Portafolio

Física Computacional II

Franco Nicolás Giovine López
Ciencias Físicas
Despartamento de Física
Facultad de Ciencias Físicas y Matemáticas
UNIVERSIDAD DE CONCEPCIÓN

Agosto-Diciembre 2022
Concepción, Chile

Profesor Guía: Dr. Roberto Navarro Maldonado

Índice general

Introducción

Este portafolio deberá incluir muestras o evidencias de las actividades desarrolladas en la asignatura de Física Computacional II (510240), dictada en el segundo semestre de 2022 en el departamento de física, facultad de Ciencias Físicas y Matemáticas, de la Universidad de Concepción.

Esta asignatura se enfoca en la resolución de problemas en Física mediante lenguajes de programación especializados (en este caso `python`), contribuyendo a conocimientos generales de construcción y aplicación de algoritmos computacionales, optimización de simulaciones numéricas y organización de códigos mediante herramientas computacionales. Además, la metodología y objetivos del curso se enmarcan en el proyecto de docencia "Mejorando el aprendizaje de competencias en computación científica y la elaboración de informes que comuniquen resultados científicos para estudiantes de pregrado de Licenciatura en Física", financiado por el fondo concursable Innovando en Red UCO 20102 de la Universidad de Concepción, cuyo objetivo es mejorar el aprendizaje en competencias de programación científica utilizando herramientas actuales para el control de cambios en códigos computacionales, como el uso de Git, Github y LaTeX y mejorar las competencias en elaboración de informes académicos, lo cual se espera conseguir con la implementación de este portafolio.

Esta asignatura contribuye al logro de las siguientes competencias del perfil de egreso "Aplicar lenguajes de programación avanzados tales como Fortran, C++, Python, etc., a problemas de la Física", y se espera obtener los siguientes resultados de aprendizaje:

1. Aplicar las herramientas computacionales en la resolución numérica de problemas en Física.
2. Generar programas computacionales apoyándose en algoritmos y conceptos de la física matemática y estadística.
3. Diferenciar, integrar y resolver ecuaciones diferenciales ordinarias, en forma numérica.

La evaluación se realizará a través de este portafolio. El profesor evaluará cada una de las muestras y el conjunto del portafolio a través de una pauta que se entregará oportunamente. El/la estudiante deberá auto-evaluar su portafolio y seleccionará un conjunto de evidencias de este portafolio, las cuales serán expuestas ante sus compañeros a final del semestre, quienes también lo evaluarán a través de una rúbrica. El documento de portafolio contendrá una sección de conclusiones donde autoevaluará su desempeñoa través de una reflexión final y de sus respuestas a una serie de preguntas que serán consensuadas con el profesor.

Como criterios de evaluación, se considerará:

1. Adecuación de las muestras incluidas a los resultados del aprendizaje y a los contenidos de la asignatura.
2. Coherencia de las evidencias con las actividades realizadas en clases.
3. Competencias comunicativas: escritura (ortografía, gramática, sintaxis, estilo) y presentaciones orales (habla correctamente y con claridad; utiliza términos técnicos apropiadamente, etcétera).
4. Presentación: Claridad, limpieza y orden del documento.
5. Autoevaluación: Seriedad y reflexión de las justificaciones dadas al principio y al final del documento.

Capítulo 1

Presentación del estudiante

Datos personales

Nombre completo	Franco Nicolás Giovine López
Matrícula	2021431560
Fecha de Nacimiento	07 de Junio de 2002
Nacionalidad	Chilena

Breve biografía académica

Soy Franco, estudiante de segundo año de la carrera de Cs. Físicas. La educación media la realicé en el colegio de los Sagrados corazones de la ciudad de Concepción. He vivido toda mi vida en Hualpén y mi comida favorita es el pastel de choclo.

Visión general e interés sobre la asignatura

El ramo de Física Computacional II nos dará un un conocimiento mas sólido sobre la herramientas computacionales que necesitaremos en un futuro cuando nos desarrollemos como profesionales. Mi interés en este ramo es reforzar los vagos conocimientos que ya tengo en computación.

Resultados esperados de este portafolio

Espero que a fin de semestre cuando este portafolio esté completo y lo lea me de cuenta de todo lo que he aprendido durante este curso. También espero poder sentirme orgulloso de los avances que he logrado programando.

Capítulo 2

Actividades de laboratorio

2.1. Derivada numérica

Fecha de la actividad: 19 de agosto de 2022

2.1.1. Objetivo de la clase

El objetivo de esta actividad fue aprender a calcular derivadas numéricas de forma analítica. También se buscó aprender como calcular los errores y minimizarlos con el fin de mejorar los resultados.

2.1.2. Desarrollo del laboratorio

Se demostró que la segunda derivada centrada puede ser escrita como:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2). \tag{2.1}$$

Su demostración es la siguiente:

Se tiene que su segunda derivada adelantada es:

$$\begin{aligned} f(x+h) &= f(x) + f'(x)h + \frac{f''(x)h^2}{2!} + \frac{f'''(x)h^3}{3!} + \frac{f^4(x)h^4}{4!} + \cdots \\ f(x+h) &= f(x) + f'(x)h + \frac{f''(x)h^2}{2!} + \frac{f'''(x)h^3}{3!} + \frac{f^4(\xi^+)h^4}{4!}. \end{aligned} \tag{2.2}$$

Por otra parte, su segunda derivada atrasada es:

$$\begin{aligned} f(x-h) &= f(x) - f'(x)h + \frac{f''(x)h^2}{2!} - \frac{f'''(x)h^3}{3!} + \frac{f^4(x)h^4}{4!} + \cdots \\ f(x-h) &= f(x) - f'(x)h + \frac{f''(x)h^2}{2!} - \frac{f'''(x)h^3}{3!} + \frac{f^4(\xi^-)h^4}{4!}, \end{aligned} \tag{2.3}$$

El valor de ξ^+ y ξ^- que se obtuvo en la ecuación 2.2 y 2.3, respectivamente, son número desconocidos que se encuentran en el intervalo $x < \xi^+ < x+h$ y en el intervalo $x-h < \xi^- < x$. Luego, sumando la ecuación 2.2 con la ecuación 2.3 queda:

$$f(x+h) + f(x-h) = 2f(x) + \frac{2f''(x)h^2}{2} + \frac{2f^4(\xi)h^4}{4 \cdot 3 \cdot 2}$$

Donde $x - h < \xi < x + h$. Ahora despejando $f''(x)$

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{f^4(\xi)h^2}{12} \tag{2.4}$$

$$= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2). \tag{2.5}$$

De la ecuación 2.4 a la 2.5 se reemplaza $-\frac{f^4(\xi)h^2}{12}$ por $O(h^2)$ el cual representa el error de grado 2. Quedando así demostrada la formula 2.1 para la segunda derivada centrada.

Luego, en la línea 4 del código 1 se escribió la función `deriv2(f,x,h)` que calcula la segunda derivada centrada de la función $f(x) = \sqrt{x}$, definida en la línea 1, según la formula 2.1 anteriormente demostrada.

```
1 def ddf(x):
2     return -0.25*x**(-1.5)
3 def f(x):
4     return np.sqrt(x)
5
6 def deriv2(f,x,h):
7     return (f(x+h) - 2*f(x) + f(x-h))/(h**2)
```

Código 1: Se definió la función f y su segunda derivada.

Después, se graficó el error relativo entre la segunda derivada analítica y numérica de f en función de h para los valores $x = \{0{,}1, 1, \pi/3\}$ quedando los gráficos 2.1, donde se puede apreciar que en los 3 gráficos hay una punta hacia abajo la cual representa el valor mínimo del error relativo. Para ello se definió la derivada analítica en la línea 1 del código 2, el error relativo como el valor absoluto de la diferencia entre la segunda derivada analítica y segunda derivada numérica en la linea 8 y los valores de h en la linea 5. Finalmente la línea 9 se usó para encontrar el valor de h para que el error sea el menor con ayuda de la función `argmin()` de *numpy*, obteniéndose los siguiente resultados expuestos en la tabla 2.1, donde h presenta los valores necesarios para que el error sea el mínimo para los valores de x.

Por otra parte, para calcular el valor de h cuando el error analítico es mínimo se usó la ecuación 2.6, donde ε es el error de truncamiento. De este modo se obtuvieron los valores de la tercera columna de la tabla 2.1.

$$\varepsilon_r = \frac{h^2}{12}|f^4(x)| + \frac{4\varepsilon}{h^2}|f(x)| \tag{2.6}$$

x	Mínimo numérico h	Mínimo analítico h
0,1	$1{,}781 \times 10^{-5}$	$3{,}199 \times 10^{-5}$
1	0,000159	0,00033
$\pi/3$	0,000214	0,00033

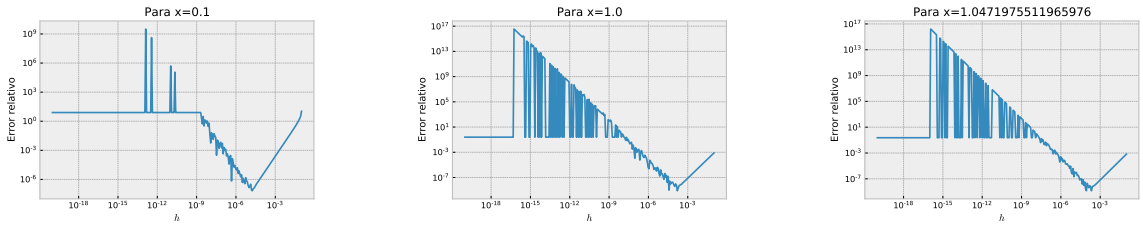
Cuadro 2.1: Valores de h donde el error relativo es menor.

```
def ddf(x):
    return -0.25*x**(-1.5)

y= [0.1,1,np.pi/3]
h = np.geomspace(0.1,1e-20,300)

for i in np.array(y):
    err = np.abs( ddf(i) - deriv2(f,i,h) )
```

Código 2: Fragmento de código creado para crear gráfico del error relativo en función de h



(a) (b) (c)

Figura 2.1: Gráficos de error relativo v/s h para los distintos valores de $x = \{0.1, 1, \pi/3\}$.

2.1.3. Conclusión

En este laboratorio se demostró que la segunda derivada centrada de una función puede ser escrita como 2.1, luego se estudio el error en función de h de la segunda derivada numérica y analítica de la función $f(x) = \sqrt{x}$, obteniéndose el valor más óptimo para h para distintos valores de x.

Antes de realizar este laboratorio pensaba que mientras más pequeño era el valor de h que se le entregaba al computador más exacto iba a ser el resultado que se iba a obtener, pero esto no es así, pues el computador no trabaja bien con números extremadamente pequeños. Por otra parte, aprendí sobre las derivadas adelantadas y retrasadas y que se pueden aplicar en métodos numéricos para programar.

Creo que faltó trabajar un poco en los errores y cómo estos se calculan, ya que no me quedaron muy claros. También creo que entender este laboratorio en clases se me hizo más complicado de lo que debería ya que recordaba muy poco de python y específicamente de cómo utilizar `matplotlib`, pues si mal no recuerdo en el ramo de computación científica no se alcanzó a ver bien este paquete.

2.2. Integración numérica básica

Fecha de la actividad: 02 de septiembre de 2022

2.2.1. Objetivo de la clase

La finalidad de este laboratorio fue aprender a resolver integrales numéricamente con la regla del punto medio y para ello se tuvo que recurrir a las series de Taylor.

En este laboratorio trabajé junto a Jeremías Martínez, Mauricio Bastías y Eduardo Escudero.

2.2.2. Desarrollo del Laboratorio

Se partió demostrando la regla del punto medio 2.7.

$$\int_a^b f(x)dx = hf\left(\frac{a+b}{2}\right) + O(h^3). \tag{2.7}$$

Para ello se consideró una expansión en series de Taylor de $f(x)$, donde $c = \frac{(a+b)}{2}$:

$$\begin{aligned} f(x) &= f(c) + f'(c)(x-c) + f''(x)\frac{(x-c)^2}{2} + \dots \\ &= f(c) + f'(c)(x-c) + f''(\xi)\frac{(x-c)^2}{2} \end{aligned} \tag{2.8}$$

En la ecuación 2.8 se define ξ, este es un valor que por el teorema del valor medio hace que la expansión en series sea exacta.

Luego, integrando en ambos lados:

$$\int_a^b f(x)dx = \int_a^b f(c)dx + \int_a^b f'(c)(x-c)dx + \int_a^b f''(\xi)\frac{(x-c)^2}{2}dx \tag{2.9}$$

$$= f(c)(b-a) + f'(c)\left[\frac{x^2}{2} - cx\right]_a^b + O(h^3) \tag{2.10}$$

De la ecuación 2.9 a la ecuación 2.10 la integral $\int_a^b f''(\xi)\frac{(x-c)^2}{2}dx$ se iguala a $O(h^3)$ la cual indica el grado del error, ya que implícitamente en la integral se encuentra un h^2 y al momento de integrar esta quedará elevado al cubo. Siguiendo con el cálculo:

$$\int_a^b f(x)dx = f(c)(b-a) + f'(c)\left(\frac{b^2}{2} - cb - \frac{a^2}{2} + ca\right) + O(h^3),$$

Finalmente, reemplazando $h = (b-a)$ y c:

$$\begin{aligned} \int_a^b f(x)dx &= f(c)(b-a) + f'(c)(0) + O(h^3) \\ &= hf\left(\frac{a+b}{2}\right) + O(h^3). \end{aligned} \tag{2.11}$$

Así, la regla del punto medio queda demostrada, donde $O(h^3)$ es el error cometido al calcular la derivada y el h^3 indica que el error es de tercer orden, este se obtiene al integrar $\int_a^b f''(\xi)\frac{(x-c)^2}{2}dx$.

Luego, usando la ecuación 2.11 anteriormente demostrada se calculó I:

```
def f(x):
    return (x-1)*np.exp(x)

def midpoint_simple(f,a,b):
    return (b-a) * f((a+b)*0.5)

print(midpoint_simple(f, 0, 3))
```

Código 3: Fragmento de código donde se define una función y su integral a través de la regla del punto medio

$$I = \int_0^3 (x-1)e^x dx$$

Con $f(x) = (x-1)e^x$ se tiene:

$$\begin{aligned} I &= \frac{3}{2} e^{\frac{3}{2}} \\ &\approx 6,722 \end{aligned} \tag{2.12}$$

Por otra parte calculando la integral I de forma analítica, se tiene:

$$\begin{aligned} I &= \int_0^3 (x-1)e^x dx \\ &= [(x-1)e^x]_0^3 - \int_0^3 e^x dx \\ &= e^3 + 2 \\ &\approx 22,085 \end{aligned} \tag{2.13}$$

En resumen, vemos que el resultado de la ecuación 2.12 obtenida usando la regla del punto medio es $I \approx 6,722$, mientras que resolviendo la integral analíticamente en la ecuación 2.13 se obtuvo $I \approx 22,085$, por lo que existe un error claro al usar el método del punto medio. Error que ocurre debido a que el intervalo que ocupan los límites de integración es muy grande.

En la primera linea del código 3 se definió una función f, en la cuarta linea se definió `midpoint_simple(f,a,b)` que calculaba la integral de f con límites de integración a y b con la regla del punto medio. Luego se imprimió `midpoint_simple(f,0,3)` dando como resultado aproximadamente 6,77.

Seguidamente se evaluó la integral de la siguiente forma:

$$\int_0^3 f(x)dx = \int_0^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_N}^3 f(x)dx,$$

donde $0 < x_1 < x_2 < ... < x_N < 3$, con 100 intervalos. Para ello se desarrolló el código 4.

```
limites = np.linspace(0,3,100)

integ = 0

for i in range( len(limites)-1) :
    integ = integ + midpoint_simple(f, limites[i], limites[i+1])
```

Código 4: Mejora del código 3 al implementar el ciclo `for`.

En la primera línea de 4 se creó un array con 100 elementos con el fin de evaluar la integral como la suma de 100 integrales, luego se definió `integ` para que cuando se ejecutara el ciclo `for` se sumaran todas las integrales de limites de integración `limites[i]` y `limites[i+1]`. Finalmente se ejecutó el programa, notando que el valor de `integ` fue aproximadamente 22,083 que al comparar con la ecuación 2.13 se observa el mismo resultado.

2.2.3. Conclusiones

Este laboratorio se comenzó haciendo la demostración de la regla del punto medio, para luego ocuparla calculando la integral de $f(x) = (x-1)e^x$ y comparar su resultado con la misma integral resuelta analíticamente. Al comparar ambos resultados se observó que la regla del punto medio fallaba. Luego de separar los límites de integración en varios intervalos muy pequeños se calculó la integral, pero ahora como la suma de varias integrales con intervalos de límites de integración más pequeños, de este modo se obtuvo un resultado mucho más exacto. Lo anterior ocurría debido a que este método falla cuando el intervalo que encierran los límites de integración es muy grande.

En esta actividad aprendí a utilizar y a programar la regla del punto medio para calcular integrales. También aprendí que al momento hacer la expansión en series de Taylor e integrar, el exponente de h obtenido indica el orden del error y que mientras mayor sea el orden menor sera el error cometido al calcular la integral siempre y cuando h sea menor a 1.

Para esta actividad no tengo preguntas y tampoco consejos para mejorarlo, pues pienso que este laboratorio fue enseñado y desarrollado correctamente.

2.3. Ceros de Funciones

Fecha de la actividad: 09 de septiembre de 2022

2.3.1. Objetivo de la clase

El objetivo de esta clase fue aprender a encontrar ceros de funciones a través del *Método de bisección*. Los ceros de una función son los valores de x cuando una función f se iguala a 0, es decir, $f(x) = 0$.

En este laboratorio trabajé junto a Mauricio Bastías y Jeremías Martínez.

2.3.2. Desarrollo del Laboratorio

En este laboratorio se trabajó estudiando la energía potencial de un péndulo invertido de masa m, largo l y que está conectado a un resorte de constante k a una altura d. La función que define su energía potencial es la ecuación 2.14, donde $\beta = kd^2/2mgl$ y θ representa el ángulo con respecto al eje y.

$$u(\theta) = cos(\theta) + \beta sin^2(\theta) \tag{2.14}$$

En la primera línea del código 5 se definió la función $u(\theta)$, en la línea 4 se definió su derivada adelantada, pues más adelante se ocupó el método de bisección para encontrar sus raíces y en la línea 7 se creó el gráfico 2.2a. Este gráfico se creó ejecutando un ciclo `for` para crear 10 curvas en un gráfico, donde *be* es β, t es θ y du es la derivada de la función $u(\theta)$.

La gráfica de la derivada numérica de la función 2.14 está representado en la imagen 2.2a, donde $\theta \in [0, \frac{\pi}{2}]$. En la gráfica se logran observar 10 curvas, en la curva del extremo inferior β toma el valor de 0,1, luego en la curva que está sobre esta $\beta = 0{,}2$, la siguiente es cuando $\beta = 0{,}3$ y así sucesivamente hasta la curva azul del extremo superior donde $\beta = 1$.

```
1 def u(t):
2     return np.cos(t) + beta* (np.sin(t))**2
3
4 def du(u,t,h):
5     return (u(t+h) - u(t)) / h
6
7 for beta in  np.linspace(0.1,1,10):
8     plt.plot(t,du(u,t,h))
```

Código 5: Se definió la función $u(t)$, su derivada y se creó un gráfico en el ciclo for

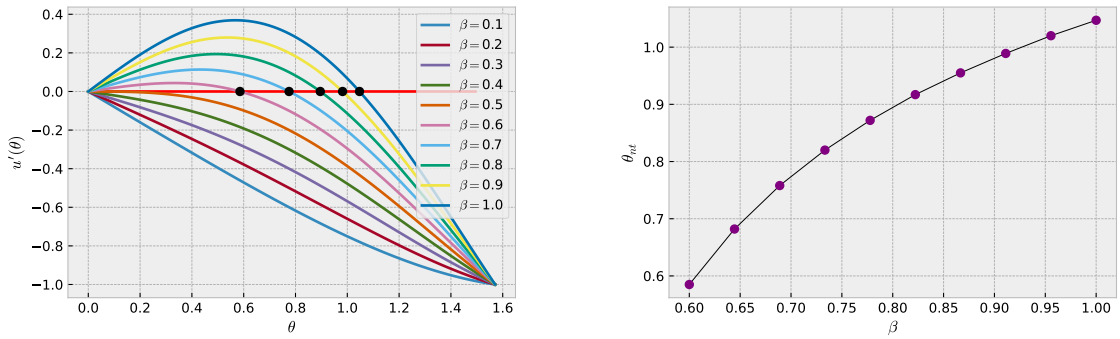
Luego para encontrar los ceros no triviales se utilizo el fragmento del código 6. Lo que se hizo ahí fue utilizar el método de la bisección. Después de observar que dentro del intervalo $[0.5, 1.2]$ se encontraban todos los ceros, en la línea 2 y 3 del código 6 se eligió a $a = 0.5$, pues para todo $u'(a)$ con $\beta > \frac{1}{2}$ resultaba que $u'(a) > 0$. Similar con $b = 1.2$, pero resultando $u'(b) < 0$. Seguidamente se crea un ciclo `for` para los valores de beta mayor a 0,5, dentro de este ciclo `for` se creó otro ciclo `for` para encontrar los ceros no triviales a través del método de la bisección. Obteniéndose así los resultados expuestos en la tabla 2.2.

β	θ_{nt}
0.6	0.585
0.7	0.775
0.8	0.895
0.9	0.981
1	1.047

Cuadro 2.2: Tabla de los valores de los ceros no triviales para cada valor de beta usados en el gráfico 2.2a.

```
1  for beta in np.linspace(0.6,1,5):
2      a=0.5
3      b=1.2
4      for i in range(20):
5          c= 0.5*(a+b)
6          condicion = du(u,a,h)*du(u,c,h)
7
8          if condicion<0:
9              b=c
10         else:
11             a=c
12     plt.plot(c,du(u,c,h),"o", color="black")
```

Código 6: Fragmento de código creado para encontrar los ceros no triviales y graficarlos



(a) Gráfico $u'(\theta)$ v/s θ.

(b) θ_{nt} v/s β

Figura 2.2

Como se logra ver en 2.2a, las curvas cortan el eje x (sin contar el origen) solo cuando $\beta > \frac{1}{2}$. Luego, lo que se hizo en la línea 12 fue graficar los ceros no triviales. Estos se puede observar como puntos negros en el gráfico 2.2a.

En el gráfico 2.2b se pueden observar los ceros no triviales θ en función de β.

Finalmente, se definió la segunda derivada de $u''(\theta)$ y se graficaron los puntos $u''(0)$ en función de

β para sus 10 valores quedando así el gráfico de la izquierda en la figura 2.3. Para ello se escribió el código 7.

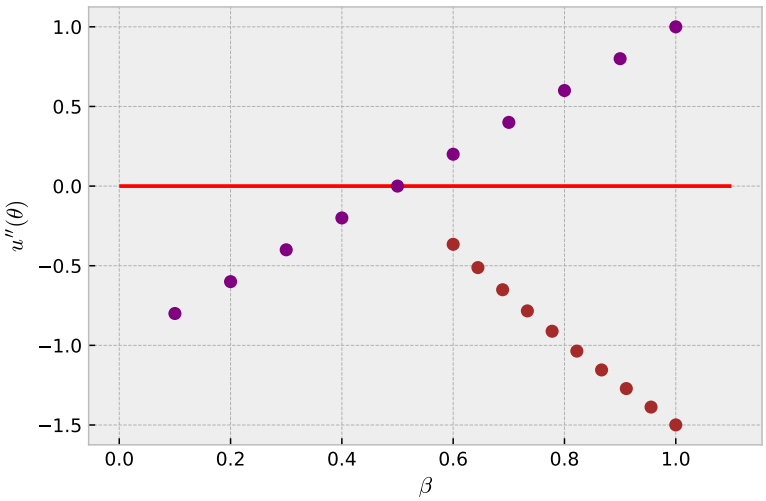


Figura 2.3: Los puntos morados representan $u''(0)$ con respecto a los valores de β. Por otra parte, los puntos rojizos representan $u(\theta_{nt})$ con respecto a los valores de $\beta > 0{,}5$

```
1  def ddu(u,t,h):
2      return ( u(t+h) - 2*u(t) + u(t-h)) / (h**2)
3
4  for beta in np.linspace(0.1,1,10): #con theta igual a 0
5      plt.plot(beta,ddu(u,0,h),"o")
```

Código 7: Se definió la segunda derivada de u y se creó el gráfico izquierdo de la figura 2.3

En el fragmento del código 7 se definió la segunda derivada centrada de u en la linea 1 y luego a través de un ciclo `for` en la linea 4 se creó un gráfico con los valores de beta en el eje x y los valores de $u''(0)$ en el eje y.

```
1  c_nt=[0.585, 0.682, 0.758, 0.820, 0.872, 0.917, 0.955, 0.989, 1.020, 1.047]
2
3  a= np.linspace(0.6,1,10)
4
5  for i in range(0,5):            #con los ceros no triviales
6      beta= a[i]
7      plt.plot(beta,ddu(u, c_nt[i], h), "o" )
```

Código 8: Código para crear el gráfico derecho del la figura 2.3

También se graficaron 10 puntos de $u''(\theta_{nt})$ para los valores de $\beta > \frac{1}{2}$ quedando el gráfica de la

derecha en la figura 2.3. Para esto se redactó el fragmento de código 8. En la primera linea del código 8 se define una lista con los ceros no triviales relacionados a $\beta = 0{,}6$ hasta $\beta = 1$, respectivamente. Luego se creó un ciclo `for` para para graficar $u''(\theta_{nt})$ v/s β.

Al analizar los gráficos presentes en la figura 2.3 se puede establecer que $\theta = \theta_{nt}$ corresponde a un equilibrio inestable del sistema para todo valor de $\beta > \frac{1}{2}$, pues la segunda derivada de u es negativa. Por otro lado, para $\theta = 0$ el sistema está en un equilibrio inestable cuando $\beta < \frac{1}{2}$, para $\beta > \frac{1}{2}$ está en equilibrio estable, porque u'' es positivo, y para $\beta = \frac{1}{2}$ se encuentra en equilibrio indiferente o neutral, pues la segunda derivada es igual a cero [4].

2.3.3. Conclusiones

En este laboratorio se trabajó encontrando los puntos de equilibrio de un péndulo invertido, para ello se debió encontrar las raíces de la derivada de la función que definía su energía potencial. La función de la energía potencial del péndulo es 2.14. Para encontrar los ceros no triviales de la derivada de 2.14 se utilizó el método de la bisección y finalmente se determinaron los valores de θ para cuando correspondía a un equilibrio estable o inestable.

En resumen puedo decir que en el desarrollo de esta actividad aprendí a encontrar ceros de funciones usando el método de la bisección y para ello es recomendable graficar la función con anticipación para definir un intervalo $[a, b]$, tal que $f(a) \cdot f(b) < 0$.

Al finalizar la actividad no me quedaron dudas de cómo utilizar y ni de cómo programar este método, ya que creo haberlo entendido bien. Por otra parte, me hubiera gustado haber trabajado con algún método más, pues creo que el método de la bisección no presenta una mayor dificultad aprender a usarlo y quizás nos hubiera dado tiempo para trabajar con otro método más.

2.4. Ecuaciones diferenciales ordinarias

Fecha de la actividad: 23 de septiembre de 2022

2.4.1. Objetivo de la clase

En este laboratorio se aprendió a usar el método de Euler-Cromer para resolver ecuaciones diferenciales ordinarias, específicamente se trabajó con las ecuaciones 2.15 y 2.16 de Lotka-Volterra. Estas son un par de ecuaciones diferenciales de primer orden no lineales que se usan para describir dinámicas de sistemas biológicos en el que dos especies interactúan, una como presa y otra como depredador[3].

$$\frac{dP}{dt} = \alpha P - \beta PD \tag{2.15}$$

$$\frac{dD}{dt} = -\gamma D + \delta PD. \tag{2.16}$$

En este laboratorio trabajé junto a Mauricio Bastías.

2.4.2. Desarrollo del laboratorio

Las ecuaciones 2.15 y 2.16 de Lotka-Volterra se pudieron reducir a las ecuaciones 2.17 y 2.18 que dependen de una solo constante μ.

$$\frac{d\Pi}{d\tau} = \Pi(1 - \Delta) \tag{2.17}$$

$$\frac{d\Delta}{d\tau} = \mu\Delta(\Pi - 1). \tag{2.18}$$

Para ello se definió $t = \tau/\alpha$, $P = \Pi\gamma/\delta$ y $D = \alpha\Delta/\beta$. Luego, reemplazando estos valores en 2.15:

$$\begin{aligned}
\frac{d\frac{\Pi\gamma}{\delta}}{d\frac{\tau}{\alpha}} &= \alpha\left(\frac{\Pi\gamma}{\delta}\right) - \beta\left(\frac{\Pi\gamma}{\delta}\right)\left(\frac{\alpha\Delta}{\beta}\right) \\
\frac{d\Pi}{d\tau}\left(\frac{\alpha\gamma}{\delta}\right) &= \alpha\left(\frac{\Pi\gamma}{\delta}\right) - \left(\frac{\Pi\gamma}{\delta}\right)\alpha\Delta \\
\frac{d\Pi}{d\tau} &= \Pi - \Pi\Delta \\
\frac{d\Pi}{d\tau} &= \Pi(1 - \Delta)
\end{aligned}$$

Ahora reemplazando en la ecuación 2.16 se tiene:

$$\begin{aligned}
\frac{d\frac{\alpha\Delta}{\beta}}{d\frac{\tau}{\alpha}} &= -\gamma\left(\frac{\alpha\Delta}{\beta}\right) + \delta\left(\frac{\Pi\gamma}{\delta}\right)\left(\frac{\alpha\Delta}{\beta}\right) \\
\frac{d\Delta}{d\tau}\left(\frac{\alpha^2}{\beta}\right) &= -\gamma\left(\frac{\alpha\Delta}{\beta}\right) + \Pi\gamma\left(\frac{\alpha\Delta}{\beta}\right) \\
\frac{d\Delta}{d\tau} &= -\frac{\gamma}{\alpha}\Delta + \frac{\gamma}{\alpha}\Pi\Delta
\end{aligned}$$

Finalmente, definiendo $\mu = \frac{\gamma}{\alpha}$ queda:

$$\frac{d\Delta}{d\tau} = \mu\Delta\left(\Pi - 1\right) \tag{2.19}$$

Para resolver estas ecuaciones diferenciales ordinarias se creó el código 9 con una función `LV` con variables `P0, V0, mu, tmax` y `h` donde las primeras dos variables correspondían a $\Pi(0)$ y a $\Delta(0)$, respectivamente, `tmax` corresponde al tiempo máximo, `mu` al valor de la constante μ y `h` el paso del tiempo.

```
1  def Lotka_Volterra(P0,D0,mu,tmax,h):
2      t= np.arange(0,tmax,h)
3      P= np.zeros(len(t))
4      D= np.zeros(len(t))
5      P[0]=P0
6      D[0]=D0
7      for n in range(len(t) - 1):
8          P[n+1]= P[n] + h * P[n] * (1- D[n])
9          D[n+1]= D[n] + h * mu * D[n] * (P[n+1] - 1)
10     return t, P, D
```

Código 9: Función que resuelva le ecuación diferencial ordinaria de Lotka-Volterra a través del método de Euler-Cromer

En la tercera y cuarta linea del código 9 se definió `P` y `D` como un `arange` de ceros de un número de dimensiones igual al número de valores que tiene `t`, luego en las siguientes dos lineas se define el primer valor de los `arange` de `P` y `D`. Finalmente de la línea 7 se ejecuta un ciclo `for` que resuelve la ecuación diferencial ordinaria de Lotka-Volterra retornando los valores de `t, P` y `D`. Además, en la línea 9 se hace el cálculo con `P[n+1]`, debido a que este ya fue calculado en la línea anterior.

Luego, se evaluó la función `LV` para $tmax = 40$, $P0 = 1$, $V0 = 5$, $\mu = 1$ y $h = 0{,}001$ obteniendo así los valores de D y P. Luego estos se graficaron en función del tiempo resultando el gráfico 2.4

Del gráfico 2.4 se pudo interpretar que cuando el número de presas es mínimo el número de depredadores poco tiempo después comienza a decaer, pues estos se encuentran sin alimento, luego con la disminución de depredadores más adelante el número de presas vuelve a aumentar hasta llegar a su máximo y el proceso se vuelve a repetir cíclicamente.

Se resolvió la ecuación diferencial de Lotka-Volterra pero ahora con $tmax = 0$ y con $\Delta(0)$ variando entre $1,1$ y 10. Para ello se creó un ciclo `for` donde `V0` tomaba 20 valores distintos y para cada valor que tomaba `V0` se graficaba una curva, obteniéndose así el gráfico 2.5. De este gráfico se puede deducir que mientras mayor sea la cantidad inicial de depredadores la caída del número de presas va a ser más brusco al igual que el aumento de depredadores.

Las ecuaciones de Lotka-Volterra admiten una forma constante que se puede apreciar en la ecuación 2.20.

$$C = \ln\Delta - \Delta + \mu(\ln\Pi - \Pi). \tag{2.20}$$

El gráfico 2.6 muestra a C en función del tiempo para todos los valores que tomó $\Delta(0)$ en el gráfico 2.5. Además, La curvas son aproximadamente constante por lo tanto se puede decir que el método

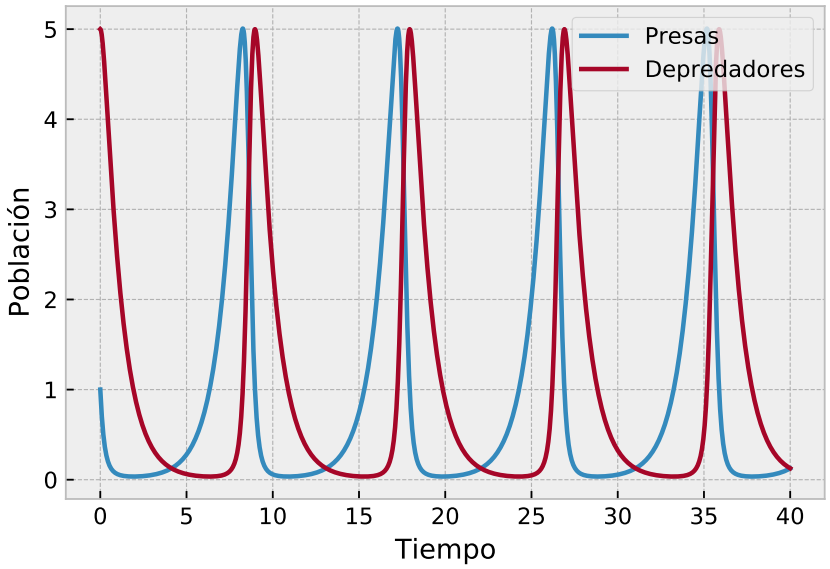


Figura 2.4: Gráfico de la población de presas y depredadores en el modelo de Lotka-Volterra con la constante $\mu = 1$.

numérico desarrollado entrega soluciones aproximadamente tolerables a las ecuaciones diferenciales.

2.4.3. Conclusiones

En este laboratorio se partió reduciendo las ecuaciones de Lotka-Volterra para que dependieran de una sola constante μ, luego se resolvieron estas ecuaciones diferenciales mediante con el método de Euler-Cromer. Seguidamente se estudió el número de la población de los depredadores y de las presas y como variaban en función del tiempo. Luego se graficó $\Delta(t)$ versus $\Pi(t)$ observando su relacionaban y como variaba la curva del gráfico según el valor inicial de los depredadores. Finalmente se graficó C en función del tiempo concluyendo que el método numérico entregaba soluciones aproximadas tolerables.

Al desarrollar esta actividad aprendí que para resolver ecuaciones diferenciales ordinarias a través del método de Euler-Cromer son necesarios los valores iniciales y que si se tienen ecuaciones de orden mayor, entonces se pueden simplificar en un sistema de ecuaciones diferenciales ordinarias de primer orden con el fin de poder utilizar este método. Lo que no entendí muy bien fue la demostración del método y en cómo se programó el método.

Me hubiera gustado que se hubiera dispuesto un poco más de tiempo en cómo se programó el método, quizás desarrollando algún ejemplo simple utilizándolo antes de comenzar a desarrollar el laboratorio.

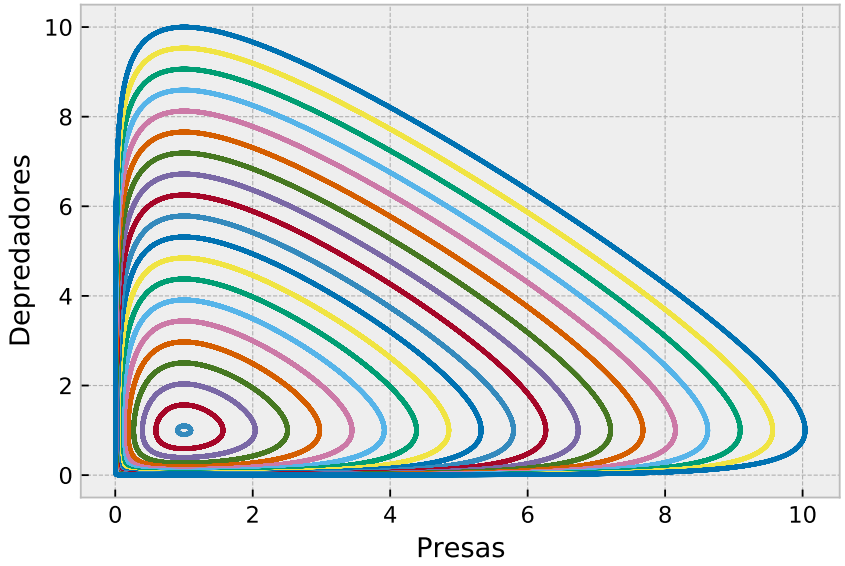


Figura 2.5: Espacio de fases del modelo de Lotka-Volterra

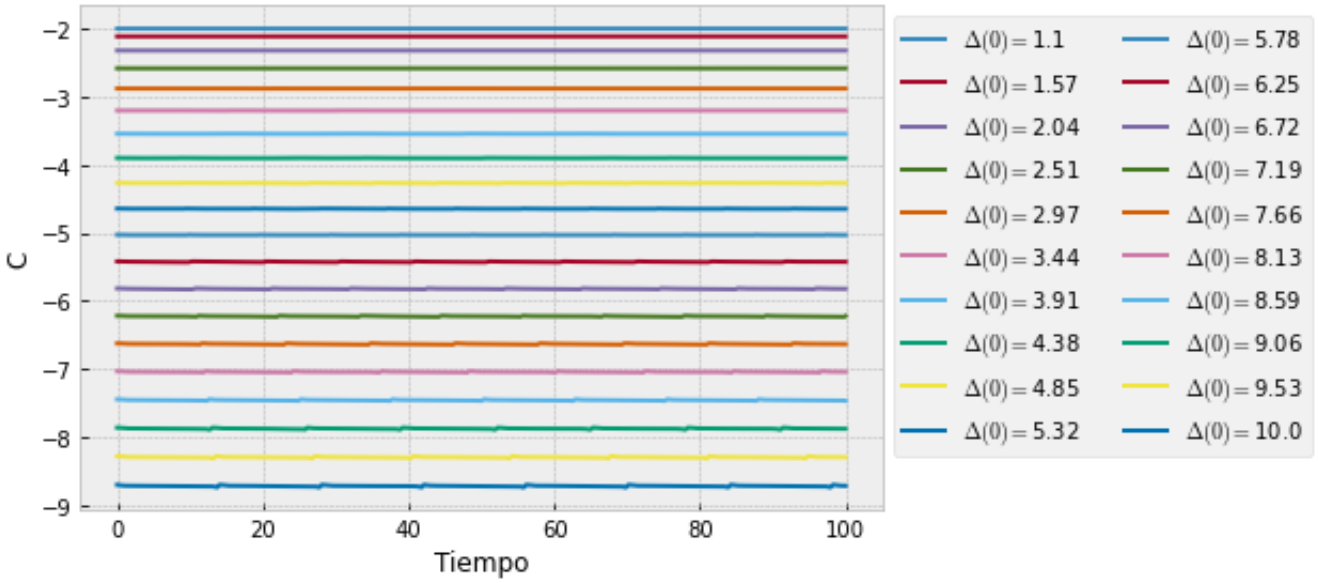


Figura 2.6: C en función del tiempo para valores de $1{,}1 < \Delta(0) < 10$ usando $h = 0{,}001$, donde mientras mayor es $\Delta(0)$ la curva se hace mas negativa.

2.5. Método de Runge-Kutta

Fecha de la actividad: 30 de septiembre de 2022

2.5.1. Objetivo de la clase

La finalidad de este laboratorio fue aprender a usar el método de Runge-Kutta para resolver sistemas de ecuaciones diferenciales ordinarias, específicamente las ecuaciones de Lorenz 2.21-2.23. Estas ecuaciones modelan un sistema dinámico determinista tridimensional no lineal derivado de las ecuaciones simplificadas de rollos de convección que se producen en las ecuaciones dinámicas de la atmósfera terrestre [2].

$$\dot{x} = \sigma(y - x), \tag{2.21}$$
$$\dot{y} = \rho x - y - xz, \tag{2.22}$$
$$\dot{z} = xy - \beta z. \tag{2.23}$$

En las ecuaciones de Lorenz 2.21-2.23 σ, ρ y β son parámetros asociados al número de Prandtl y al número de Rayleigh, a estos parámetros se les asignaron los siguientes valores: $\sigma = 10$, $\rho = 28$ y $\beta = 8/3$.

2.5.2. Desarrollo del laboratorio

Antes de comenzar a resolver las ecuaciones diferenciales ordinarias se asignaron los valores iniciales de x, y y z, estos fueron $x = 0$, $y = 1$ y $z = 1{,}05$. Con estos valores ya definidos se resolvió la ecuación diferencial ordinaria con ayuda del método de Runge-Kutta, para ello se creó el código 10.

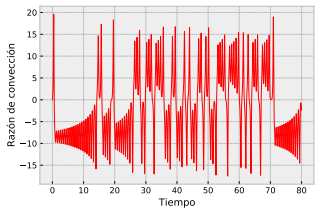
```
1  def f(u,t, rho=28, beta= 8/3 , sig=10):
2      x,y,z = u
3      return np.array([sig*(y-x), rho*x-y-x*z, x*y - beta*z ])
4
5  tmax=80
6  h=0.01
7  t= np.arange(0,tmax,h)
8
9  u=np.empty([t.size,3])
10 u[0]=[0,1,1.05]
11
12 for n in range(t.size-1):
13     K1= f(u[n], t[n])
14     K2= f(u[n] + h*0.5*K1, t[n]+0.5*h )
15     K3= f(u[n] + h*0.5*K2, t[n]+0.5*h )
16     K4= f(u[n] + h*K3, t[n]+h )
17     u[n+1]= u[n] + (h/6)*(K1 + 2*K2 + 2*K3 + K4)
```

Código 10: Fragmento de código que resuelve las ecuaciones de Lorenz a través del método de Runge-Kutta.

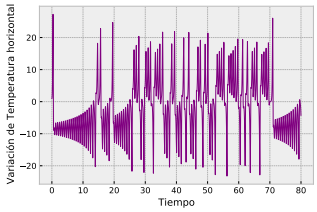
En la primera línea del código 10 se creó la función f que retornó un `array` de `numpy` del sistema de las ecuaciones de Lorenz, luego en la línea 5, 6 y 7 se creó un `arange` de 30000 números que iban del 0 al 300. En la línea 9 se definió a u como una matriz vacía de 30000x3 y en la línea 10 se definieron

los valores de la primera fila de u. Finalmente, en la línea 12 se creó un ciclo `for` que dio los valores restantes a u, es decir, resolvió el sistema de ecuaciones diferenciales ordinarias a través del método de Runge-Kutta.

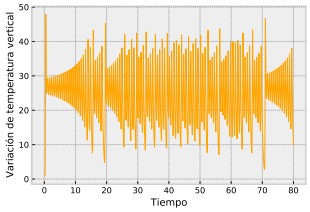
Se graficó a x, y y z con los mismo valores iniciales anteriores con respecto al tiempo dando como resultados los gráficos de la figura 2.8. Luego, se cambiaron solo los valores iniciales quedando $x = 0{,}4$, $y = 1{,}2$ y $z = 1$. Ahora con estos valores iniciales se volvió a graficar con respecto al tiempo quedando así los gráficos de la figura 2.7



(a) Gráfica de la Razón de convección con respecto al tiempo para los valores iniciales y los valores de las constantes utilizados en el código 10.



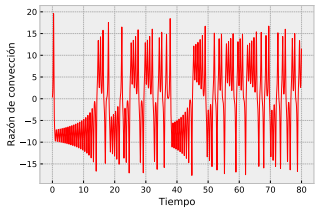
(b) Gráfico de la variación de la temperatura horizontal con respecto al tiempo para los valores iniciales y los valores de las constantes utilizados en el código 10.



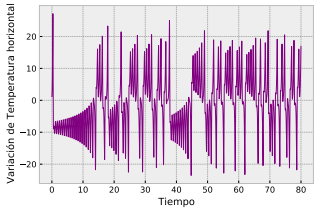
(c) Gráfico de la variación de la temperatura vertical con respecto al tiempo para los valores iniciales y los valores de las constantes utilizados en el código 10.

Figura 2.7: Gráficos de x, y y z con respecto al tiempo para los valores inciales $x = 0$, $y = 1$ y $z = 1{,}05$.

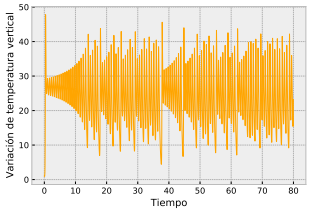
Al comparar los gráficos 2.7a, 2.7b, 2.7c con los gráficos 2.8a, 2.8b y 2.8c, respectivamente, se pudo observar claras diferencias entre ellos, de esto se pudo deducir que si las condiciones iniciales cambian, aunque sea un cambio muy pequeño, entonces se producirán cambios importantes, es decir, se está en presencia de un sistema caótico.



(a) Gráfica de la Razón de convección con respecto al tiempo para los valores iniciales $x = 1$, $y = 2$, $z = 1$ y los valores de las constantes utilizados en el código 10.



(b) Gráfico de la variación de la temperatura horizontal con respecto al tiempo para los valores iniciales $x = 1$, $y = 2$, $z = 1$ y los valores de las constantes utilizados en el código 10.



(c) Gráfico de la variación de la temperatura vertical con respecto al tiempo para los valores iniciales $x = 1$, $y = 2$, $z = 1$ y los valores de las constantes utilizados en el código 10.

Figura 2.8: Gráficos de x, y y z con respecto al tiempo para los valores iniciales $x = 0{,}4$, $y = 1{,}2$ y $z = 1$.

El código 11 se utilizó para crear el gráfico 2.9. En este fragmento de código se tuvieron que utilizar el paquete `matplotlib.pyplot` y la función `axes3d` del paquete `mpl_toolkits.mplot3d`, para ello se importaron en la línea 1 y 2 del código 11. En la línea 8 se definió un `array` de numpy para los valores que se graficarían en el eje x, lo mismos para el eje y y z en las líneas 9 y 10, respectivamente. Finalmente, en la línea 12 se creó el gráfico 2.9.

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d

fig=plt.figure()

ax1 = fig.add_subplot(111, projection="3d")

xx= np.array([u[:,0]])
yy= np.array([u[:,1]])
zz= np.array([u[:,2]])

ax1.plot_wireframe(xx, yy, zz, color="green", linewidth=0.2)
```

Código 11: –

La figura 2.9 es un gráfico tridimensional donde $x(t)$ está representado en el eje x, $y(t)$ en el eje y y $z(t)$ en el eje z, además sus valores iniciales son los mismo que se utilizaron en la figura 2.8

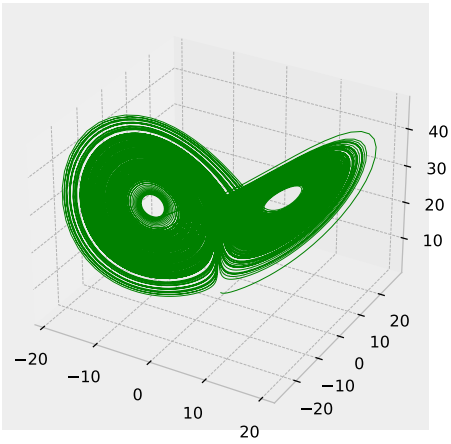


Figura 2.9: Gráfico tridimensional paramétrico de $x(t)$, $y(t)$ y $z(t)$

2.5.3. Conclusiones

En este laboratorio se inició resolviendo las ecuaciones de Lorenz con el método de Runge-Kutta, para ello se le asignaron ciertos valores a las constantes y a los valores iniciales, luego se graficaron las 3 ecuaciones diferenciales ordinaraias ya resueltas con respecto al tiempo. Después, se cambiaron levemente las condiciones iniciales obteniendo gráficos que diferían de los anteriores. De esto se concluyó que se estaba en presencia de un sistema complejo. Finalmente, se creó un gráfico tridimensional paramétrico de $x(t)$, $y(t)$ y $z(t)$ obteniendo las típicas alas de mariposa que caracterizan a las ecuaciones de Lorenz.

En esta actividad aprendí a utilizar el método de Runge-Kutta para resolver ecuaciones diferenciales ordinarias, pero esto no implica que haya entendido bien la demostración de este método, pues realmente no entendí casi nada de la demostración. Recomendaría poner quizás un poco más de énfasis en la demostración de este método para futuras realizaciones de esta actividad.

2.6. Interpolación

Fecha de la actividad: 28 de octubre de 2022

2.6.1. Objetivo de la clase

La interpolación es la obtención de nuevos puntos partiendo del conocimiento de un conjunto de puntos [5]. El objetivo de este laboratorio fue aprender a hacer interpolaciones polinomiales con 3 métodos distintos: construyendo polinomios de Lagrange, usando la interpolación *spline* cúbica y usando la inversa de la matriz de Vandermonde. Para ello, se trabajó con datos de la cantidad de manchas solares por año.

En este laboratorio trabajé junto a Eduardo Escudero y Mauricio Bastías.

2.6.2. Desarrollo del laboratorio

Se comenzó trabajando usando la inversión de la matriz de Vandermonde, para ello en la línea 9 y 10 del código 12 se creó la matriz y en la línea 12 se resolvió el sistema $M \times a = B$, donde M es la matriz de Vandermonde y B es `Manchas2` obteniéndose de esta forma `a`. Luego, en la línea 14 se creó una función que representaba el polinomio que graficaba la curva de la interpolación. Finalmente, se introdujo todo esto en un ciclo `for` para interpolar todos los datos en intervalos de 5 puntos, pues este método daba error cuando se intentaba interpolar con más de cinco nodos. El número 51 que aparece en la línea 1 del código hacer referencia a que se hicieron 51 interpolaciones que estaban unidas entre sí por un nodo.De este modo se obtuvo el gráfico 2.10a.

```
1  for m in range(51):
2      año2 = año[m*4:5+m*4]
3      manchas2 = manchas[m*4:5+m*4]
4
5      dim = año2.size
6
7      vandermonde = np.ones( [dim,dim] )
8
9      for i in range(1,dim):
10         vandermonde[:,i]= año2**i
11
12     a = Resolver(vandermonde, manchas2)
13
14     def p(x,a):
15         pol=0.0
16         for n in range(len(a)):
17             pol= pol + a[n] * x**n
18         return pol
```

Código 12: Fragmento de código en el cual se interpolaron los datos usando la inversión de la matriz de Vandermonde.

El segundo método utilizado fue el de los polinomios de Lagrange. En este caso, al igual que en el anterior, se tuvo que interpolar por intervalos de 5 nodos, para ello se escribió el código 13 donde se creó una función dentro de una iteración en la línea 7 que construía los polinomios de Lagrange. De este modo, se creó el gráfico 2.10b.

```
for m in range(51):
    año2 = año[m*4:5+m*4]
    manchas2 = manchas[m*4:5+m*4]

    N= len(año2)

    def f(x):
        pol = 0.0
        for i in range(N):
            Li = manchas2[i]
            for j in range(N):
                if j != i:
                    Li = Li * (x-año2[j]) / (año2[i] - año2[j])
            pol= pol + Li
        return(pol)
```

Código 13: Fragmento de código utilizado para construir los polinomios de Lagrange y posteriormente crear el gráfico 2.10b con los polinomios obtenidos.

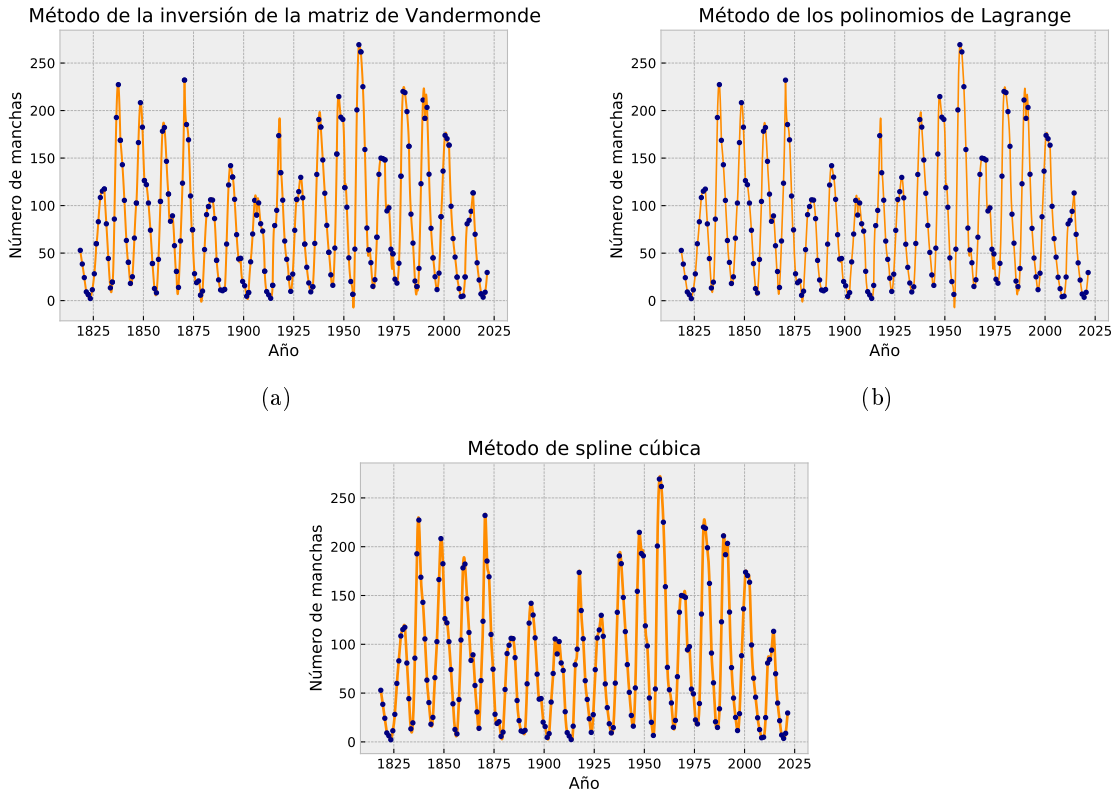


Figura 2.10: Gráficos del promedio de número de manchas solares con respecto al año, donde los punto azules representan los datos y la curva naranja se obtuvo al interpolar con los métodos que aparecen en sus respectivos títulos.

```
1 csp = sp.cspline(año, manchas)
2 x = np.linspace( np.min(año), np.max(año), 10000, endpoint=False)
3 plt.plot(x, csp(x))
```

Código 14: Código utilizado para interpolar usando spline cúbico.

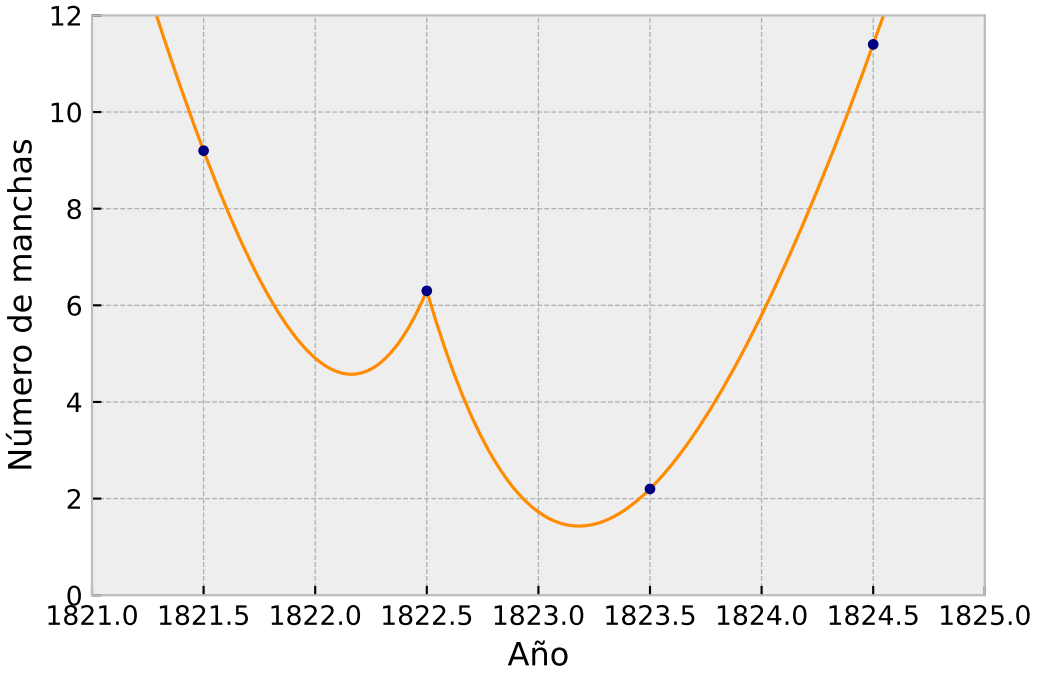


Figura 2.11: Acercamiento del gráfico 2.10a y 2.10b

Para el último caso, se interpoló utilizando splines cúbica. Este método es un conjunto de polinomios de tercer grado que se genera a partir de un conjunto de puntos [1]. Al utilizar este método, a diferencia de la matriz de Vandermonde y los polinomio de Lagrange, no se tuvieron que interpolar los datos utilizando intervalos, pues bastó con interpolar todos los datos en un solo intervalo.

Para interpolar usando spline cúbica se creó el código 14. Con ayuda de la función cspline del programa *spline* se creó en la primera línea la función `csp` que creó una función que interpolaba todos los puntos, luego para graficar la curva en la línea 2 y 3 se evaluó con la función `csp` 10000 valores ubicados en el intervalo $[1818, 2021[$. La función cspline interpola utilizando polinomio cúbicos y retorna una función que pasa por todos los nodos, de este modo se obtuvo el gráfico 2.10c.

Finalmente, al comparar los gráficos 2.10a y 2.10b se puede observar en ambos casos que al hacer un acercamiento se obtiene el gráfico de la figura 2.11 donde se aprecia que la curva no es suave, esto se debe a que al interpolar se tuvo que separar en 51 intervalos, quedando de este modo 51 interpolaciones donde en las conexiones de dichas interpolaciones quedaron puntas. Por otro lado, al usar el método spline cúbica la curva quedó suave en todas sus partes.

2.6.3. Conclusión

En este laboratorio se trabajó con datos que entregaban la información del promedio de las manchas solares desde el año 1818 hasta el 2021. Estos datos se interpolaron con 3 métodos distintos obteniéndose los gráficos 2.10a, 2.10b y 2.10c que corresponden a la curva obtenida al interpolar con los métodos de la matriz de Vandermonde, polinomio de Lagrange y spline cúbica, respectivamente. Analizando y comparando los tres gráficos se pudo concluir que al utilizar el método de spline cúbica se pueden obtener resultados más útiles, al menos en este caso, para predecir que sucede entre los nodos, pues al utilizar los polinomios de Lagrange y la inversa de la matriz de Vandermonde se obtienen curvas que no son suaves en todas sus partes y por ende no estimará tan bien el comportamiento de los datos como cuando se usa spline cúbica.

En esta actividad aprendí que cuando se tienen ciertos datos en los que se presentan variables independientes y variables dependientes, se puede intentar estimar el comportamiento entre dos nodos (datos). Siendo esto muy útil para predecir ciertos acontecimientos en diversas áreas, tales como la economía, la sociología, etc.

En el desarrollo de esta actividad me hubiera gustado que se explicara un poco más el método de interpolación *spline* cúbica, ya que al leer el código de esta función no entendí casi nada y solo aprendí a utilizarla copiando la función, por lo que no supe como explicar el desarrollo cuando utilicé este método.

2.7. Números aleatorios

Fecha de la actividad: 18 de noviembre de 2022

2.7.1. Objetivo de la clase

En este laboratorio se trabajó con números aleatorios para encontrar el número π, para ello se utilizó la función `numpy.random.rand(N)` que retornaba N números aleatorios ubicados en el intervalo $]0,1[$.

En este laboratorio trabajé junto a Eduardo Escudero y Jeremías Martínez.

2.7.2. Desarrollo del laboratorio

Se trabajó creando una circunferencia inscrita en un cuadrado de lado 1 con el fin de relacionar de forma analítica el área del cuadrado con el de la circunferencia y de forma numérica encontrar el valor de π.

El área de una circunferencia es $A_{circunf} = \pi r^2$ y el de un cuadrado de lado $2r$ es $A_{cuad} = 4r^2$, al relacionarlo se obtiene la ecuación 2.24.

$$4\frac{A_{circunf}}{A_{cuad}} = \pi \tag{2.24}$$

Ahora, si se toman en cuenta la cantidad de puntos que se encuentran dentro de un cuadrado de lado $2r$ y la cantidad de puntos que se encuentran dentro de una circunferencia de radio r inscrita en el cuadrado, entonces al relacionar la cantidad de puntos que se encuentran dentro de las figuras, como si fueran el área de ellas, se debería cumplir la relación expuesta en la ecuación 2.24.

Para encontrar el número π se comenzó creando el código 15. En las líneas 1 y 2 del código se generaron $N = 20000$ números aleatorios entre $]0,1[$ para los valores de x y para los valores de y, luego, en la línea 4 y 5 se crearon la misma cantidad de números aleatorios entre el intervalo $]-1,0[$ para mx y my. Seguidamente, en la línea 7 se concatenaron los `arrays` mx, x, mx y x y en la línea 8 se concatenaron los `arrays` my, y, y e my, y se dividieron por 2, de este modo, se obtuvo que xx e yy correspondían a 80000 números aleatorios entre $]-0{,}5, 0{,}5[$. Finalmente se graficó yy con respecto a xx generando un cuadrado de lado 1 con centro en el origen.

```
1 x=np.random.rand(int(N))
2 y=np.random.rand(int(N))
3
4 mx=-np.random.rand(int(N))
5 my=-np.random.rand(int(N))
6
7 xx=np.concatenate( [ mx,x,mx,x] ) /2
8 yy=np.concatenate( [ my,y,y,my] ) /2
```

Código 15: Fragmento de código que crea dos `arrays` de N números aleatorios en el intervalo $]-0{,}5, 0, 5[$

Luego, el código 16 se ocupó para encontrar que valores de xx y de yy se encontraban dentro de la circunferencia inscrita dentro del cuadrado de lado 1 centrado en el origen. Con la ayuda de un ciclo `for` en la línea 4 se encontró la distancia de los puntos con respecto al origen, si la distancia era menor que 0,5 entonces los valores se añadían a la lista xa y ya. En el gráfico 2.12 se puede apreciar una

circunferencia de radio 0,5 con centro en el origen, este se consiguió graficando ya con respecto a xa obtenidos en la línea 7 y 8 del código 16.

```
1 xa= []
2 ya= []
3 for n in range(len(xx)):
4     r = np.sqrt(xx[n]**2 + yy[n]**2)
5 
6     if r<0.5:
7         xa.append(xx[n])
8         ya.append(yy[n])
```

Código 16: Fragmento de código que crea una lista con los valores que se encuentran dentro de una circunferencia de radio 0,5 centrada en el origen.

Con la circunferencia ya encontrada se procedió a calcular el valor de π, para ello, siguiendo la ecuación 2.24, se dividieron todos los punto que se encontraban dentro de la circunferencia, es decir, el tamaño de xa con todos los puntos que se encontraban dentro del cuadrado, es decir, $4N = 80000$ y el resultado se multiplicó por 4. De este modo, se obtuvo que $\pi \approx 3{,}1424$.

Para observar cómo variaba el valor de π con respecto al número de valores generados aleatoriamente se creó que gráfico 2.13. En el eje x de este gráfico se muestra la cantidad de números generados y el eje y representa el valor que toma π. En resumen, se puede decir que mientras mayor sea la cantidad de números generados aleatoriamente el valor obtenido al resolver la ecuación 2.24, en general, tenderá al valor real de π.

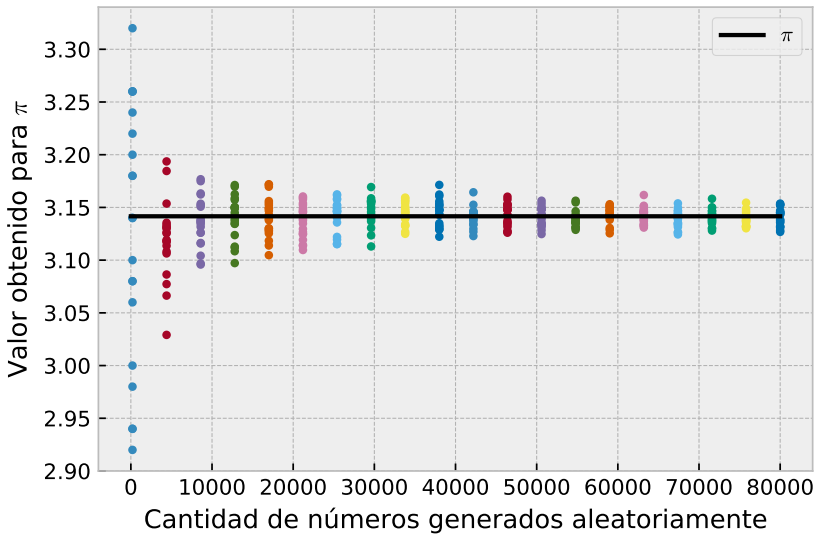


Figura 2.13: Este gráfico relaciona la cantidad de números generados aleatoriamente con el valor obtenido para π, además está graficada una recta que indica el valor real de π

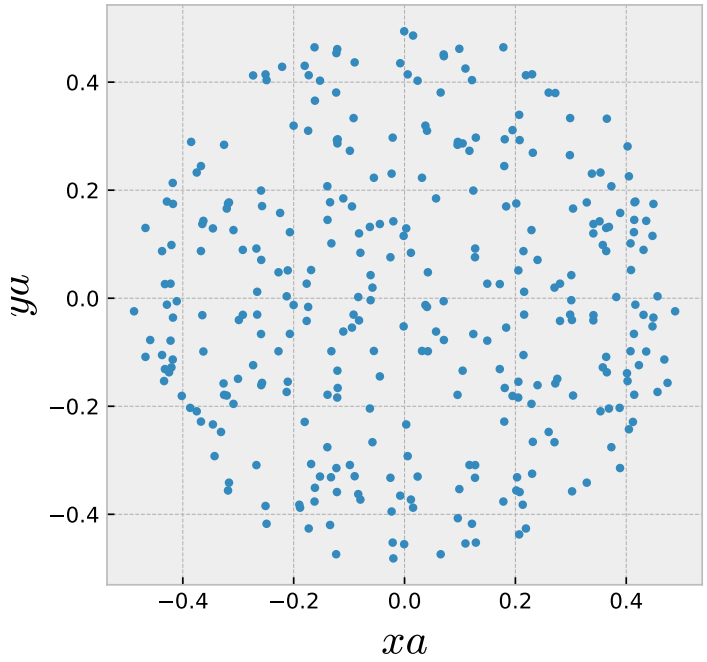


Figura 2.12: 100 números generados aleatoriamente dentro de una circunferencia de radio 0,5.

2.7.3. Conclusiones

En este laboratorio se trabajó con números aleatorios, ocupándolos para encontrar el número π. Primero se crearon número aleatorios del $]-0{,}5\,0{,}5[$ para el eje x y para el eje y, luego se graficaron obteniéndose de esta manera un cuadrado de largo 1 con centro en el origen. Después se separaron los puntos generados aleatoriamente que se encontraban dentro de una circunferencia de radio 0,5 inscrita dentro del cuadrado y a través de un relación analítica de sus áreas, se encontró un valor aproximado de π. Finalmente se concluyó que mientras mayor era la cantidad de números generados aleatoriamente, entonces la aproximación al número π sería mejor.

En este laboratorio aprendí a que los números aleatorios pueden ser útiles para trabajar con métodos numéricos como lo fue en este caso encontrando el número π. También aprendí que los números aleatorios generados por un computador no son realmente aleatorios y que nunca lo serán, pues necesitarán de todos algún algoritmo que los genere.

Por otra parte, me hubiese gustado trabajar un poco más en crear números aleatorios, quizás creando alguna función más compleja que se asemejara a `numpy.random.rand()`.

Capítulo 3

Conclusiones

Fecha de presentación: 14 de diciembre de 2022

En el presente portafolio se trabajaron en distintos problemas para los cuales se necesitaron conocimientos en cálculo numérico y en programación para desarrollar programas computacionales capaces de resolver derivadas, integrales, ecuaciones diferenciales ordinarias, etc. También se trabajaron con herramientas útiles en el área de la computación científica tales como Git, Github y LaTeX.

En las actividades de este laboratorio aprendí diversos métodos numéricos para resolver distintos problemas, como por ejemplo: calcular derivadas numéricas, la regla del punto medio para resolver integrales, el método de la bisección para encontrar ceros en funciones trabajando con la energía potencial de un péndulo, el método de Euler-Cromer y el método de Runge-Kutta para resolver ecuaciones diferenciales ordinarias como por ejemplo las ecuaciones de Lotka-Volterra y las ecuaciones de Lorenz, construir polinomios de Lagrange, usar la interpolación spline cúbica y usar la inversa de la matriz de Vandermonde para interpolar datos del promedio de manchas solares por año o usar funciones que generan números aleatorios para encontrar el numero π numéricamente. Por otra parte, creo que en el laboratorio de derivadas me faltó aprender cómo calcular los errores, pero lo demás creo haberlo entendido bien. En general me costó entender la demostración de los métodos en la mayoría de las actividades, a excepción de las actividades de integrales, números aleatorios y de ceros de funciones donde creo haber entendido todo.

En futuras versiones de este curso recomendaría seguir con la metodología que se estaba siguiendo en las clases de los días viernes, pues siento que las clases se impartían y se desarrollaban de buena manera y que los laboratorios eran útiles para practicar lo aprendido. No obstante, comenzar a trabajar con lo recién enseñado en la segunda clase de los viernes, al menos a mí y a un compañero con el que hablé, se nos hacía un tanto complicado, por ende, creo que dedicar los días miércoles para introducir y/o complementar la primera clase del viernes podría ser más provechoso, ya que personalmente no creo haberle sacado mucho provecho a la clase de los días miércoles.

Como ya dije, personalmente creo que en los laboratorios en los que aprendí mejor son los laboratorios de ceros de funciones, integrales y números aleatorios, esto queda evidenciado en el desarrollo de dichos laboratorio, ya que en estos expliqué más detenidamente los pasos de como fueron programados los respectivos métodos, en contraste, en otros laboratorios, como por ejemplo el de Runge-Kutta, solo se señaló en el código cuando se usó el respectivo método y no se explicó a fondo.

Luego, al comparar las evidencias expuestas en este portafolio con los objetivos iniciales del curso y con las primera ideas que tenía del curso antes de iniciarlo, por una parte creo que los objetivos del curso, tales como aprender a utilizar herramientas computacionales para resolver problemas aplicados a la física, se vieron concretados en el desarrollo de este portafolio y de sus actividades. Por otra parte, al comparar los resultados con mis expectativas iniciales, siento que mis expectativas se vieron sobrepasadas por los resultados, pues pensé que iba a ser un curso menos exigente y en donde iba a

aprender menos cosas. En resumen, diría que la creación de este portafolio fue sumamente útil, ya que no solo sirvió para desarrollar las actividades y que estas fueran evaluadas, si no que también sera de gran utilidad en un futuro cuando necesite programar.

Para finalizar, creo que el trabajo que realicé durante este curso se adecuó a la exigencia que este requería y si pudiera haber cambiado algo en mi método de estudio, creo que hubiese sido estudiar por mi mismo las demostraciones de los métodos enseñados en las clases. Por otra parte, espero recurrir a este portafolio en un futuro de la carrera o ya egresado cuando necesite resolver ciertos problemas que requieran programación.

Bibliografía

[1] Cátedra de Métodos Numéricos - Dpto. de Matemática - Facultad de Ingeniería - U.N.M.d.P., *Método de spline*, http://www3.fi.mdp.edu.ar/metodos/apuntes/spline.pdf, 2022, [Online; accessed 9-December-2022].

[2] Wikipedia contributors, *Atractor de lorenz — Wikipedia, the free encyclopedia*, https://es.wikipedia.org/wiki/Atractor_de_Lorenz, 2022, [Online; accessed 28-October-2022].

[3] _____, *Ecuaciones lotka–volterra — Wikipedia, the free encyclopedia*, https://es.wikipedia.org/wiki/Ecuaciones_LotkaâĂŞVolterra, 2022, [Online; accessed 1-September-2022].

[4] _____, *Equilibrio mecánico — Wikipedia, the free encyclopedia*, https://es.wikipedia.org/wiki/Equilibrio_mecÃąnico#Estabilidad_del_equilibrio, 2022, [Online; accessed 1-September-2022].

[5] _____, *Interpolación — Wikipedia, the free encyclopedia*, https://es.wikipedia.org/wiki/InterpolaciÃşn, 2022, [Online; accessed 9-December-2022].