Apuntes: Programación Orientada a la Química

Prof. Jhon Fredy Pérez Torres

August 21, 2025

Contenido

1 El Sistema Operativo Linux **5**
- 1.1 Instalando Linux a través de Unetbootin . . . 5
- 1.2 Familiarizandose con Linux . . . 7
- 1.3 Tarea . . . 8

2 El Lenguaje de Programación Python **11**
- 2.1 Introducción . . . 11
- 2.2 ¡Hola Mundo! . . . 11
- 2.3 Tipos de variables . . . 12
- 2.4 Configuración electrónica . . . 13
- 2.5 Tarea . . . 13
- 2.6 Introducción a los Bots . . . 14
  - 2.6.1 Python-Telegram-Bot . . . 14

3 Matrices, Vectores y Balance de Ecuaciones Químicas **17**
- 3.1 Introducción . . . 17
- 3.2 Definiendo matrices y vectores . . . 17
- 3.3 Balance de ecuaciones químicas . . . 18
- 3.4 Ejercicios . . . 20
- 3.5 Espacio nulo y nulidad de una matriz . . . 20
- 3.6 Tarea . . . 21

4 La librería MatPlotLib, Distribución Maxwell-Boltzmann y Diagramas Tanabe-Sugano **23**
- 4.1 Introducción . . . 23
- 4.2 Distribución de Maxwell-Boltzmann . . . 23
- 4.3 Cantidades estadísticas de `NumPy` . . . 24
- 4.4 Tarea . . . 25
- 4.5 Cálculo de valores propios (diagramas de Tanabe-Sugano) . . . 25
  - 4.5.1 Introducción . . . 25
  - 4.5.2 Estados electrónicos de complejos de metales de transición . . . 25

5 Ecuaciones Diferenciales Ordinarias **27**
- 5.1 El azúcar invertido . . . 27
- 5.2 Reacciones de los NO_x en la atmósfera terreste . . . 28
- 5.3 Interacción radiación-materia . . . 30
- 5.4 Ejercicios . . . 32

6 Método de Discretización del Hamiltoniano en una Malla de Fourier (FGHM) 33
6.1 Descripción del problema . . . 33
6.2 Vibraciones de moléculas diatómicas (ecuación de Schrödinger) . . . 33
6.3 Estructura elecrtónica del átomo de hidrógeno (ecuación de Dirac) . . . 36
6.4 Algoritmo FGHM en Fortran . . . 38
6.5 Tarea . . . 39

7 Estructura electrónica (método de Hartree-Fock) 41
7.1 Introducción . . . 41
7.2 Conjunto de funciones base . . . 41

8 Introducción a la Computación Cuántica 43
8.1 Qiskit . . . 43

9 El Lenguaje de Programación Fortran 45
9.1 Introducción al lenguaje de programación `Fortran` . . . 45
9.2 La librería de álgebra lineal `LAPACK` . . . 45
9.3 Implementación del método de campo autoconsistente de Hartee-Fock . . . 45

10 Introducción a los Microcontroladores 47
10.1 El microcontrolador `Arduino` . . . 47
10.2 Uso de fotorresistores `LDR` . . . 47
10.3 Construcción de un colorímetro . . . 47
10.3.1 Ley de Beer-Lambert . . . 47
10.4 Uso del medidor de humedad y temperatura `DTH22` . . . 47
10.4.1 Determinación de la rapidez de hidratación de una muestra café comercial . . . 47

Capítulo 1

El Sistema Operativo Linux

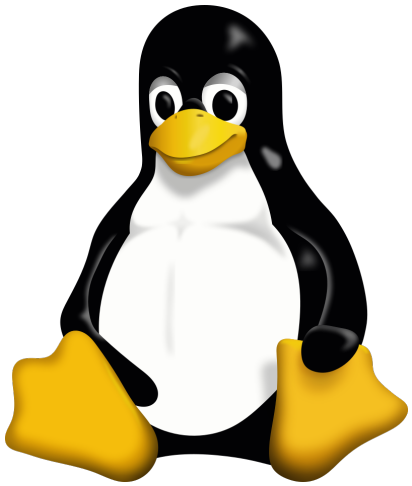


Figura 1.1: Tux, mascota oficial de Linux.

1.1 Instalando Linux a través de Unetbootin

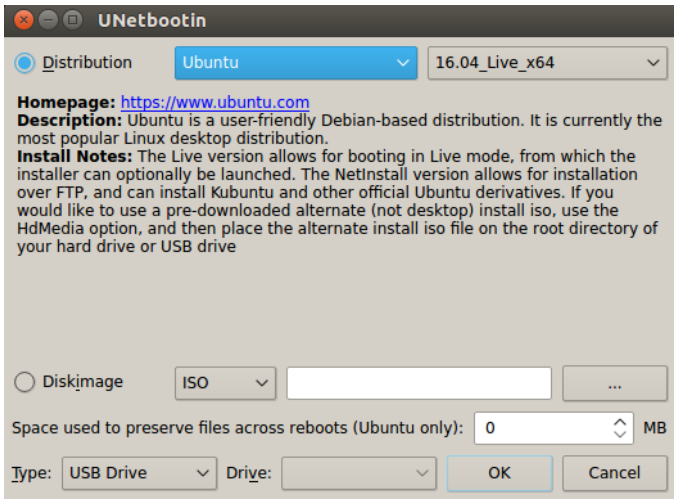
Si usted desea instalar Linux, es porque probablemente tiene Windows como sistema operativo. En caso de tener Mac como sistema operativo, no es necesario instalar Linux. Existen diversas formas de instalar Linux como sistema operativo, una de ellas y la que utilizaremos, es a través de Unetbootin. Primero necesitamos una usb (de 4GB o más). Es aconsejable borrar toda la informacióm que se encuentre almacenada en la usb.

Guia para instalar Linux–Ubuntu a través de Unetbootin:

1. Descargue en su computador el programa Unetbootin del siguiente enlace: https://unetbootin.github.io/

2. Con la usb conectada a su PC ejecute el programa unetbootin-windows-677.exe

3. Seleccione la distribución de Linux que desea instalar. Se aconseja instalar la distribución Ubuntu versión 16.04_Live_x64.



Al dar click en el botón OK unetbootin instalará la distribución Ubuntu versión 16.04_Live_x64 en la unidad usb.

4. Ahora que tenemos el sistema operativo Ubuntu en la usb, necesitamos arrancar el computador desde la usb. Para ello es necesario reiniciar o apagar/encender el computador. Una vez inicia el computador y se encuentre en la pantalla de arranque, el menú de arranque le debería indicar qué tecla necesita oprimir para entrar al menú de arrangue. Generalmente esta tecla es una de las siguientes teclas: F12, F1, F2, F8, F9, F10, TAB o ESC. También cabe la posibilidad de que la pantalla de arranque de Windows no te diga qué tecla presionar para el menú de arranque (Boot Menu), en tal caso no queda otra opción más que ir probando cada tecla cada vez que reinicias. Hacer una búsqueda en internet de qué tecla puede servir de acuerdo a las especificaciones de su versión de Windows y/o marca de computador también ayuda.

5. Una vez tengas acceso al sistema de arranque del PC, aveces llamado Boot Menu, deberá navegar a través de él como se le indica hasta poder seleccionar la usb como unidad de arranque primario, es decir que empiece por reconocer primero la usb antes que el disco duro. Esto se logra en muchos casos cambiando de posición en la lista de unidades de las que se puede arrancar. Si su PC cuenta con unidad de CD, ésta te aparecerá como una de las opciones. Finalmente se sale del menú del sistema de arraque guardando los cambios.

6. Ahora que su PC inicia desde la unidad usb, debe reiniciar nuevamente el computador con la usb conectada. En la pantalla de arranque le aparecerá un menú indicándole que ahora es posible iniciar el sistema operativo Linux, en nuestro caso Ubuntu. Puedes elegir entre iniciar el sistema operativo o instalarlo. Si desea puedes inicia el sistema operativo, explorarlo un poco, y luego reiniciar de nuevo pero ahora dando la opción de instalar el Ubuntu en el menú de inicio.

7. Ya casi habrá terminado. Una vez inicia la instalación del nuevo sistema operativo, deberá seguir unos pasos que se le irán indicando. Importante es que el PC esté conectado a la corriente, también a la Internet (en lo posible a través de un cable de red). Durante la instalación se le preguntará por Zona Horaria (recomendado Bogotá), Idioma (Recomendado Inglés), entre otras cosas. **Durante la instalación se detectará que su PC cuenta con Windows como sistema operativo, a lo cuál deberá decidir si haces una partición (en tal caso tu PC quedará con los dos sistemas operativos, Linux y Wondows), o si prefiere eliminar de forma definitiva el sistema operativo Windows de su PC.**

Si su decición es conservar el Windows, el sistema de instalación le recomedará una partición, la cual es recomendable respetar.

8. Durante la instalación es posible que se pregunte por el nombre (username) de una cuenta y una clave (password o pwd). Deberá recordar la clave que ingrese, pues la necesitará para acceder a la cuenta (username) que habrá creado cuando termine la instalación.

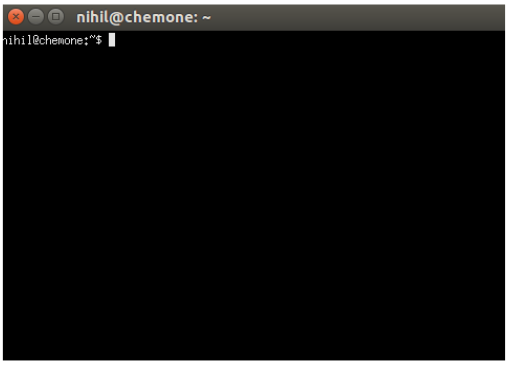
9. Una vez termine la instalación ya tendrá el sistema operativo Linux en su computador y estará preparado para aprender muchas cosas.

10. Para finalizar, apague el computador a través del botón de inicio. Una vez apagado el computador retire la usb. La próxima vez que encienda el PC aparecerá una pantalla de inicio ofreciendole iniciar Linux o Windows (si ha conservado el sistema operativo Windows en su computador), en caso contario iniciará el sistema operativo Linux.

1.2 Familiarizandose con Linux

Una de las primeras cosas que es útil aprender del sistema operativo Linux es a instalar programas. Algunos programas ya vienen instalados, por ejemplo ya cuenta con un navegador de interner (probablemente Firefox), también cuenta con un navegador de archivos desde el que podrá navegar a través del sistema de archivos. Probablemente cuente ya con los directorios `Downloads`, `Documents`, `Desktop`, `Music`, `Videos`, entre otros.
Para instalar programas puede hacerse uso de un *gestor de paquetes* o de una *terminal*. A continuación vamos describir cómo emplear una terminal para instalar programas a la vez que aprovechamos el procedimiento para aprender comandos de terminal.

1. Presione las teclas `Alt` y `F2` al mismo tiempo, es decir: `Alt+F2`. Se debe desplegar una ventana (Run a command). Escriba en ella `xterm` y presione `enter`. Le debe aparecer una terminal, algo como esto:



2. A continuación escriba en la terminal:

```
sudo apt-get install konsole
```

se le pedirá el pwd (clave de tu usuarion). Ingrese el pwd y presione `enter`.

3. Si no le apareció ningún error, es porque acabó de instalar otra terminal, que se llama konsole. Cierre la terminal xterm escribiendo exit y presionando la tecla `enter`.

4. Ahora vamos a abrir la nueva terminal. Presione `Alt+F2`, escriba konsole en la ventana que se despliega y presione la tecla `enter` para abrirás la nueva terminal llamada konsole.

5. Hasta aquí hemos aprendido a instalar programas través de la instucción `sudo apt-get install`. Por ejemplo podemos instalar la aplicación smplayer que sirve para reproducir archivos de audio y video mediante la siguiente instrucción:

   ```
   sudo apt-get install smplayer
   ```

   también podemos instalar la aplicación vlc que también sirve para reproducir archivos de audio y video,

   ```
   sudo apt-get install vlc
   ```

   Podemos ahora instalar los lenguajes de programación Python y Fortan que aprenderemos durante esta asignatura:

   ```
   sudo apt-get install python
   ```

   ```
   sudo apt-get install gfortran
   ```

Es posible que estés interesado en una aplicación en particular, puede consultar en un motor de búsqueda, por ejemplo *google* o *DuckDuckGo*, cómo instalarlo en su sistema operativo a través de `sudo apt-get install`.
Una terminal es una aplicación muy útil para muchas cosas, no sólo para instalar otras aplicaciones. Por ejemplo el comando `cal` (escribir `cal` en una terminal como konsole y presionar la tecla `enter`) imprime una parte del calendario correspondiente al mes en el que estamos. Compruébelo usted mismo. El comando `date` imprime día/mes/hora/zona horaria/año en el que estamos actualmente. Note que una terminal también puede interpretar una secuencia de comandos, de hecho el primer comando que aprendimos fue realmente la secuencia de comandos `sudo apt-get install`.

1.3 Tarea

1. Describir el resultado que arroja cada uno de los siguientes comandos ejecutados en una terminal:

   (a) `ls`
   (b) `cal 2018`
   (c) `cal 12 2019`
   (d) `pwd`
   (e) `echo hola`
   (f) `echo 2+2`
   (g) `echo 2+2 | bc`
   (h) `who`
   (i) `who am i`

2. ¿Qué resultado produce presionar las teclas `ctrl+l` estando la terminal y después de haber aplicado uno de los comandos del inciso anterior?

3. ¿Qué resultado produce presionar las teclas `ctrl++` estando la terminal?

4. ¿Qué resultado produce presionar las teclas `ctrl+-` estando la terminal?

5. Cristobal Colón llegó a América el 12 de octubre de 1492, ¿qué día de la semana fue?

Capítulo 2

El Lenguaje de Programación Python

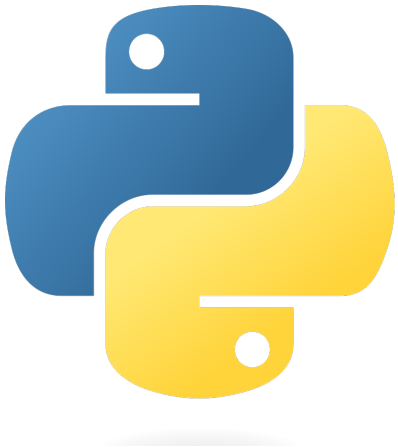


Figura 2.1: Logo (tal vez oficial) de `Python`.

2.1 Introducción

En esta clase daremos comienzo a la programación empleando el lenguaje Python. El objetivo es familiarizarnos con los elementos básicos de un programa. Comenzaremos por analizar algunas instrucciones y tipo de variables que pueden intervenir. Estas notas pueden complementarse con el tutorial `tutorial2.mkv`.

2.2 ¡Hola Mundo!

Nuestro primer programa Python (como el primer programa que se suele escribir en cualquier lenguaje de programación) será escribir "Hola Mundo!" en la pantalla del computador. Para ello abrimos un editor de texto (como `gedit`, aunque recomiendo aprender `vim` pero es más difícil.) y creamos el archivo "hola.py". La extensión .py nos indicará que se trata de un programa Python. Así, en una terminal digitamos:

```
gedit hola.py &
```

y presionamos la tecla enter. Escribir el símbolo ampersand (&) después de una secuencia de comandos nos permite mantener la terminal habilitada para ejecutar otros comandos. Una vez ejecutado el comando anterior, se despliega una nueva ventana en la que podremos editar. Escribimos entonces en el editor de texto los siguiente:

```
print("¡Hola Mundo!")
```

y guardamos el archivo. Para regresar a la terminal empleamos la combinación `alt+Tab`. Esta combinación nos permite ir de una ventana/aplicación a otra. Si ejecutamos el comando `ls` (que vimos en la sesión anterior) conseguiremos una lista de los directorios y archivos, entre ellos debe estar el archivo "hola.py". A continuación ejecutamos nuestro programa Python desde la terminal con la siguiente instrucción:

```
python hola.py
```

como resultado conseguimos que en pantalla aparezca ¡Hola Mundo!. Se recomienda analizar el resultado que produce al ejecutar de nuevo el programa hola.py con las siguientes variaciones:

1. ```
   for i in range(0,3):
       print("¡Hola Mundo!")
   ```

2. ```
   for i in range(0,3):
       print("¡Hola Mundo!",i)
   ```

3. ```
   for i in range(2,4):
       print("¡Hola Mundo!",i)
       print("¡Chao Mundo!",i)
   ```

4. ```
   for i in range(2,4):
       print("¡Hola Mundo!",i)
     print("¡Chao Mundo!",i)
   ```

5. ```
   texto=("¡Hola Mundo!")
     for i in range(2,4):
       print(texto,i)
   ```

2.3 Tipos de variables

En la sección anterior empleamos dos tipos de variables (sin saberlo), la variable `texto` que es una variable tipo string `str` o cadena de caracteres y la variable `i` que es de tipo integer `int` o numérica entera. Existen muchos tipos de variables, algunas las iremos aprendiendo a través de la asignatura. Un concepto interesante en Python son las *tuplas*. El siguiente ejemplo ilustra el uso de tuplas:

```
names = ("Carolina","John","Andres","Diana")
for k in names:
  print("Hola",k)
```

En sí una tupla es un conjunto ordenado e inmutable de elementos del mismo o diferente tipo. Una tupla es similar a un vector, la diferencia está en que los vectores son conjuntos de elementos pero del mismo tipo. Así, en el ejemplor anterior, `names` es tupla y vector a la vez. En el ejemplo:

```
z=int(17)
orbitales = [("1s", 2), ("2s", 2), ("2p", 6), ("3s", 2),("3p",6),("4s",2)]
conf=str("Configuración: ")
for orbital, maxe in orbitales:
  k = min(maxe, z)
```
```

```
  conf=str(conf)+str(orbital)+str("(")+str(k)+str(")")
  z = z-k
  if z <= 0:
    break
print(conf)
```

`z` es una variable tipo entero (tipo `int`), `orbitales` es una tupla que se compone de variables tipo cadena de caracteres (`str`) y tipo entero (`int`). El programa anterior escribe la configuración electrónica para un átomo de $Z = 17$. Compruébelo usted mismo.

2.4 Configuración electrónica

La configuración electrónica de un átomo hace referencia al orden de llenado de los orbitales atómicos según su energía respetando el principio de exclusión de Pauli. Este principio establece que dos electrones dentro del átomo no pueden tener el mismo conjunto de números cuánticos. Los números cuánticos son 4 y están dados por:

- Número cuántico principal n, toma valores desde 1 hasta ∞ (en la práctica llega hasta 7).
- Número cuántico de momento angular ℓ, toma valores desde 0 hasta $n-1$ y se suele representar mediante el siguiente código de letras $\ell = 0 \rightarrow$ s, $\ell = 1 \rightarrow$ p, $\ell = 2 \rightarrow$ d, $\ell = 3 \rightarrow$ f, $\ell = 4 \rightarrow$ g, $\ell = 5 \rightarrow$ h, $\ell = 6 \rightarrow$ i, etc...
- Número cuántico de poyección del momento angular (sobre el eje z) m_ℓ, toma valores desde $-\ell$ hasta $+\ell$ con variaciones de 1. Ejemplo: para $\ell = 1$, m_ℓ toma valores de $-1, 0, 1$.
- Número cuántico de proyección de espín m_s, puede tomar dos valores $-1/2$ y $+1/2$.

Así, cada electrón en el átomo se representa mediante la tupla (n, ℓ, m_ℓ, m_s). Recuerde que el orden de llenado de orbitales viene dado por el orden creciente de la cantidad $n + \ell$, y en caso de haber igualdad entre dos o más elementos de la serie se llenará primero el de mayor valor de ℓ. Ejemplo: entre los orbitales 2s y 2p, se llena primero el orbital 2s ya que $n + \ell = 1 + 1 = 2$ mientras que para 2p $n + \ell = 2 + 1 = 3$. En la serie (3d, 4s, 4p) el orden de llenado es (4s, 3d, 4p) ya que ($n + \ell = 4 + 0$, $n + \ell = 3 + 2$, $n + \ell = 4 + 1$)=(4, 5, 5). Note que el orbital 3d se llena primero que el orbital 4p por tener un valor mayor de ℓ.

2.5 Tarea

1. Escribir un programa Python tal que al ejecutarse solicite un número atómico, y que con esta información escriba en pantalla la configuración electrónica del átomo correspondiente.
2. ¿Cuál es la configuración electrónica del Berkelio ${}_{97}$Bk?
3. ¿Cuál es la configuración electrónica del Meitnerio ${}_{109}$Mt?
4. ¿Cuál es la configuración electrónica del Oganesón ${}_{118}$Og?

Figura 2.2: @BotFather, padre de los Bots.

2.6 Introducción a los Bots

2.6.1 Python-Telegram-Bot

En esta sección explicaremos cómo crear un Bot de Telegram usando Python. Emplearemos el código que construye la configuración electrónica a partir de un valor z como programa funcional del bot. Para construir el Bot en Python necesitamos instalar una librería, la Python-Telegram-Bot. Para instalar la librería ejecutamos la siguiente secuencia de comandos en una terminal (konsole):

```
sudo apt update
sudo apt install python3-pip
pip install python-telegram-bot==13.13
```

luego escribimos en un arichivo (por ejemplo mybot.py) el siguiente código Python:

```
from telegram.ext import Updater, InlineQueryHandler, CommandHandler
my_token = '???'
def get_config(z):
  code=['s','p','d','f','g','h','i']
  orb=[]
  for n in range(1,8):
    for l in range(0,n):
      orb.append((n+l,n,2*(2*l+1)))
  orb=sorted(orb)
  config=str( )
  for nl, n, maxe in orb:
    k = min(maxe, z)
    config=str(config)+str(n)+str(code[nl-n])+str('(')+str(k)+str(')')
    z = z-k
    if z <= 0:
      break
  return config
def elec_config(update, context):
  zstr = " ".join(context.args)
  zvec = zstr.split()
  for z in zvec:
    config = get_config(int(z))
    answer = str("C.E. Z=")+str(int(z))+str(": ")+str(config)
    update.message.reply_text(answer)
def main():
  updater = Updater(my_token, use_context=True)
  dp = updater.dispatcher
  dp.add_handler(CommandHandler('aufbau',elec_config, pass_args=True))
  updater.start_polling()
  updater.idle()
if __name__=='__main__':
  main()
```

Ahora debemos conseguir el `token` que va en la línea 4 de nuestro programa Bot. Para ello buscamos en Telegram al padre de los Bots como si se tratara de un contacto. Buscamos entonces a @BotFather el cual tiene el "Avatar" que aparece en la Figura 2.2.

Para iniciar una "conversación" con @BotFather le escribimos

```
/start
```

Podemos seguir sus instrucciones o bien podemos escribirle

```
/newbot
```

y @BotFather nos preguntará por el nombre que queremos ponerle a nuestro Bot, debe terminar en Bot. Por ejemplo podemos llamarlo `ConfigElecBot`. Usted puede escoger el nombre que más le guste para su Bot pero debe terminar en Bot. Una vez le demos el nombre de nuestro Bot a @BotFather, éste nos regresará el `token` justo después de `Use this token to access the HTTP API:`. Debemos copiar ese `token` e ingresarlo en la línea 4 de nuestro programa Python, es decir `my_token = "token"`. Finalmente ejecutamos nuestro programa Python en un terminal y buscamos en Telegram nuestro Bot, en este ejemplo buscamos a `@ConfigElecBot` entre los contactos y le escribimos

```
/aufbau 59
```

Nuestro Bot nos regresará la configuración electrónica para 59 electrones. Tenga en cuenta que si nuestro programa Python no se está ejecutando, o si el computador donde se está ejecutando no está conectado a la red, es como si nuestro Bot no estuviera disponible y por tanto no respoderá a nuestros requerimientos en Telegram. Lo ideal sería ejecutar el programa Bot en un servidor.

Capítulo 3

Matrices, Vectores y Balance de Ecuaciones Químicas

3.1 Introducción

En esta clase continuaremos con la programación en Python. Aprenderemos a definir y manipular matrices y vectores. En particular nos centraremos en determinar el espacio nulo de una matriz. Encontraremos una aplicación inmediata de estas operaciones en el balance de ecuaciones químicas a través de la matriz de reacción. Estas notas deben complementarse con el tutorial `tutorial3.mkv`.

3.2 Definiendo matrices y vectores

Antes de entrar a declarar matrices y vectores en Python es conveniente primero discutir las cantidades reales, también llamadas cantidades de punto flotante o simplemente floats. Vimos que las cantidades cadena de caracteres se especifican por `str( )` y las cantidades enteras por `int( )`. Para definir una cantidad de punto flotante empleamos `float( )`. Es posible convertir cantidades enteras a cantidades de punto flotante y viceversa. El programa Python:

```
x = 27.21
print(x,x/2)
y = int(x)
print(y)
x = float(y)
print(x)
```

escribe en pantalla la cantidad real 27.21, su mitad 13.605, el entero 27 y el real 27.0. Compruébelo usted mismo. Note del ejemplo anterior como se pasa de una cantidad de punto flotante a un entero y luego de nuevo a punto flotante. En el proceso se pierde la parte decimal 0.21.

Pasemos ahora a la declaración de matrices. Para manipular matrices es conveniente instalar el módulo `numpy` (Numerical Python), es posible que ya lo tenga, de cualquier forma ejecute en una terminal (konsole) las siguiente instrucción:

sudo apt-get install python-numpy

ahora cuenta con el módulo `numpy`. Este módulo nos permitirá multiplicar fácilmente matrices entre otras muchas cosas. Para importar `numpy` y declarar un acrónimo para él empleamos la instrucción

`import numpy as np` dentro del programa que vamos a escribir (name.py). Luego las matrcies que son arreglos de elementos (en general esos elementos son números) se declaran a través de la instrucción np.array( ). Por ejemplo, la instrucción `A = np.array([[1,1],[1,-1]])` declara la matriz:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.1}$$

Por otro lado un vector puede declararse como un arrelgo matricial, ya sea de una única columna (vector columna) o una única fila (vector fila). Por ejemplo, la instrucción `x = np.array([[7],[2]])` declara el vector columna:

$$\mathbf{x} = \begin{pmatrix} 7 \\ 2 \end{pmatrix} \tag{3.2}$$

mientras que la instrucción `y = np.array([[3,0]])` declara el vector fila:

$$\mathbf{y} = \begin{pmatrix} 3 & 0 \end{pmatrix}. \tag{3.3}$$

El programa que se muestra a continuación declara las matrices A y los vectores $\mathbf{x}$ e $\mathbf{y}$. Además escribe en pantalla la matriz A y los vectores columna y fila $\mathbf{x}$ y $\mathbf{y}$ respectivamente.

```
import numpy as np
A = np.array([[1,1],[1,-1]])
print("A=\n",A,"\n")
x = np.array([[7],[2]])
print("x=\n",x,"\n")
y = np.array([[3,0]])
print("y=\n",y,"\n")
```

Compruebe el funcionamiento del programa anterior y analice el efecto de utilizar `print(A)`, `print(x)`, y `print(y)` en lugar de lo que se tiene en el programa.

Ejercício: complete el programa anterior para calcular las cantidades $B = \text{np.dot}(A, X)$, $C = \text{dp.dot}(A, y)$, $D = \text{np.dot}(B, C)$, $S = \text{np.dot}(x, A)$, $T = \text{dp.dot}(y, A)$, $U = \text{np.dot}(S, T)$. La multiplicación de matrices y vectores se lleva a cabo a través del producto interno (np.dot). ¿Qué puede infereir de las operaciones $\text{dp.dot}(A, x)$ y $\text{np.dot}(x, A)$?, ayuda: trate de hacer las multiplicaciones usted mismo (lápiz y papel).

3.3 Balance de ecuaciones químicas

Considere la siguiente reacción química

$$a\,\mathrm{H_2} + b\,\mathrm{O_2} \longrightarrow c\,\mathrm{H_2O} \tag{3.4}$$

donde los coeficientes a, b y c garantizan la conservación de materia (ausencia de reacciones nucleares) y balancean la ecuación química. Así, el conjunto de coeficientes satisfacen las suiguientes ecuaciones algebraicas:

$$2a + 0b = 2c \quad \text{conservación del elemento H} \tag{3.5}$$

$$0a + 2b = 1c \quad \text{conservación del elemento O} \tag{3.6}$$

que pueden reordenarse de la siguiente manera:

$$2a + 0b - 2c = 0 \tag{3.7}$$
$$0a + 2b - 1c = 0 \tag{3.8}$$
$$\tag{3.9}$$

para finalmente escribirlas de la siguiente forma matricial

$$\begin{pmatrix} 2 & 0 & -2 \\ 0 & 2 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.10}$$

Si definimos ahora

$$A = \begin{pmatrix} 2 & 0 & -2 \\ 0 & 2 & -1 \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{3.11}$$

nos encontramos con la siguiente ecuación matricial

$$A\mathbf{x} = 0. \tag{3.12}$$

A la matrix A la llamaremos *matriz de reacción*, dado que contiene información sobre la relación en cantidad que existe entre los elementos que constituyen las sustancias que participan en la reacción. En matemáticas al vector $\mathbf{x}$ se le llama *espacio nulo* de la matriz A, dado que al multiplicar la matriz A por el vector $\mathbf{x}$ se obtiene un vector nulo o de componentes cero.
Nuestro problema de balancear la ecuación química lo hemos transformado en un problema de álgebra lineal que consiste en encontrar el espacio nulo de una matriz. Para resolver este problema con Python necesitamos, además de manipular matrices, poder resolver problemas de álgebra lineal. Para ello empleamos el paquete `linalg` del módulo `scipy` (Scintific Python). Para importat `linalg` a nuestro programa python empleamos la instrucción `from scipy import linalg`. Para instalar el módulo `scipy` utilizamos en una terminal la siguiente instrucción:

sudo apt-get install python-scipy

Hasta este momento hemos instalado en nuestro computador dos módulos de python, el `numpy` y el `scipy`. El siguiente programa Python encuentra los coeficientes para la reacción (3.4):

```
import numpy as np
from scipy import linalg
print("H2 + O2 -> H2O")
A=np.array([[ 2, 0,-2],
            [ 0, 2,-1]])
x=linalg.null_space(A)
print(x)
xmin=min(x)
x=x/xmin
print(x)
```

las instrucciones `import numpy as py` y `from scipy import linalg` nos permiten operar con matrices y encontrar el espacio nulo de una matriz respectivamente. La matriz de reacción de la ecaución (3.4) la definimos mediante la instrucción `A=np.array([[2,0,-2],[0,2,-1]])`. La instrucción `x=linalg.null_space(A)` encuentra el espacio nulo de A y lo almacena en $\mathbf{x}$. El

resultado es $\mathbf{x} = (0.666.., 0.333..., 0.666...)$. No debemos sorprendernos ya que cualquier múltiplo de $\mathbf{x}$ es también solución a la ecuación matricial. Es decir, $\mathbf{x}' = \lambda\mathbf{x}$ con λ un número real es también solución a la ecaución (3.12). Para encontrar la solución que solemos utilizar en las ecuaciones químicas basta con encontrar el número más pequeño (valor mínimo) de los elementos de $\mathbf{x}$ y dividir por ese número. Para ello empleamos la instrucción `xmin=min(x)`. La función `min` la vimos en el capítulo sobre configuración electrónica. Así, la solución que nos interesa es `x = x/xmin` que nos proporciona el resultado $\mathbf{x} = (a, b, c) = (2, 1, 2)$. Compruébelo usted mismo. Finalmente la reacción (3.4) la escribimos como:

$$2\,H_2 + O_2 \longrightarrow 2\,H_2O. \tag{3.13}$$

3.4 Ejercicios

Para cada una de las siguientes reacciones químicas, escriba la matriz de reacción y mediante un programa Python encuentre el espacio nulo que balancea la ecuación.

1. $CO + O_2 \longrightarrow CO_2$
2. $NH_3 + O_2 \longrightarrow NO + H_2O$
3. $KI + KClO_3 + HCl \longrightarrow I_2 + H_2O + KCl$
4. $Cu + HNO_3 \longrightarrow Cu(NO_3)_2 + NO + H_2O$
5. $Ni(NO_3)_2 + NaOH \longrightarrow Ni(OH)_2 + NaNO_3$
6. $[Cr(N_2H_4CO)_6]_4[Cr(CN)_6]_3 + MnO_4^- + H^+ \longrightarrow Cr_2O_7^{2-} + Mn^{2+} + CO_2 + NO_3^- + H_2O$

3.5 Espacio nulo y nulidad de una matriz

Vimos que el problema de balancear una ecuación química puede transformarse en un problema de álgebra lineal. De esta forma los coeficientes estequiométricos que balancean la ecuación química se pueden encontrar determinando el espacio nulo $\mathbf{x}$ de la matriz de reacción A, mediante la ecuación $A\mathbf{x} = 0$. Formalmente pueden haber varios vectores $\mathbf{x}$ que satisfacen $A\mathbf{x} = 0$. Al número de vectores que satisfacen esta ecuación se le llama *nulidad de* A o dimensión del espacio nulo de A. Se puede determinar la nulidad de A previo al cálculo de los vectores $\mathbf{x}$ mediante la siguiente ecuación

$$\text{Nulidad}(A) = \text{Columnas}(A) - \text{Rango}(A) \tag{3.14}$$

donde el Rango de A es el número de columnas linealmente independientes. Para calcular la nulidad de una matriz en Python empleamos las isugientes instrucciones:

```
ran = np.linalg.matrix_rank(A)
col = A.shape[1]
nul = col - ran
```

donde `ran`, `col`, y `nul` corresponden al rango, número de columnas y nulidad de A respectivamente. Si $nul(A) = 0$ entonces la ecuación química asociada a la matriz de reacción A no se puede balancear. Si $nul(A) \geq 2$ entonces la ecuación química se puede balancear de tantas formas como la nulidad de A. Si $nul(A) = 1$ entonces hay una única solución o manera de balancear la ecuación química. Compurebe que la nulidad de todas las matrices de los ejercicios propuestos es igual a uno.

Ejemplo: Considere la siguiente reacción química,

$$CO + CO_2 + H_2 \longrightarrow CH_4 + H_2O \tag{3.15}$$

la matriz de reacción asociada es:

$$A = \begin{pmatrix} 1 & 1 & 0 & -1 & 0 \\ 1 & 2 & 0 & 0 & -2 \\ 0 & 0 & 2 & -4 & -2 \end{pmatrix} \tag{3.16}$$

puede demostrarse que $nul(A) = 2$. Compurébelo usted mismo. También puede demostrarse[1] (aunque está por fuera del alcance de este curso) que las dos soluciones linealmente independientes son:

$$CO_2 + CH_4 \longrightarrow 2\,CO + 2\,H_2 \tag{3.17}$$
$$CO + H_2O \longrightarrow CO_2 + H_2 \tag{3.18}$$

Combinaciones lineales de estas dos soluciones también son soluciones al problema inicial, por ejemplo la combinación de la reacción (3.17) multiplicada por 2 más la reacción (3.18) multiplicada por 3 nos da

$$CO + CO_2 + 7\,H_2 \longrightarrow 2\,CH_4 + 3\,H_2O. \tag{3.19}$$

3.6 Tarea

Para cada una de las siguientes reacciones químicas, escriba la matriz de reacción y mediante un programa Python encuentre la nulidad de la matriz. Para aquellos casos en que $nul(A) = 1$ encuentre el espacio nulo que balancea la ecuación. Recuerde importar los módulos `scipy`, `numpy`, y `math` a su programa.

1. $HIO_3 + FeI_2 + HCl \longrightarrow FeCl_3 + ICl + H_2O$
2. $S + HNO_3 \longrightarrow H_2SO_4 + NO_2 + H_2O$
3. $FeS + HNO_3 \longrightarrow Fe_2(SO_4)_3 + NO + H_2SO_4$
4. $NaHCO_3 + H_3C_6H_5O_7 \longrightarrow CO_2 + H_2O + Na_3C_6H_5O_7$
5. $CuSCN + KIO_3 + HCl \longrightarrow CuSO_4 + KCl + HCN + ICl + H_2O$
6. $Cr_7N_{66}H_{96}C_{42}O_{24} + KMnO_4 + H_2SO_4 \longrightarrow K_2Cr_2O_7 + K_2SO_4 + MnSO_4 + CO_2 + KNO_3 + H_2O$

Las reacciones 1, 5, y 6 se conocen como Redox Challenges: Good Times for Puzzle Fanatics[2],[3].

[1] L. R. Thorne, *Chem. Educator* **2010**, 15, 304
[2] R. Stout, *J. Chem. Educ.***1995**, 72, 1125
[3] David M. Hart, *J. Chem. Educ.* **1996**, 73, A226

Capítulo 4

La librería MatPlotLib, Distribución Maxwell-Boltzmann y Diagramas Tanabe-Sugano

4.1 Introducción

Esta sesión la dedicaremos al aprendizaje de la librería o módulo `MatPlotLib` de `Python`. La librería `MatPlotLib` nos permite hacer gráficos que nos ayudan a visualizar los resultados de un programa. Como escenario utilizaremos la Distribución de Maxwell-Boltzmann. La librería cuenta con una amplia descripción que podemos encontrar en el sitio `https://matplotlib.org/`. Para instalar `MatPlotLib` empleamos el siguiente comando en una consola:

```
apt-get install python-matplotlib
```

o alternativamente los siguientes comandos:

```
python -m pip install -U pip
python -m pip install -U matplotlib
```

4.2 Distribución de Maxwell-Boltzmann

La distribución de Maxwell-Boltzmann fue formulada originalmente para describir la distribución de velocidades de las moléculas (o átomos) que conforman un gas ideal. Así, para un gas compuesto de moléculas de masa m y que se mueven en las tres direcciones del espacio x, y, z con velocidades $\vec{v}_x$, $\vec{v}_y$ y $\vec{v}_z$, el módulo de la velocidad

$$v = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{4.1}$$

podrá tomar valores de acuerdo a la siguiente función de distribución

$$f(v) = 4\pi \left(\frac{m}{2\pi kT}\right)^{3/2} v^2 \exp\left(\frac{-mv^2}{2kT}\right). \tag{4.2}$$

La función $f(v)$ también puede interpretarse como la fracción de moléculas que tienen una velocidad v y por tanto cumple con la condición de normalidad $\int_0^\infty f(v)\mathrm{d}v = 1$.

El siguiente código `Python` crea una gráfica de $f(v)$ vs. v para la molécula N_2 a diferentes temperaturas (100, 200 y 298 K):

```
import numpy as np
import matplotlib.pyplot as plt
m=28.00; KB=1.381e-23; uma=1.661e-27
def disf(m,KB,T,v):
    f=4.0*np.pi*(m/(2.0*np.pi*KB*T))**(1.5)*v**2
    f=f*np.exp(-m*v**2/(2.0*KB*T))
    return f
v=np.arange(0,1000,1)
f1=disf(m*uma,KB,100,v)
f2=disf(m*uma,KB,200,v)
f3=disf(m*uma,KB,298,v)
plt.xlabel("v [m/s]")
plt.ylabel("f(v)")
plt.plot(v,f1,v,f2,v,f3)
plt.show()
```

El siguiente código `Python` crea una gráfica de $f(T, v)$, es decir, la función de distrubución en función de la temperatura y la velocidad para la molécula de N_2:

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
m=28.00; KB=1.381e-23; uma=1.661e-27
def disf(m,KB,T,v):
    f=4.0*np.pi*(m/(2.0*np.pi*KB*T))**(1.5)*v**2
    f=f*np.exp(-m*v**2/(2.0*KB*T))
    return f
v=np.arange(0,1000,1)
T=np.arange(1,300,1)
T,v=np.meshgrid(T,v)
fTv=disf(m*uma,KB,T,v)
fig = plt.figure()
ax=fig.add_subplot(111, projection="3d")
ax=plt.axes(projection="3d")
plt.xlabel('Temperatura [K]')
plt.ylabel('Velocidad [m/s]')
ax.plot_surface(T,v,fTv,cmap='viridis', edgecolor='none')
ax.set_title("f(T,v)")
plt.show()
```

4.3 Cantidades estadísticas de `NumPy`

En la sección anterior vimos cómo calcular la frecuencia de apariencia de una molécula en función de su velocidad. Si consideramos un conjunto de N moléculas, es decir una población, podemos aplicar los conceptos de estadística a esa población. El módulo `NumPy` nos permite calcular algunas

cantidades estadísticas, como el valor medio, la media geométrica, la varianza, la desviación estandar, entre otras. El siguiente código `Python` crea una población de N moléculas y las distribuye de acuerdo a su velocidad como lo determina la distribución de Maxwell-Boltzmann.

```
import numpy as np
import matplotlib.pyplot as plt
m=28.00; KB=1.381e-23; uma=1.661e-27
def disf(m,KB,T,v):
    f=4.0*np.pi*(m/(2.0*np.pi*KB*T))**(1.5)*v**2
    f=f*np.exp(-m*v**2/(2.0*KB*T))
    return f
v=np.arange(0,1000,2)
f=disf(m*uma,KB,298,v)
velocity=[]
for i in range(len(v)):
    x=v[i]
    y=10000*disf(m*uma,KB,298,x)
    for j in range(int(y)):
        velocity.append(v[i])
print(np.mean(velocity),np.median(velocity),np.std(velocity))
plt.bar(v,10000*f,align='center')
plt.show()
```

4.4 Tarea

1. Construya una gráfica de $f(v)$ vs. v que muestre la distribución de velocidades para las moléculas O_2, N_2, H_2O, He, y H_2 a 25 °C. A partir de la gráfica estime la velocidad más probable para cada una de las moléculas mencionadas.

2. Calcule la media, la mediana y la desviación estandar para la velocidad en cada uno de los casos anteriores.

3. Cambie la temperatura a 700 K y repita el procedimiento anterior. Comente sobre los resultados.

4.5 Cálculo de valores propios (diagramas de Tanabe-Sugano)

4.5.1 Introducción

4.5.2 Estados electrónicos de complejos de metales de transición

Capítulo 5

Ecuaciones Diferenciales Ordinarias

5.1 El azúcar invertido

Se le llama azúcar invertido al producto de la hidrólisis de la sacarosa. Durante la hidrólisis la sacarosa se disgrega en glucosa y fructuosa:

$$\underbrace{C_{12}H_{22}O_{11(ac)}}_{\text{Sacarosa}} + H_2O_{(l)} \longrightarrow \underbrace{C_6H_{12}O_{6(ac)}}_{\text{Glucosa}} + \underbrace{C_6H_{12}O_{6(ac)}}_{\text{Fructuosa}} \tag{5.1}$$

El nombre de *invertido* hace referencia a que durante la hidrólisis se invierte el poder rotatorio de la solución frente a la luz polarizada, ya que la sacarosa y la glucosa son dextrógiras mientras que la fructuosa es levógira (una mezcla de cantidades iguales de glucosa y fructuosa resulta levógira). Cuando la solución de azúcar se encuentra diluida, la velocidad de hidrólisis es proporcional a la concentración de la sacarosa, es decir,

$$\frac{\mathrm{d}[S]}{\mathrm{d}t} = -k[S] \tag{5.2}$$

donde $[S]$ es la concentración de sacarosa y k es la constante de proporcionalidad que a 25 °C toma el valor de $k = 0.2283\ \mathrm{h}^{-1}$.

1. Escriba un programa `python` que emplee el paquete de integración `odeint` para encontrar la concentración de sacarosa a cualquier tiempo.

2. Si la concentración inicial de sacarosa es de 1.2 mol/L, determine la concentración de sacarosa para $t = 3, 6$ y 12 horas.

```
import numpy as np
from scipy.integrate import odeint
def rxn(S,t):
    k = 0.2283
    dSdt = -k*S
    return dSdt
S0 = 1.2
S = odeint(rxn,S0,[0,3,6,12])
print(S)
```

3. Use la librería `matplotlib.pyplot` para graficar $[S]$ en el rango $t = [0, 12]$.

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
def rxn(S,t):
    k = 0.2283
    dSdt = -k*S
    return dSdt
S0 = 1.2
t = np.linspace(0,12,100)
S = odeint(rxn,S0,t)
plt.xlabel('Tiempo / h')
plt.ylabel('Conc. / mol/L')
plt.plot(t,S)
plt.show()
```

5.2 Reacciones de los NO_x en la atmósfera terreste

El ozono reacciona con el dióxido de nitrógeno para producir pentóxido de dinitrógeno y oxígeno según se representa en la siguiente ecuación química:

$$2\,NO_2 + O_3 \longrightarrow N_2O_5 + O_2. \tag{5.3}$$

El conjunto de reacciones elementales propuesto para el mecanísmo de la reacción es el siguiente:

- Paso 1: $NO_2 + O_3 \xrightarrow{k_1} NO_3 + O_2$
- Paso 2: $NO_3 + NO_2 \xrightarrow{k_2} N_2O_5$

Escriba un programa `Python` para calcular la concentración de cada especie en el tiempo. Utilice los valores de la tabla 5.1 para estimar los valores de k_1 y k_2 a 300 K y calcule la concentración de NO_2, O_3, NO_3, O_2 y N_2O_5 en el tiempo para rango $t = [0, 1 \times 10^{-5}\text{s}]$ en incrementos de $\Delta t = 1 \times 10^{-7}\text{s}$, cuando se hacen reaccionar 0.01 mol/cm^3 de NO_2 con 0.005 mol/cm^3 de O_3.
El sistema de ecuaciones diferenciales a resolver es:

$$\frac{\mathrm{d}[NO_2]}{\mathrm{d}t} = -k_1[NO_2][O_3] - k_2[NO_3][NO_2] \tag{5.4}$$
$$\frac{\mathrm{d}[O_3]}{\mathrm{d}t} = -k_1[NO_2][O_3] \tag{5.5}$$
$$\frac{\mathrm{d}[NO_3]}{\mathrm{d}t} = k_1[NO_2][O_3] - k_2[NO_3][NO_2] \tag{5.6}$$
$$\frac{\mathrm{d}[O_2]}{\mathrm{d}t} = k_1[NO_2][O_3] \tag{5.7}$$
$$\frac{\mathrm{d}[N_2O_5]}{\mathrm{d}t} = k_2[NO_3][NO_2] \tag{5.8}$$

El siguiente programa (`odeNOx.py`) calcula las concentraciones de cada especie en función del tiempo de la reacción $2\,NO_2 + O_3 \longrightarrow N_2O_5 + O_2$:

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
```

```
def rxn(X,t):
    T  = 300
    k1 = (5.9e+12)*np.exp(-3500/T)
    k2 = (2.3e+12)*np.exp(    0/T)
    r1 = k1*X[0]*X[1]
    r2 = k2*X[2]*X[0]
    dNO2dt  = -r1-r2
    dO3dt   = -r1
    dNO3dt  = r1-r2
    dO2dt   = r1
    dN2O5dt = r2
    return [dNO2dt,dO3dt,dNO3dt,dO2dt,dN2O5dt]
t = np.arange(0,1e-5,1e-7)
X0 = [0.01,0.005,0,0,0]
X = odeint(rxn,X0,t)
fig, axs=plt.subplots(2, sharex=True)
axs[0].set_title('2NO$_2$+O$_3$ -> N$_2$O$_5$+O$_2$')
axs[0].plot(t,X[:,0],'b-',label='NO$_2$')
axs[0].plot(t,X[:,1],'r-',label='O$_3$')
axs[0].plot(t,X[:,3],'g-',label='O$_2$')
axs[0].plot(t,X[:,4],'--',color='orange',label='N$_2$O$_5$')
axs[0].legend(loc='upper right')
axs[0].set_ylabel('Conc. / mol$\cdot$cm$^{-3}$')
axs[1].plot(t,X[:,2],'b--',label='NO$_3$')
axs[1].legend(loc='upper right')
axs[1].set_xlabel('Tiempo / s')
axs[1].set_ylabel('Conc. / mol$\cdot$cm$^{-3}$')
plt.show()
```

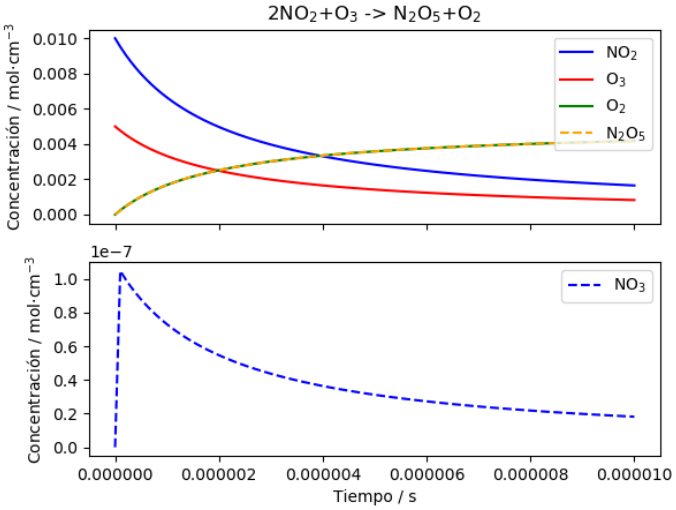


Figura 5.1: Evolución de las especies químicas involucradas en la reacción de ozono con dióxido de nitrógeno.

Tabla 5.1: Tabla de parámetros de Arrhenius para reacciones químicas que ocurren en la combustión. Los valores de temperaturas (en K) indican el rango de validéz de los parámetros de Arrhenius. Los valores de A deben entenderse como $x(\pm y) = x \times 10^{\pm y}$. Las constantes de velocidad se obtienen como $k = AT^B \exp(-E/RT)$. Las unidades de k corresponden a: para reacciones de primer orden $[k]$=s^{-1}, para reacciones de segundo orden $[k]$=cm^{3}mol^{-1}s^{-1}, y para reacciones de tercer orden $[k]$=cm^{6}mol^{-2}s^{-1}. Los factores f y F sirven para determinar las incertidumbres de k según $fk_0 < k < Fk_0$. Datos tomados de: *Table of Recommended Rate Constants for Chemical Reactions Ocurring in Combustion*, Francis Westley, Chemical Kinetics Information Center, National Msasurement Laboratory, National Bureau of Standards, Washintong, D.C. 20234 (Arpil 1980).

Reacción Química	T (en K)	A	B	E/R (en K)	Factores k f	F
$H_2 + O_2 \rightarrow OH + OH$	1500-2500	2.5(+12)	0	19630±5000	0.1	10.
$OH + H_2 \rightarrow H_2O + H$	300-2500	2.2(+13)	0	2590±100	0.8	1.2
$H + H_2O \rightarrow OH + H_2$	300-2500	9.3(+13)	0	10250±100	0.5	1.5
$O + H_2 \rightarrow OH + H$	400-2000	1.8(+10)	1.0	4480±150	0.7	1.3
$H + OH \rightarrow H_2 + O$	400-2000	8.3(+9)	1.0	3500±150	0.7	1.3
$H + O_2 \rightarrow OH + O$	700-2500	2.2(+14)	0	8450±350	0.7	1.3
$O + OH \rightarrow O_2 + H$	1500-2500	2.5(+13)	0	0	0.5	2.0
		6.3(+11)	0.5	0		
$OH + H + N_2 \rightarrow H_2O + N_2$	1000-3000	2.2(+22)	-2.0	0	0.5	2.0
$NO_2 + O_3 \rightarrow NO_3 + O_2$	286-302	5.9(+12)	0	3500	0.5	2.0
$NO_3 + NO_2 + M \rightarrow N_2O_5 + M$	300	2.3(+12)	-	-	0.4	2.5
(bajas presiones)	300	1.0(+18)	-	-	0.5	2.0
$NO_3 + NO_2 \rightarrow NO_2 + NO + O_2$	300-850	1.4(+11)	0	1600±500	0.4	2.5

5.3 Interacción radiación-materia

Cuando la materia interactúa con la radiación electromagnética (luz), se pueden producir cambios en la estructura de la materia debido a la absorción y/o emisión de fotonoes (partículas de luz). La interacción entre la radiación y la materia está governada por la ecuación de Schrödigner:

$$\mathrm{i}\hbar\partial_t \Psi(x,t) = \mathcal{H}\Psi(x,t). \tag{5.9}$$

En el siguiente ejemplo resolveremos la ecuación de Schrödinger para un átomo de hidrógeno inmerso en un campo de radiación electromagnética. En el átomo de hidrógeno las funciones de onda que representan los estados estacionarios corresponen a los orbitales atómicos que dependen de los números cuánticos n, ℓ, m_ℓ, y m_s:

$$\psi_{n\ell m_\ell m_s}(\mathbf{r},t) = R_{n\ell}Y_{\ell m_\ell}(\theta,\phi)e^{-\mathrm{i}E_n t/\hbar}. \tag{5.10}$$

La función de onda de la ecuación (5.9) puede escrirse como

$$\Psi(\mathbf{r},t) = \sum_n c_n(t)\psi_{n\ell m_\ell m_s}(\mathbf{r},t)e^{-\mathrm{i}E_n t/\hbar} \tag{5.11}$$

donde los coeficientes satisfacen el siguiente conjunto de ecuaciones lineales acopladas:

$$\frac{\mathrm{d}}{\mathrm{d}t}c_m(t) = -\frac{\mathrm{i}}{\hbar}\sum_n c_n(t)e^{(E_m-E_n)t/\hbar}\int_{t_0}^{t_f}\psi^*_{m\ell m_\ell m_s}(\mathbf{r})\mathcal{H}^{(\mathrm{int})}(t)\psi_{n\ell m_\ell m_s}(\mathbf{r}). \tag{5.12}$$

Si sólo consideramos los dos primeros niveles del átomo de hidrógeno, el estado fundamental 1s y por ejemplo el estado excitado $2p_0$, entonces el conjunto de ecauciones acopladas se reduce a:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}c_1(t)\\c_2(t)\end{pmatrix} = -\frac{\mathrm{i}}{\hbar}\begin{pmatrix}0 & H_{12}(t)\\H_{21}(t) & 0\end{pmatrix}\begin{pmatrix}c_1(t)\\c_2(t)\end{pmatrix} \tag{5.13}$$

con

$$H_{12} = 0.745(\mathrm{e}a_0)e^{\mathrm{i}(E_1-E_2)t/\hbar}\mathcal{E}(t) \tag{5.14}$$
$$H_{21} = 0.745(\mathrm{e}a_0)e^{\mathrm{i}(E_2-E_1)t/\hbar}\mathcal{E}(t) \tag{5.15}$$
$$\mathcal{E}(t) = \mathcal{E}_0\sin\left(\frac{\pi t}{\tau}\right)\cos\left(\omega\left(t-\tau/2\right)\right) \tag{5.16}$$

$\mathcal{E}(t)$ es el campo eléctrico de la onda electromagnética, $\mathcal{E}_0$ la amplitud de la onda, τ la duración del pulso electromagnético, $\omega = 2\pi\nu = 2\pi hc/\lambda$ la frecuencia angular y λ la longitud de onda. Así, el conjunto de ecuaciones (5.13) se resuelve fácilmente mediante el siguiente programa `tdse.py`:

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
t0 =0; tau = 413.41*1.91; E0 = 0.058802; w = 0.375
def field(t):
    field = E0 * np.sin(np.pi*t/tau)**2 * np.cos(w*(t-tau/2))
    return field
def deriv(t, y):
    dip = 0.745
    E = field(t)
    Hint = np.array([[0+0j, 0-1j*dip*E*np.exp(1j*(-0.5+0.125)*t)],
                     [0-1j*dip*E*np.exp(1j*(-0.125+0.5)*t), 0+0j]])
    return Hint @ y
tf = tau
t = np.linspace(t0, tf, 1001)
result = solve_ivp(deriv, [t0, tf], np.array([1+0j, 0+0j]), t_eval=t)
c1 = result.y[0,:]; c2 = result.y[1,:]
p1 = (c1*np.conj(c1)).real; p2 = (c2*np.conj(c2)).real; norma = p1 + p2
print(c1[0],c2[0],c1[-1],c2[-1])
print(p1[-1],p2[-1],norma[-1])
plt.plot(t,p1,t,p2)
plt.plot(t,field(t))
plt.show()
```

El estado final corresponde a:

$$\Psi(\mathbf{r},t_f) = (-0.70866 + 0.00082\ \mathrm{i})\psi_{100m_s}(\mathbf{r}) + (-0.27702 + 0.65547\ \mathrm{i})\psi_{210m_s}(\mathbf{r}) \tag{5.17}$$

compruébelo usted mismo.

5.4 Ejercicios

1. Considere el siguiente sistema de reacciones elementales entre el metileno y el radical metilo:[1]

$$CH_2 + CH_3 \longrightarrow C_2H_4 + H \quad k_1 = 1.0 \times 10^{-10}\ \text{cm}^3\ \text{molecule}^{-1}\ \text{s}^{-1} \tag{5.18}$$
$$CH_3 + CH_3 \longrightarrow C_2H_6 \quad k_2 = 9.5 \times 10^{-11}\ \text{cm}^3\ \text{molecule}^{-1}\ \text{s}^{-1} \tag{5.19}$$
$$CH_2 + CH_2 \longrightarrow C_2H_2 + H_2 \quad k_3 = 5.3 \times 10^{-11}\ \text{cm}^3\ \text{molecule}^{-1}\ \text{s}^{-1} \tag{5.20}$$
$$CH_3 + H \longrightarrow CH_4 \quad k_4 = 3.4 \times 10^{-10}\ \text{cm}^3\ \text{molecule}^{-1}\ \text{s}^{-1} \tag{5.21}$$

   (a) Escriba el sistema de ecuaciones diferenciales que representa el el proceso de reacciones químicas acopladas. Escriba un programa `Python` que calcule la concentración de cada especie en el tiempo a partir de las concentraciones iniciales.

   (b) Calcule las cantidades de todas las especies después de 100 microsegundos cuando se dejan 8×10^{-9} mol/cm^3 de CH_3 con 4×10^{-9} mol/cm^3 de CH_2. Estime el reactivo limitante de la reacción.

   (c) Repita el numeral anterior pero ahora iniciando con $4 \times 10^{-}9$ mol/cm^3 de CH_3 y $8 \times 10^{-}9$ mol/cm^3 de CH_2.

   (d) Escriba una ecuación química para una de las posibles reacciones globales. ¿Corresponde la estequimetría de la reacción global planteada con alguna de las situaciones de los incisos (b) y (c)? Explique su respuesta.

2. Si se mide la energía del átomo de hidrógeno después de la interacción con el pulso electromagnético, (ver programa `tdse.py`) ¿cuál es la probabilidad de obtener E_1?

3. Modifique los parámetros τ y/o ω del pulso electromagnético en el programa `tdse.py` para preparar un estado $\Psi(\mathbf{r}, t_f)$ en el que la probabilidad de obtener E_2 sea de 1/4. Reporte los valores de τ, ω, y $c_2(t_f)$. Comente sobre cómo se mide esta probabilidad.

[1] A. H. Laufer, and A. M. Bass, "Mechanism and rate constant of the reaction between methylene and methyl radicals", *J. Phys. Chem.* **79**, 1635 (1975)

Capítulo 6

Método de Discretización del Hamiltoniano en una Malla de Fourier (FGHM)

"When one has a particular problem to work out in quantum mechanics, one can minimize the labor by using a representation in which the representatives of the most important abstract quantities occurring in the problem are as simple as possible." P.A.M. Dirac, The Principles of Quantum Mechanics, 1958.

6.1 Descripción del problema

Considere una partícula de masa m que se mueve a lo largo del eje x y confinada en un espacio bajo la acción del potencial $V(x)$, el Hamiltoniano asociado a la partícula viene dado por:

$$\hat{\mathcal{H}} = \frac{\hat{p}^2}{2m} + V(\hat{x}). \tag{6.1}$$

Se desea calcular las funciones propias $\psi_n(x)$ y valores propios E_n del Hamiltoniano resolviendo la ecuación de Schrödinger independiente del tiempo (ecuación de valores propios),

$$\hat{\mathcal{H}}\psi_n(x) = E_n\psi_n(x). \tag{6.2}$$

6.2 Vibraciones de moléculas diatómicas (ecuación de Schrödinger)

Este método emplea el principio variacional en el que las funciones propias del Hamiltoniano $\hat{H}$ se calculan directamente en una malla $x \in [x_{\min}, x_{\max}]$ de N_x puntos.[1] El Hamiltnoniano se discretiza en la malla de puntos según las siguientes ecuaciones:

$$H_{ij} = T_{ij} + V(x_i)\delta_{ij} \tag{6.3}$$

donde $V(x_i)\delta_{ij}$ son los elementos de la matriz de energía potencial, $\delta_{ij} = 0$ si $i \neq j$ y $\delta_{ij} = 1$ si $i = j$, conocida como la delta de Kronecker, T_{ij} corresponde a los elementos de la matriz de energía

[1] C. Clay Marston and Gabriel Balint-Kurti, The Fourier grid Hamiltonian method for bound states eigenvalues and eigenfunctions, *J. Chem. Phys.* **91**, 3571 (1989)

cinética y estan determinados por

$$T_{ij} = \frac{1}{(N_x - 1)m} \sum_{k=1}^{N_p} (k\Delta p)^2 \cos\left(k\Delta p(x_j - x_i)\right) \tag{6.4}$$

con

$$x_i = x_{\min} + (i-1)\Delta x, \qquad \Delta x = \frac{x_{\max} - x_{\min}}{N_x - 1}, \qquad N_p = \frac{N_x - 1}{2}, \qquad \Delta p = \frac{2\pi}{(N_x - 1)\Delta x}. \tag{6.5}$$

```
import numpy as np
import matplotlib.pyplot as plt
#Variables de entrada
mass = 1
xmin = -5.0
xmax =  5.0
Nx = 101
#Cantidades escalares
dx =(xmax-xmin)/(Nx-1)
Np =int((Nx-1)/2)
dp =(2.0*np.pi)/((Nx-1)*dx)
#Cantidades vectoriales
x = np.empty(Nx)
V = np.empty(Nx)
for i in range(Nx):
    x[i] = xmin+i*dx
#Cantidades matriciales
T = np.empty((Nx,Nx))
H = np.empty((Nx,Nx))
#Funci'on de energ'ia potencial
def potosc(ka,x):
    V = 0.5*ka*x**2
    return V
def pot2well(k1,k2,x):
    V = -k1*x**2 + k2*x**4
    return V
#FGH-method
for i in range(Nx):
    for j in range(Nx):
        T[i,j]=0.0
        for k in range(Np):
            T[i,j] = T[i,j] + ((k+1)*dp)**2*np.cos((k+1)*dp*(x[j]-x[i]))
        T[i,j] = T[i,j] /((Nx-1)*mass)
        H[i,j] = T[i,j]
        if i == j:
            V[i] = pot2well(0.5,0.025,x[i])
            H[i,i] = H[i,i] + V[i]
E,C = np.linalg.eig(H)
idx = np.argsort(E)
E = E[idx]
```

```
C = C[:,idx]
F = 3*C+E
for i in range(5):
    print("{0:.3f}".format(E[i]))
plt.plot(x,V[:],x,F[:,0],x,F[:,1],x,F[:,2],x,F[:,3],x,F[:,4],x,F[:,5])
plt.show()
```

6.3 Estructura elecrtónica del átomo de hidrógeno (ecuación de Dirac)

A continuación se presenta el algoritmo FGHM relativista escrito en `Python`:

```
# E. Layton and S-I Chu, Chem. Phys. Lett. 186, 100 (1991)
import numpy as np
from scipy.linalg import eig
import matplotlib.pyplot as plt
from sympy.physics.quantum.cg import wigner_3j

la = np.exp(1.0); c = 137.0359990840

def eigen(Z,kappa,N,m):
    print('kappa',kappa)
    tmin = -16.0
    tmax =   6.0
    dt = (tmax-tmin)/(N-1)
    t   = np.empty((N))
    h11 = np.empty((N,N)); h11[:,:] = 0.0
    h12 = np.empty((N,N)); h12[:,:] = 0.0
    h21 = np.empty((N,N)); h21[:,:] = 0.0
    h22 = np.empty((N,N)); h22[:,:] = 0.0
    H = np.empty((2*N,2*N)); H[:,:] = 0.0
    S = np.empty((2*N,2*N)); S[:,:] = 0.0
    for i in range(N):
        t[i] = tmin + i*dt
        r = la**t[i]
        h11[i,i] = -Z/c + m*c*r
        h12[i,i] = kappa*1.0
        h21[i,i] = kappa*1.0
        h22[i,i] = -Z/c - m*c*r
    for i in range(N):
        for j in range(N):
            acum = 0.0
            for k in range(int((N-1)/2)):
                acum = acum + k*np.sin(2*np.pi*k*(i-j)/N)
            h12[i,j] = h12[i,j] + (4*np.pi/(dt*N**2))*acum
            h21[i,j] = h21[i,j] - (4*np.pi/(dt*N**2))*acum
    for i in range(N):
        for j in range(N):
            H[i,j] = h11[i,j]
            H[i,j+N] = h12[i,j]
            H[i+N,j] = h21[i,j]
            H[i+N,j+N] = h22[i,j]
        S[i,i] = la**t[i]/c
        S[i+N,i+N] = la**t[i]/c
    H = H*np.log(la)
```

```
    S = S*np.log(la)
    E,C = eig(H,S)
    idx = np.argsort(E)
    E = E[idx] - m*c**2
    C = C[:,idx]
    return t,E,C

Z = 1; N = 201; m = 1.0
k = -1; t,E,C1s = eigen(Z,k,N,m); print('E',E[N])
k = +1; t,E,C2pm = eigen(Z,k,N,m); print('E',E[N+1])
k = -2; t,E,C2pp = eigen(Z,k,N,m); print('E',E[N])
```

6.4 Algoritmo FGHM en Fortran

1. Se define la masa m de la partícula, los valores extremos $x_{\min}$ y $x_{\max}$, y el número de puntos de la malla N_x (numero impar). Estos son valores de entrada para el programa.

2. Se calculan las cantidades escalares Δx, N_p y Δp.

3. Se calculan los elementos x_i en un arreglo vectorial:

$$x_i = x_{\min} + (i-1)\Delta x \tag{6.6}$$

4. Se calculan los elementos T_{ij} y V_{ij} y se almacenan en un arreglo matricial de $N_x \times N_x$.

5. Se calculan las matrices H y S según

$$H = T + V \tag{6.7}$$
$$S = (\delta_{ij}) \tag{6.8}$$

6. Se resuelve el sistema de ecuaciones

$$(H - SE)\,C = 0 \tag{6.9}$$

   donde H y S son cantidades conocidas, mientras que E (vector de dimensión N_x) y C (matriz de dimensión $N_x \times N_x$) son las cantidades a calcular. Para calcular E y C a partir de H y S se emplea la subrutina `DSYGV` de la librería LAPACK[2] de la siguiente manera

```
INTEGER :: LWORK, INFO
REAL*8, ALLOCATABLE :: WORK(:)

LWORK=3*Nx
ALLOCATE(WORK(LWORK))
CALL DSYGV(1,'V','U',Nx,H,Nx,S,Nx,E,WORK,LWORK,INFO)
C=H
DEALLOCATE(WORK)
```

El programa se compila a la vez que se enlaza a las librerías de LAPACK mediante la siguiente instrucción:

```
gfortran fghm.f -llapack -o fghm.exe
```

Las librerías de LAPACK se pueden instalar con la siguiente instrucción:

```
sudo apt-get install liblapack-dev
```

[2]LAPACK: Linear Algebra PACKage.
Para saber más sobre LAPACK se puede consultar el sitio *www.netlib.org/lapack*

6.5 Tarea

1. Reproduzca la tabla II del artículo "C. Clay Marston and Gabriel Balint-Kurti, The Fourier grid Hamiltonian method for bound states eigenvalues and eigenfunctions, *J. Chem. Phys.* **91**, 3571 (1989)" mediante un programa `Python` y un programa `Fortran`.

2. Empleando el comando `time` compare el desempeño de `Python` y `Fortran` a la hora de calcular los valores propios y vectores propios de una matriz. Reporte los tiempos encontrados para $N_x = 101$, $N_x = 301$ y $N_x = 501$.

Capítulo 7

Estructura electrónica (método de Hartree-Fock)

7.1 Introducción

7.2 Conjunto de funciones base

Capítulo 8

Introducción a la Computación Cuántica

8.1 Qiskit

```
import numpy as np
import matplotlib.pyplot as plt
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.quantum_info import Statevector
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram, plot_bloch_multivector
a=-0.5
b=np.sqrt(1-a**2)
print('|a|^2=',a**2,',|b|^2=',b**2)
qr = QuantumRegister(3, name='q')
crx, crz = ClassicalRegister(1,name='crx'), ClassicalRegister(1,name='crz')
qc = QuantumCircuit(qr, crz, crx)
qc.initialize([a,b],0) #Creates Alice's atom H*
qc.initialize([0, 1/np.sqrt(2), 1/np.sqrt(2), 0], [1,2]) #Creates EPR pair
qc.cx(0,1) #RF waves
qc.h(0) #UV-Vis
#Bob builts up H* state
qc.barrier()
qc.measure(1,1)
qc.x(2)
qc.x(2).c_if(crx,1)
qc.barrier()
qc.measure(0,0)
qc.z(2).c_if(crz,1)
##
print(qc.draw(initial_state=True,filename='AliceBob.txt',
        vertical_compression='low')) #Prints circuit
##
sim = AerSimulator()
qc.save_statevector()
job = sim.run(qc,shots=1000,memory=False)
state = job.result().get_statevector()
print(state)
```

```
counts = job.result().get_counts()
print(counts)
plot_histogram(counts)
plt.show()
```

Capítulo 9

El Lenguaje de Programación Fortran

9.1 Introducción al lenguaje de programación `Fortran`

9.2 La librería de álgebra lineal `LAPACK`

9.3 Implementación del método de campo autoconsistente de Hartee-Fock

Capítulo 10

Introducción a los Microcontroladores

Figura 10.1: Logo (probabemente oficial) de `Arduino`.

10.1 El microcontrolador `Arduino`

10.2 Uso de fotorresistores `LDR`

10.3 Construcción de un colorímetro

10.3.1 Ley de Beer-Lambert

10.4 Uso del medidor de humedad y temperatura `DTH22`

10.4.1 Determinación de la rapidez de hidratación de una muestra café comercial