.9833416647e-01	9.9833416647e-01	0.0000000000e+00
j_1	3.3300011903e-02	3.3300011903e-02	6.2512535349e-16
j_2	6.6619060845e-04	6.6619060845e-04	1.1392257878e-15
j_3	9.5185197209e-06	9.5185197209e-06	3.5595154377e-16
j_4	1.0577201502e-07	1.0577201502e-07	3.0030377615e-15
j_5	9.6163102329e-10	9.6163102329e-10	4.3009251601e-16
j_6	7.3975410936e-12	7.3975410936e-12	4.3679031004e-16
j_7	4.9318874757e-14	4.9318874757e-14	1.9194133908e-15
j_8	2.9012001025e-16	2.9012001025e-16	1.5294851906e-15
j_9	1.5269856935e-18	1.5269856935e-18	2.3963989381e-15
j_10	7.2715109967e-21	7.2715109967e-21	4.9661186614e-15
j_11	3.1615815052e-23	3.1615815052e-23	3.7180580317e-16
j_12	1.2646513379e-25	1.2646513379e-25	4.5385778182e-15
j_13	4.6839536653e-28	4.6839536653e-28	6.7014082214e-15
j_14	1.6151744028e-30	1.6151744028e-30	1.3013750669e-15
j_15	5.2102909410e-33	5.2102909410e-33	8.4046318154e-15
j_16	1.5788897129e-35	1.5788897129e-35	2.7085004743e-15
j_17	4.5111483007e-38	4.5111483007e-38	8.7946235683e-15
j_18	1.2192377198e-40	1.2192377198e-40	3.3449713916e-15
j_19	3.1262701152e-43	3.1262701152e-43	8.5355035003e-15
j_20	7.6250923124e-46	7.6250923124e-46	9.7934792491e-15
j_21	1.7732864463e-48	1.7732864463e-48	1.2851484112e-14
j_22	3.9406551793e-51	3.9406551793e-51	4.2168732861e-15
j_23	8.3844091285e-54	8.3844091285e-54	1.1059831688e-15
j_24	1.7111107510e-56	1.7111107510e-56	6.7476583961e-15
j_25	3.3551315322e-59	3.3551315322e-59	1.3047181832e-14

```
l = 25
l = 26
l = 27
l = 28
```

Bessel esférico método down con ajuste: x = 1			
j_l	Down	Scipy	Error relativo
j_0	8.4147098481e-01	8.4147098481e-01	0.0000000000e+00
j_1	3.0116867894e-01	3.0116867894e-01	5.5295741337e-16
j_2	6.2035052011e-02	6.2035052011e-02	2.2370881232e-16
j_3	9.0065811171e-03	9.0065811171e-03	0.0000000000e+00
j_4	1.0110158084e-03	1.0110158084e-03	1.0723889414e-15
j_5	9.2561158611e-05	9.2561158611e-05	8.7850199972e-16
j_6	7.1569363101e-06	7.1569363101e-06	1.3018646549e-15
j_7	4.7901341987e-07	4.7901341987e-07	1.1051790411e-15
j_8	2.8264988022e-08	2.8264988022e-08	1.9900338047e-15
j_9	1.4913765026e-09	1.4913765026e-09	2.7732118990e-16
j_10	7.1165526400e-11	7.1165526400e-11	3.9955207240e-15
j_11	3.0995518548e-12	3.0995518548e-12	2.9970868226e-15
j_12	1.2416625970e-13	1.2416625970e-13	1.6264353314e-15
j_13	4.6046376777e-15	4.6046376777e-15	5.4822009318e-15

j_14	1.5895759875e-16	1.5895759875e-16	1.7059325147e-15
j_15	5.1326861154e-18	5.1326861154e-18	2.4014621909e-15
j_16	1.5567082706e-19	1.5567082706e-19	4.1754859823e-15
j_17	4.4511775038e-21	4.4511775038e-21	4.9014390310e-15
j_18	1.2038557422e-22	1.2038557422e-22	3.9057648175e-15
j_19	3.0887423635e-24	3.0887423635e-24	3.5678791695e-16
j_20	7.5377957222e-26	7.5377957222e-26	1.2640210849e-14
j_21	1.7538825776e-27	1.7538825776e-27	6.3406209487e-15
j_22	3.8993613099e-29	3.8993613099e-29	6.0373513326e-15
j_23	8.3001189151e-31	8.3001189151e-31	1.0551795091e-14
j_24	1.6945801738e-32	1.6945801738e-32	0.0000000000e+00
j_25	3.3239363664e-34	3.3239363664e-34	2.4457391015e-13

```
l = 25
l = 26
l = 27
l = 28
l = 29
l = 30
l = 31
l = 32
```

Bessel esférico método down con ajuste: x = 10

j_l	Down	Scipy	Error relativo
j_0	-5.4402111089e-02	-5.4402111089e-02	0.0000000000e+00
j_1	7.8466941799e-02	7.8466941799e-02	7.0744634567e-16
j_2	7.7942193629e-02	7.7942193629e-02	1.7805231238e-16
j_3	-3.9495844984e-02	-3.9495844984e-02	1.2298067644e-15
j_4	-1.0558928512e-01	-1.0558928512e-01	6.5715890549e-16
j_5	-5.5534511621e-02	-5.5534511621e-02	3.7484225761e-16
j_6	4.4501322334e-02	4.4501322334e-02	6.2370226681e-16
j_7	1.1338623066e-01	1.1338623066e-01	4.8957577045e-16
j_8	1.2557802365e-01	1.2557802365e-01	6.6306766444e-16
j_9	1.0009640955e-01	1.0009640955e-01	9.7050948274e-16
j_10	6.4605154493e-02	6.4605154493e-02	4.5109952333e-15
j_11	3.5574414886e-02	3.5574414886e-02	4.4862174206e-15
j_12	1.7215999745e-02	1.7215999745e-02	4.6350651183e-15
j_13	7.4655844766e-03	7.4655844766e-03	4.2987102224e-15
j_14	2.9410783418e-03	2.9410783418e-03	4.4236924549e-15
j_15	1.0635427146e-03	1.0635427146e-03	4.2815855554e-15
j_16	3.5590407351e-04	3.5590407351e-04	4.7218154903e-15
j_17	1.1094072797e-04	1.1094072797e-04	4.6420826810e-15
j_18	3.2388474389e-05	3.2388474389e-05	4.6028039766e-15
j_19	8.8966272694e-06	8.8966272694e-06	4.5699994208e-15
j_20	2.3083719613e-06	2.3083719613e-06	4.9536837995e-15
j_21	5.6769777198e-07	5.6769777198e-07	4.8491595607e-15
j_22	1.3272845821e-07	1.3272845821e-07	4.1879893668e-15
j_23	2.9580289943e-08	2.9580289943e-08	8.9484517064e-15
j_24	6.2989045263e-09	6.2989045263e-09	2.9626095365e-13
j_25	1.2843422360e-09	1.2843422360e-09	6.7998798626e-12

Como podemos ver, todos los erorres relativos son menores a 10^{-10}. Además note que antes de cada tabla se imprimen los valores que l va tomando. Estos valores indican a partir de que l se comenzó a usar la relación de recurrencia *down* para obtener esos errores relativos. Dado que se tuvo que comenzar forzosamente con $l = 25$ para obtener los primeros 25

valores, la primer iteración se hace con 25, luego con 26 y así hasta llegar al l que cumple con la condición. Lo sorprendente es que requieren muy pocas iteraciones para logar ese error relativo. Por ejemplo, en la tabla $x = 0.1$ se comienza con $l = 25$. No se cumple la condición y ahora lo realiza con $l = 26$. Tampoco se cumple, así que ahora lo hace con $l = 27$. Pero en esta si se cumple. Así que, de la relación de recurrencia *down* bastó tomar $j_{27}(x) = 1$ y $j_{28}(x) = 2$ como valores iniciales del mecanismo de Miller. Esto permitió que $j_{25}(x)$ al ser normalizado tuviera un erorr relativo menor a 10^{-10}. Por lo que el ciclo while, ¡solo se repitió 3 veces!. Esto es algo demasiado sorprendente. Yo pensé que tomaría mucho más en lograr ese error relativo, pero no fue así. Por lo que el mecanismo de Miller es una herramienta sumamente poderosa para lograr aproximaciones muy precisas de las funciones esféricas de Bessel.

(c) Para este inciso se creo una función que imprima la tabla deseada. Se apoyó de la función *bessel_esf_up* y *bessel_esf_dow_ajuste* para imprimir esta nueva tabla. Esta función es muy similar a todas las anteriores que también imprimen una tabla. Al igual se usa la libreía *prettytable* para crear la tabla. La función y el resultado son los siguientes:

[59]:
```
def tabla_up_vs_down(x: float):
    """Esta función imprime una tabla que compara los valores numérocos
    para las primeras 25 funciones de bessel calculadas por el método up
    y por el método down ajustado"""

    tabla = PrettyTable() #Crea la tabla
    tabla.title = (f"Bessel up vs down: x = {x}")#Agrega título
    tabla.field_names = ['l', 'j_l up', 'j_l down', 'Error']#Agrega el nombre de␣
 ↪las columnas
    j = bessel_esf_down_ajuste(x)

    for i in range(0, 26):
        error = abs( bessel_esf_up(i, x) - j[i] ) / ( abs(bessel_esf_up(i, x)) +␣
 ↪abs(j[i]) )
        tabla.add_row([f"{i}",
                    f"{bessel_esf_up(i,x):<.10e}",
                    f"{j[i]:<.10e}",
                    f"{error:<.10e}"]) #Inserta un registro en la tabla. El número␣
 ↪de elementos debe coincidir con el número de columnas
    print(tabla)


tabla_up_vs_down(0.1)
tabla_up_vs_down(1)
tabla_up_vs_down(10)
```

```
l = 25
l = 26
l = 27
+-------------------------------------------------------------------+
|                  Bessel up vs down: x = 0.1                       |
+----+-------------------+------------------+-------------------+
| l  |       j_l up      |     j_l down     |       Error       |
+----+-------------------+------------------+-------------------+
| 0  |  9.9833416647e-01 | 9.9833416647e-01 | 0.0000000000e+00 |
| 1  |  3.3300011903e-02 | 3.3300011903e-02 | 2.4484076345e-14 |
| 2  |  6.6619060840e-04 | 6.6619060845e-04 | 3.6778032976e-11 |
| 3  |  9.5185172724e-06 | 9.5185197209e-06 | 1.2861706541e-07 |
| 4  |  1.0560066988e-07 | 1.0577201502e-07 | 8.1063050260e-04 |
| 5  | -1.4456983610e-08 | 9.6163102329e-10 | 1.0000000000e+00 |
| 6  | -1.6958688670e-06 | 7.3975410936e-12 | 1.0000000000e+00 |
```

7	-2.2044849573e-04	4.9318874757e-14	1.0000000000e+00
8	-3.3065578490e-02	2.9012001025e-16	1.0000000000e+00
9	-5.6209278948e+00	1.5269856935e-18	1.0000000000e+00
10	-1.0679432344e+03	7.2715109967e-21	1.0000000000e+00
11	-2.2426245830e+05	3.1615815052e-23	1.0000000000e+00
12	-5.1579297467e+07	1.2646513379e-25	1.0000000000e+00
13	-1.2894600104e+10	4.6839536653e-28	1.0000000000e+00
14	-3.4814904488e+12	1.6151744028e-30	1.0000000000e+00
15	-1.0096193356e+15	5.2102909410e-33	1.0000000000e+00
16	-3.1297851253e+17	1.5788897129e-35	1.0000000000e+00
17	-1.0328189952e+20	4.5111483007e-38	1.0000000000e+00
18	-3.6148351852e+22	1.2192377198e-40	1.0000000000e+00
19	-1.3374786904e+25	3.1262701152e-43	1.0000000000e+00
20	-5.2161307440e+27	7.6250923124e-46	1.0000000000e+00
21	-2.1386002303e+30	1.7732864463e-48	1.0000000000e+00
22	-9.1959288288e+32	3.9406551793e-51	1.0000000000e+00
23	-4.1381465870e+35	8.3844091285e-54	1.0000000000e+00
24	-1.9449196999e+38	1.7111107510e-56	1.0000000000e+00
25	-9.5300651483e+40	3.3551315322e-59	1.0000000000e+00

l = 25
l = 26
l = 27
l = 28

Bessel up vs down: x = 1

l	j_l up	j_l down	Error
0	8.4147098481e-01	8.4147098481e-01	0.0000000000e+00
1	3.0116867894e-01	3.0116867894e-01	0.0000000000e+00
2	6.2035052011e-02	6.2035052011e-02	1.0626168585e-15
3	9.0065811171e-03	9.0065811171e-03	3.7847120744e-14
4	1.0110158084e-03	1.0110158084e-03	2.2904083010e-12
5	9.2561158570e-05	9.2561158611e-05	2.2147050060e-10
6	7.1569358637e-06	7.1569363101e-06	3.1183670799e-08
7	4.7900765821e-07	4.7901341987e-07	6.0141358091e-06
8	2.8179009348e-08	2.8264988022e-08	1.5232562895e-03
9	3.5500713480e-11	1.4913765026e-09	9.5349892826e-01
10	-2.7504495792e-08	7.1165526400e-11	1.0000000000e+00
11	-5.7762991235e-07	3.0995518548e-12	1.0000000000e+00
12	-1.3257983488e-05	1.2416625970e-13	1.0000000000e+00
13	-3.3087195729e-04	4.6046376777e-15	1.0000000000e+00
14	-8.9202848634e-03	1.5895759875e-16	1.0000000000e+00
15	-2.5835738908e-01	5.1326861154e-18	1.0000000000e+00
16	-8.0001587767e+00	1.5567082706e-19	1.0000000000e+00
17	-2.6374688224e+02	4.4511775038e-21	1.0000000000e+00
18	-9.2231407196e+03	1.2038557422e-22	1.0000000000e+00
19	-3.4099245974e+05	3.0887423635e-24	1.0000000000e+00
20	-1.3289482789e+07	7.5377957222e-26	1.0000000000e+00
21	-5.4452780190e+08	1.7538825776e-27	1.0000000000e+00
22	-2.3401405999e+10	3.8993613099e-29	1.0000000000e+00
23	-1.0525187422e+12	8.3001189151e-31	1.0000000000e+00
24	-4.9444979475e+13	1.6945801738e-32	1.0000000000e+00
25	-2.4217514755e+15	3.3239363664e-34	1.0000000000e+00

l = 25
l = 26

```
l = 27
l = 28
l = 29
l = 30
l = 31
l = 32
```

Bessel up vs down: x = 10			
l	j_l up	j_l down	Error
0	-5.4402111089e-02	-5.4402111089e-02	0.0000000000e+00
1	7.8466941799e-02	7.8466941799e-02	3.5372317283e-16
2	7.7942193629e-02	7.7942193629e-02	8.9026156192e-17
3	-3.9495844984e-02	-3.9495844984e-02	6.1490338220e-16
4	-1.0558928512e-01	-1.0558928512e-01	3.2857945274e-16
5	-5.5534511621e-02	-5.5534511621e-02	6.2473709602e-17
6	4.4501322334e-02	4.4501322334e-02	4.6777670010e-16
7	1.1338623066e-01	1.1338623066e-01	2.4478788522e-16
8	1.2557802365e-01	1.2557802365e-01	2.2102255481e-16
9	1.0009640955e-01	1.0009640955e-01	2.0796631773e-16
10	6.4605154493e-02	6.4605154493e-02	4.2961859365e-16
11	3.5574414886e-02	3.5574414886e-02	7.8021172533e-16
12	1.7215999745e-02	1.7215999745e-02	2.0152457036e-15
13	7.4655844766e-03	7.4655844766e-03	8.1907856940e-15
14	2.9410783418e-03	2.9410783418e-03	4.4679293794e-14
15	1.0635427146e-03	1.0635427146e-03	3.0073041401e-13
16	3.5590407351e-04	3.5590407351e-04	2.4168841062e-12
17	1.1094072797e-04	1.1094072797e-04	2.2701555631e-11
18	3.2388474374e-05	3.2388474389e-05	2.4560248198e-10
19	8.8966272156e-06	8.8966272694e-06	3.0251708214e-09
20	2.3083717673e-06	2.3083719613e-06	4.2024944301e-08
21	5.6769703033e-07	5.6769777198e-07	6.5320696803e-07
22	1.3272546314e-07	1.3272845821e-07	1.1282813646e-05
23	2.9567553789e-08	2.9580289943e-08	2.1532744843e-04
24	6.2420396684e-09	6.2989045263e-09	4.5343362561e-03
25	1.0184405863e-09	1.2843422360e-09	1.1546970348e-01

Como ya se vió antes, el método *up* no fue adecuado para obtener las aproximaciones deseadas, porque había valores en los que j_l^{up} arrojaba un valor muy grande que no tenía nada que ver con la respuesta correcta. Mientras que con el mecanismo de Miller fue sencillo controlar las aproximaciones y obtener estimaciones muy cercanas. Estás diferencias se notan claramente en las columnas 2 y 3 de las tablas anteriores.

Ahora analicemos más a detalle la columna 4. Esta columan contiene el siguiente valor:

$$\frac{|j_l^{up} - j_l^{down}|}{|j_l^{up}| + |j_l^{down}|} \tag{5}$$

Podemos ver de las tablas que para algunas $l's$, se cumple que $|j_l^{up}| \gg |j_l^{down}|$. Cuando esto sucede, en el cálculo de la fórmula (5), la computadora puede interpretar lo siguiente:

$$|j_l^{up} - j_l^{down}| \approx |j_l^{up}| \quad ; \quad |j_l^{up}| + |j_l^{down}| \approx |j_l^{up}|$$

Por lo que ese cálculo arrojaría como respuesta 1. En efecto, esto sucede en las tablas para $x = 0.1$ y $x = 1$. Todo debido a que los valores calculados por el método up divergen muy rápido y arroja valores grandes para x pequeñas como lo son 0.1 y 1.