



Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ingeniería Mecánica e Industrial



Temas Selectos de Programación I

Profesor: Ing. Barrón Velázquez Gersain

Grupo 3

Memoria de tareas y apuntes

Alumno:

- Goytia González Jorge Hadamard

Semestre 2021-1

Tarea: Reporte de la película “*The Imitation Game*” (“*El Código Enigma*”)

Como se pude ver durante la película, está ambientada en uno de los más grandes conflictos de la historia, como lo es la gran segunda guerra mundial, en la cual, sumergidos en desesperación para terminar con el conflicto, se recurrió a la formación de un equipo de mentes brillantes multidisciplinarias.

Nuestro personaje principal es Alan Turing, matematico de Cambridge, a quien se le concede el título del padre de las computadoras actuales, que con su peculiar forma de razonamiento, se ofrece para dicho equipo para decifrar el código de cifrado aleman más dificil, al que se referian como “Enigma”. La peculiaridad de este código de cifrado es que se reiniciaba a las 6 de la mañana de cada día, por lo que tenian las horas contadas para darle solución, aunque a pesar de ser un equipo de 4 personas principalmente, se veian frustrados al no conseguir decifrar el código a tiempo, ya que sería diferente al día siguiente.

Para dar solución al problema, recurrieron a usar otra máquina Enigma, que fue robada por la fuerza de inteligencia polaca durante un asedio, para ello, se reunieron todos los mensaje alemanes interceptados anteriormente por equipos de criptografos esperando encontrar el secreto paraa desbaratar este problemático cifrado de mensajes.

Una vez teniendo la máquina Enigma en operación, a la que Alan bautizo con el nombre de “Cristopher” en honor de alguien importante para él durante se época de estudiante, cuyo funcionamiento era el opuesto de la Enigma original, no tenian ningún resultado en cuanto a decifrar los mensajes que tenian, ya que lo que estaban haciendo era buscar practicamente a siegas, sin ningún indicio, por lo que recurrieron a un análisis, buscando patrones o algún idicio común en los mensajes interceptados anteriormente.

Una vez que terminaron el análisis de los mensajes encontraron que en todos, en todos ellos aparecian las siguientes palabras “Heil Hitler”, por lo que, con el tiempo en contra dado que era de noche, corren hacia Cristopher para ingresar esas palabras. Despues de unos cuantos segundos de operación de la maquina, esta se detuvo, proporcionandole el significado real de uno de los mensajes que ingresaron el cual les decia la proxima flota aliada que sería atacada.

Algo un tanto cruel, pero necesario, fue que mediante la estadística y la probabiidad, tomaban la decision de si se podia o no intervenir para salvar vidas, ya que si lo hacian de forma recurrente con todos, los alemanes sabrian que si cifrado ya no era un enigma.

Dado a este desarrollo, se dio pie a futuras investigaciones lo que llevó al concepto de la máquina de Turing, el que nos represente una idealización de computación capaz de almacenar información infinita. Esta fue la aportación de Turing en la guerra, cosa que significó la victoria de esta por ataques previamente calculados y estudiados para que el ejército Nazi, asegurando la victoria.

Tipos de inteligencia artificial

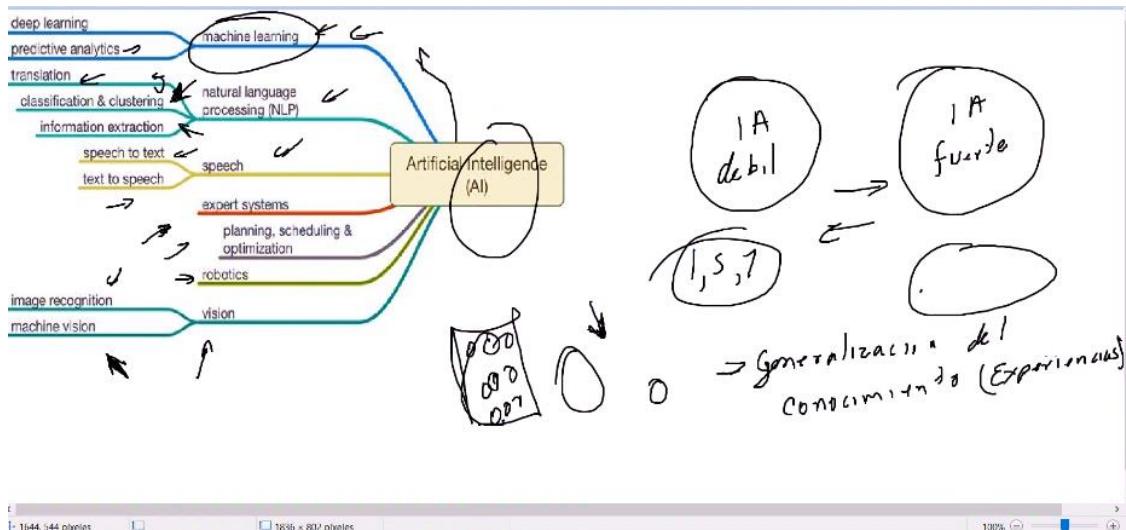
Primero, ¿Qué es la inteligencia artificial?. Es la combinación de algoritmos planteados con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano. Es una tecnología que aun no se ha perfeccionado dado el nivel de complejidad al que se quiera llegar durante su desarrollo/implementación, pero que desde hace ya varios años se encuentra presente prácticamente en todas nuestras herramientas tecnológicas de nuestro día a día, permitiendo trabajar con grandes cumulos de información (Big Data).

Existen diversas clasificaciones para la inteligencia artificial, siendo:

- **Machine Learning:** esta rama de la inteligencia artificial tiene como principal objetivo el imitar el comportamiento humano, o en otras palabras, tratan de crear un razonamiento semejante al nuestro al tratar de resolver algún problema, siendo este muy bien definido, esta rama tiene dos subconjuntos:
 - **Deep learning:** es un subconjunto del machine learning, en el que los programas empleados “aprenden” de un gran conjunto de datos, realizando una tarea repetitiva para mejorar el resultado de manera progresiva.
 - **Predictive Analytics:** este subconjunto emplea métodos estadísticos en un conjunto de datos de hechos sucedidos, para obtener un modelo del comportamiento de esos datos y tratar de predecir el comportamiento a futuro.
- **Natural Language Processing (NLP):** esta rama trabaja mediante métodos lógicos basados en la gramática, teniendo como componentes un análisis de léxico, sintáctico, semántico y pragmático, esta división de la IA es la que se usa en los traductores.
- **Speech:** Va de la mano con el NLP, ya que el speech se encarga de convertir mediante métodos NLP texto en voz, o viceversa, voz en una cadena de texto.
- **Expert Systems:** esta rama emula a un experto o profesional humano en la resolución de problemas en cierto campo, mediante reglas previamente establecidas, basado en casos o modelos probabilísticos para obtener una solución. Permiten aumentar la eficiencia y reducir los tiempos comparando con el experto humano.
- **Planning, scheduling and optimization:** esta rama de la inteligencia artificial se enfoca en realizar estrategias o secuencias de acción, como sería en robots autónomos y vehículos no tripulados, que principalmente sería la aplicación en procesos automatizados (industria).
- **Robotics:** esta rama ha desarrollado desde el principio de la informática con diversidad de objetivos: la automatización de procesos industriales, las aplicaciones militares y la exploración espacial. La evolución de la robótica también ha pasado por su intento de construir robots con forma humana e imitar su comportamiento. Un claro ejemplo sería en los robots de exploración/reconocimiento.
- **Vision:** esta rama trabaja con lo que es el reconocimiento de imágenes y la visión computacional, que trata de interpretar y extraer el significado de elementos visuales reales, como caras (reconocimiento facial) y objetos.

También tenemos otras dos clasificaciones, siendo la Inteligencia Artificial Fuerte y la Inteligencia Artificial Débil:

- IA Fuerte: En este caso se considera que un ordenador puede tener una mente y unos estados mentales, y que, por tanto, un día será posible construir uno con todas las capacidades de la mente humana. Este ordenador será capaz de razonar, imaginar, así como ser consciente de sí mismo (teóricamente hablando).
- IA Débil: Este tipo de sistemas son capaces de resolver problemas muy bien definidos y acotados. La inteligencia artificial débil es la que ha provocado la verdadera explosión de esta disciplina en los últimos tiempos: se han aplicado distintas técnicas como lo es el **machine learning / deep learning** para lograr resolver problemas específicos, y los resultados han sido excepcionales.



Referencias:

- <https://ovrmind.com/ramas-inteligencia-artificial/>
- https://www.fqcsic.es/lychnos/es_es/articulos/inteligencia_artificial
- <https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>
- <https://www.ibm.com/mx-es/cloud/deep-learning>
- <https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/>
- https://es.wikipedia.org/wiki/Sistema_experto

Aprendizaje supervisado, no supervisado y reforzado

- **Aprendizaje supervisado:** es una rama del machine learning donde partimos de un conjunto de datos etiquetados previamente en la resolución de un problema, es decir, conocemos el valor deseado para el conjunto de datos que disponemos, el cual se emplea para el entrenamiento de un algoritmo.
- **Aprendizaje no supervisado:** de igual manera, esta rama parte de un conjunto de datos los cuales no se encuentran etiquetados, es decir, no sabe a qué respuesta debe de llegar con ese conjunto de datos. De esta manera el algoritmo trata de interpretar y dar sentido a los datos mediante la búsqueda de patrones y características.
- **Aprendizaje reforzado:** este aprendizaje es similar al entrenamiento de una mascota, ya que mediante estímulos (refuerzo), se premia o se castiga de cierta manera si la acción que se realiza es correcta o no. De esta manera, el algoritmo va aprendiendo desde cero.

- Regresión (lineal o logística)
- Árboles de decisión
- Clasificación
- Clusterización
- Redes neuronales

*****Clusterización es diferente de clasificación*****

La clasificación de datos se basa en la discriminación por atributos en objetos, mientras que la clusterización se basa en la identificación de similitudes y en base a ellas realiza la agrupación.

Referencias:

- <https://www.aprendemachinelearning.com/aprendizaje-por-refuerzo/>
<https://healthdataminer.com/data-mining/aprendizaje-supervisado-y-no-supervisado/>
<https://www.tibco.com/es/reference-center/what-is-supervised-learning>
<https://www.apd.es/algoritmos-del-machine-learning/>
<https://www.grapheverywhere.com/diferencias-entre-clusterizacion-y-clasificacion/>
<https://www.clubdetecnologia.net/blog/2018/los-10-modelos-mas-populares-de-inteligencia-artificial/>

7 / Septiembre / 2021

Ley de Coulomb

El físico e ingeniero francés Charles-Augustin de Coulomb (1736-1806), es a quién se le atribuye esta ley debido a las conclusiones que presentó, dados sus experimentos de cuerpos cargados:

1. Los cuerpos cargados sufren una fuerza de atracción o repulsión al aproximarse.
2. El valor de dicha fuerza es proporcional al producto del valor de sus cargas.
3. La fuerza que es de atracción si las cargas son de signos opuestos y de repulsión si son del mismo signo.
4. La fuerza es inversamente proporcional al cuadrado de la distancia que los separa.



Estas conclusiones son la esencia de la ley de Coulomb. Siendo la siguiente ecuación quién la caracteriza:

$$F = k * \frac{q_1 * q_2}{r^2}$$

En donde tenemos que k es nuestra constante de proporción $k = 9 * 10^9 [N * m^2/C^2]$. Los valores q_1 y q_2 corresponden a dos cargas puntuales, cuya unidad son Coulombs [C]. Finalmente, r el valor de la distancia entre las cargas, en metros [m].

Temas Selectos de Programación I

Goytia González Jorge Hadamard

Para comenzar con la introducción del curso revisamos algunos videos con ejercicios resueltos, para maquetarlos en Excel, y una vez comprobado el funcionamiento y la metodología realizada, una vez comprendido eso, podíamos pasar a nuestro ambiente de programación (Visual Studio).

TODAS LAS CARGAS DE MISMO SIGNO (+)			
		q1	q3
		q2	
$q1=$	8.00E-06		
$q3=$	7.00E-06		
$q2=$	6.00E-06		
$k=$	9.00E+09		
$d=$	1.20E-01		
d es entre $q1$ y $q2$			
Ley de Coulomb (3 cargas equidistantes)			
		$f1$	$f2$
		$\frac{k \cdot q1 \cdot q3}{(d/2)^2 \cdot (d/2)}$	$\frac{k \cdot q3 \cdot q2}{(d/2)^2 \cdot (d/2)}$
		1.40E+02	1.05E+02
			$F = f1 - f2$
			35.00
$q1=$	8.00E-06		
$q3=$	7.00E-06		
$q2=$	6.00E-06		
$k=$	9.00E+09		
$d=$	1.20E-01	$r = 1.00E-02$	
d es entre $q1$ y $q2$			
r es entre $q1$ y $q3$			
Coulomb (3 cargas, dando la distancia entre dos cargas y extremos)			
		$f1$	$f2$
		$\frac{k \cdot q1 \cdot q3}{r^2}$	$\frac{k \cdot q3 \cdot q2}{(d-r)^2 \cdot (d-r)}$
		5.04E+03	3.12E+01
			$F = f1 - f2$
			5008.76

Tarea 2: Excel – Ejercicio de 3 cargas con resultante igual a 0

3 cargas (fuerza resultante siempre 0, obtener distancia)			
		proponiendo	
		$q1$	$q2$
$q1=$	8.00E-06		
$q3=$	7.00E-06		
$q2=$	6.00E-06		
$k=$	9.00E+09		
$d=$	6.00E-02		
d es entre $q1$ y $q2$			
		$r1$	$r2$
		$\frac{d}{\sqrt{\frac{q1}{q2} - 1}}$	$\frac{d}{\sqrt{\frac{q1}{q2} - 1}}$
		3.88E-01	-2.78E-02
		$r1$	$r2$
		$\frac{d}{\sqrt{\frac{q2}{q1} + 1}}$	$\frac{d}{\sqrt{\frac{q2}{q1} + 1}}$
		3.22E-02	4.48E-01
proponiendo			
$q1$			
$q2$			
$q3$			

Referencias:

<https://www.fisicalab.com/apartado/ley-de-coulomb>

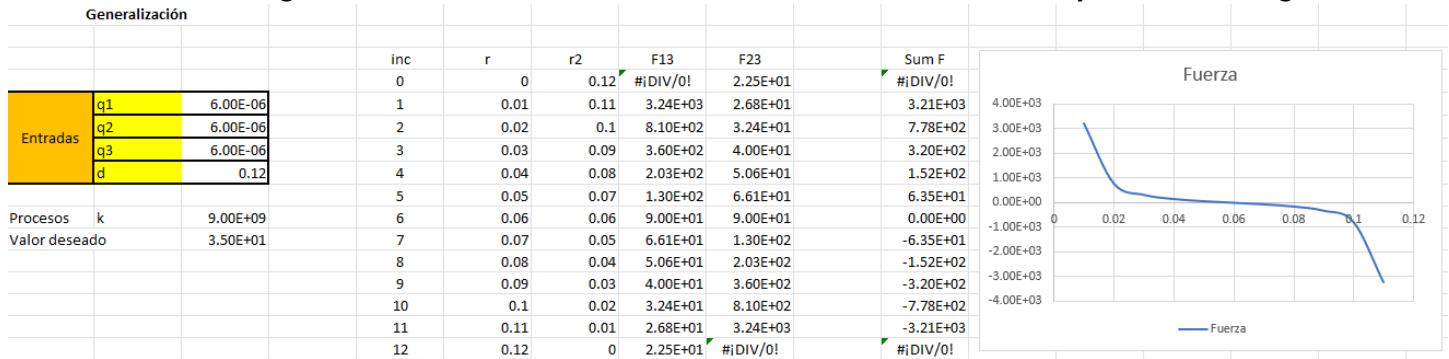
http://www.sc.ehu.es/sbweb/fisica/electromagnet/campo_electrico/fuerza/fuerza.htm

<https://es.khanacademy.org/science/fisica-pe-pre-u/x4594717deeb98bd3:electrostatica/x4594717deeb98bd3:ley-de-coulomb/a/coulombs-law-and-electric-force-ap-physics-1>

https://youtu.be/nVoWS69u_yQ

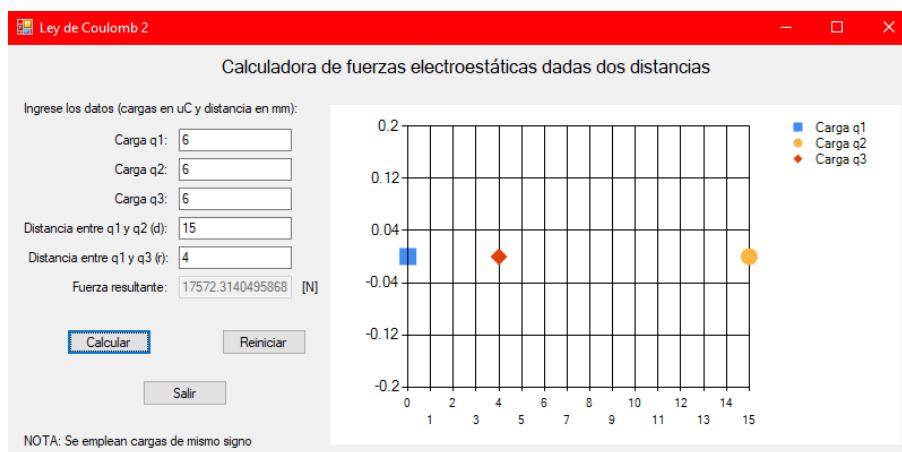
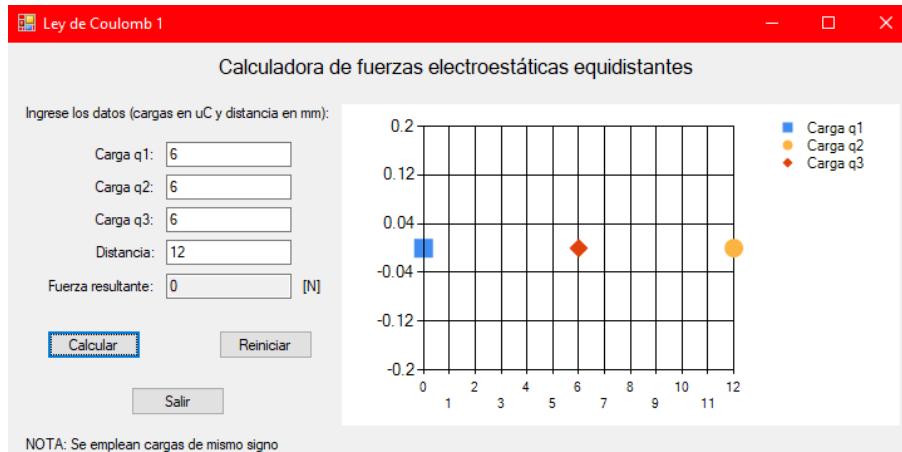
9 / Septiembre / 2021

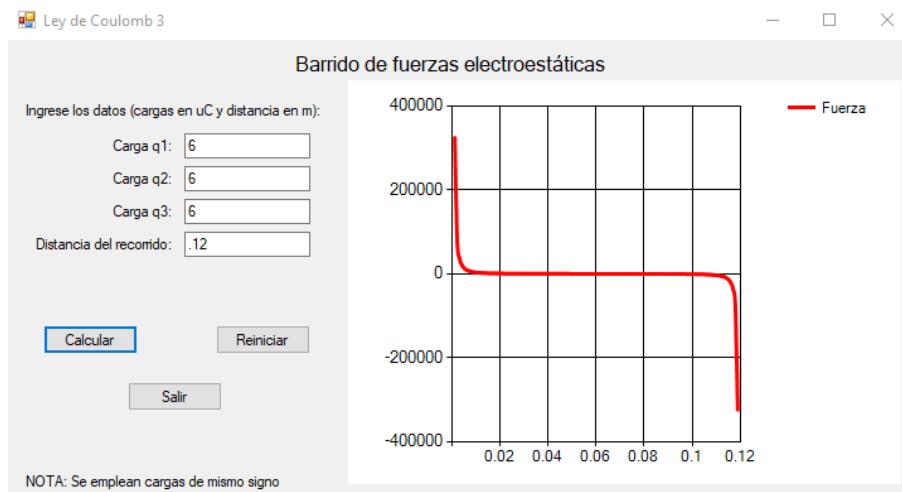
Continuamos trabajando con lo que es el tema de ley de coulomb, ahora mediante lo que sería el barrido de una sola carga, entre otras dos (el caso general de 3 cargas colineales), obteniendo la gráfica del comportamiento de la fuerza resultante, haciendo la introducción a lo que es el error, ya que buscábamos un valor específico entre todos los obtenidos (una fuerza de 35 Newtons), de manera que el error fuese 0. Primero realizamos un barrio sencillo para pocos datos y posteriormente lo extendimos a intervalos aún más pequeños.



El documento de Excel que corresponde a las tareas de este tema contiene dichos ejercicios de clase y tareas acerca del tema.

Tarea 3: C# - Pasar los ejemplos de Excel a C# (Ley de Coulomb)





Tarea 4: Investigar error y que es el error cuadrático medio

Tarea 5: Inicio de nuestra librería de Matlab, métodos de aritmética básica y linspace (Anexo al final)

14 / Septiembre / 2021

Error

Una actividad frecuente del profesional de la ingeniería consiste en trabajar con modelos matemáticos representativos de un fenómeno físico. Estos modelos son abstracciones matemáticas que distan mucho de representar exactamente al fenómeno bajo estudio debido principalmente a las carencias y dificultades que aún posee el humano de la comprensión total de la naturaleza.

Como consecuencia de esto existen diferencias entre los resultados obtenidos experimentalmente y los emanados propiamente del modelo matemático.

A las diferencias cuantitativas entre los dos modelos se les denomina **errores**.

Estos errores se cuantifican de dos maneras diferentes:

- **Error Absoluto:** el error absoluto es la diferencia absoluta entre un valor real y un aproximado. Está dado por la siguiente fórmula:

$$E = |V_{real} - V_{aprox}|$$

El error absoluto recibe este nombre ya que posee las mismas dimensiones que la variable bajo estudio.

- **Error relativo:** corresponde a la expresión en porcentaje de un error absoluto; en consecuencia, este error es adimensional. Expresado por la fórmula:

$$e = \frac{|V_{real} - V_{aprox}|}{V_{real}} * 100\%$$

Error Cuadrático Medio (MSE o ECM)

En estadística, el error cuadrático medio (MSE o Mean Square Error en inglés) es una forma de evaluar la diferencia entre un estimador y el valor real de la cantidad que se quiere calcular. El MSE mide el promedio del cuadrado del "error", siendo el error el valor en la que el estimador difiere de la cantidad a ser estimada. El MSE se obtiene con la siguiente ecuación:

$$MSE = \sum_{i=1}^n (x_i - \bar{x})^2$$

Donde x_i corresponde a cada uno de nuestros valores obtenidos.

Nuestro valor deseado es \bar{x} , o, dicho de otra forma, el valor pronosticado.

Finalmente n el número total de datos.

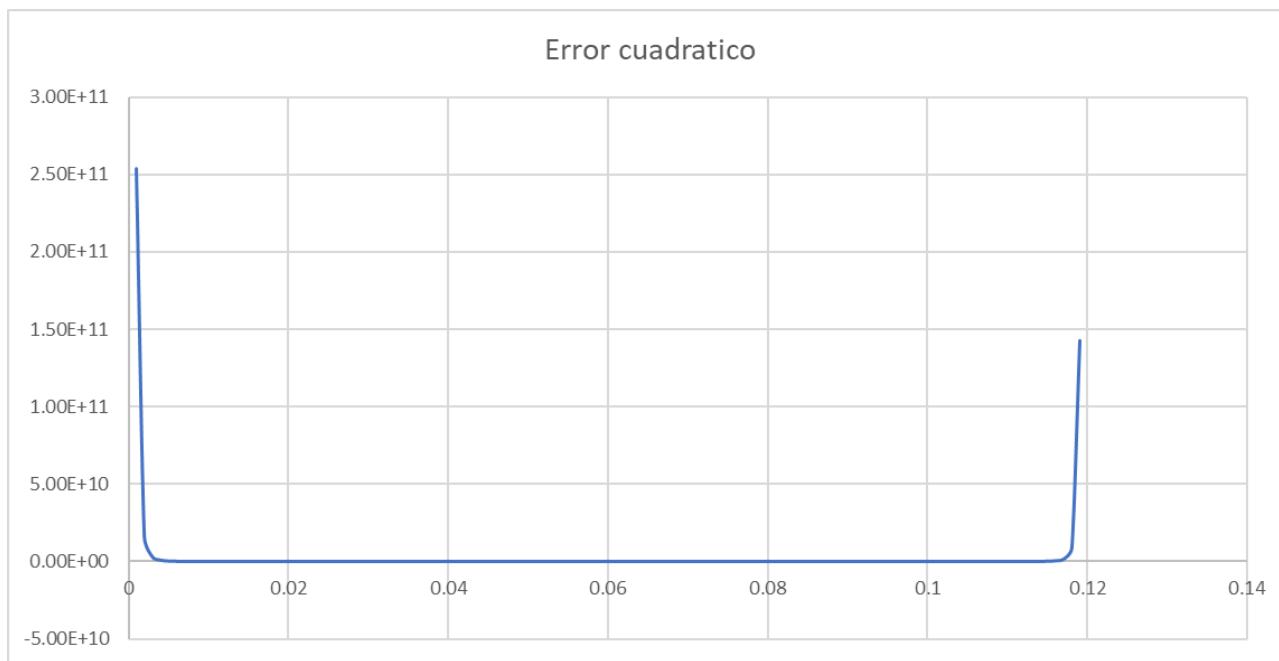
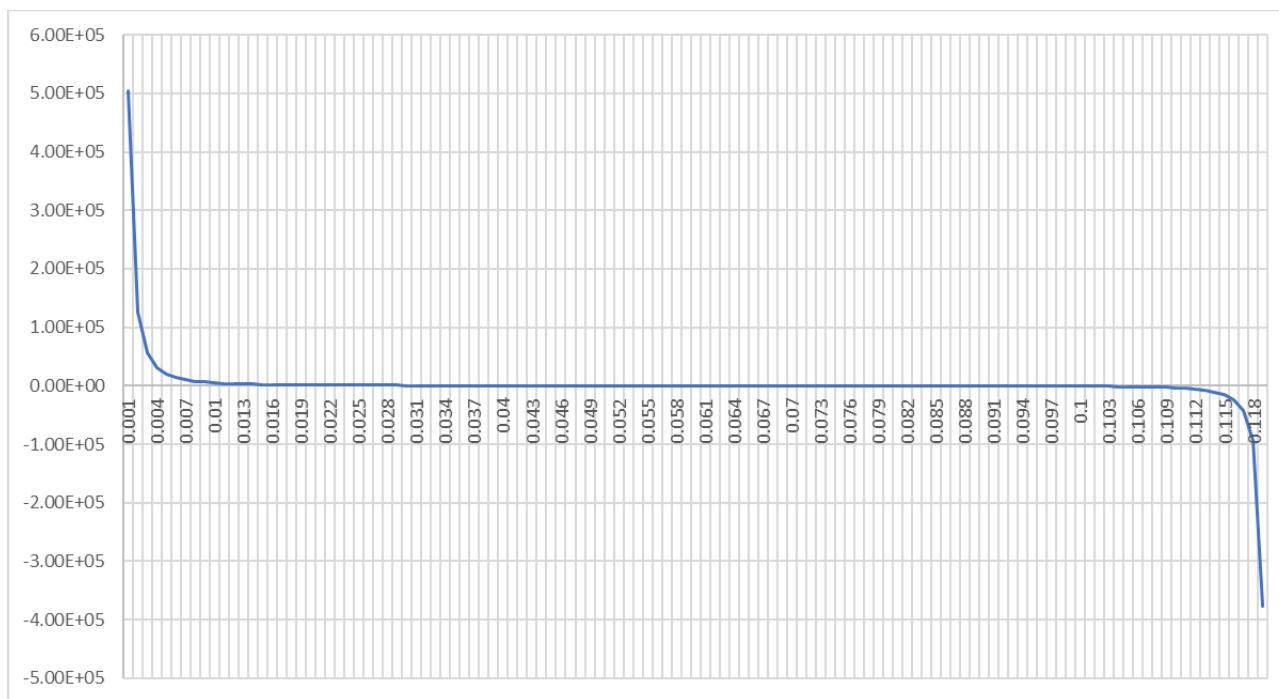
Este cálculo de errores es de gran utilidad cuando nuestros conjuntos de datos y nuestro valor deseado presentan grandes errores.

Durante la clase, nosotros solo trabajamos con los errores cuadráticos de cada fuerza resultante, un detalle importante que encontré acerca de este cálculo de error es que puede emplearse el valor absoluto, pero no es recomendable, ya que lo que se busca al obtener el comportamiento del error (asemejándose a una parábola), es el derivar dicho valor, a manera de obtener una pendiente.

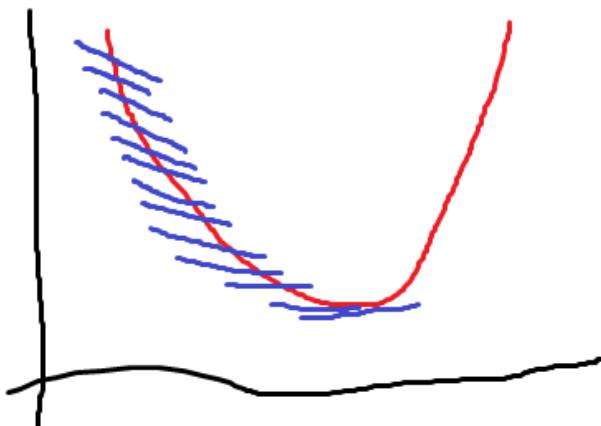
La hoja de cálculo que se desarrolló durante la clase es la siguiente, que de igual manera la distancia máxima con la que trabajamos era de .12 metros:

		inc	r	I2	F13	F23	Sum F	Error	ErrorCuadrático
Entradas	q1	8.00E-06	0	0.12	#DIV/0!	2.63E+01	#DIV/0!	#DIV/0!	#DIV/0!
	q2	6.00E-06	0.1	0.001	0.119	5.04E+05	2.67E+01	5.04E+05	2.54E+11
	q3	7.00E-06	0.2	0.002	0.118	1.26E+05	2.71E+01	1.26E+05	1.59E+10
	d	0.12	0.3	0.003	0.117	5.60E+04	2.76E+01	5.60E+04	3.13E+09
Procesos	k	9.00E+03	0.4	0.004	0.116	3.15E+04	2.81E+01	3.15E+04	9.90E+08
	Valor deseado	3.50E+01	0.5	0.005	0.115	2.02E+04	2.86E+01	2.01E+04	4.05E+08
Valores calculados			0.6	0.006	0.114	1.40E+04	2.91E+01	1.40E+04	1.95E+08
			0.7	0.007	0.113	1.03E+04	2.96E+01	1.03E+04	1.05E+08
			0.8	0.008	0.112	7.88E+03	3.01E+01	7.84E+03	6.15E+07
			0.9	0.009	0.111	6.22E+03	3.07E+01	6.19E+03	6.16E+03
			1	0.01	0.11	5.04E+03	3.12E+01	5.01E+03	4.97E+03
			1.1	0.011	0.109	4.17E+03	3.18E+01	4.13E+03	4.10E+03
			1.2	0.012	0.108	3.50E+03	3.24E+01	3.47E+03	3.43E+03
			1.3	0.013	0.107	2.98E+03	3.30E+01	2.95E+03	2.91E+03
			1.4	0.014	0.106	2.57E+03	3.36E+01	2.54E+03	2.50E+03
			1.5	0.015	0.105	2.24E+03	3.43E+01	2.21E+03	2.17E+03
			1.6	0.016	0.104	1.97E+03	3.49E+01	1.93E+03	1.90E+03
			1.7	0.017	0.103	1.74E+03	3.56E+01	1.71E+03	1.67E+03
			1.8	0.018	0.102	1.56E+03	3.63E+01	1.52E+03	1.48E+03
			1.9	0.019	0.101	1.40E+03	3.71E+01	1.36E+03	1.32E+03
			2	0.02	0.1	1.26E+03	3.78E+01	1.22E+03	1.19E+03
			2.1	0.021	0.099	1.14E+03	3.86E+01	1.10E+03	1.07E+03
			2.2	0.022	0.098	1.04E+03	3.94E+01	1.00E+03	9.57E+02
			2.3	0.023	0.097	9.53E+02	4.02E+01	9.13E+02	8.78E+02
			2.4	0.024	0.096	8.75E+02	4.10E+01	8.34E+02	7.99E+02
			2.5	0.025	0.095	8.06E+02	4.19E+01	7.65E+02	7.30E+02
			2.6	0.026	0.094	7.46E+02	4.28E+01	7.03E+02	6.68E+02
			2.7	0.027	0.093	6.91E+02	4.37E+01	6.48E+02	6.13E+02
			2.8	0.028	0.092	6.43E+02	4.47E+01	5.98E+02	5.63E+02

Y graficando tanto las fuerzas resultantes como los errores cuadráticos medios, obtuvimos las siguientes gráficas:



Siendo esta la imagen a la que se trata de llegar, en la que marcamos lo que vendría siendo el comportamiento de las pendientes entre cada dos puntos para el ECM.



Tarea 6: Obtener la tangente entre dos puntos para el Excel de la clase

Generalización										
	inc	r	r2	F13	F23	Sum F	Error	ErrorCuadrático	Pendiente	
Entradas	q1	6.00E-06	0	0.12	#DIV/0!	2.25E+01	#DIV/0!	#DIV/0!	#DIV/0!	
	q2	6.00E-06	0.1	0.001	0.119	3.24E+05	2.29E+01	3.24E+05	1.05E+11	
	q3	6.00E-06	0.2	0.002	0.118	8.10E+04	2.33E+01	8.10E+04	6.56E+09	
	d	0.12	0.3	0.003	0.117	3.60E+04	2.37E+01	3.60E+04	1.29E+09	
Procesos	k	3.00E+09	0.5	0.005	0.115	1.30E+04	2.45E+01	1.29E+04	1.67E+08	
	Valor deseado	0.00E+00	0.6	0.006	0.114	9.00E+03	2.49E+01	8.98E+03	8.06E+07	
			0.7	0.007	0.113	6.61E+03	2.54E+01	6.53E+03	4.34E+07	
			0.8	0.008	0.112	5.06E+03	2.58E+01	5.04E+03	2.54E+07	
			0.9	0.009	0.111	4.00E+03	2.63E+01	3.97E+03	1.58E+07	
			1	0.01	0.11	3.24E+03	2.68E+01	3.21E+03	1.03E+07	
			1.1	0.011	0.109	2.68E+03	2.73E+01	2.65E+03	7.02E+06	
			1.2	0.012	0.108	2.25E+03	2.78E+01	2.22E+03	4.94E+06	
			1.3	0.013	0.107	1.92E+03	2.83E+01	1.89E+03	3.57E+06	
			1.4	0.014	0.106	1.65E+03	2.88E+01	1.62E+03	2.64E+06	
			1.5	0.015	0.105	1.44E+03	2.94E+01	1.41E+03	1.99E+06	
			1.6	0.016	0.104	1.27E+03	3.00E+01	1.24E+03	1.53E+06	
			1.7	0.017	0.103	1.12E+03	3.05E+01	1.09E+03	1.19E+06	
			1.8	0.018	0.102	1.00E+03	3.1E+01	9.69E+02	9.39E+05	
			1.9	0.019	0.101	8.98E+02	3.18E+01	8.66E+02	8.66E+02	
			2	0.02	0.1	8.10E+02	3.24E+01	7.78E+02	7.50E+05	
			2.1	0.021	0.099	7.35E+02	3.31E+01	7.02E+02	4.92E+05	
			2.2	0.022	0.098	6.69E+02	3.37E+01	6.36E+02	4.04E+05	
			2.3	0.023	0.097	6.12E+02	3.44E+01	5.78E+02	3.34E+05	
			2.4	0.024	0.096	5.63E+02	3.52E+01	5.27E+02	2.78E+05	
			2.5	0.025	0.095	5.18E+02	3.59E+01	4.82E+02	2.33E+05	
			2.6	0.026	0.094	4.79E+02	3.67E+01	4.43E+02	1.96E+05	
			2.7	0.027	0.093	4.44E+02	3.75E+01	4.07E+02	1.66E+05	

Referencias:

https://www.ingenieria.unam.mx/pinilla/PE105117/pdfs/tema1/1_aproximacion_numerica_y_errores.pdf
<https://www.iartificial.net/error-cuadratico-medio-para-regresion/>

16 / Septiembre / 2021

*****NO HUBO CLASE*****

Realizamos un programa de consola en C# en el cual tratamos de replicar los ejercicios de errores con los ejemplos de cargas de ley de Coulomb, a manera de recapitulación de todas las iteraciones que habíamos hecho sobre el barrido de fuerzas para la ley de Coulomb.

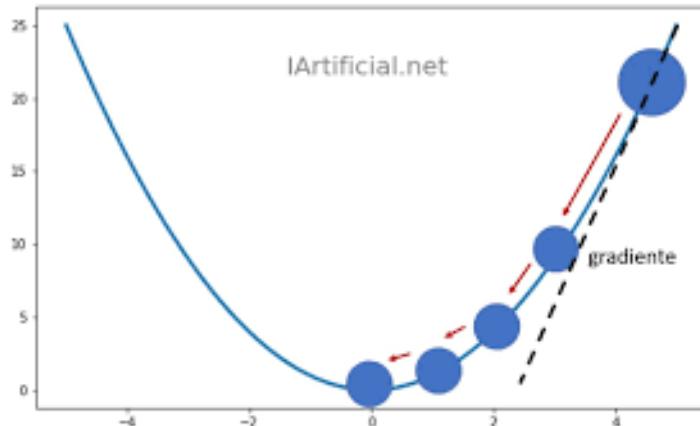
Corroboramos el funcionamiento correcto de los métodos de nuestra librería, que en mi caso la función Linspace tenía un error al crear el arreglo de forma decreciente.

Código de la clase (que se modificó de tarea):

```
1  using System;
2  [using MATLAB_Library;
3
4  namespace Pruebas
5  {
6      class BarridoyErroresCargas
7      {
8          static void Main(string[] _)
9          {
10             //Llamamos a nuestra biblioteca
11             Basics prueba = new Basics();
12
13             //Console.WriteLine("Hola Mundo");
14             //Console.ReadLine();
15
16             //Entradas
17             double q1 = 8 * Math.Pow(10, -6);
18             double q2 = 6 * Math.Pow(10, -6);
19             double q3 = 7 * Math.Pow(10, -6);
20
21             Console.WriteLine("\nEl valor de q1 es: {0} C", q1);
22             Console.WriteLine("\nEl valor de q2 es: {0} C", q2);
23             Console.WriteLine("\nEl valor de q3 es: {0} C", q3);
24
25             double[] r1 = prueba.Linspace(.0, .12, .001); //Se prueba la función de linspace con primer y ultimo dato de vector, con ese resultado se crea el vector resultante
26             double[] r2 = prueba.Linspace(.12, 0, .001);
27
28             double[] F13 = FuerzaVect(r1, q1, q3);
29             double[] F23 = FuerzaVect(r2, q2, q3);
30
31             Console.WriteLine("\nVector resultante r1: ");
32             Basics.PrintArray(r1);
33
34             Console.WriteLine("\nVector resultante r2: ");
35             Basics.PrintArray(r2);
36             Console.WriteLine("Fuerza en 3 debido a 1: ");
37             Basics.PrintArrayE(F13);
38             Console.WriteLine("Fuerza en 3 debido a 2: ");
39             Basics.PrintArrayE(F23);
40             //Console.WriteLine("Vector resultante: [" + String.Join(", ", F13) + "]");
41             //Console.WriteLine("Vector resultante: [" + String.Join(", ", F23) + "]");
42             //Console.WriteLine("Vector resultante: " + String.Join(" ", array4));
43
44             double[] FT = prueba.DiffArray(F13, F23);
45             Console.WriteLine("Arreglo de fuerzas totales: ");
46             Basics.PrintArrayE(FT);
47
48             double[] EC = prueba.ECM(FT, 35);
49             Console.WriteLine("Arreglo de errores cuadráticos: ");
50             Basics.PrintArrayE(EC);
51
52             Console.Read();
53
54             //Clase martes 14 de septiembre
55             private static double Fuerza(double q, double q3, double r)
56             {
57                 double k = (9 * Math.Pow(10, 9));
58                 double FT = (k * q * q3) / (Math.Pow(r, 2));
59
60                 return FT;
61             }
62
63             private static double[] FuerzaVect(double[] a, double q, double q3)
64             {
65                 double[] output = new double[a.Length];
66                 for (int i = 0; i < a.Length; i++)
67                 {
68                     output[i] = Fuerza(q, q3, a[i]);
69                 }
70                 return output;
71             }
72         }
73     }
74 }
```

Tarea 7: Corregir método Linspace, barrido de cargas con ECM en consola (modificación de la clase).

Realizamos la maqueta del método de descenso por gradiente de manera sencilla en Excel, especificando los pasos a realizar.



Teniendo la grafica anterior como idea para el maquetado, donde empezamos desde un punto cualquiera y comparamos tanto adelante como atrás el ECM, según en donde nos encontremos, recorremos el punto a donde el ECM sea menor.

Una vez dejando claro lo que debíamos hacer, llegamos a tener la siguiente hoja de cálculo en Excel para comprender este método.

			r1	r2	F31	F32	FT	Error
q1	8.00E-06	Paso 1: Elegir valor aleatorio	0.11	0.01	4.17E+01	3.78E+03	-3.74E+03	1.42E+07
q2	6.00E-06							
q3	7.00E-06	Paso 2: Elegir un punto superior e inferior	0.111	0.009	4.09E+01	4.67E+03	-4.63E+03	2.17E+07
d	0.12		0.109	0.011	4.24E+01	3.12E+03	-3.08E+03	9.71E+06
k	9.00E+09	Paso 3: Comparar los errores		0.109				
Valor	3.50E+01							
Paso	1.00E-03		r1	r2	F31	F32	FT	Error
		Paso 1: Elegir valor aleatorio	0.109	0.011	4.24E+01	3.12E+03	-3.08E+03	9.71E+06
		Paso 2: Elegir un punto superior e inferior	0.11	0.01	4.17E+01	3.78E+03	-3.74E+03	1.42E+07
			0.108	0.012	4.32E+01	2.63E+03	-2.58E+03	6.85E+06
		Paso 3: Comparar los errores		0.108				

Tarea 8: Realizar la función de descenso para nuestra librería, y probarla en consola.

```

1  using System;
2
3  namespace Descenso
4  {
5      class ProgramDescenso
6      {
7          private static void Main(string[] args)
8          {
9              //Entradas
10             double q1 = 8 * Math.Pow(10, -6);
11             double q2 = 6 * Math.Pow(10, -6);
12             double q3 = 7 * Math.Pow(10, -6);
13             double d = 0.12;
14             double r1 = 0.11; //Este es el valor aleatorio
15             double valor = 35;
16             double paso = .001;
17
18             //Se imprimen los datos en pantalla
19             Console.WriteLine("\nEl valor de q1 es: {0} [C]", q1);
20             Console.WriteLine("\nEl valor de q2 es: {0} [C]", q2);
21             Console.WriteLine("\nEl valor de q3 es: {0} [C]", q3);
22             Console.WriteLine("\nLa distancia total es: {0} [m]", d);
23             Console.WriteLine("\nEl valor aleatorio es: {0} [m]", r1);
24             Console.WriteLine("\nEl valor deseado es: {0} [N]", valor);
25             Console.WriteLine("\nEl valor del paso es: {0} [m]", paso);
26
27             //Hacemos el llamado de las funciones
28             //Se hace la primera iteración de la función descenso
29             Descenso(q1, q2, q3, d, r1, valor, paso);
30
31             Console.Read();
32         }
33     }
34     private static double Fuerzas(double q1, double q2, double q3, double d, double r1)
35     {
36         //Se obtiene la diferencia entre d y el valor aleatorio
37         double r2 = d - r1;
38
39         double k = (9 * Math.Pow(10, 9));
40         //Se calculan ambas fuerzas
41         double F31 = (k * q1 * q3) / (Math.Pow(r1, 2));
42         double F32 = (k * q2 * q3) / (Math.Pow(r2, 2));
43         //Se obtiene la fuerza total
44         double FT = F31 - F32;
45
46         return FT;
47     }
48
49     private static void Descenso(double q1, double q2, double q3, double d, double r1, double valor, double paso)
50     {
51         //Error cuadrático inicial
52         double FT1;
53         double EC1;
54
55         do
56         {
57             //Error cuadrático inicial
58             FT1 = Fuerzas(q1, q2, q3, d, r1);
59             EC1 = Math.Pow((FT1 - valor), 2);
60
61             //Elegimos un punto superior y se calcula la fuerza en ese punto
62             double r1S = r1 + paso;
63             double FT2 = Fuerzas(q1, q2, q3, d, r1S);
64             double EC2 = Math.Pow((FT2 - valor), 2); //Se calcula el error en este punto
65
66             //Elegimos un punto inferior y se calcula la fuerza en ese punto
67             double r1I = r1 - paso;
68             double FT3 = Fuerzas(q1, q2, q3, d, r1I);
69             double EC3 = Math.Pow((FT3 - valor), 2); //Se calcula el error en este punto
70
71             //Calculamos las pendientes asociadas
72             double pend1 = Math.Truncate(((EC2 - EC1) / (r1S - r1)) * 1000) / 1000;
73             double pend2 = Math.Truncate(((EC1 - EC3) / (r1 - r1I)) * 1000) / 1000;
74
75             //Comparamos los errores
76             if (EC1 > EC2)
77             {
78                 r1 = Math.Truncate(r1S * 1000) / 1000;
79                 Console.WriteLine("\nEl valor de pendiente es: {0}", pend1);
80                 Console.WriteLine("\nEl valor de r1 ahora es: {0} [mm]", r1);
81             }
82
83             else if (EC1 > EC3)
84             {
85                 r1 = Math.Truncate(r1I * 1000) / 1000;
86                 Console.WriteLine("\nEl valor de pendiente es: {0}", pend2);
87                 Console.WriteLine("\nEl valor de r1 ahora es: {0} [mm]", r1);
88             }
89         } while (EC1 > 1E-25);
90     }
91 }
92 
```

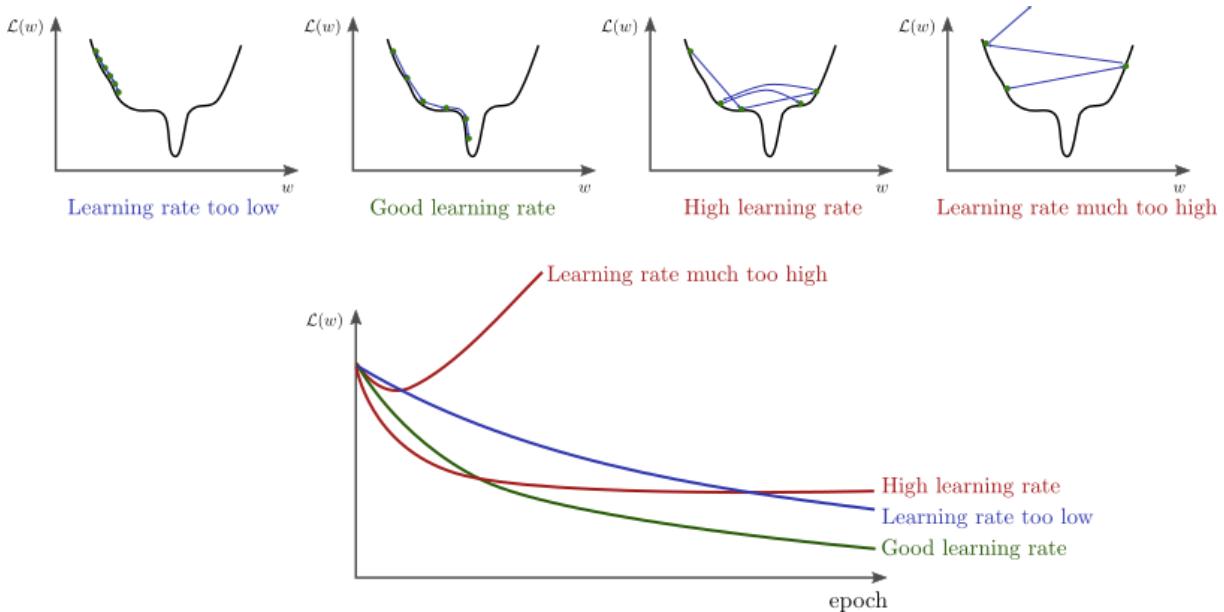
Teniendo la siguiente respuesta mediante la consola, hasta llegar a $r_1 = .06 \text{ mm}$, que es donde se encuentra nuestro ECM en 0:

Consola de depuración de Microsoft Visual Studio	Consola de depuración de Microsoft Visual Studio
El valor de q1 es: 8E-06 [C]	El valor de pendiente es: 894954788.444
El valor de q2 es: 6E-06 [C]	El valor de r1 ahora es: 0.105 [mm]
El valor de q3 es: 7E-06 [C]	El valor de pendiente es: 640392734.772
La distancia total es: 0.12 [m]	El valor de r1 ahora es: 0.104 [mm]
El valor aleatorio es: 0.11 [m]	El valor de pendiente es: 467927100.01
El valor deseado es: 35 [N]	El valor de r1 ahora es: 0.103 [mm]
El valor del paso es: 0.001 [m]	El valor de pendiente es: 348269920.293
El valor de pendiente es: 4525287737.117	El valor de r1 ahora es: 0.102 [mm]
El valor de r1 ahora es: 0.109 [mm]	El valor de pendiente es: 263487781.129
El valor de pendiente es: 2865270105.872	El valor de r1 ahora es: 0.1 [mm]
El valor de r1 ahora es: 0.108 [mm]	El valor de pendiente es: 157346826.845
El valor de pendiente es: 1885098847.684	El valor de r1 ahora es: 0.099 [mm]
El valor de r1 ahora es: 0.107 [mm]	El valor de pendiente es: 123856313.591
El valor de pendiente es: 1281022118.174	El valor de r1 ahora es: 0.098 [mm]
El valor de r1 ahora es: 0.106 [mm]	El valor de pendiente es: 98550896.783

28 / Septiembre / 2021

*****NO HUBO CLASE*****

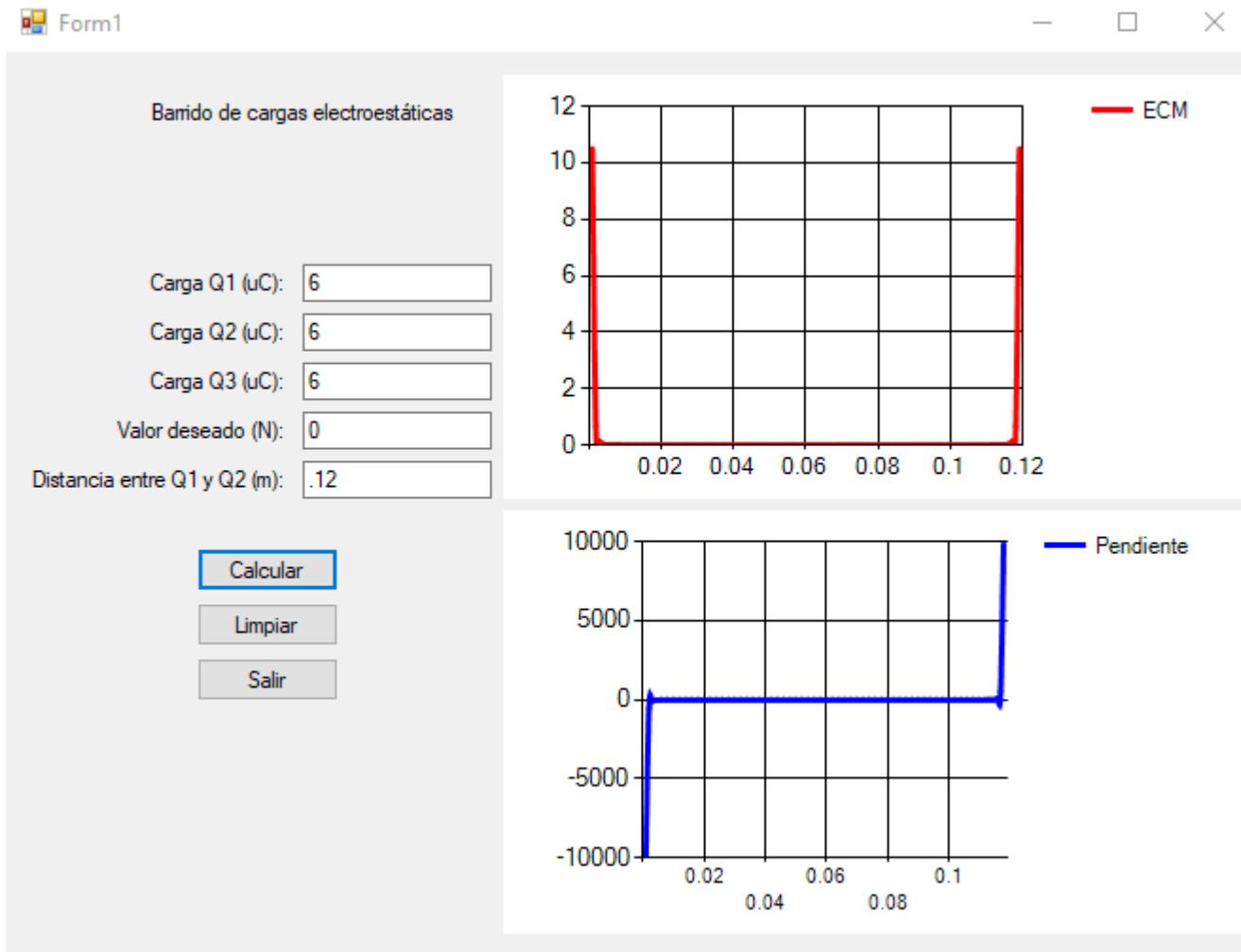
Durante la clase se revisó lo que se hizo de tarea y continuamos viendo acerca del descenso por gradiente, y vimos otras configuraciones o maneras de programar este método. Lo que nosotros denominamos “paso” durante las clases es lo que se le conoce como factor de aprendizaje, y puede ser tanto de valores muy grandes, como muy pequeños, según la curva del error que se tenga como las que vimos durante la clase.



Como se ve en la imagen, si tenemos un factor de aprendizaje demasiado alto, puede converger más rápido a lo que nosotros estamos buscando, el error mínimo, pero dado el comportamiento de la función en la que se trabaja, puede que nunca converja, y solo pase de un lado a otro en la curva.

El método que nosotros realizamos, al utilizar pocos datos, tenemos un buen factor de aprendizaje, ya que si puede converger, aunque usualmente se emplea un factor dinámico, es decir, si de un punto a otro la diferencia del error es grande, pude aumentar el factor de aprendizaje para converger más rápido, aunque puede darse el caso de que encuentre uno más grande, por lo que el factor se reduciría para la siguiente iteración, a manera de buscar la mejor manera de reducir el tiempo de convergencia.

Tarea 9: Investigar probabilidad y estadística, ver el video del Teorema de Bayes, programar descenso por gradiente y pendiente en forms



5 / Octubre / 2021

Probabilidad

La probabilidad nos permite determinar qué tan posible sea que un evento determinado suceda dentro de un conjunto de eventos posibles, y este valor está entre 0 y 1, el ejemplo más habitual suele ser el de lanzar un dado o una moneda.

Para el ejemplo del dado sería que tan probable caiga cierto número (por ejemplo, un 2), y dividirlo entre el número de eventos posibles (que serían 6 dado el número de caras del dado). Dado que el 2 solo aparece una vez, el evento determinado es 1.

$$P = \frac{\text{evento determinado}}{\text{eventos posibles}} = \frac{1}{6} = .1667$$

El valor de la probabilidad de obtener un 2 al lanzar un dado, es de 1/6, bien, .1667, que bien concuerda con la definición de que el valor puede ser entre 0 y 1.

Cuando hablamos de estadística nos referimos a la rama de la matemática que se encarga de analizar y estudiar datos, así como también buscar las explicaciones de los fenómenos que alteran los resultados.

La estadística es considerada una ciencia ya que estudia a una población de forma específica, a través de la recopilación de datos y con el objetivo de determinar un problema y buscar su solución.

Su ejecución se manifiesta de diversas maneras, tanto dentro de la física como de la ciencia, y además suele emplearse en los negocios y en el campo gubernamental.

Tipos de estadística

Se puede clasificar a la estadística en 4 tipos:

- **Descriptiva:** también conocida como deductiva, es aquella estadística que se encarga de mostrar el resultado de los datos estudiados de forma específica, sin generalizaciones.
- **Inferencial:** también conocida como inductiva, es aquella estadística que, a diferencia de la descriptiva, sí ofrece resultados junto con datos generales de investigación amplia.
- **Aplicada:** luego de investigar, estudiar y analizar con los métodos anteriores, se utiliza la estadística aplicada para proporcionar resultados específicos y generalizados sobre la investigación.
- **Matemática:** además de realizar los procesos de estadística deductiva o inferencial, la estadística matemática utilizará el álgebra y ciertos análisis más profundos para ofrecer un punto de vista enfocado y formal.

Su estudio, uso y aplicación es fundamental para la **toma de decisiones de diferentes ámbitos**.

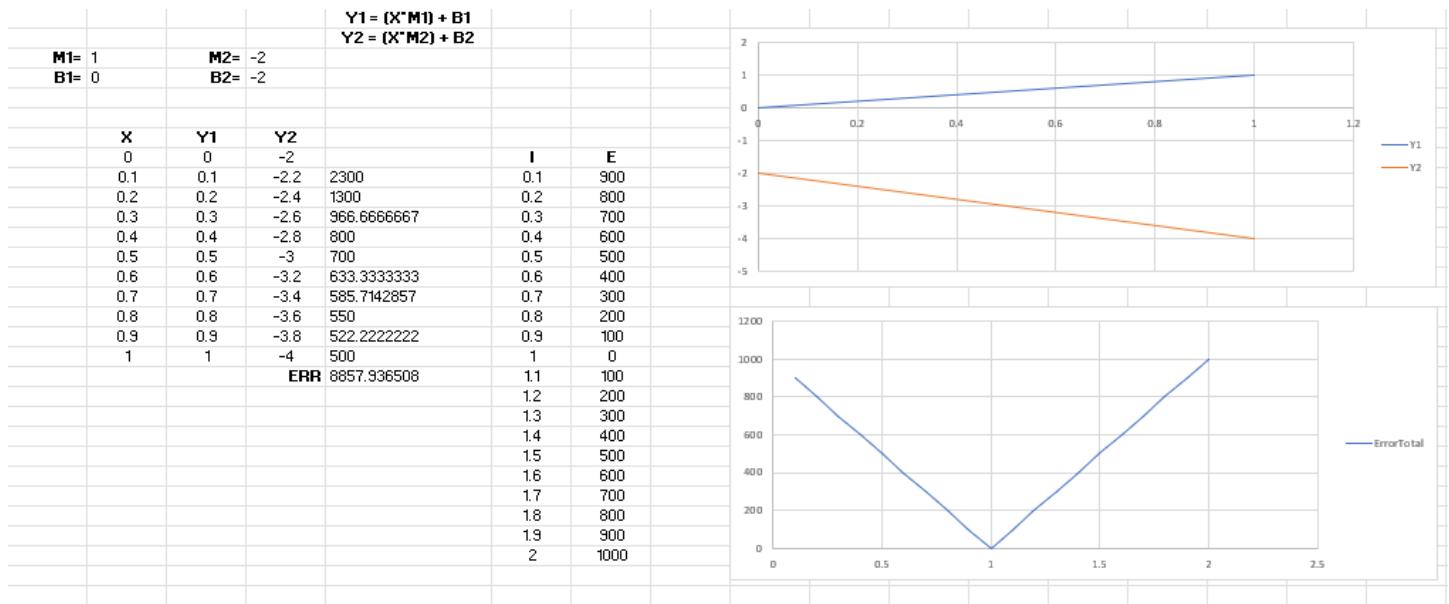
Teorema de Bayes

En términos más generales y menos matemáticos, el teorema de Bayes es de enorme relevancia puesto que vincula la probabilidad de **A** dado **B** con la probabilidad de **B** dado **A**. Es decir, por ejemplo, que sabiendo la probabilidad de tener un dolor de cabeza dado que se tiene gripe, se podría saber (si se tiene algún dato más), la probabilidad de tener gripe si se tiene un dolor de cabeza. Muestra este sencillo ejemplo la alta relevancia del teorema en cuestión para la ciencia en todas sus ramas, puesto que tiene vinculación íntima con la comprensión de la probabilidad de aspectos causales dados los efectos observados.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Realizamos un ejercicio en Excel para calcular los errores entre dos rectas, mediante el manejo de arreglos y la ecuación canónica de la recta, y después se trató de replicar el ejercicio mediante un programa de consola en C#.

Tal como habíamos visto durante el descenso por gradiente, este método no solo nos ayuda a llegar al error mínimo, sino que también nos permite durante cada iteración, a determinar los parámetros de la función con la que se trabaja, tal y como lo hicimos durante la clase, que dábamos valores diferentes para la pendiente y la ordenada de una segunda recta, tratando de llegar a que esta tenga el mínimo error o que llegue a los valores exactos de la recta ideal.



Referencias:

<https://es.khanacademy.org/math/probability/probability-geometry/probability-basics/a/probability-the-basics>

<https://academicos.fciencias.unam.mx/wp-content/uploads/sites/30/2015/04/Probabilidad.pdf>

<https://enciclopediaeconomica.com/estadistica/>

<https://youtu.be/D7KKIC0LOyw> - Cómo escapar de la trampa Bayesiana | Veritasium en español

<https://youtu.be/FoimdndwdgU> - Probabilidad condicional explicada de manera visual (Teorema de Bayes) | Khan Academy en español

Continuamos con el ejercicio de la clase anterior, ahora mostrando y almacenando los errores en arreglos para una futura aplicación en el mismo programa.

```

6  class ProgramPendiente
7  {
8      static void Main(string[] _)
9      {
10         //Declaracion de MATLAB para el uso de la libreria
11         Basics MATLAB = new Basics();
12         //Creamos el arreglo de elementos de X entre .1 y 1
13         double[] X = MATLAB.Linspace(.1, 1, .1);
14
15         //Declaramos los valores de pendientes y ordenadas al origen de cada una de las rectas
16         //Primera recta (ideal)
17         double m1 = 1;
18         double b1 = 0;
19
20         //Segunda recta
21         double m2 = .1;
22         double b2 = 0;
23
24         //Creamos el arreglo de la primera recta
25         double[] y1 = MATLAB.ProdDot(X, m1); //Producto de la pendiente
26         double[] Y1 = MATLAB.SumDot(y1, b1); //Sumando la ordenada
27
28         //Creamos el arreglo de la segunda recta
29         double[] y2 = MATLAB.ProdDot(X, m2); //Producto de la pendiente
30         double[] Y2 = MATLAB.SumDot(y2, b2); //Sumando la ordenada
31
32         //Creamos el arreglo de los errores porcentuales
33         double[] errores = MATLAB.ErrorPrc(y1, y2);
34
35         //Suma total de los errores porcentuales
36         double sumaErrores = MATLAB.SumList(errores);
37
38     }
39
40     //Suma total de los errores porcentuales
41     double sumaErrores = MATLAB.SumList(errores);
42     /*
43     Console.WriteLine("Arreglo de X: ");
44     Basics.PrintArray(X);
45     Console.WriteLine("Arreglo de Y1: ");
46     Basics.PrintArray(Y1);
47     Console.WriteLine("Arreglo de Y2: ");
48     Basics.PrintArray(Y2);
49     Console.WriteLine("Arreglo de errores: ");
50     Basics.PrintArray(errores);
51     Console.WriteLine("Errores totales: {0}\n", sumaErrores);
52     */
53
54     //Prueba Descenso
55     double punto = -2; //Valor para la pendiente
56     double paso = .1;
57     //Mostramos el arreglo buscado
58     Console.WriteLine("Arreglo ideal: ");
59     Basics.PrintArray(Y1);
60
61 }
```

12 / Octubre / 2021

Realizamos la aplicación del descenso por gradiente para el programa que se estuvo trabajando desde la clase pasada, siendo esta la última modificación que se realizó al programa.

Tarea 10: Implementar el método de descenso por gradiente a este programa

```

35     //Suma total de los errores porcentuales
36     double sumaErrores = MATLAB.SumList(errores);
37     /*
38     Console.WriteLine("Arreglo de X: ");
39     Basics.PrintArray(X);
40     Console.WriteLine("Arreglo de Y1: ");
41     Basics.PrintArray(Y1);
42     Console.WriteLine("Arreglo de Y2: ");
43     Basics.PrintArray(Y2);
44     Console.WriteLine("Arreglo de errores: ");
45     Basics.PrintArray(errores);
46     Console.WriteLine("Errores totales: {0}\n", sumaErrores);
47     */
48
49     //Prueba Descenso
50     double punto = -2; //Valor para la pendiente
51     double paso = .1;
52     //Mostramos el arreglo buscado
53     Console.WriteLine("Arreglo ideal: ");
54     Basics.PrintArray(Y1);
55
56     //Mostramos los valores ideales para el arreglo de datos que se busca
57     Console.WriteLine("Pendiente ideal: {0}\nOrdenada ideal: {1}\nOrdenada de recta de prueba: {2}\n", m1, b1, b2);
58     //Hacemos el descenso
59     MATLAB.Descenso(punto, b2, X, Y1, paso);
60
61 }
```

Con lo que una vez que corremos el programa, nos retorna la siguiente información:

```
C:\Users\jorge\Documents\Facultad\SEM2022-1\Temas Selectos de Programación I\Pruebas\Pendiente\bin\Debug\netcoreapp3.1\Prueba.exe

Arreglo ideal:
[0.1000, 0.2000, 0.3000, 0.4000, 0.5000, 0.6000, 0.7000, 0.8000, 0.9000, 1.0000]

Pendiente ideal: 1
Ordenada ideal: 0
Ordenada de recta de prueba: 0

Valor del error: 32.3785, valor del punto: -1.9
[-0.1900, -0.3800, -0.5700, -0.7600, -0.9500, -1.1400, -1.3300, -1.5200, -1.7100, -1.9000]

Valor del error: 30.184, valor del punto: -1.8
[-0.1800, -0.3600, -0.5400, -0.7200, -0.9000, -1.0800, -1.2600, -1.4400, -1.6200, -1.8000]

Valor del error: 28.0665, valor del punto: -1.7
[-0.1700, -0.3400, -0.5100, -0.6800, -0.8500, -1.0200, -1.1900, -1.3600, -1.5300, -1.7000]

Valor del error: 26.026, valor del punto: -1.6
[-0.1600, -0.3200, -0.4800, -0.6400, -0.8000, -0.9600, -1.1200, -1.2800, -1.4400, -1.6000]

Valor del error: 24.0625, valor del punto: -1.5
[-0.1500, -0.3000, -0.4500, -0.6000, -0.7500, -0.9000, -1.0500, -1.2000, -1.3500, -1.5000]

Valor del error: 22.176, valor del punto: -1.4
[-0.1400, -0.2800, -0.4200, -0.5600, -0.7000, -0.8400, -0.9800, -1.1200, -1.2600, -1.4000]

Valor del error: 20.3665, valor del punto: -1.3
[-0.1300, -0.2600, -0.3900, -0.5200, -0.6500, -0.7800, -0.9100, -1.0400, -1.1700, -1.3000]

Valor del error: 18.634, valor del punto: -1.2
[-0.1200, -0.2400, -0.3600, -0.4800, -0.6000, -0.7200, -0.8400, -0.9600, -1.0800, -1.2000]

C:\Users\jorge\Documents\Facultad\SEM2022-1\Temas Selectos de Programación I\Pruebas\Pendiente\bin\Debug\Prueba.exe

Valor del error: 3.1185, valor del punto: 0.1
[0.0100, 0.0200, 0.0300, 0.0400, 0.0500, 0.0600, 0.0700, 0.0800, 0.0900, 0.1000]

Valor del error: 2.464, valor del punto: 0.2
[0.0200, 0.0400, 0.0600, 0.0800, 0.1000, 0.1200, 0.1400, 0.1600, 0.1800, 0.2000]

Valor del error: 1.8865, valor del punto: 0.3
[0.0300, 0.0600, 0.0900, 0.1200, 0.1500, 0.1800, 0.2100, 0.2400, 0.2700, 0.3000]

Valor del error: 1.386, valor del punto: 0.4
[0.0400, 0.0800, 0.1200, 0.1600, 0.2000, 0.2400, 0.2800, 0.3200, 0.3600, 0.4000]

Valor del error: 0.9625, valor del punto: 0.5
[0.0500, 0.1000, 0.1500, 0.2000, 0.2500, 0.3000, 0.3500, 0.4000, 0.4500, 0.5000]

Valor del error: 0.616, valor del punto: 0.6
[0.0600, 0.1200, 0.1800, 0.2400, 0.3000, 0.3600, 0.4200, 0.4800, 0.5400, 0.6000]

Valor del error: 0.3465, valor del punto: 0.7
[0.0700, 0.1400, 0.2100, 0.2800, 0.3500, 0.4200, 0.4900, 0.5600, 0.6300, 0.7000]

Valor del error: 0.154, valor del punto: 0.8
[0.0800, 0.1600, 0.2400, 0.3200, 0.4000, 0.4800, 0.5600, 0.6400, 0.7200, 0.8000]

Valor del error: 0.0385, valor del punto: 0.9
[0.0900, 0.1800, 0.2700, 0.3600, 0.4500, 0.5400, 0.6300, 0.7200, 0.8100, 0.9000]

Valor del error: 0, valor del punto: 1
[0.1000, 0.2000, 0.3000, 0.4000, 0.5000, 0.6000, 0.7000, 0.8000, 0.9000, 1.0000]
```

Hasta que converge en el error mínimo, que, para el caso particular de este programa, se podía llegar sin ningún problema a obtener un error de 0.

Tarea 11: Realizar un programa para guardar los errores del ejercicio de la clase, pero con valores de -2 a 2 para m2 y b2, y que se guardaran en un archivo de Excel

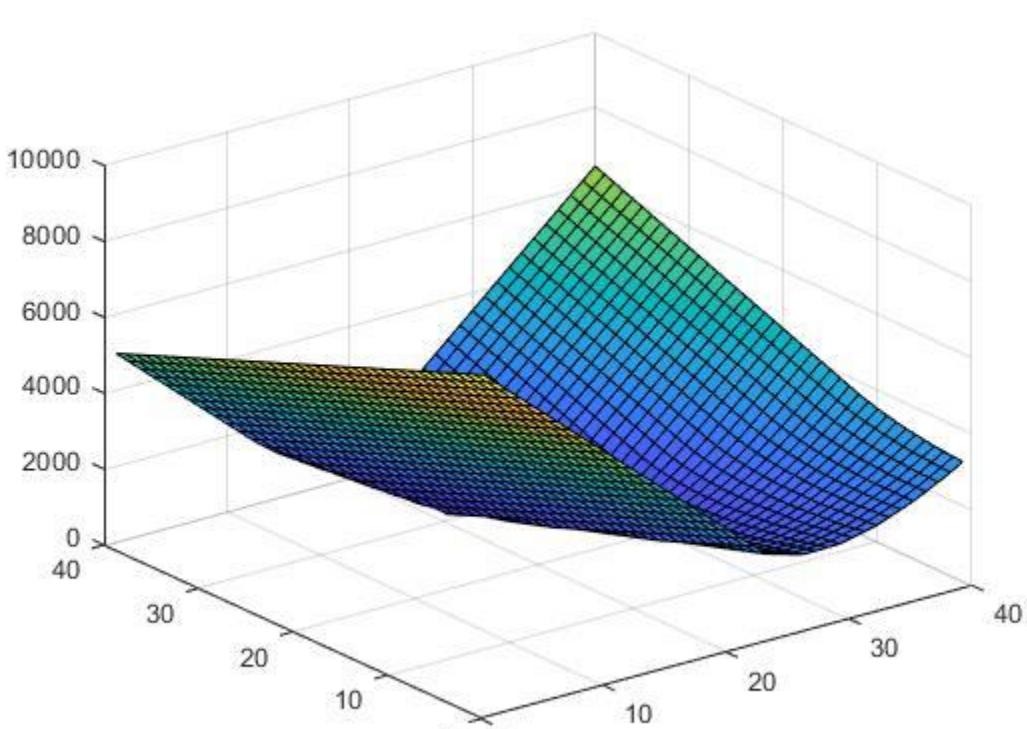
```

1  using MATLAB_Library;
2  using SpreadsheetLight;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace MatrizErrores
10 {
11     class MatrizErrores
12     {
13         static void Main(string[] _)
14         {
15             //Declaracion de MATLAB para el uso de la libreria
16             Basics MATLAB = new Basics();
17             //Hacemos el llamado de la API/libreria del MIT para el manejo de archivos en Excel
18             SLDocument sl = new SLDocument();
19             //Creamos el arreglo de elementos de X entre .1 y 1
20             double[] X = MATLAB.Linspace(.1, 1, .1);
21             //Declaramos los valores de pendientes y ordenadas al origen de cada una de las rectas
22             //Primera recta (ideal)
23             double m1 = 1;
24             double b1 = 0;
25             //Creamos el arreglo de la primera recta
26             double[] y1 = MATLAB.ProdDot(X, m1); //Producto de la pendiente
27             double[] Y1 = MATLAB.SumDot(y1, b1); //Sumando la ordenada
28             //Tenemos la recta ideal creada
29             //Creamos los indices de renglon y columna
30             int iRow = 2;
31             int jColumn = 2;
32             //Indicamos la distribucion de nuestra hoja de calculo
33             sl.SetCellValue("A1", "m2/b2");
34             //Creamos las variables de pendiente y ordenada
35             double m2;
36             double b2;
37             //Creamos las instancias para la creacion de la segunda recta
38             double[] y2 = new double[Y1.Length];
39             double[] Y2 = new double[Y1.Length];
40             double error;
41             //Creamos un ciclo para la creacion de la segunda recta, teniendo dos ciclos,
42             //uno para pendientes y otro para ordenadas
43             //Comenzamos a llenar la matriz
44             for (m2 = -2; m2 <= 2; m2 += .1)
45             {
46                 for(b2 = -2; b2 <= 2; b2 += .1)
47                 {
48                     //Creamos el arreglo de la segunda recta
49                     y2 = MATLAB.ProdDot(X, m2); //Producto de la pendiente
50                     Y2 = MATLAB.SumDot(y2, b2); //Sumando la ordenada
51                     error = MATLAB.ErrAccumPrc(Y1, Y2); //Se crea el error acumulado
52                     sl.SetCellValue(iRow, jColumn, error);
53                     jColumn++;
54                 }
55                 jColumn = 2;
56                 iRow++;
57             }
58             //Guardado y nombrado de nuestro Excel creado
59             sl.SaveAs("MatrizDeErrores.xlsx");
60             //Mensaje final del programa
61             Console.WriteLine("Programa terminado, archivo creado con exito");
62             Console.ReadLine();
63         }
64     }
}

```

Teniendo el siguiente archivo generado en Excel

m2/b2	8857.93	8565.04	8272.14	7979.25	7686.35	7393.46	7100.56	6807.65	6514.76	6221.86	5928.97	5636.07	5343.18	5050.28	4757.38	4464.49	4171.58	3878.69	3585.79	3292.9	3000	2707.1	2414.21	2121.31	2028.42
8557.93	8465.04	8172.14	7879.25	7586.35	7293.46	7000.56	6707.65	6414.76	6121.86	5828.97	5536.07	5243.18	4950.28	4657.38	4364.49	4071.58	3778.69	3485.79	3192.9	2900	2607.1	2314.21	2041.31	1948.42	
8657.93	8365.04	8072.14	7779.25	7486.35	7193.46	6900.56	6607.65	6314.76	6021.86	5728.97	5436.07	5143.18	4850.28	4557.38	4264.49	3971.58	3678.69	3385.79	3092.9	2800	2507.1	2214.21	1961.31	1868.42	
8557.93	8265.04	7972.14	7679.25	7386.35	7093.46	6800.56	6507.65	6214.76	5921.86	5628.97	5336.07	5043.18	4750.28	4457.38	4164.49	3871.58	3578.69	3285.79	2992.9	2700	2407.1	2114.21	1881.31	1788.42	
8457.93	8165.04	7872.14	7579.25	7286.35	6993.46	6700.56	6407.65	6114.76	5821.86	5528.97	5236.07	4943.18	4650.28	4357.38	4064.49	3771.58	3478.69	3185.79	2892.9	2600	2307.1	2014.21	1801.31	1708.42	
8357.93	8065.04	7772.14	7479.25	7186.35	6893.46	6600.56	6307.65	6014.76	5721.86	5428.97	5136.07	4843.18	4550.28	4257.38	3964.49	3671.58	3378.69	3085.79	2792.9	2500	2207.1	1914.21	1721.31	1628.42	
8257.93	7965.04	7672.14	7379.25	7086.35	6793.46	6500.56	6207.65	5914.76	5621.86	5328.97	5036.07	4743.18	4450.28	4157.38	3864.49	3571.58	3278.69	2985.79	2692.9	2400	2107.1	1814.21	1641.31	1548.42	
8157.93	7865.04	7572.14	7279.25	6986.35	6693.46	6400.56	6107.65	5814.76	5521.86	5228.97	4936.07	4643.18	4350.28	4057.38	3764.49	3471.58	3178.69	2885.79	2592.9	2300	2007.1	1714.21	1561.31	1468.42	
8057.93	7765.04	7472.14	7179.25	6886.35	6593.46	6300.56	6007.65	5714.76	5421.86	5128.97	4836.07	4543.18	4250.28	3957.38	3664.49	3371.58	3078.69	2785.79	2492.9	2200	1907.1	1614.21	1481.31	1388.42	
7957.93	7665.04	7372.14	7079.25	6786.35	6493.46	6200.56	5907.65	5614.76	5321.86	5028.97	4736.07	4443.18	4150.28	3857.38	3564.49	3271.58	2978.69	2685.79	2392.9	2100	1807.1	1514.21	1401.31	1308.42	
7857.93	7565.04	7272.14	6979.25	6686.35	6393.46	6100.56	5807.65	5514.76	5221.86	4928.97	4636.07	4343.18	4050.28	3757.38	3464.49	3171.58	2878.69	2585.79	2292.9	2000	1707.1	1414.21	1321.31	1228.42	
7757.93	7465.04	7172.14	6879.25	6586.35	6293.46	6000.56	5707.65	5414.76	5121.86	4828.97	4536.07	4243.18	3950.28	3657.38	3364.49	3071.58	2778.69	2485.79	2192.9	1900	1607.1	1334.21	1241.31	1168.42	
7657.93	7365.04	7072.14	6779.25	6486.35	6193.46	5900.56	5607.65	5314.76	5021.86	4728.97	4436.07	4143.18	3850.28	3557.38	3264.49	2971.58	2678.69	2385.79	2092.9	1800	1507.1	1254.21	1161.31	1108.42	
7557.93	7265.04	6972.14	6679.25	6386.35	6093.46	5800.56	5507.65	5214.76	4921.86	4628.97	4336.07	4043.18	3750.28	3457.38	3164.49	2871.58	2578.69	2285.79	1992.9	1700	1407.1	1174.21	1081.31	1048.42	
7457.93	7165.04	6872.14	6579.25	6286.35	5993.46	5700.56	5407.65	5114.76	4821.86	4528.97	4236.07	3943.18	3650.28	3357.38	3064.49	2771.58	2478.69	2185.79	1892.9	1600	1307.1	1094.21	1001.31	988.42	
7357.93	7065.04	6772.14	6479.25	6186.35	5893.46	5600.56	5307.65	5014.76	4721.86	4428.97	4136.07	3843.18	3550.28	3257.38	2964.49	2671.58	2378.69	2085.79	1792.9	1500	1207.1	1014.21	921.31	928.42	
7257.93	6965.04	6672.14	6379.25	6086.35	5793.46	5500.56	5207.65	4914.76	4621.86	4328.97	4036.07	3743.18	3450.28	3157.38	2864.49	2571.58	2278.69	1985.79	1692.9	1400	1107.1	934.21	861.31	868.42	
7157.93	6865.04	6572.14	6279.25	5986.35	5693.46	5400.56	5107.65	4814.76	4521.86	4228.97	3936.07	3643.18	3350.28	3057.38	2764.49	2471.58	2178.69	1885.79	1592.9	1300	1007.1	854.21	801.31	815.08	
7057.93	6765.04	6472.14	6179.25	5886.35	5593.46	5300.56	5007.65	4714.76	4421.86	4128.97	3836.07	3543.18	3250.28	2957.38	2664.49	2371.58	2078.69	1785.79	1492.9	1200	907.1	774.21	741.31	775.08	
6957.93	6665.04	6372.14	6079.25	5786.35	5493.46	5200.56	4907.65	4614.76	4321.86	4028.97	3736.07	3443.18	3150.28	2857.38	2564.49	2271.58	1978.69	1685.79	1392.9	1100	807.1	694.21	681.31	735.08	
6857.93	6565.04	6272.14	5979.25	5686.35	5393.46	5100.56	4807.65	4514.76	4221.86	3928.97	3636.07	3343.18	3050.28	2757.38	2464.49	2171.58	1878.69	1585.79	1292.9	1000	707.1	614.21	621.31	695.08	
6757.93	6465.04	6172.14	5879.25	5586.35	5293.46	5000.56	4707.65	4414.76	4121.86	3828.97	3536.07	3243.18	2950.28	2657.38	2364.49	2071.58	1778.69	1485.79	1192.9	900	627.1	554.21	581.31	675.08	
6657.93	6365.04	6072.14	5779.25	5486.35	5193.46	4900.56	4607.65	4314.76	4021.86	3728.97	3436.07	3143.18	2850.28	2557.38	2264.49	1971.58	1678.69	1385.79	1092.9	800	547.1	494.21	541.31	655.08	
6557.93	6265.04	5972.14	5679.25	5386.35	5093.46	4800.56	4507.65	4214.76	3921.86	3628.97	3336.07	3043.18	2750.28	2457.38	2164.49	1871.58	1578.69	1285.79	992.9	700	467.1	434.21	511.31	655.08	
6457.93	6165.04	5872.14	5579.25	5286.35	4993.46	4700.56	4407.65	4114.76	3821.86	3528.97	3236.07	2943.18	2650.28	2357.38	2064.49	1771.58	1478.69	1185.79	892.9	600	387.1	387.55	491.31	668.42	
6357.93	6065.04	5772.14	5479.25	5186.35	4893.46	4600.56	4307.65	4014.76	3721.86	3428.97	3136.07	2843.18	2550.28	2257.38	1964.49	1671.58	1378.69	1085.79	792.9	500	307.1	347.55	491.31	702.7	
6257.93	5965.04	5672.14	5379.25	5086.35	4793.46	4500.56	4207.65	3914.76	3621.86	3328.97	3036.07	2743.18	2450.28	2157.38	1864.49	1571.58	1278.69	985.79	692.9	400	247.1	327.55	517.03	771.58	



Práctica 1: Programación concurrente (Ejemplo de clase)

La entrega oficial de esta práctica se realizará la próxima clase, pero se realizó un ejemplo durante la clase (El proyecto de consola se encuentra en la carpeta “**Pruebas**”).

Programación concurrente

Hace referencia a las técnicas de programación que son utilizadas para expresar la concurrencia entre tareas y solución de los problemas de comunicación y sincronización entre procesos. La programación concurrente es la ejecución simultánea de múltiples tareas interactivamente. Estas tareas pueden ser un conjunto de procesos o hilos de ejecución creados por un único programa. Las tareas se pueden ejecutar en una sola CPU (multiprogramación), en varios procesadores, o en una red de computadores distribuidos.



Para realizar esta práctica también se consultaron videos además de la introducción que se nos proporcional en el manual de prácticas.

Referencias:

- http://ferestrepoca.github.io/paradigmas-de-programacion/progconcurrente/concurrente_teoria/index.html
- <https://youtu.be/Ch2HtAcCxFO> - ¿Qué diablos es Task en C# .Net? Programación asíncrona | hdeleon.net
- <https://youtu.be/jvfSC0HDUlo> - Programación concurrente en C# .Net utilizando Expresiones Lambda Asíncronas | hdeleon.net
- <https://youtu.be/-nNJF7g4JNk> - ¿Cómo asegurarse que una colección de Hilos ha terminado su proceso en C# .Net? | Task, WhenAll | hdeleon.net

19 / Octubre / 2021

Comenzamos revisando la tarea que teníamos pendiente de la clase anterior, que era terminar la modificación de los arreglos para las rectas generadas e implementar el método de descenso para llegar al arreglo con los valores ideales y nuestro error generado igual a 0.

También realizamos la entrega formal de la primera práctica, dado que en la clase anterior se nos dio una introducción de lo que es la programación concurrente, y se realizó un ejemplo sobre lo mismo.

Esta clase la dedicamos a lo que fue una introducción de lo que es una API, para poder realizar correctamente la práctica de la próxima semana, que no es más que un programa realizado por un tercero, para uso público. Investigamos un poco de ello durante la clase, y vimos que prácticamente todo software que usamos hoy en día hace uso de una o más APIs.

Una vez aclarado lo que es en general una API, el profesor nos proporcionó algunos enlaces a modo de referencia, para poder consultar más información, ya que una sola clase no bastaba para poder bien este tema, ya que existen muchísimas API, y no podemos centrarnos solo en una para realizar la práctica, ya que se trata buscar una en específico para satisfacer una función dentro de nuestro programa a desarrollar.

También comentamos un poco acerca de lo que son las bases de datos, a manera de introducción de la idea fundamental para realizar nuestra práctica correspondiente al mismo tema.

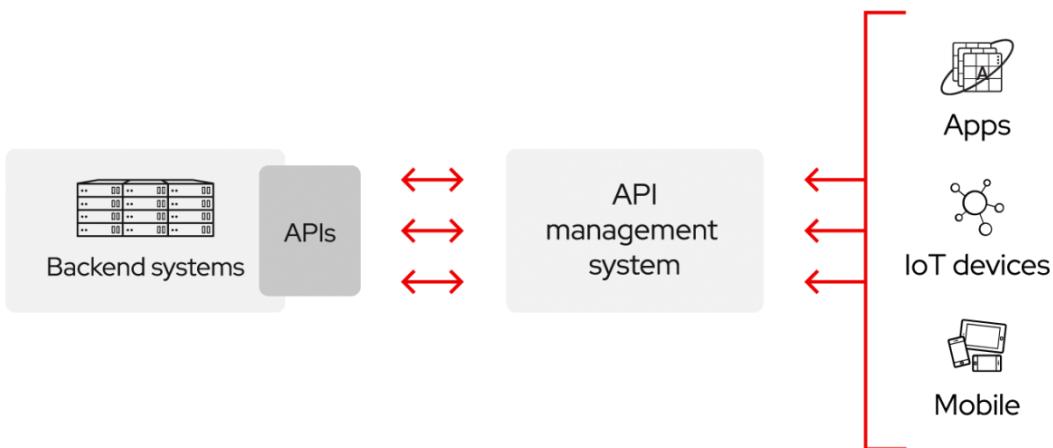
Resumen

¿Qué es una API?

Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. API significa **Interfaz de Programación de Aplicaciones**.

Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. Las API le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos (o de gestionar algunos ya existentes).

Las API son un medio simplificado para conectar su propia infraestructura a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos. Las API públicas representan un valor comercial único porque simplifican y amplían la forma en que se conecta con sus partners y, además, pueden rentabilizar sus datos (un ejemplo conocido es la API de Google Maps).



A medida que se han difundido las API, se desarrolló una especificación de protocolo para permitir la estandarización del intercambio de información; se llama **Protocolo de Acceso a Objetos Simples**, más conocido como **SOAP**. Las API diseñadas con SOAP usan XML para el formato de sus mensajes y reciben solicitudes a través de HTTP o SMTP. Con SOAP, es más fácil que las aplicaciones que funcionan en entornos distintos o están escritas en diferentes lenguajes comparten información.

Otra especificación es la **Transferencia de Estado Representacional (REST)**. Las API web que funcionan con las limitaciones de arquitectura REST se llaman API de **RESTful**. La diferencia entre REST y SOAP es básica: SOAP es un protocolo, mientras que REST es un estilo de arquitectura. Esto significa que no hay ningún estándar oficial para las API web de RESTful.

Tarea 12: Comenzar a ver los cursos que se nos proporcionaron durante la clase, para comenzar a comprender los tipos de API y elegir con cuál trabajar.

Tarea 13: Ver el curso de manejo de bases de datos para la práctica 3

Referencias:

<https://www.businessinsider.es/api-sirve-todo-necesitas-saber-861403>

<https://platzi.com/clases/api-rest/> - Curso de Api Rest | platzi

<https://youtu.be/6PFNShHE0zE> - Crear API en C# MVC .Net y consumirla con una solicitud POST desde Windows Forms | hdeleon.net

<https://youtu.be/TKE2Rr9alf0> - ¿Qué diablos es un web service Api Rest? | ejemplo práctico en C# .Net | hdeleon.net

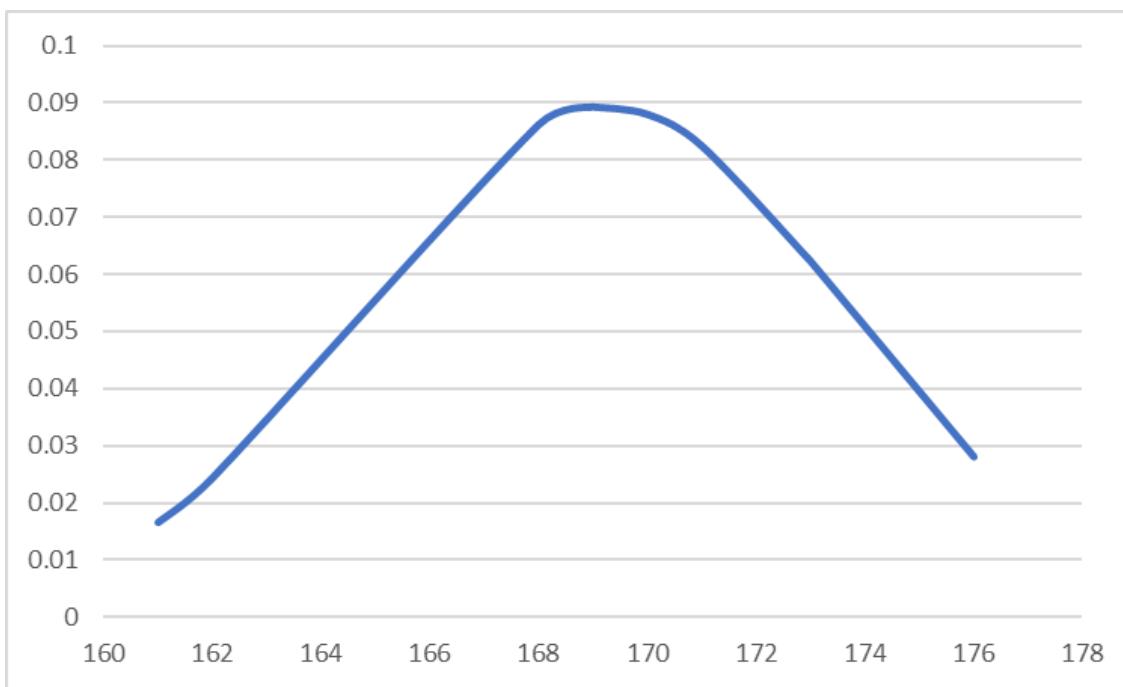
<https://youtube.com/playlist?list=PLWYKfSbdsJjZks0eLbsrCNc2hxTVs-i> – Curso de SQL Sever para novatos | hdeleon.net

26 / Octubre / 2021

Retomamos una parte de los conceptos de que es una API, en que consiste, como pueden emplearse y otras cosas más. Una vez que terminamos de comentar acerca de las API, comenzamos la clase con lo que es el tema de estadística descriptiva, de manera más concreta comenzamos con lo que todos ya saben por nuestras clases de probabilidad y estadística en la facultad. Los conceptos con los que comenzamos a trabajar fue el de promedio, varianza, desviación estándar y la función de densidad de probabilidad en una hoja de cálculo en Excel, esto con las estaturas del grupo.

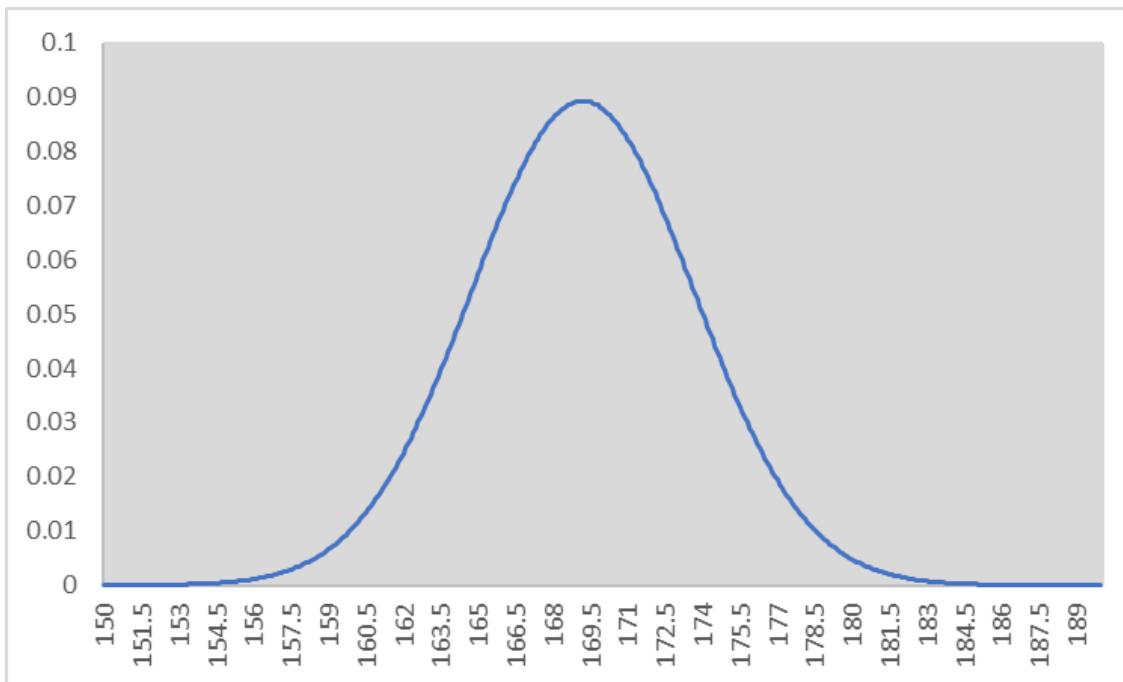
x	y	Media	Varianza	Desviación	mu-4sigma	mu+4sigma	Distribución
1	161	169.2	19.96	4.4676	151.3296	187.0704	0.01656928
2	162						0.0243695
3	168						0.08613297
4	169						0.08920734
5	169						0.08920734
6	170						0.08787654
7	171						0.08233537
8	173						0.06219235
9	173						0.06219235
10	176						0.02803957

También realizamos las graficas de la distribución que obtuvimos de los valores anteriores.



Para apreciar mejor la campana de distribución, aumentamos los datos e indicamos los límites de las graficas a como se especifica en la distribución, desde $\mu - 4\sigma$ hasta $\mu + 4\sigma$.

La distribución de nuestras estaturas con el rango ampliado y más datos:



Tarea 14: Realizar los métodos Mean, Var, Std y Normpdf de Matlab para nuestra biblioteca y probarlos en un proyecto de consola.

```

8  static void Main(string[] args)
9  {
10
11     //Prueba para obtener un promedio, desviación estandar, varianza y distribución gaussiana de un conjunto de datos
12     double[] datos_prom = {172, 170, 165, 170, 175, 170, 160, 178, 173, 160};
13     double promedio = Math.Round(Mean(datos_prom), 5);
14     double desviacion = Math.Round(Std(datos_prom), 5);
15     double varianza = Math.Round(Var(datos_prom), 5);
16     double[] distribucion = Normpdf(datos_prom, promedio, desviacion);
17
18     //Prueba para obtener el likelihood de un valor
19     double dato = datos_prom[new Random().Next(0, datos_prom.Length)];
20     double likelihood = Normpdf(dato, promedio, desviacion);
21     /*
22     Console.WriteLine("El promedio es: {0} cm\n", promedio);
23     Console.WriteLine("La varianza es: {0} cm\n", varianza);
24     Console.WriteLine("La desviación estandar es: {0} cm\n", desviacion);
25     Console.WriteLine("La distribución gaussiana que se obtiene para estos datos es: \n");
26     Basics.PrintArray(datos_prom);
27     Basics.PrintArray(distribucion);
28     Console.WriteLine("\nEl likelihood para el dato {0} cm, es: {1}", dato, likelihood);
29     */
30     double[] y = {3, 5, 9, 10, 20, 21, 24, 24, 27, 35};
31     double[] x = {100, 90, 80, 45, 50, 50, 60, 40, 25, 25};
32     var recta = Regression(x, y);
33     Console.WriteLine("Arreglo de x:");
34     Basics.PrintArray(x);
35     Console.WriteLine("Arreglo de y:");
36     Basics.PrintArray(y);
37     Console.WriteLine("La ecuación de la recta obtenida por regresión lineal es: y = {0} + {1}x", recta.Item1, recta.Item2);
38     Console.Read();
39 }
```

Con la siguiente salida de la consola:

```

C:\Users\jorge\Documents\Facultad\SEM2022-1\Temas Selectos de Programación I\Pruebas\PruebasEstadística\bin\Debug\netcoreapp3.1\PruebasEstadística
El promedio es: 169.3 cm
La varianza es: 32.21 cm
La desviación estandar es: 5.6754 cm
La distribución gaussiana que se obtiene para estos datos es:
[172.0000, 170.0000, 165.0000, 170.0000, 175.0000, 170.0000, 160.0000, 178.0000, 173.0000, 160.0000]
[0.0628, 0.0698, 0.0528, 0.0698, 0.0425, 0.0698, 0.0184, 0.0217, 0.0568, 0.0184]
```

28 / Octubre / 2021

Práctica 2: Implementación de una Interfaz de Programación de Aplicaciones (API)

Para realizar esta práctica, se consultó sobre la API “SpreadsheetLight”.

SpreadsheetLight es básicamente una librería de terceros (**en este caso, publicada por licencia del MIT**), que nos permite trabajar con hojas de cálculo Open XML de código abierto para .NET Framework escrita en C#.

Esta librería tiene gran variedad de métodos para trabajar con las hojas de cálculo, desde crearlas, abrirlas, agregar valores por renglón-columna, métodos de gráficos, métodos para estilos y personalización de fuente, entre muchos otros.

Nosotros para esta práctica empleamos métodos como el de insertar valores por índices en renglón-columna, ingresar información en una celda específica (por ejemplo: C4, “Fecha”) y métodos para

crear una clase de gráfico en específico, así como su estilo, también que nos guardara todo ello en un archivo con un nombre determinado.

Algo que nos fue de mucha ayuda fue su documentación, ya que además de dar una explicación del funcionamiento de cada uno de los métodos, nos brinda un código de muestra, en el cuál, nos señala el cómo se están empleando los métodos y el resultado que se obtendría en la hoja de cálculo.

Referencias:

<https://spreadsheetlight.com/>

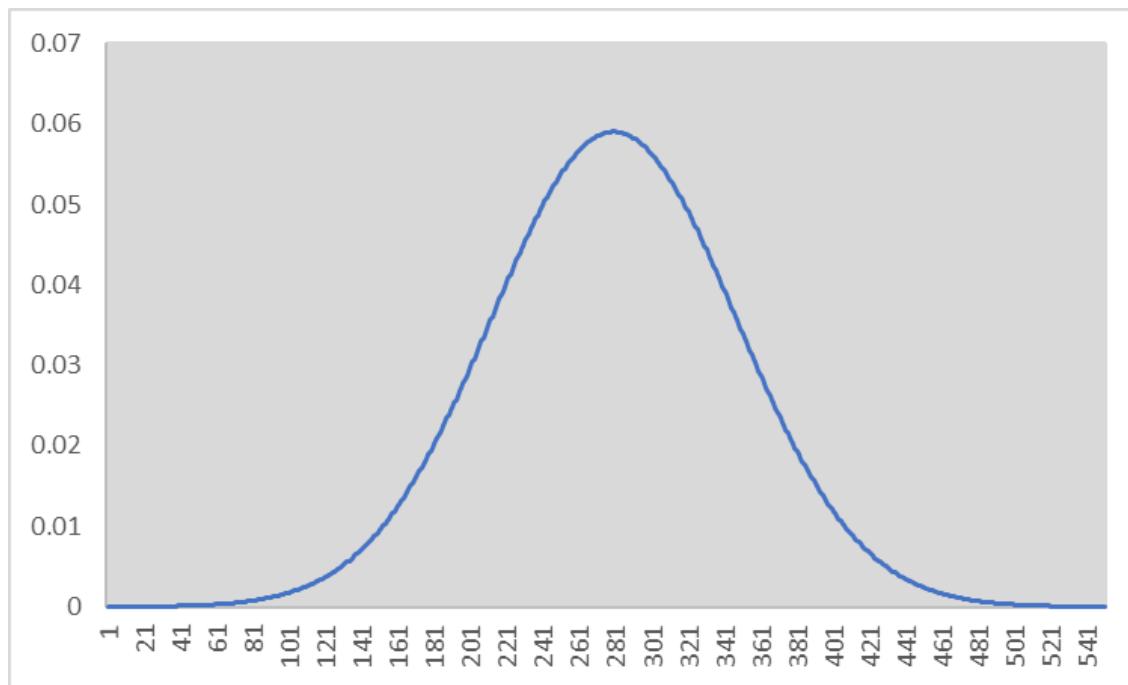
2 / Noviembre / 2021

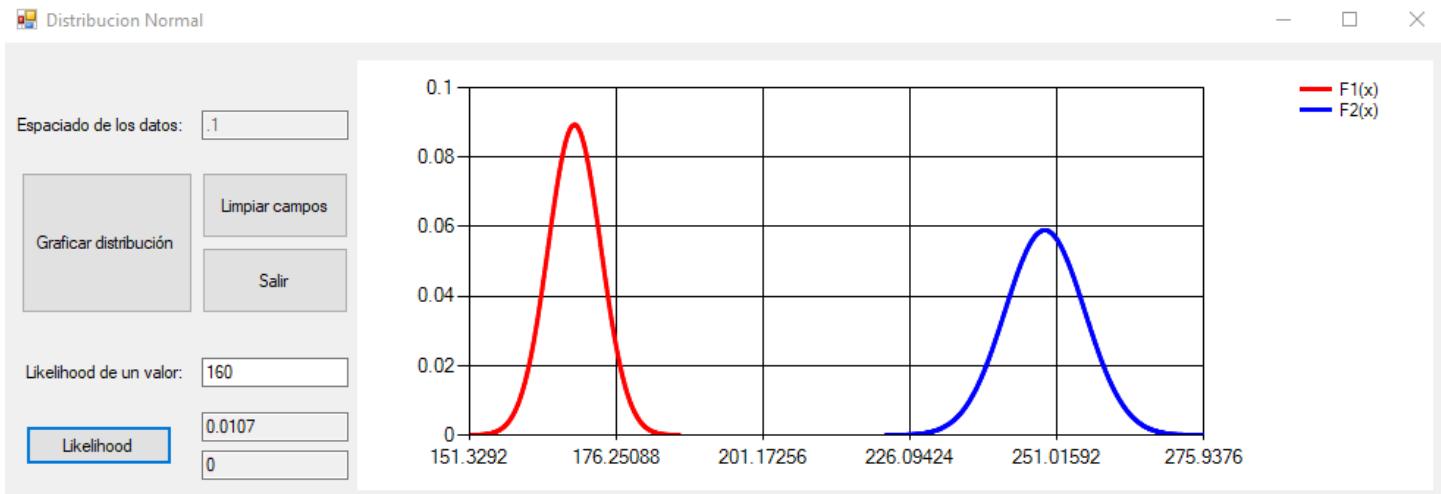
*****NO HUBO CLASE*****

4 / Noviembre / 2021

Continuamos con la activada de Excel de la clase de los principios de estadística, ya que podemos interpretar nuestros datos para esa distribución, agregamos una distribución más, también de estaturas, solo que ahora, serían de elefantes, aunque sea el mismo parámetro, los objetos son totalmente distintos. También vimos el concepto de Likelihood, que nos es más que la misma distribución de probabilidad, pero solamente para un dato en específico.

Esta es la gráfica que obtuvimos para nuestros elefantes en Excel.





Como se puede ver, tenemos las dos gráficas juntas, pero separadas por una brecha, si se desea saber a que objeto pertenece un dato, se calcula el Likelihood, que como podemos ver, para un 160, nos indica que pertenece a la curva roja (personas), y no a la azul (elefantes).

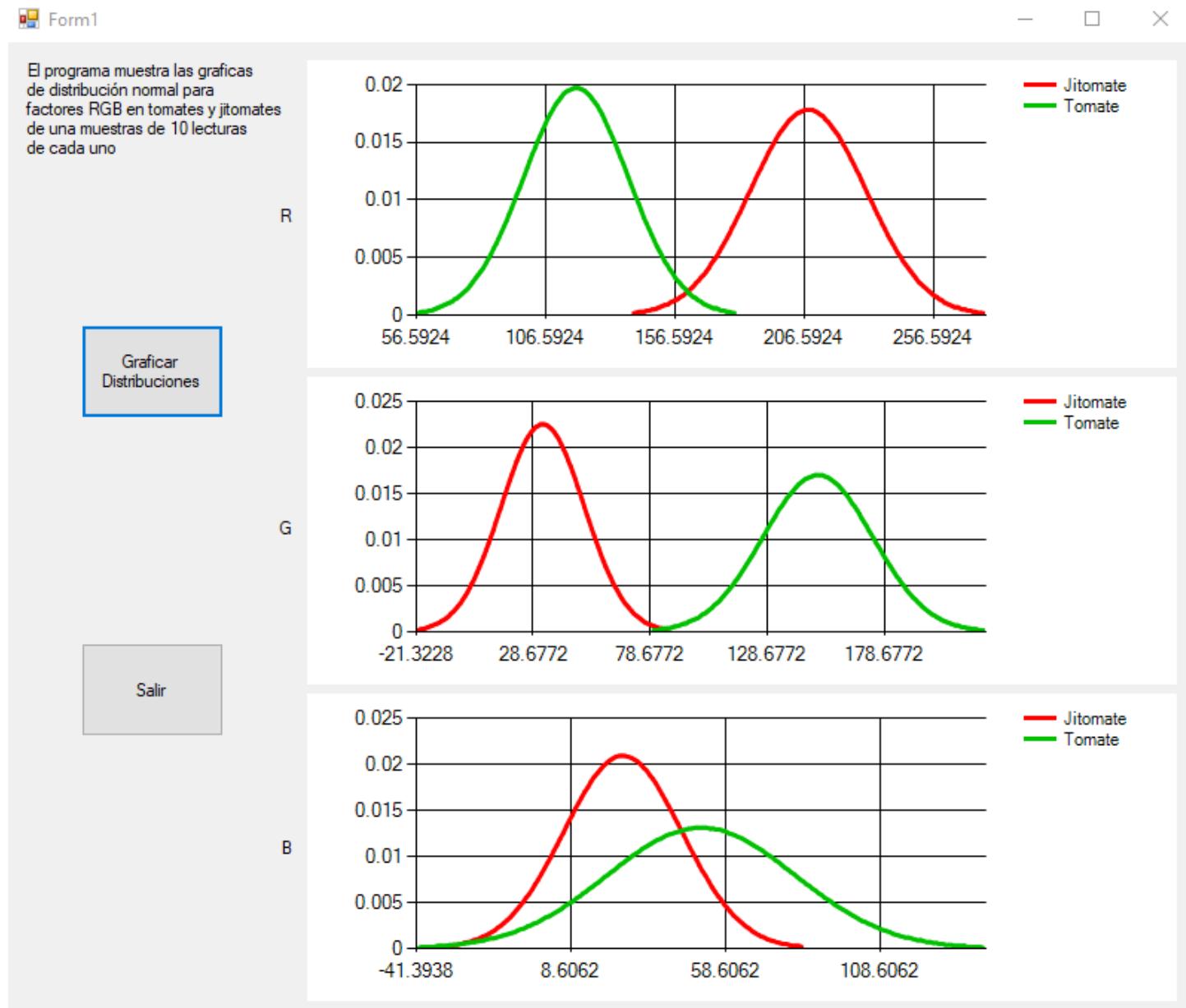
9 / Noviembre / 2021

Continuando con la temática de la clase pasada, ahora mediante Matlab, el profesor descargo algunas imágenes con las que trabajamos, siendo fotos de tomates y jitomates, para obtener sus valores RGB y guardarlos en un Excel. Esto para comprender mejor el como interpretar los datos de las distribuciones cuando estas se encuentran traslapadas un poco, ya que, si están encimadas en mas de la mitad de cada una de ellas, determinamos que es un “atributo basura”.

Estos son los datos que rescatamos durante la clase.

jitomates	R	G	B		tomates	R	G	B	
193	4	8			137	181	60		
195	47	45			147	168	75		
189	3	6			114	144	46		
195	46	42			129	127	48		
210	26	28			127	176	87		
192	47	42			140	177	108		
188	51	58			109	142	29		
254	47	5			108	151	9		
241	25	4			91	128	15		
224	52	16			82	109	30		
254	52	58	Max		147	181	108	Max	
188	3	4	Min		82	109	9	Min	

Tarea 16: Realizar las distribuciones de tomates y jitomates en Windows Forms



Práctica 3: Base de Datos (Manejo y Conexión)

Solo se realizó la entrega de la práctica, en el caso del equipo al que pertenezco, se volverá a entregar la siguiente clase debido a que no se realizó correctamente.

De la página de Microsoft para el **namespace System.Data.SqlClient** revisamos muy bien lo que corresponde a las clases de **SqlConnection**, **ConnectionStringBuilder**, **SqlDataAdapter** y **SqlDataReader**. A partir de esas clases es lo que rige completamente la manipulación de nuestra base de datos, ya que durante mientras se realizaban los métodos correspondientes a cada uno de los comando empleados, por ejemplo, cuando creábamos nuestra consulta y nuestro comando, olvidábamos el terminar tanto la consulta como el comando uno vez que terminaran su función, y gracias al manejo de excepciones, nos percatamos de ellos por los mensajes de error programados en caso de que se presentara dicha excepción.

Comparado con la primera entrega que se realizó, el consultar la documentación directa de la página de Windows y el revisar bien el cómo se manejaba una consulta, logramos realizar un mejor trabajo.

Para realizar la práctica de nueva cuenta, empleamos diversas instrucciones de SQL Server, siendo los comandos de “INSERT INTO” (insertar en), “CREATE TABLE” (crear tabla), “DROP TABLE” (eliminar tabla) y “TRUNCATE TABLE” (limpiar tabla) de la variedad que existen, tanto para la base en sí misma, como para las tablas.

Referencias:

https://www.youtube.com/watch?v=Zmrt_yS4ggE&list=PLWYKfSbdsjjZks0eLbrsrCNc2hxTVs-i

<https://www.w3schools.com/sql/default.asp>

<https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient?view=dotnet-plat-ext-5.0>

16 / Noviembre / 2021

Introducción al Machine Learning

El aprendizaje automático (machine learning) nos ayuda en 3 aspectos importantes en nuestro desempeño como ingenieros que manejan software. En primera instancia tendremos una herramienta que nos ayude a reducir los tiempos de programación, como segundo lugar, permite la personalización basada en las necesidades de un grupo específico y, por último, nos permite realizar problemas que no podríamos realizar a mano, como reconocer la voz y el rostro de una persona.

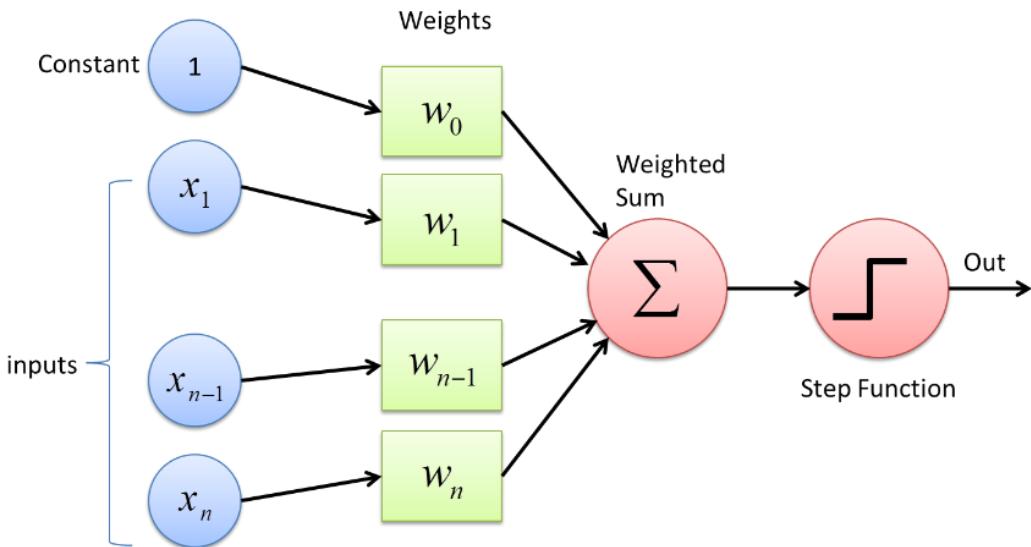
Un claro ejemplo que nos dan en el curso de Machine Learning de Google es cuando se desarrolla un programa de correcciones ortográficas, si los ingenieros realmente realizaran un corrector ortográfico por cada idioma existe desde cero, serían años y años de trabajo, mientras que si se emplea machine learning incluyendo ejemplos y reglas gramaticales, pueden tener programas más confiables para los idiomas en una fracción menor de tiempo.

Introducción a las Redes Neuronales

La Neurona

Es la unidad básica de procesamiento de una red neuronal, esta neurona tiene conexiones de entrada, por donde reciben estímulos externos (valores de entrada), con esos valores, la neurona realiza un cálculo de manera interna (función matemática), produciendo así una salida. La función matemática

que se realiza no es más que una suma ponderada de cada uno de los valores de entrada, en la que a cada entrada tiene un peso (weight), con el cual, mediante su ajuste, se regula de forma positiva o negativa la salida de la neurona.



De manera más abstracta, la función matemática que realiza una neurona no es más que una regresión lineal multivariable, en donde la función matemática estaría dada de la siguiente manera:

$$y = b + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

El término b (w_0), se le conoce como **sesgo o bias** (este parámetro siempre se multiplica por 1).

El término w es lo que se conoce como peso.

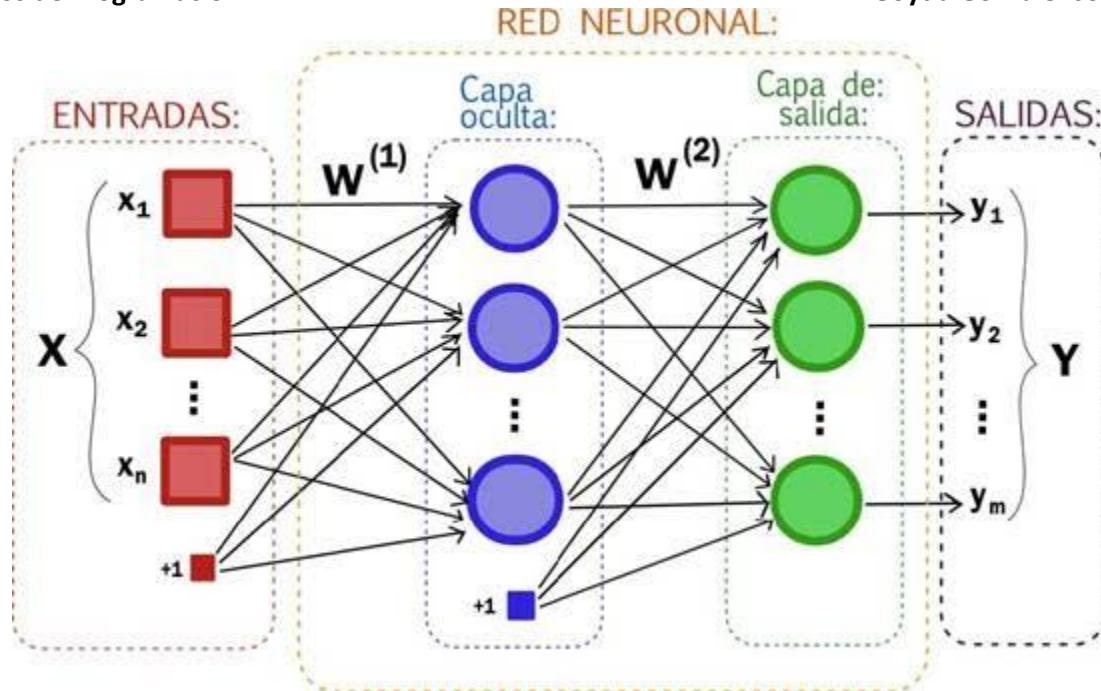
El término x corresponde a los valores de entrada.

La Red

La forma en la que se organizan las neuronas en una red, si se acomodan a manera de columna y se agrupan, se le denomina capa.

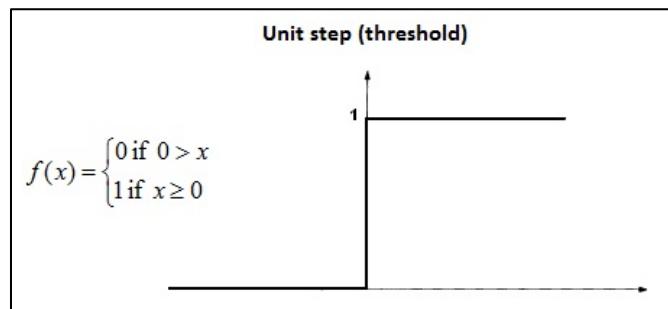
Los valores de entrada de una capa corresponden a la salida de las neuronas de la capa anterior, y las salidas que produce, son los valores de entrada para las neuronas de la capa siguiente. La primera capa de una red neuronal se le llama capa de entrada, a la última capa, capa de salida, y aquellas capas que se encuentran entre las dos anteriores, se les llaman capas ocultas.

El que la salida de una neurona sea la entrada de una consecuente, se le llama **conocimiento jerarquizado**. De igual manera, entre más capas se vayan añadiendo a nuestra red, el conocimiento que se elabora durante el proceso se va haciendo mucho más complejo, esto es lo que se conoce como aprendizaje profundo (**Deep Learning**).



La función de activación

Esta función nos permite deformar la salida de una neurona, a manera de que, en una red neuronal de capas muy extensas, no colapse, es la no linealidad.



Existen diferentes funciones de activación, la más básica de ellas, pero menos favorable, es la función escalonada:

$$f(x) = \begin{cases} 0 & \text{para } x < 0 \\ 1 & \text{para } x \geq 0 \end{cases}$$

La siguiente función sería la función sigmoide:

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

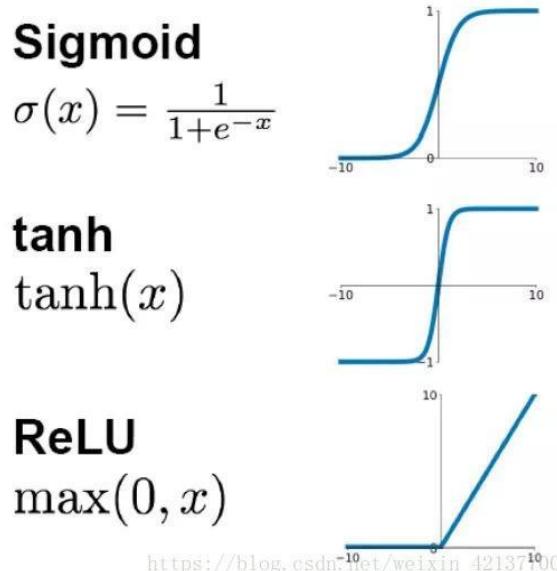
donde los valores muy grandes, se saturan en 1, mientras que los valores muy pequeños en 0, esta función nos es muy útil para representar probabilidades.

Otra función, muy similar a la sigmoide, pero su rango de valores es de -1 a 1, es la tangente hiperbólica:

$$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

Por último, otra función de activación muy utilizada es la Unidad Rectificada Lineal (RELU), que se comporta como una función lineal, cuando es positiva y como una constante a 0 cuando es negativo

$$f(x) = \begin{cases} 0 & \text{para } x < 0 \\ x & \text{para } x \geq 0 \end{cases}$$



Referencias:

<https://developers.google.com/machine-learning/crash-course/ml-intro>

<https://youtu.be/MR1v2lwFTPg> - ¿Qué es una red neuronal? Parte 1: La Neurona | DotCSV

<https://youtu.be/uwbHOpp9xkc> - ¿Qué es una red neuronal? Parte 2: La Red | DotCSV

Tarea 17: Programar el método de regresión lineal

```

8     static void Main(string[] args)
9     {
10
11         //Prueba para obtener un promedio, desviacion estandar, varianza y distribución gaussiana de un conjunto de datos
12         double[] datos_prom = {172, 170, 165, 170, 175, 170, 160, 178, 173, 160};
13         double promedio = Math.Round(Mean(datos_prom), 5);
14         double desviacion = Math.Round(Std(datos_prom), 5);
15         double varianza = Math.Round(Var(datos_prom), 5);
16         double[] distribucion = Normpdf(datos_prom, promedio, desviacion);
17
18         //Prueba para obtener el likelihood de un valor
19         double dato = datos_prom[new Random().Next(0, datos_prom.Length)];
20         double likelihood = Normpdf(dato, promedio, desviacion);
21         /*
22         Console.WriteLine("El promedio es: {0} cm\n", promedio);
23         Console.WriteLine("La varianza es: {0} cm\n", varianza);
24         Console.WriteLine("La desviación estandar es: {0} cm\n", desviacion);
25         Console.WriteLine("La distribución gaussiana que se obtiene para estos datos es: \n");
26         Basics.PrintArray(datos_prom);
27         Basics.PrintArray(distribucion);
28         Console.WriteLine("\nEl likelihood para el dato {0} cm, es: {1}", dato, likelihood);
29         */
30         double[] y = { 3, 5, 9, 10, 20, 21, 24, 24, 27, 35 };
31         double[] x = { 100, 90, 80, 45, 50, 50, 60, 40, 25, 25 };
32         var recta = Regression(x, y);
33         Console.WriteLine("Arreglo de x:");
34         Basics.PrintArray(x);
35         Console.WriteLine("Arreglo de y:");
36         Basics.PrintArray(y);
37         Console.WriteLine("La ecuación de la recta obtenida por regresión lineal es: y = {0} + {1}x", recta.Item1, recta.Item2);
38         Console.Read();
39     }

```

Que nos regresa lo siguiente para la regresión de los datos "X" y "Y".

```
C:\Users\jorge\Documents\Facultad\SEM2022-1\Temas Selectos de Programación I\Pruebas\PruebasEstadística\bin\Debug\netcoreapp3.1\PruebasEsta...
Arreglo de x:
[100.0000, 90.0000, 80.0000, 45.0000, 50.0000, 50.0000, 60.0000, 40.0000, 25.0000, 25.0000]

Arreglo de y:
[3.0000, 5.0000, 9.0000, 10.0000, 20.0000, 21.0000, 24.0000, 24.0000, 27.0000, 35.0000]

La ecuación de la recta obtenida por regresión lineal es: y = 37.7489 + -0.3531x
```

18 / Noviembre / 2021

Regresión lineal

La regresión lineal es una técnica de modelado estadístico que se emplea para describir una variable de respuesta continua como una función de una o varias variables predictoras. Puede ayudar a comprender y predecir el comportamiento de sistemas complejos o a analizar datos experimentales, financieros y biológicos.

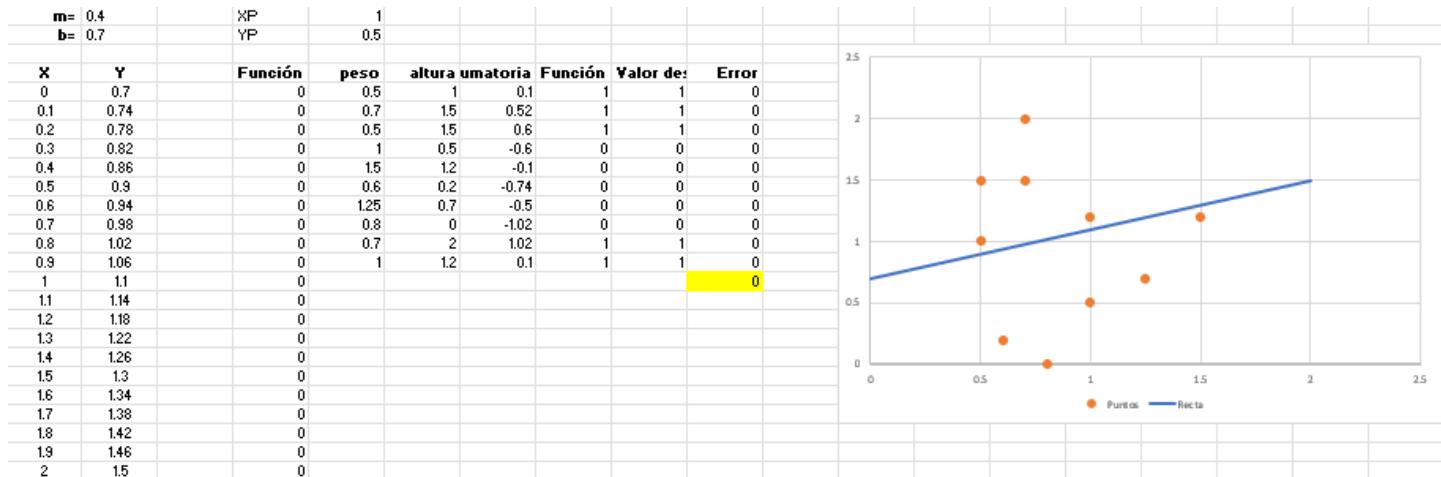
Las técnicas de regresión lineal permiten crear un modelo lineal. Este modelo describe la relación entre una variable dependiente y (también conocida como la respuesta) como una función de una o varias variables independientes X_i (denominadas predictores). La ecuación general correspondiente a un modelo de regresión lineal es:

$$Y = \beta_0 + \sum \beta_i X_i + \epsilon$$

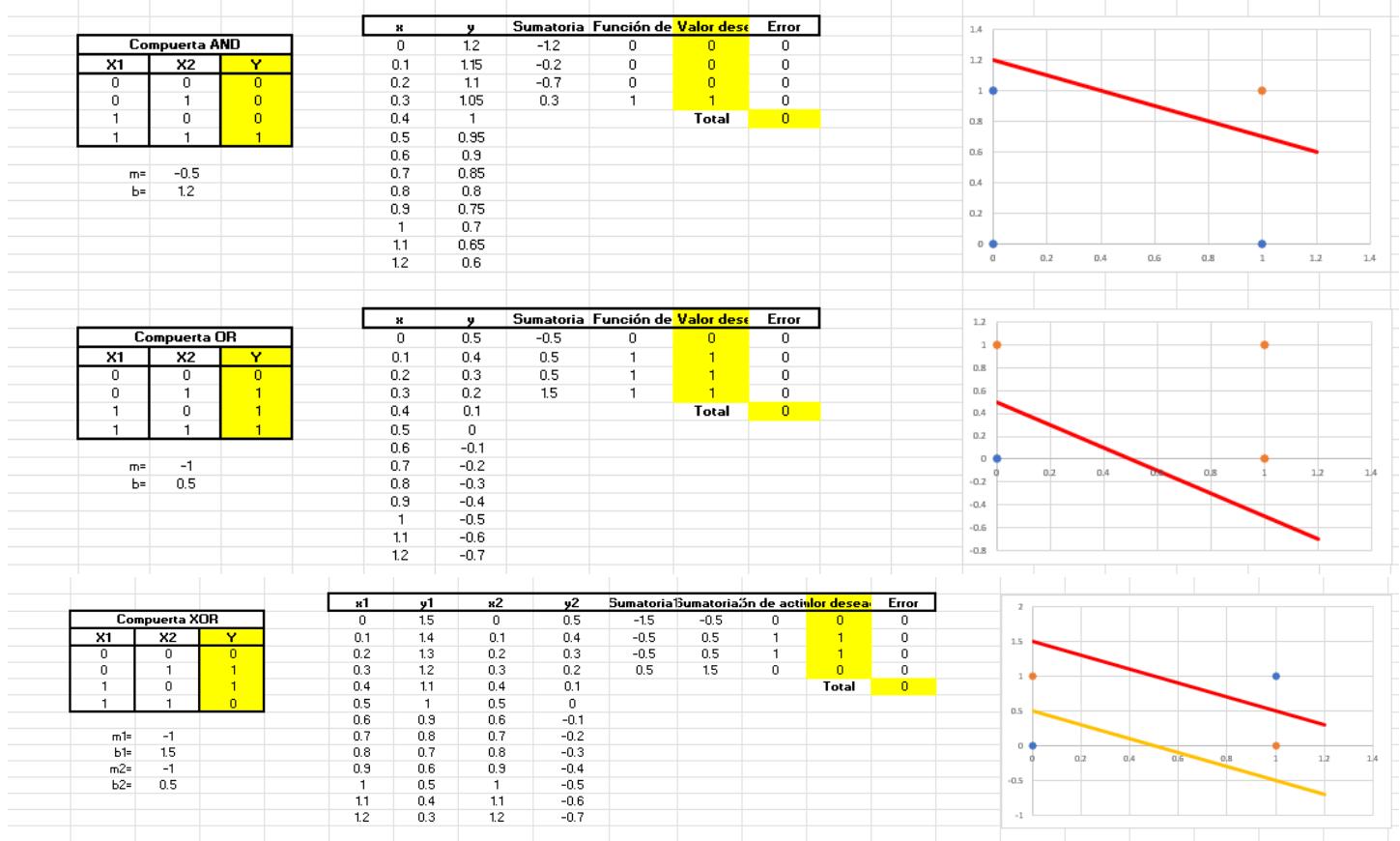
donde β representa las estimaciones de parámetros lineales que se deben calcular y ϵ representa los términos de error.

Existen dos tipos de regresión, la simple y la múltiple. La regresión simple es aquella en la que solo se emplea una sola variable de predicción (ecuación canónica de la recta), mientras que la múltiple es de dos o más predictores (permitiendo obtener superficies, dimensiones, etc.).

Una vez que presentamos nuestros métodos de regresión lineal, retomamos los videos de DotCSV, siendo el de la explicación de la neurona y la red, además de profundizar un poco más en su composición. Para comprender como se entrena un perceptrón, vimos un video durante la clase, y al mismo tiempo, recreamos en una hoja de cálculo de Excel básica.



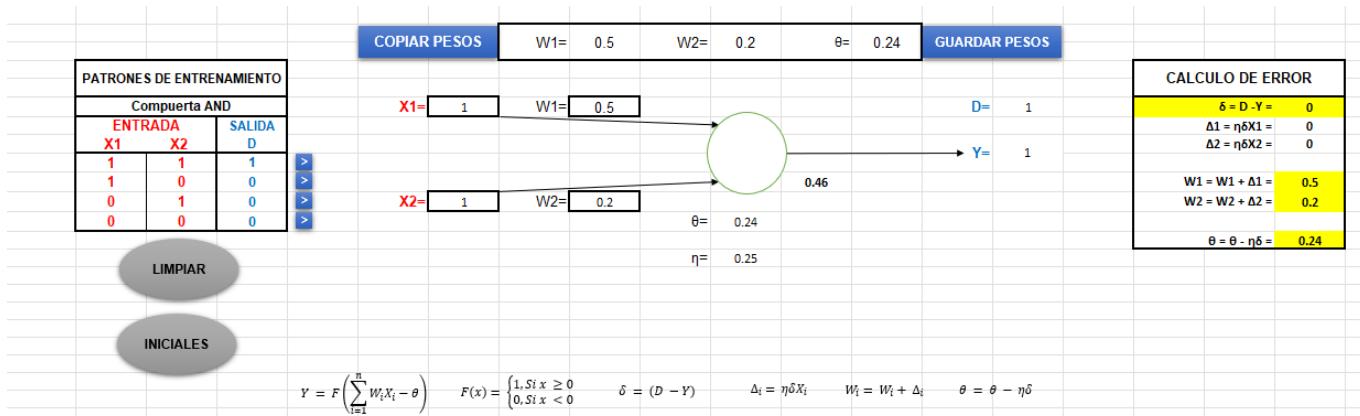
Referencias:

https://es.wikipedia.org/wiki/Regresión_lineal<https://la.mathworks.com/discovery/linear-regression.html><https://youtu.be/fNeXC8d5En8> – Video con datos de ejemplo**Tarea 18:** Realizar las compuertas AND, OR y XOR en Excel mediante el perceptrón**23 / Noviembre / 2021****El Perceptrón**

Como vimos en las clases anteriores, el perceptrón es la parte básica de la neurona, y es en donde se lleva a cabo una operación, que en el caso particular que hemos visto, es una suma ponderada con la característica de que es muy similar a la ecuación de la recta.

El video que vimos durante la clase corresponde a un perceptrón mediante macros, lo que se hizo durante la clase fue analizar su funcionamiento paso a paso, para tratar de recrearlo para la siguiente clase. En cierta forma este entrenamiento ya nos era familiar, puesto que se trataba de una compuerta AND.

Tarea 19: Realizar la macro del perceptrón visto en clase (video de YouTube)



Referencias:

<https://youtu.be/Woe8fXttC6E> - Entrenamiento de Perceptrón

25 / Noviembre / 2021

Práctica 4: Interfaz Software – Hardware

Para esta práctica se investigó acerca de lo que es la comunicación serial y se vieron algunos videos de ejemplo para comprender su manejo.

La comunicación serie o comunicación secuencial, en telecomunicaciones e informática, es el proceso de envío de datos de un bit a la vez, de forma secuencial, sobre un canal de comunicación o un bus.

En cambio, en la “comunicación en paralelo” todos los bits de cada símbolo se envían al mismo tiempo, y por ello debe haber al menos tantas líneas de comunicación como bits tenga la información a transmitir.

La ventaja de la comunicación serie es que necesita un número más pequeño de líneas de transmisión que una comunicación paralela que transmita la misma información. Esta última necesita tantas líneas de transmisión como la cantidad de bits que componen la información, mientras que la primera se puede llevar a cabo con una sola línea de transmisión. Por otra parte, surgen una serie de problemas en la transmisión de un gran número de bits en paralelo, como los problemas de interferencia o desincronización.

A la misma frecuencia de transmisión, la comunicación paralela tiene un mayor rendimiento. La comunicación serie tiene que compensar esta debilidad con una frecuencia más alta.

Para esta práctica simplemente se mostrará en el reporte lo que corresponde a la programación del proyecto, ya que es necesario el uso de esta comunicación.

Referencias:

https://es.wikipedia.org/wiki/Comunicaci%C3%B3n_serie<https://youtu.be/G5n20Wfm19I> - Enviar Datos y Leer PUERTO SERIAL con ARDUINOhttps://youtu.be/rJNC_gJnbPc - ¿Cómo recibir información desde Arduino en C# .Net? | Windows Forms, WPF<https://youtu.be/mlw3APOUt8U> - Ejemplo con HC-SR04 Sensor Ultrasónico<https://youtu.be/CvO5adh12a8> - Ejemplo con sensor de color TCS230

Retomamos el tema de redes neuronales recapitulando los videos que consultamos de tarea anteriormente, para comenzar a ver lo que sería la librería Encog de Jeff Heaton, para poder realizar redes neuronales en C# (paquete NuGet).

Nos introducimos un poco en el tema de la librería revisando un ejemplo de red neuronal para solucionar una compuerta XOR, como la que realizamos de tarea para esta clase, viendo código de muestra para consola, y comprendiendo los métodos que se lograban ver en el ejemplo, sin ver de pies a cabeza la librería.

Una vez que terminamos de leer el código de muestra, nos percatamos de que los pasos que se realizaron en la macro son esencialmente los mismos que se llevan a cabo en el código muestra.

Tarea 20: Copiar y pegar el código de ejemplo de Jeff Heaton para una compuerta XOR y replicar el perceptrón de macros en Forms.

```
1  using Encog;
2  using Encog.Engine.Network.Activation;
3  using Encog.ML.Data;
4  using Encog.ML.Data.Basic;
5  using Encog.ML.Train;
6  using Encog.Neural.Networks;
7  using Encog.Neural.Networks.Layers;
8  using Encog.Neural.Networks.Training.Propagation.Resilient;
9  using System;
10
11 namespace Perceptron_NET_
12 {
13     class PerceptronXOR
14     {
15         /// <summary>
16         /// Entradas de para funcion XOR.
17         /// </summary>
18         public static double[][] XORInput =
19         {
20             new[] {0.0, 0.0},
21             new[] {0.0, 1.0},
22             new[] {1.0, 0.0},
23             new[] {1.0, 1.0}
24         };
25
26         /// <summary>
27         /// Salidas deseadas para la funcion XOR.
28         /// </summary>
29         public static double[][] XORIdeal =
30         {
31             new[] {0.0},
32             new[] {1.0},
33             new[] {1.0},
34             new[] {0.0}
35         };
36
37         static void Main(string[] _)
38         {
39             // Se crea una red nueronal basica de la clase BasicNetwork
40             var network = new BasicNetwork();
41             //Al ser la capa de entradas no existe funcion de activacion pero existe un bias de 1
42             network.AddLayer(new BasicLayer(null, true, 2));
43             //La configuración de la compuerta XOR, consta de una capa oculta de 3 nueronas
44             network.AddLayer(new BasicLayer(new ActivationSigmoid(), true, 3));
45             //Se agrega la capa de salida de la red, siendo una sola neurona
46             network.AddLayer(new BasicLayer(new ActivationSigmoid(), false, 1));
47             //Se termina de construir la red
48             network.Structure.FinalizeStructure();
49             network.Reset();
50
51             // Se crea el conjunto de datos de entrenamiento
52             IMLDataSet trainingSet = new BasicMLDataSet(XORInput, XORIdeal);
53
54             // Se entrena la red nueronal con los datos para entrenamiento
55             IMLTrain train = new ResilientPropagation(network, trainingSet);
56 }
```

```

57     //Epoca es las veces en las que se recorre todas las entradas para la normalización de los datos
58     int epoch = 1;
59
60     do
61     {
62         //Se inicia el entrenamiento de la red nueronal
63         train.Iteration();
64         //Se muestra la epoca y el error para la iteracion actual
65         Console.WriteLine(@"Epoca #" + epoch + @" Error:" + train.Error);
66         epoch++;
67     } while (train.Error > 0.01);
68
69     //Termina el entrenamiento cuando el error es menor a .01
70     train.FinishTraining();
71
72     //Se realiza la prueba de la nuerona entrenada
73     Console.WriteLine(@"Resultados de la red nuronal:");
74     foreach (IMLDataPair pair in trainingSet)
75     {
76         //Se computan los pares de entradas para la red entrenada
77         IMLData output = network.Compute(pair.Input);
78         Console.WriteLine(pair.Input[0] + @", " + pair.Input[1]
79                         + @", actual=" + output[0] + @", ideal=" + pair.Ideal[0]);
80     }
81     EncogFramework.Instance.Shutdown();
82     Console.Read();
83
84 }
85
86

```

Y tenemos la siguiente respuesta de la consola:

```

C:\Users\jorge\Documents\Facultad\SEM2022-1\Temas Selectos de Programación I\Perceptron(NET)\PerceptronXOR\bin\Debug\PerceptronXOR.exe

Epoca #60 Error:0.0443130562713986
Epoca #61 Error:0.0504801561243656
Epoca #62 Error:0.0407333029188926
Epoca #63 Error:0.0396420405513696
Epoca #64 Error:0.0498803113391959
Epoca #65 Error:0.0366218357748813
Epoca #66 Error:0.0356269595417679
Epoca #67 Error:0.0453793052694164
Epoca #68 Error:0.0316262038384001
Epoca #69 Error:0.0304247122854229
Epoca #70 Error:0.0327804670432504
Epoca #71 Error:0.0252052988353348
Epoca #72 Error:0.0225662846901869
Epoca #73 Error:0.0200368461617795
Epoca #74 Error:0.0184953527884182
Epoca #75 Error:0.0166283017332818
Epoca #76 Error:0.0152586977034735
Epoca #77 Error:0.0149226766716344
Epoca #78 Error:0.014602786584073
Epoca #79 Error:0.0138799402390907
Epoca #80 Error:0.0126198265636021
Epoca #81 Error:0.0113939639965737
Epoca #82 Error:0.010233830828828
Epoca #83 Error:0.00884014691523191
Resultados de la red nuronal:
0,0, actual=0.0681112523345273, ideal=0
0,1, actual=0.919540385045215, ideal=1
1,0, actual=0.90711554301396, ideal=1
1,1, actual=0.101677219470564, ideal=0

```

Y, por último, el forms correspondiente al perceptrón de macros.

COMPUERTA AND

ENTRADAS		SALIDA	
X1	X2	D	->
1	1	1	->
1	0	0	->
0	1	0	->
0	0	0	->

Perceptron

X1	W1	F(x)	D
1	.5	0.46	1

X2	W2	θ	Y
1	.2	.24	1

η : .25

Calculo errores

$$\delta = D - Y = 0$$

$$\Delta_1 = \eta \delta X_1 = 0$$

$$\Delta_2 = \eta \delta X_2 = 0$$

$$W_1 = W_1 + \Delta_1 = 0.5$$

$$W_2 = W_2 + \Delta_2 = 0.2$$

$$\theta = \theta - \eta \delta = 0.24$$

Actualizar

Formulas:

$$Y = F\left(\sum_{i=1}^n W_i X_i - \theta\right)$$

$$F(x) = \begin{cases} 1, & \text{Si } x \geq 0 \\ 0, & \text{Si } x < 0 \end{cases}$$

$$\delta = (D - Y)$$

$$\Delta_i = \eta \delta X_i$$

$$W_i = W_i + \Delta_i$$

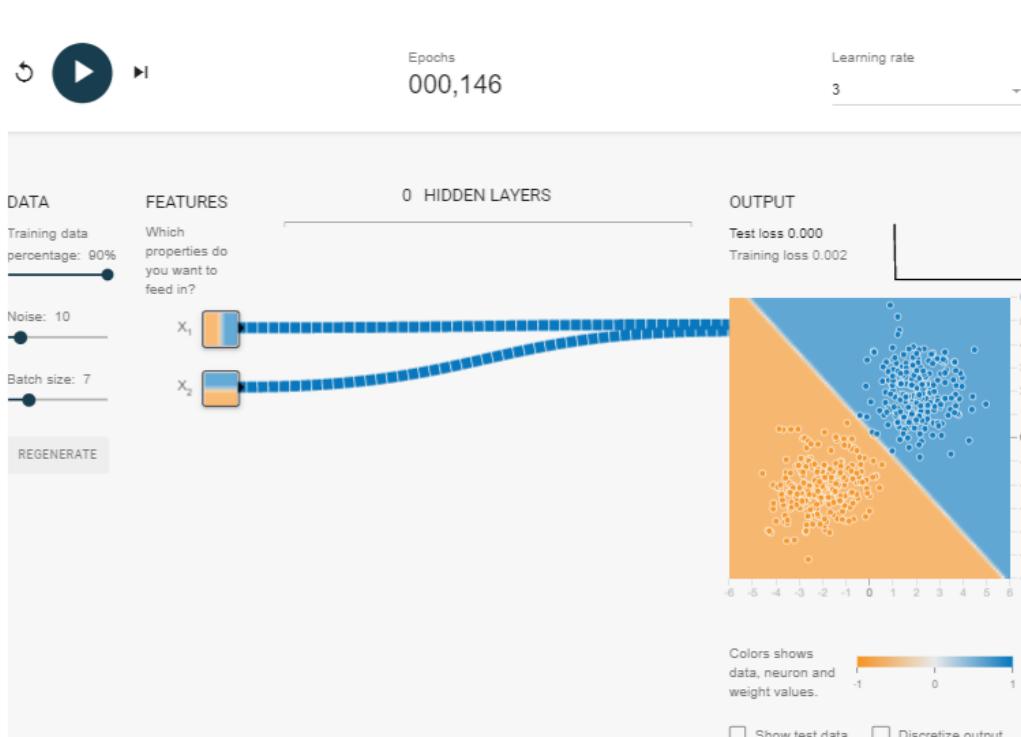
$$\theta = \theta - \eta \delta$$

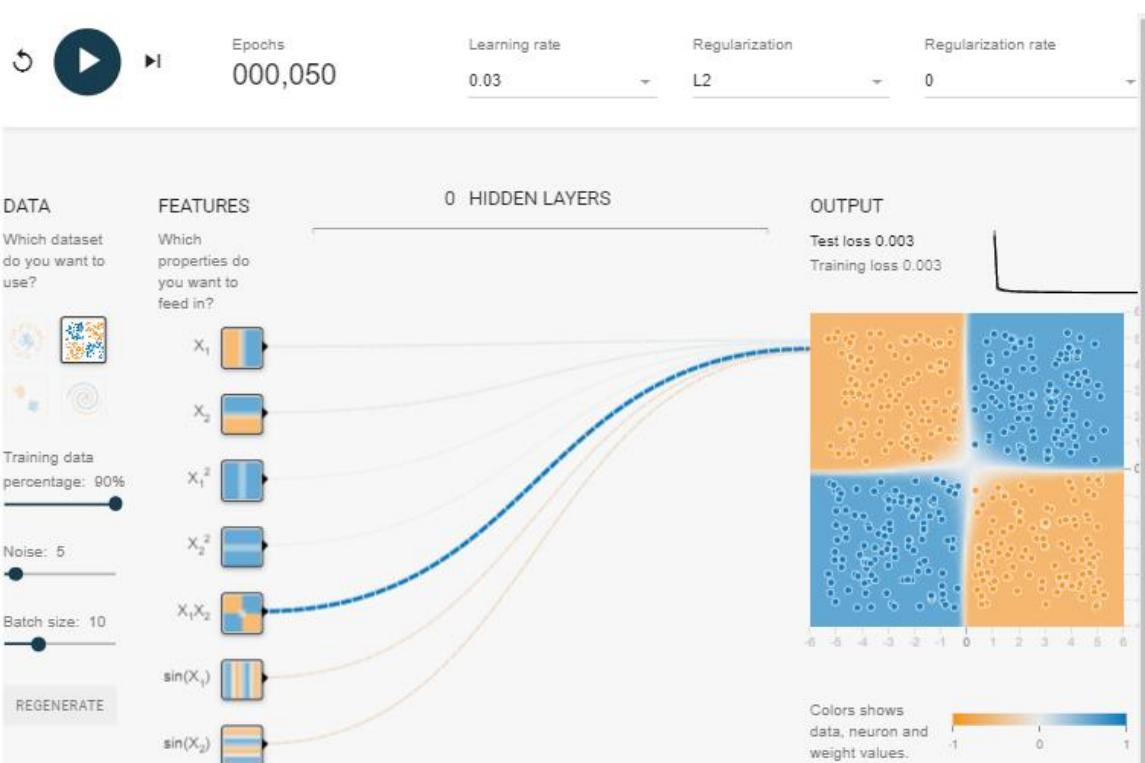
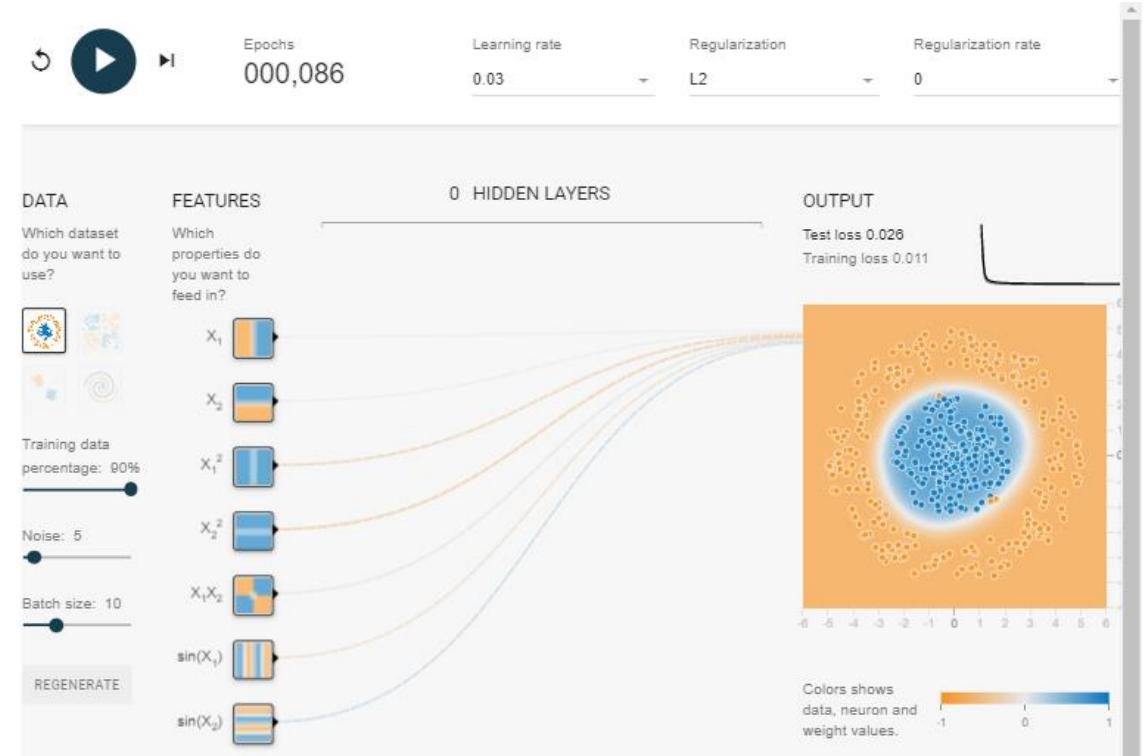
Sair

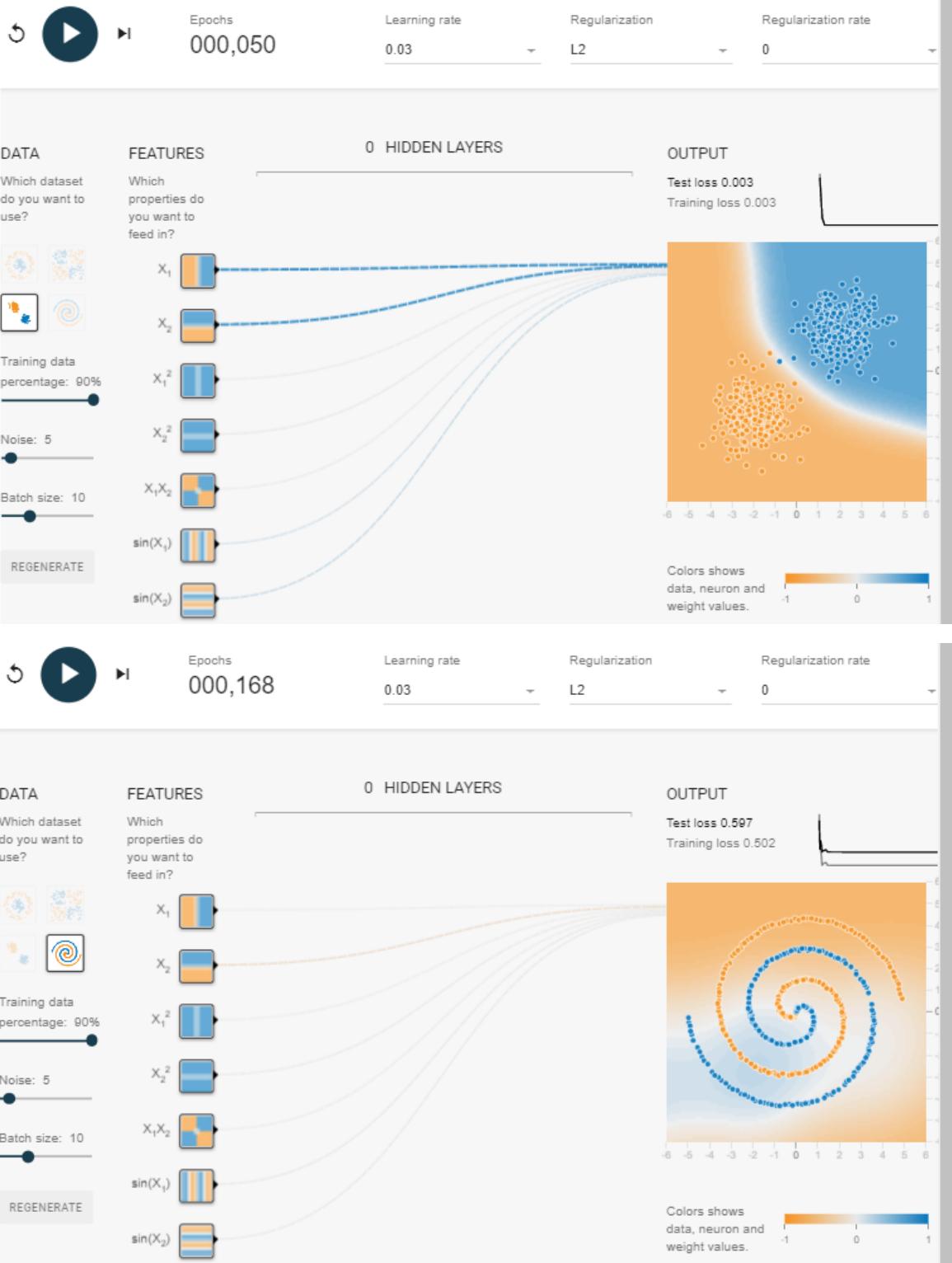
2 / Diciembre / 2021

Vimos algunos ejemplos de clasificación mediante redes neuronales, los cuales se encontraban en el curso intensivo de Machine Learning de Google. Finalmente nos quedamos trabajando con un apartado para comprender que no es necesario tener una gran cantidad de capas y neuronas en las mismas, sino que dependen de la suma ponderada interna de cada neurona, así como de su función de activación.

De esto nos dimos cuenta en unos ejemplos, ya que la línea de conexión de las neuronas, según el peso que tienen asignado, tendrán un tono de color más fuerte entre más grande sea el valor del peso.

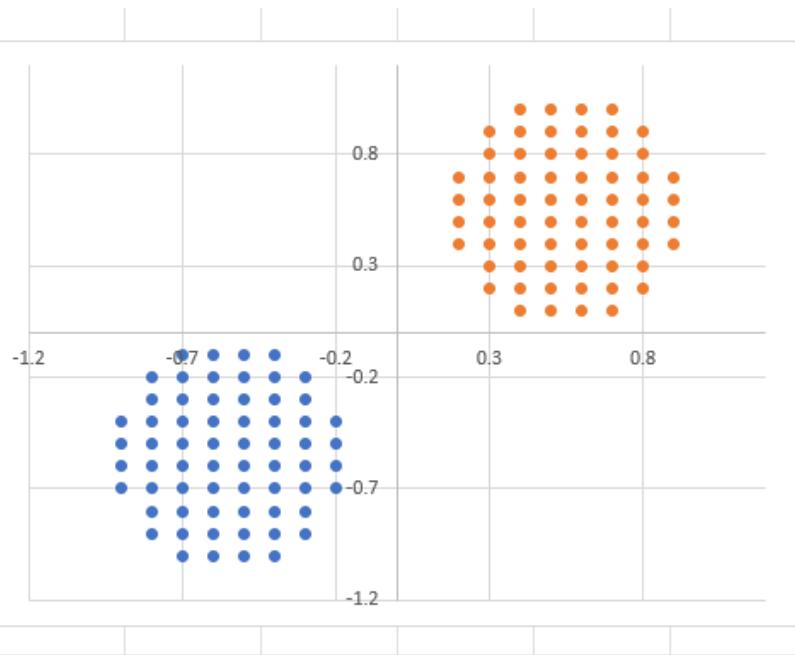




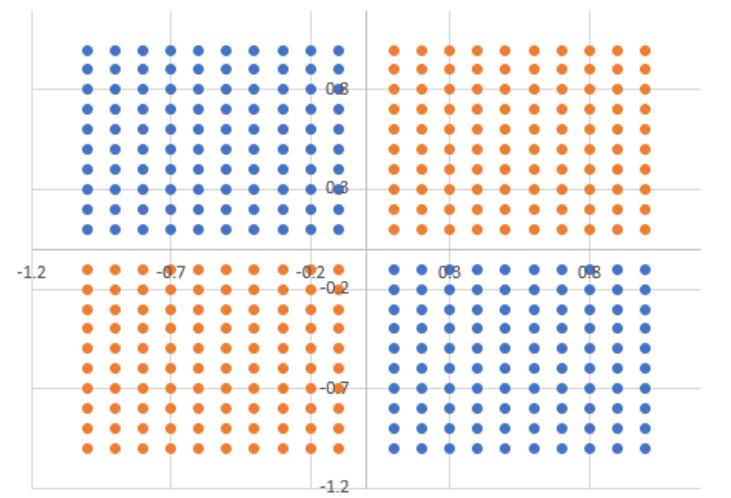


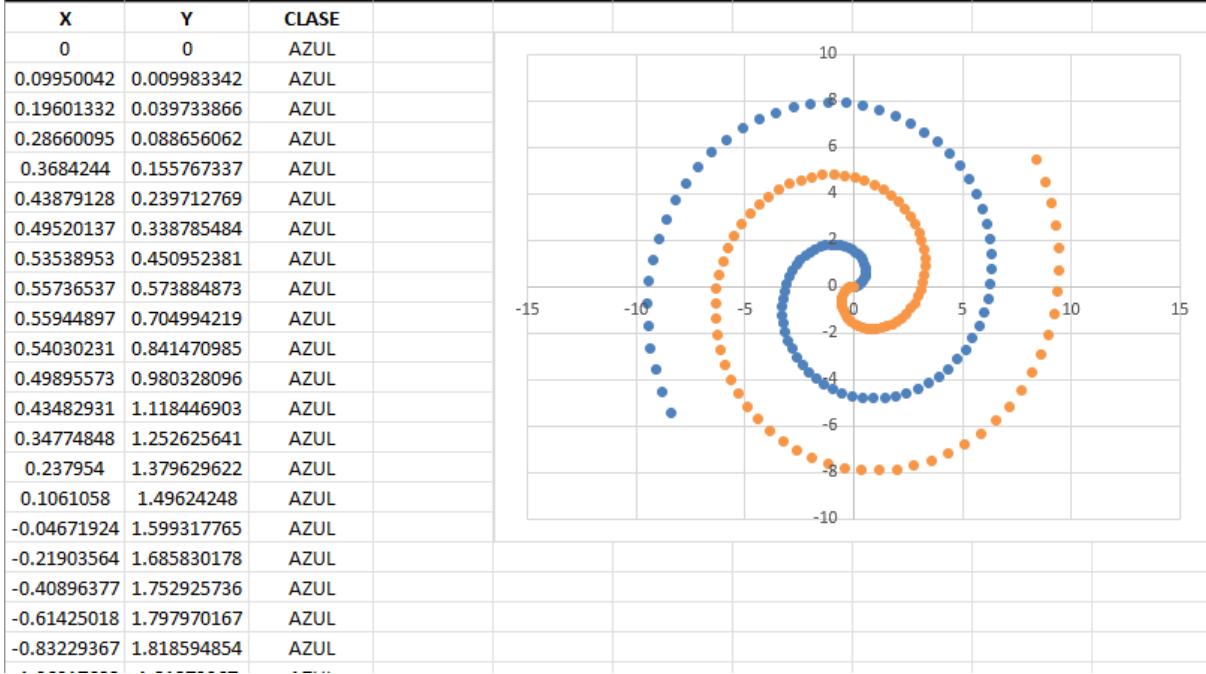
Tarea 21: Ejemplos del curso de Machine Learning con Encog (datos en Excel (CSV))

X	Y	CLASE
-0.9	-0.7	AZUL
-0.9	-0.6	AZUL
-0.9	-0.5	AZUL
-0.9	-0.4	AZUL
-0.8	-0.9	AZUL
-0.8	-0.8	AZUL
-0.8	-0.7	AZUL
-0.8	-0.6	AZUL
-0.8	-0.5	AZUL
-0.8	-0.4	AZUL
-0.8	-0.3	AZUL
-0.8	-0.2	AZUL
-0.7	-1	AZUL
-0.7	-0.9	AZUL
-0.7	-0.8	AZUL
-0.7	-0.7	AZUL
-0.7	-0.6	AZUL
-0.7	-0.5	AZUL
-0.7	-0.4	AZUL
-0.7	-0.3	AZUL
-0.7	-0.2	AZUL



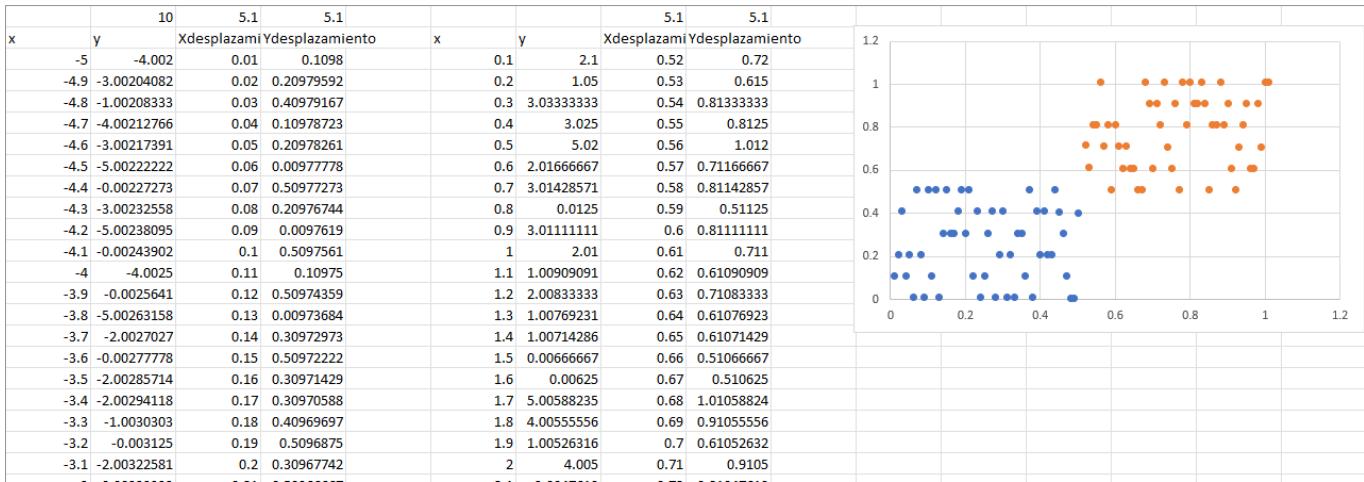
X	Y	CLASE
-1	1	AZUL
-1	0.9	AZUL
-1	0.8	AZUL
-1	0.7	AZUL
-1	0.6	AZUL
-1	0.5	AZUL
-1	0.4	AZUL
-1	0.3	AZUL
-1	0.2	AZUL
-1	0.1	AZUL
-0.9	1	AZUL
-0.9	0.9	AZUL
-0.9	0.8	AZUL
-0.9	0.7	AZUL
-0.9	0.6	AZUL
-0.9	0.5	AZUL
-0.9	0.4	AZUL
-0.9	0.3	AZUL
-0.9	0.2	AZUL
-0.9	0.1	AZUL
-0.8	1	AZUL





7 / Diciembre / 2021

Con los datos de Excel que se nos pidió la clase pasada, hicimos un nuevo archivo de dispersión de datos, ya que ahora, lo que buscábamos era que nuestros datos de muestra se encontraran en un rango de 0 y 1, por lo que tuvimos que normalizarlos. Una vez que hicimos esa normalización, tratamos de recorrerlos del cuadrante que se encontraban, teniendo nuestra hoja de cálculo de la siguiente forma.



De los ejemplos de Jeff Heaton realizamos nuevamente el programa de la compuerta XOR, para ir comprendiendo mejor el como funciona la librería de Encog, ya que la vamos a requerir para la segunda etapa de nuestro proyecto. Utilizamos algunos datos de la hoja de cálculo anterior para nuestro ejemplo, y recrear la compuerta XOR, dado que no alcanzó el tiempo, terminar el código y obtener los pesos quedó como tarea.

Tarea 22: Obtener los pesos de cada neurona del ejemplo de clase y guardarlos, además de probar ahora con unos diferentes valores.

```
13
14     namespace EjemploXOR2
15     {
16         class Program
17         {
18             /// <summary>
19             /// Input for the XOR function.
20             /// </summary>
21             public static double[][] XORInput =
22             {
23                 new[] {0.01, 0.1098},
24                 new[] {0.02, 0.2097},
25                 new[] {0.03, 0.4097},
26                 new[] {0.04, 0.1097},
27                 new[] {0.05, 0.2097},
28                 new[] {0.52, 0.72},
29                 new[] {0.53, 0.615},
30                 new[] {0.54, 0.813},
31                 new[] {0.55, 0.8125},
32                 new[] {0.56, 1.0}
33             };
34
35             /// <summary>
36             /// Ideal output for the XOR function.
37             /// </summary>
38             public static double[][] XORIdeal =
39             {
40                 new[] {0.0},
41                 new[] {0.0},
42                 new[] {0.0},
43                 new[] {0.0},
44                 new[] {0.0},
45                 new[] {1.0},
46                 new[] {1.0},
47                 new[] {1.0},
48                 new[] {1.0},
49                 new[] {1.0}
50             };
51
52             static void Main(string[] _)
53             {
54                 // create a neural network, without using a factory
55                 var network = new BasicNetwork();
56                 network.AddLayer(new BasicLayer(null, true, 2));
57                 network.AddLayer(new BasicLayer(new ActivationSigmoid(), true, 4));
58                 network.AddLayer(new BasicLayer(new ActivationSigmoid(), false, 1));
59                 //ActivationReLU
60                 network.Structure.FinalizeStructure();
```

```

61     network.Reset();
62
63     // create training data
64     //Para entrenar la red neuronal, se construye un objeto IMLDataSet.
65     //Este objeto contiene las entradas y las salidas esperadas.
66     //Para construir este objeto, se crean dos matrices.
67     //La primera matriz contendrá los valores de entrada para el operador XOR.
68     //La segunda matriz contendrá las salidas ideales para cada uno de los cuatro valores de entrada correspondientes.
69     IMLDataSet trainingSet = new BasicMLDataSet(XORInput, XORIdeal);
70
71     // train the neural network
72     IMLTrain train = new ResilientPropagation(network, trainingSet);
73
74
75     int epoch = 1;
76     do
77     {
78         train.Iteration();
79         Console.WriteLine(@"Epoch #" + epoch + @" Error:" + train.Error);
80         epoch++;
81         if (epoch > 1000)
82         {
83             epoch = 0;
84             network.Reset();
85         }
86     } while (train.Error > 0.001);
87
88     Console.WriteLine("-----");
89     for (int i = 0; i < network.Flat.Weights.Length; i++)
90     {
91         Console.WriteLine(network.Flat.Weights[i]);
92     }
93
94     double[][] pesos = new double[network.LayerCount - 1][]{};
95     int capa = network.LayerCount - 2;
96     for (int k = 0; k <= network.LayerCount - 2; k++)
97     {
98         pesos[k] = new double[network.Flat.LayerFeedCounts[k], network.Flat.LayerCounts[k + 1]];
99         for (int i = 0; i < pesos[k].GetLength(0); i++)
100        {
101            for (int j = 0; j < pesos[k].GetLength(1); j++)
102            {
103                pesos[k][i, j] = network.GetWeight(capa, j, i);
104            }
105        }
106        capa--;
107    }
108
109
110    // test the neural network
111    Console.WriteLine(@"Neural Network Results:");
112
113    //El método Compute acepta una clase IMLData y también devuelve otro objeto IMLData.
114    //Solo necesito pasar el arreglo de entrada a una entrada de tipo IMLData
115    //El objeto devuelto contiene la salida de la red neuronal, que se muestra al usuario.
116    //Con la ejecución del programa, primero se muestran los resultados del entrenamiento.
117
118    //Salidas necesitando IMLDataPair
119    double[] Entrada = { 0.045, 0.2500 };
120    double[] Salida = { 0 };
121    IMLData IN = new BasicMLData(Entrada);
122    IMLData OUT = new BasicMLData(Salida);
123
124    //Entradas y salidas
125    IMLDataPair Prueba = new BasicMLDataPair(IN, OUT);
126    IMLData Teoria = network.Compute(Prueba.Input);
127    Console.WriteLine("{0} XOR {1} ---->Esperado:{2}---->Obtenido:{3}", Prueba.Input[0], Prueba.Input[1], Prueba.Ideal[0], Teoria[0]);
128
129    //Salidas sin necesitar IMLDataPair
130    double[] Entrada2 = { 0.60, 0.7500 };
131    double[] Salida2 = { 1 };
132    IMLData IN2 = new BasicMLData(Entrada2);
133    IMLData Teoria2 = network.Compute(IN2);
134    Console.WriteLine("{0} XOR {1} ---->Esperado:{2}---->Obtenido:{3}", Entrada2[0], Entrada2[1], Salida2[0], Teoria2[0]);
135
136    //Imprimir pesos
137    Escribir_pesos(pesos);
138
139    //Transportar pesos
140
141

```

```
142
143     /*
144      foreach (IMLDataPair pair in trainingSet)
145      {
146
147          IMLData output = network.Compute(pair.Input);
148          Console.WriteLine(pair.Input[0] + @"," + pair.Input[1]
149                         + @", actual=" + output[0] + @",ideal=" + pair.Ideal[0]);
150      }
151
152      */
153      EncogFramework.Instance.Shutdown();
154      Console.ReadLine();
155  }
156
157  /// <summary>
158  /// Imprime una matriz de pesos
159  /// </summary>
160  /// <param name="W"></param>
161  public static void Imprimirmatriz(double[,] W)
162  {
163
164      /* Peso asociado de la neurona m a la neurona m
165      *   N1| N2| N3|...|Nm
166      *   -----
167      *   | N1 | | | | |
168      *   -----
169      *   | N2 | | | | |
170      *   -----
171      *   | N3 | | | | |
172      *   .
173      *   .
174      *   .
175      *   .
176      *   .
177      *   .
178      *   .
179      *   .
180      *   .
181      *   .
182      *   .
183      *   .
184      *   .
185      *   .
186      *   .
187      *   .
188      *   .
189      *   .
190      *   .
191      *   .
192      *   .
193      *   .
194      *   .
195      *   .
196      *   .
197
198      //Pesos de capa oculta y capa de salida
199      for (int j = 0; j < W.GetLength(0); j++)
200      {
201          Guardar_pesos(W[j], ruta);
202          Console.WriteLine("-----Pesos" + j + " -----");
203          Imprimirmatriz(W[j]);
204      }
205
206
207  }
208
209  public static void Guardar_pesos(double[,] W, string ruta)
210  {
211
212      string cadena = "";
213      List<string> lista = new List<string>();
214      for (int j = 0; j < W.GetLength(1); j++)
215      {
216          cadena = cadena + "    |    N" + Convert.ToString(j + 1);
217      }
218      lista.Add(cadena);
219      for (int i = 0; i < W.GetLength(0); i++)
220      {
221          cadena = "N" + Convert.ToString(i + 1) + "|";
222          for (int j = 0; j < W.GetLength(1); j++)
223          {
224              if (j == W.GetLength(1) - 1)
225              {
226                  cadena += Convert.ToString(W[i, j]);
227              }
228          }
229      }
230
231      File.WriteAllText(ruta, lista.ToString());
232  }
```

```

226         else
227             {
228                 cadena = cadena + Convert.ToString(W[i, j]) + "|";
229             }
230         lista.Add(cadena);
231     }
232     cadena = "-----\n";
233     lista.Add(cadena);
234     for (int j = 0; j < lista.Count; j++)
235     {
236         File.AppendAllText(ruta, lista[j] + "\n");
237     }
238 }
239 }
240 }
241 }

```

La respuesta de la consola:

```

C:\Users\jorge\Documents\Facultad\SEM2022-1\Temas Selectos de Programación I\Perceptron(NET)\EjemploXOR\bin\Debug\EjemploXOR.exe

-----
1.82394350620405
-4.49727232950618
-5.36224097622669
1.60741928964968
0.552192362275815
13.5700553049035
6.79594070175482
-4.47157891177769
-12.2386926470346
-3.35547040154926
2.44535853600421
-13.0621174542364
-3.21902181326346
1.41538493134807
12.7540397443436
6.21210732460594
-4.17256235679854
Neural Network Results:
0.045 XOR 0.25 ---->Esperado:0---->Obtenido:0.00591901487298605
0.6 XOR 0.75 ---->Esperado:1---->Obtenido:0.981644765303244
-----Pesos0 -----
| N1 | N2 | N3 | N4 | N5 |
N1| 1.82394351 | -4.49727233 | -5.36224098 | 1.60741929 | 0.55219236 |
-----Pesos1 -----
| N1 | N2 | N3 |
N1| 13.5700553 | 6.7959407 | -4.47157891 |
N2| -12.23869265 | -3.3554704 | 2.44535854 |
N3| -13.06211745 | -3.21902181 | 1.41538493 |
N4| 12.75403974 | 6.21210732 | -4.17256236 |

```

También se almacenan en un archivo llamado “Pesos.txt”.

	N1	N2	N3	N4	N5
N1	1.82394350620405	-4.49727232950618	-5.36224097622669	1.60741928964968	0.552192362275815

	N1	N2	N3
N1	13.5700553049035	6.79594070175482	-4.47157891177769
N2	-12.2386926470346	-3.35547040154926	2.44535853600421
N3	-13.0621174542364	-3.21902181326346	1.41538493134807
N4	12.7540397443436	6.21210732460594	-4.17256235679854

Dado que varios no logramos realizar lo que se pidió, nuestro compañero Adrián lo compartió durante la clase, a petición del profesor, ya que esto es necesario para el desarrollo de la segunda etapa.

Práctica 5: Servicios web

WCF es un modelo de programación para el desarrollo de aplicaciones con arquitectura orientada a servicios (SOA). Aplicaciones distribuidas basadas en la comunicación mediante mensajes.

Principales Características

Una aplicación WCF está compuesta por:

- Clientes: Son aplicaciones que inician la comunicación.
 - Servicios: Son aplicaciones que esperan los mensajes de los clientes y responden a los mismos.
- Los mensajes son enviados entre endpoints. Un endpoint (veremos más adelante un poco más detallado este concepto) es un lugar donde un mensaje es enviado, o recibido, o ambos.
- Un servicio expone uno o más application endpoints, y un cliente genera un endpoint compatible con uno de los endpoints de un servicio dado.
- La combinación de un servicio y un cliente compatibles conforman un communication stack.

Algunos ejemplos de servicios pueden ser:

- Servicio seguro para procesar transacciones comerciales
- Un panel de control que sondea los datos de uno o varios servicios y los muestra lógicamente.
- Un servicio de chat que permite a dos personas comunicarse e intercambiar datos en tiempo real.

Los conceptos básicos que se manejan en WCF son Address, Binding, Contract, Endpoint, Behavior, Hosting, Channels y Metadata,

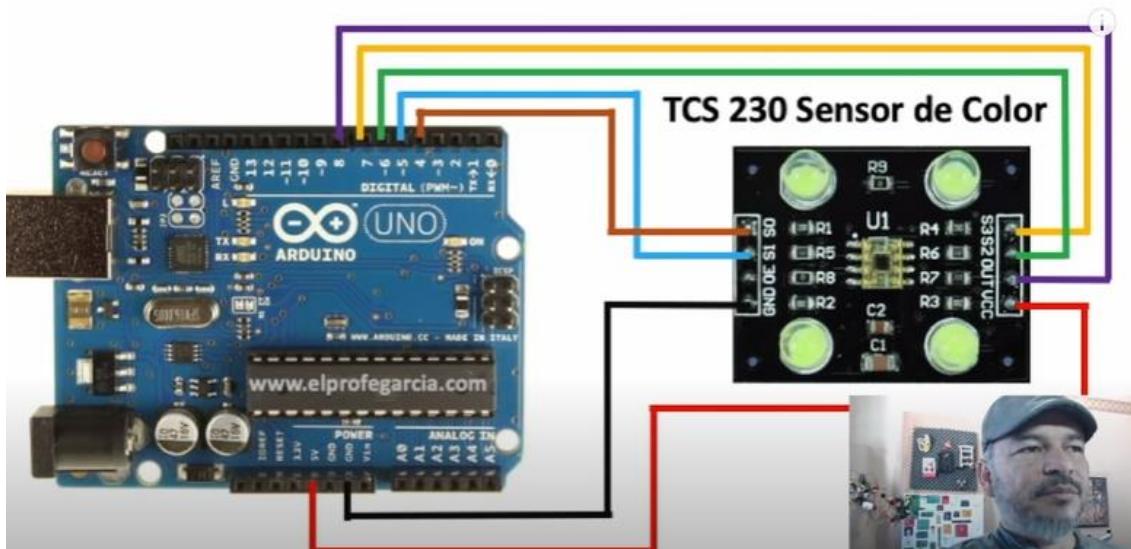
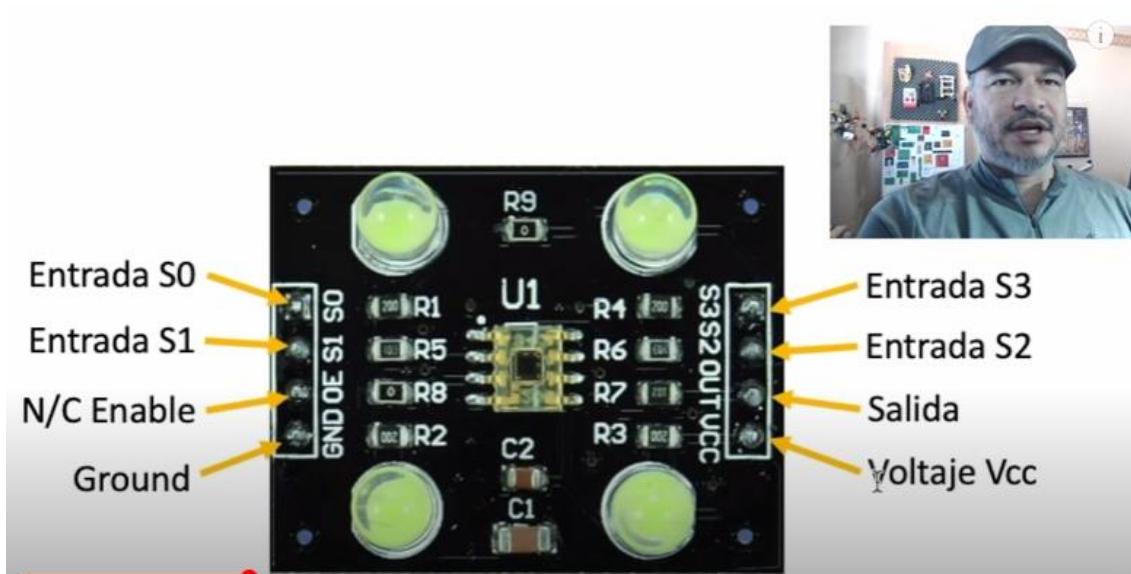
Referencias:

<http://dotnetuy.com/blog/2018/02/14/tutorial-wcf-primera-parte-conceptos-basicos-de-wcf-windows-communication-foundation/>

<https://www.kabel.es/mi-capa-de-servicios-wcf-o-web-api/>

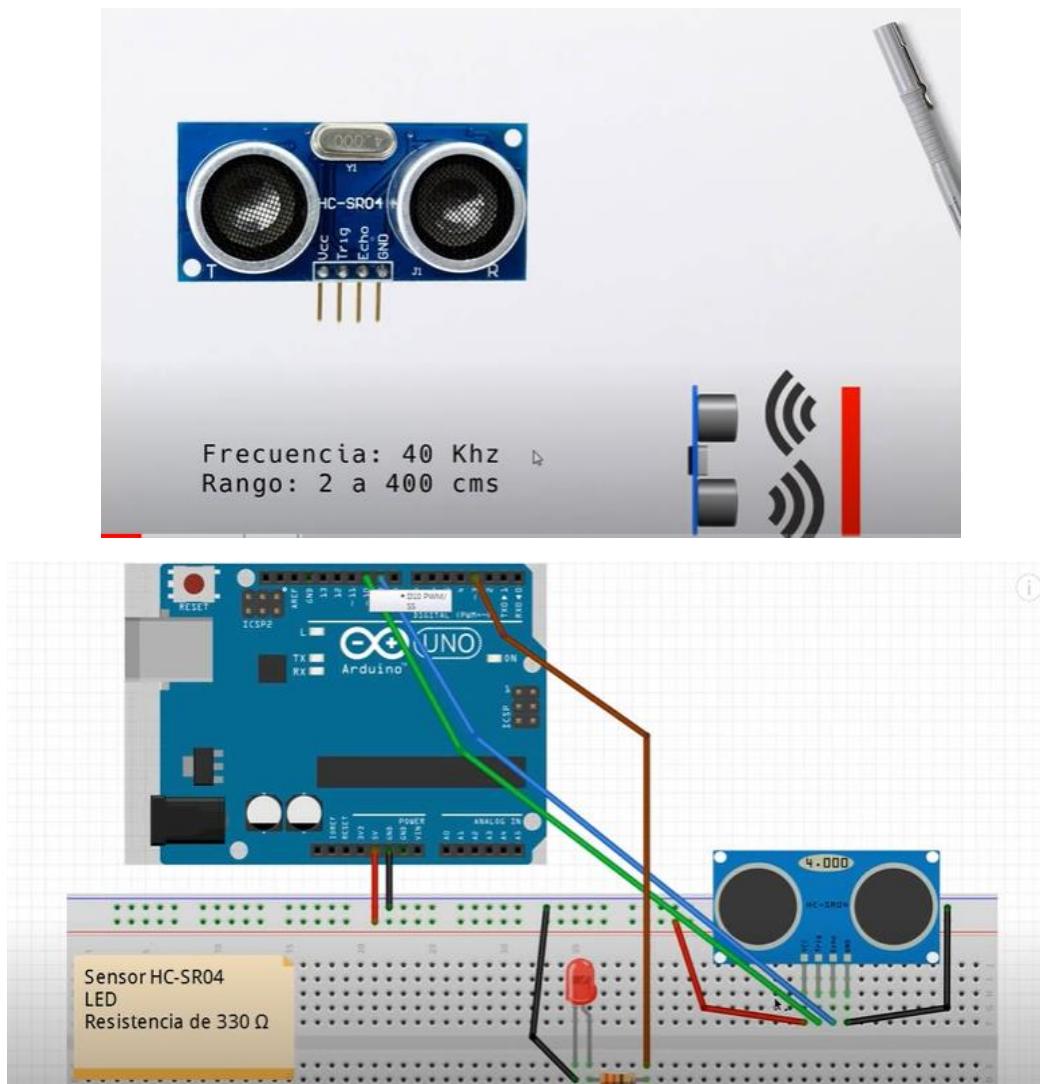
Se emplearán los siguientes sensores:

- Sensor de color TCS230



Durante el desarrollo de nuestro proyecto tuvimos ciertas problemáticas con este sensor, ya que la luz ambiental genera una gran interferencia de las mediciones, además, de la calidad de los cables, algunas veces la intensidad de los LED ya integrados, se atenuaban un poco, por lo que teníamos un mal funcionamiento, pero logramos darle una solución.

- HC-SR04 Sensor Ultrasónico



Para este sensor no tuvimos ningún problema, ya que no depende de la luz ni de otra cosa, salvo de una onda sonora de ciertas frecuencias para calcular la distancia recorrida según el tiempo de vuelo de este y obtener nuestro resultado, después de calibrar la distancia que ya se tiene sin medir el objeto.

Como vimos, podemos distinguir mediante la probabilidad, si un dato pertenece o no a un conjunto de datos, según los valores de promedio, varianza y desviación estándar de la distribución con la que se quiere comparar. Dado que para uno de nuestros colores tenía valores semejantes o compartidos con los otros dos, decidimos omitirlo, ya que, a pesar de acotar más los valores, este siempre nos indicaba que era cualquiera de los otros dos colores, siendo que empleábamos morado, verde y naranja.

Tal como estuvimos viendo, entre mayor sea el valor de la probabilidad en la distribución, se puede asumir o predecir, que realmente pertenece a esa clase, partiendo de muestras que nosotros realizamos previamente con nuestro sensor.

Referencias:

<https://www.geekfactory.mx/tutoriales-arduino/hc-sr04-con-arduino-sensor-de-distancia-ultrasonico/>

<https://hetpro-store.com/TUTORIALES/sensor-de-color-tcs3200-con-arduino/>

<https://youtu.be/mlw3APOUt8U> - Ejemplo con HC-SR04 Sensor Ultrasónico

<https://youtu.be/CvO5adh12a8> - Ejemplo con sensor de color TCS230

Proyecto: Clasificador mediante redes neuronales

Esta vez vamos a realizar una mejora de nuestro código para el clasificador de paquetes, esta vez empleando la librería Encog de Jeff Heaton, para mejorar el algoritmo de clasificación, además de entrenar nuestra red para un mejor desempeño de nuestro clasificador. Previo a esto, realizamos un ejemplo del manual de usuario, que es el ejemplo con los datos de Iris-setosa, iris-versicolor e iris-virginica, y como ya vimos durante las ultimas clases, diseñamos nuestra red neuronal según los resultados que nos dio Encog , y posteriormente, la empleamos para computar la información recibida de Arduino.

Para desarrollarlo, nos basamos en parte también del ejemplo de la compuerta XOR de la guia rápida de Encog, en cuanto a la parte de diseñar la red.

Lo que más se nos dificultó fue la parte de computar los datos normalizados, ya que, por las variaciones del sensor, no siempre da el resultado en el rango de valores aproximados que nos proporciona el reporte de análisis de Encog.

Referencias:

https://s3.amazonaws.com/heatonresearch-books/free/encog-3_3-quickstart.pdf

<https://s3.amazonaws.com/heatonresearch-books/free/Encog3CS-User.pdf>

Métodos de nuestra librería, basada en las librerías de Matlab

A lo largo del curso, desarrollamos nuestra propia librería basada en métodos del software Matlab. La librería la dividí en dos clases: **Basics** y **Stadistic**. La clase Basics, bien contiene los métodos básicos que empleamos durante el curso, siendo los siguientes métodos.

Clase Basics**Sum**

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MATLAB_Library
8  {
9      public class Basics
10     {
11         /// <summary>
12         /// Suma de dos valores, devuelve el valor de la suma
13         /// </summary>
14         /// <param name = "input1"> Sumando 1 </param>
15         /// <param name = "input2"> Sumando 2 </param>
16         /// <returns> Suma </returns>
17         public double Sum(double input1, double input2)
18         {
19             double output = input1 + input2;
20             output = Math.Round(output, 4);
21
22             return output;
23         }
24     }

```

Este método es la operación básica de la suma de dos números de entrada.

SumList

```

25  /// <summary>
26  /// Retorna la suma de los valores de un arreglo
27  /// </summary>
28  /// <param name = "input"> Arreglo de datos </param>
29  /// <returns> Suma </returns>
30  public double SumList(double[] input)
31  {
32      double output = 0;
33
34      for (int i = 0; i < input.Count(); i++)
35      {
36          output = Sum(output, input[i]);
37      }
38      output = Math.Round(output, 4);
39
40      return output;
41  }
42

```

Este método retorna la suma de un arreglo de datos de entrada.

```
43  /// <summary>
44  /// Devuelve un arreglo resultante de sumar una constante a cada uno de los elementos de un
45  /// </summary>
46  /// <param name="input1"> Arreglo de datos </param>
47  /// <param name="input2"> Constante </param>
48  /// <returns> Arreglo de sumas </returns>
49  public double[] SumDot(double[] input1, double input2)
50  {
51      double[] output = new double[input1.Length];
52
53      for (int i = 0; i < input1.Length; i++)
54      {
55          output[i] = Math.Round(Sum(input1[i], input2), 4);
56      }
57
58      return output;
59  }
60
```

Este método retorna un arreglo de datos resultado de sumar una constante a cada uno de los elementos de un arreglo de datos de entrada.

SumArray

```
61  /// <summary>
62  /// Regresa la suma de dos arreglos elemento por elemento en un nuevo arreglo, todos de la
63  /// </summary>
64  /// <param name="input1"> Arreglo de datos 1 </param>
65  /// <param name="input2"> Arreglo de datos 2 </param>
66  /// <returns> Arreglo de sumas </returns>
67  public double[] SumArray(double[] input1, double[] input2)
68  {
69      double[] output = new double[input1.Length];
70
71      for (int i = 0; i < input1.Length; i++)
72      {
73          output[i] = Math.Round(Sum(input1[i], input2[i]), 4);
74      }
75
76      return output;
77  }
78
```

Este método regresa un arreglo de valores de sumas, resultado de sumar elemento a elemento dos arreglos de datos de entrada de la misma dimensión.

Resta

```

79     /// <summary>
80     /// Resta de dos valores, devuelve el valor de la resta
81     /// </summary>
82     /// <param name = "input1"> Minuendo </param>
83     /// <param name = "input2"> Sustraendo </param>
84     /// <returns> Resta </returns>
85     public double Resta(double input1, double input2)
86     {
87         double output = input1 - input2;
88         output = Math.Round(output, 4);
89
90         return output;
91     }
92

```

Este método realiza la operación aritmética de resta de dos números que recibe como entradas.

Diff

```

93     /// <summary>
94     /// Devuleve un arreglo de las diferencias de sus elementos de manera consecutiva en un nu
95     /// </summary>
96     /// <param name = "input"> Arreglo de datos </param>
97     /// <returns> Arreglo de diferencias </returns>
98     public double[] Diff(double[] input)
99     {
100         int l = input.Length - 1;
101         double[] output = new double[l];
102
103         for (int i = 0; i < l; i++)
104         {
105             double valor = input[i + 1];
106             output[i] = Math.Round(Resta(valor, input[i]), 4);
107         }
108
109         return output;
110     }
111

```

Este método regresa un arreglo con las diferencias entre los elementos de un arreglo de entrada de manera consecutiva. Es decir, el primer elemento del arreglo resultante sería el resultado de restar el primer y el segundo elemento del arreglo de entrada; el segundo, resultado de la resta del segundo y el tercer elemento y así, hasta terminar de recorrer todo el arreglo de entrada.

DiffArray

```

112  /// <summary>
113  /// Devuelve un arreglo resultante de la resta de dos arreglos, resta y resultado por dato
114  /// </summary>
115  /// <param name="input1"> Arreglo de minuendos </param>
116  /// <param name="input2"> Arreglo de sustraendos </param>
117  /// <returns> Arreglo de restas </returns>
118  public double[] DiffArray(double[] input1, double[] input2)
119  {
120      double[] output = new double[input1.Length];
121
122      for (int i = 0; i < input1.Length; i++)
123      {
124          output[i] = Math.Round(Resta(input1[i], input2[i]), 4);
125      }
126
127      return output;
128  }
129

```

Este método realiza la operación básica de la resta, de igual manera que el método Diff, regresa un arreglo de datos, pero este arreglo de datos es de la diferencia de restar dos arreglos de datos de la misma dimensión elemento a elemento.

Prod

```

130  /// <summary>
131  /// Devuelve el producto de dos números
132  /// </summary>
133  /// <param name="input1"> Factor 1 </param>
134  /// <param name="input2"> Factor 2 </param>
135  /// <returns> Producto </returns>
136  public double Prod(double input1, double input2)
137  {
138      double output = input1 * input2;
139      output = Math.Round(output, 4);
140
141      return output;
142  }
143

```

El método Prod corresponde a la operación básica de la multiplicación de dos elementos, devolviendo el producto de los dos números de entrada.

ProdDot

```

144     /// <summary>
145     /// Devuelve el arreglo resultante de multiplicar cada elemento por una constante
146     /// </summary>
147     /// <param name="input1"> Arreglo de datos </param>
148     /// <param name="input2"> Constante </param>
149     /// <returns> Arreglo de productos </returns>
150     public double[] ProdDot(double[] input1, double input2)
151     {
152         double[] output = new double[input1.Length];
153
154         for (int i = 0; i < input1.Length; i++)
155         {
156             output[i] = Math.Round(Prod(input1[i], input2), 4);
157         }
158
159         return output;
160     }
161

```

Este método devuelve un arreglo como resultado, el cual mediante un arreglo de datos de entrada y una constante, genera un nuevo arreglo con el resultado de multiplicar cada elemento del arreglo de entrada por la constante de entrada.

ProdArray

```

162     /// <summary>
163     /// Devuelve un arreglo resultante de multiplicar elemento por elemento de dos vectores de
164     /// </summary>
165     /// <param name="input1"> Arreglo de factores 1 </param>
166     /// <param name="input2"> Arreglo de factores 2 </param>
167     /// <returns> Arreglo de productos </returns>
168     public double[] ProdArray(double[] input1, double[] input2)
169     {
170         double[] output = new double[input1.Length];
171
172         for (int i = 0; i < output.Length; i++)
173         {
174             output[i] = Math.Round(Prod(input1[i], input2[i]), 4);
175         }
176
177         return output;
178     }
179

```

Este método genera un arreglo a partir de dos arreglos de entrada, el arreglo resultante corresponde a un vector que contiene el resultado de multiplicar elemento por elemento los dos conjuntos de datos de entrada.

Div

```
180  /// <summary>
181  /// Devuelve el cociente entre dos números
182  /// </summary>
183  /// <param name="input1"> Dividendo </param>
184  /// <param name="input2"> Divisor </param>
185  /// <returns> Cociente </returns>
186  public double Div(double input1, double input2)
187  {
188      double output = input1 / input2;
189      output = Math.Round(output, 4);
190
191      return output;
192  }
193
```

El método Div corresponde a la operación básica de la división de dos números de entrada.

DivDot

```
194  /// <summary>
195  /// Devuelve un arreglo tras dividir cada elemento de un arreglo entre una constante
196  /// </summary>
197  /// <param name="input1"> Arreglo de datos </param>
198  /// <param name="input2"> Constante </param>
199  /// <returns> Arreglo de cocientes </returns>
200  public double[] DivDot(double[] input1, double input2)
201  {
202      double[] output = new double[input1.Length];
203
204      for (int i = 0; i < input1.Length; i++)
205      {
206          output[i] = Math.Round(Div(input1[i], input2), 4);
207      }
208
209      return output;
210  }
211
```

Este método divide un arreglo de datos de entrada entre una constante, generando un nuevo arreglo como resultado de esta operación.

DivArray

```

212     /// <summary>
213     /// Devuelve un arreglo resultante de dividir elemento a elemento entre dos arreglos
214     /// </summary>
215     /// <param name="input1"> Arreglo de dividendos </param>
216     /// <param name="input2"> Arreglo de divisores </param>
217     /// <returns> Arreglo de cocientes </returns>
218     public double[] DivArray(double[] input1, double[] input2)
219     {
220         double[] output = new double[input1.Length];
221
222         for (int i = 0; i < input1.Length; i++)
223         {
224             output[i] = Math.Round(Div(input1[i], input2[i]), 4);
225         }
226
227         return output;
228     }
229

```

Este método genera un arreglo de cocientes, como resultado de dividir elemento por elemento dos arreglos de entrada.

Linspace

```

230     /// <summary>
231     /// Devuelve un arreglo desde input1 a input2 con un espaciado de input3 entre todos los valores
232     /// </summary>
233     /// <param name = "input1"> Valor inicial del arreglo </param>
234     /// <param name = "input2"> Valor final del arreglo </param>
235     /// <param name = "input3"> Espaciado de los datos </param>
236     /// <returns> Arreglo dimensionado </returns>
237     public double[] Linspace(double input1, double input2, double input3)
238     {
239         double paso = input3;
240         int l = Convert.ToInt32((Math.Abs(input2 - input1)) / paso) + 1;
241         double[] output = new double[l];
242
243         if (input1 > input2) //Si el valor inicial es mayor que el final, se crea el arreglo de manera decreciente
244         {
245             for (int i = 0; i < l; i++)
246             {
247                 output[i] = Math.Round((input1 - (paso * i)), 4);
248             }
249         }
250
251         else //Se crea el arreglo de manera creciente
252         {
253             for (int i = 0; i < l; i++)
254             {
255                 output[i] = Math.Round((input1 + (paso * i)), 4);
256             }
257         }
258
259         return output;
260     }
261

```

El método Linspace genera un arreglo uniformemente espaciado, recibiendo como entradas los valores extremos (límites) del arreglo y el espacio entre los datos que se van a generar entre los extremos, puede generar arreglos tanto de manera creciente como decreciente, el mismo método calcula el tamaño del arreglo resultante.

```
262  /// <summary>
263  /// Devuelve un arreglo con el error cuadrático medio de cada elemento a partir de un arreglo
264  /// </summary>
265  /// <param name="input1"> Arreglo de datos obtenidos </param>
266  /// <param name="input2"> Valor esperado </param>
267  /// <returns> Arreglo de errores cuadráticos medios </returns>
268  public double[] ECM(double[] input1, double input2)
269  {
270      int l = input1.Length;
271      double[] output = new double[l];
272
273      for (int i = 0; i < input1.Length; i++)
274      {
275          output[i] = Math.Round(Math.Pow((input1[i] - input2), 2), 4);
276      }
277
278      return output;
279  }
```

El método ECM corresponde al cálculo del Error Cuadrático Medio a manera de arreglo de errores, respecto con un arreglo de datos de entrada y respecto a un valor deseado o esperado.

ECM (sobrecarga)

```
281  /// <summary>
282  /// Devuelve un arreglo con el error cuadrático de cada elemento a partir de un arreglo de
283  /// </summary>
284  /// <param name="input1"> Arreglo de datos obtenidos </param>
285  /// <param name="input2"> Arreglo de datos esperados </param>
286  /// <returns> Arreglo de errores cuadráticos medios</returns>
287  public double[] ECM(double[] input1, double[] input2)
288  {
289      int l = input1.Length;
290      double[] output = new double[l];
291
292      for (int i = 0; i < input1.Length; i++)
293      {
294          output[i] = Math.Round(Math.Pow((input1[i] - input2[i]), 2), 4);
295      }
296
297      return output;
298  }
```

La sobrecarga de ECM corresponde a utilizar dos arreglos de entrada, uno con valores obtenidos, y otro con valores deseados, en lugar de solo un valor esperado. Este método también genera un arreglo de errores cuadráticos como salida.

```

300  /// <summary>
301  /// Devuelve el error cuadrático medio acumulado de un arreglo de errores generado a partir
302  /// </summary>
303  /// <param name="input1"> Arreglo de datos obtenidos </param>
304  /// <param name="input2"> Arreglo de datos esperados </param>
305  /// <returns> Suma de errores </returns>
306  public double ErrAccum(double[] input1, double[] input2)
307  {
308      double[] errores = ECM(input1, input2);
309      double output = SumList(errores);
310      output = Math.Round(output, 4);
311
312      return output;
313  }
314

```

Este método devuelve el ECM acumulado, recibiendo como entrada el arreglo de valores obtenidos, y el arreglo de valores esperados.

ErrAccum (sobrecarga)

```

315  /// <summary>
316  /// Devuelve el error cuadrático medio acumulado de un arreglo de errores generado a partir
317  /// </summary>
318  /// <param name="input1"> Arreglo de datos obtenidos </param>
319  /// <param name="input2"> Valor esperado </param>
320  /// <returns> Suma de errores </returns>
321  public double ErrAccum(double[] input1, double input2)
322  {
323      double[] errores = ECM(input1, input2);
324      double output = SumList(errores);
325      output = Math.Round(output, 4);
326
327      return output;
328  }
329

```

La sobrecarga de ErrAccum devuelve el ECM acumulado de recibir como entrada un arreglo de datos y un valor esperado, en lugar de otro arreglo.

ErrorPrc

```

330  /// <summary>
331  /// Devuelve un arreglo de errores porcentuales calculados a partir de dos arreglos de datos
332  /// </summary>
333  /// <param name="input1"> Arreglo de datos esperados </param>
334  /// <param name="input2"> Arreglo de datos obtenidos </param>
335  /// <returns> Arreglo de errores porcentuales </returns>
336  public double[] ErrorPrc(double[] input1, double[] input2)
337  {
338      double[] output = new double[input1.Length];
339
340      for (int i = 0; i < input1.Length; i++)
341      {
342          output[i] = Math.Round(Div(Math.Abs(Resta(input1[i], input2[i])), input1[i]), 4) * 100;
343      }
344
345      return output;
346  }
347

```

El método ErrorPrc genera un arreglo de errores porcentuales, recibiendo como entrada un arreglo de datos esperados y un arreglo de datos obtenidos.

ErrAccumPrc

```

348  /// <summary>
349  /// Devuelve el error cuadrático medio acumulado de un arreglo de errores generado a partir de
350  /// </summary>
351  /// <param name="input1"> Arreglo de datos obtenidos </param>
352  /// <param name="input2"> Arreglo de datos esperados </param>
353  /// <returns> Suma de errores </returns>
354  public double ErrAccumPrc(double[] input1, double[] input2)
355  {
356      double[] errores = ErrorPrc(input1, input2);
357      double output = SumList(errores);
358      output = Math.Round(output, 4);
359
360      return output;
361  }
362

```

Este método devuelve el error porcentual acumulado, recibiendo como entrada un arreglo de datos obtenidos y un arreglo de datos esperados.

Vector

```

363  /// <summary>
364  /// Devuelve el arreglo de valores correspondientes a una recta
365  /// </summary>
366  /// <param name="input1"> Pendiente </param>
367  /// <param name="input2"> Ordenada </param>
368  /// <param name="input3"> Arreglo de datos </param>
369  /// <returns> Arreglo de rectas </returns>
370  public double[] Vector(double input1, double input2, double[] input3)
371  {
372      double[] vector = ProdDot(input3, input1);
373      double[] output = SumDot(vector, input2);
374
375      return output;
376  }
377

```

El método Vector, genera un arreglo de valores generados a partir de la ecuación general de una recta: $y = mx + b$, recibiendo como entrada un arreglo correspondiente a los valores de x , el valor de la pendiente y el valor para la ordenada al origen.

```

378     /// <summary>
379     /// Devuelve el valor del punto con error acumulado de cero
380     /// </summary>
381     /// <param name="input1"> Punto aleatorio </param>
382     /// <param name="input2"> Ordenada </param>
383     /// <param name="input3"> Arreglo de valores obtenidos </param>
384     /// <param name="input4"> Arreglo de valores deseados </param>
385     /// <param name="input5"> Tamaño del incremento/decremento </param>
386     /// <returns> Punto mínimo </returns>
387     public void Descenso(double input1, double input2, double[] input3, double[] input4, double input5)
388     {
389         double p = Math.Round(input1, 4);
390         double step = Math.Round(input5, 4);
391         double pSup = p + step;
392         double pInf = p - step;
393         bool noterminar = true;
394
395         do
396         {
397             //Punto base
398             double[] y = Vector(p, input2, input3);
399             double error = ErrAccum(input4, y);
400
401             //Punto superior
402             y = Vector(pSup, input2, input3);
403             double errorSup = ErrAccum(input4, y);
404
405             //Punto inferior
406             y = Vector(pInf, input2, input3);
407             double errorInf = ErrAccum(input4, y);
408
409             //Si el error del punto inferior es mayor que el error actual
410             if (errorInf > error)
411             {
412                 p = Math.Round(pSup, 4);
413                 pSup += step;
414                 if (errorSup == 0)
415                 {
416                     //Si llegamos al error minimo, terminamos el ciclo
417                     noterminar = false;
418                 }
419             }
420             //Si el error del punto inferior es menor que el actual
421             else
422             {
423                 p = Math.Round(pInf, 4);
424                 pInf -= step;
425                 if (errorInf == 0)
426                 {
427                     //Si llegamos al error minimo, terminamos el ciclo
428                     noterminar = false;
429                 }
430             }
431             //Se crea el nuevo arreglo con el punto actual
432             y = Vector(p, input2, input3);
433             //Se calcula el error acumulado con el punto actual
434             error = ErrAccum(input4, y);
435             //Mostramos el error y puntos asociados, ademas del arreglo obtenido con dicho punto
436             Console.WriteLine("Valor del error: {0}, valor del punto: {1}", error, p);
437             PrintArray(y);
438         } while (noterminar != false);
439     }
440
441 }
```

El método de descenso por gradiente recibe como entrada el valor de un punto aleatorio, una ordenada al origen, dos arreglos de datos, uno de valores esperados y otro de valores obtenidos, finalmente, el valor del paso para el descenso/ascenso dependiendo de donde se encuentre el punto aleatorio, este

método de descenso busca el valor del punto que nos genere el menor error posible (deseablemente 0) a partir de los demás valores de entrada.

Hasta aquí corresponde los métodos de la clase Basics de la librería.

Los dos últimos métodos de esta clase solo pueden ser utilizados en consola, ya que son métodos para la impresión de arreglos en nuestra consola de pruebas.

PrintArray

```

441      //-----
442      //SOLO PARA CONSOLA
443
444      /// <summary>
445      /// Imprime un arreglo en consola con formato
446      /// </summary>
447      /// <param name="input"> Arreglo de entrada </param>
448      public static void PrintArray(double[] input)
449      {
450          int len = input.Length;
451          Console.WriteLine("[{0}", input[0].ToString("0.0000"));
452          for (int i = 1; i < len; i++)
453          {
454              Console.Write(", ");
455              Console.WriteLine(input[i].ToString("0.0000"));
456          }
457          Console.WriteLine("]");
458          Console.WriteLine("\n");
459      }
460
461

```

Este método nos permite imprimir arreglos con formato, admitiendo como máximo 4 decimales después del punto, y separando los elementos del arreglo mediante el uso de comas.

PrintArrayE

```

462      /// <summary>
463      /// Imprime un arreglo en consola con formato exponencial
464      /// </summary>
465      /// <param name="input"> Arreglo de entrada </param>
466      public static void PrintArrayE(double[] input)
467      {
468          int len = input.Length;
469          Console.WriteLine("[{0}", input[0].ToString("E"));
470          for (int i = 1; i < len; i++)
471          {
472              Console.Write(", ");
473              Console.WriteLine(input[i].ToString("E"));
474          }
475          Console.WriteLine("]");
476          Console.WriteLine("\n");
477      }
478
479
480

```

Este método nos permite imprimir arreglos con formato en notación científica y separando los elementos del arreglo mediante el uso de comas.

Clase Stadistics

Para la segunda clase de nuestra librería, Stadistics, corresponde a los métodos de estadística, teniendo los siguientes métodos.

Mean

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MATLAB_Library
8  {
9      public class Stadistics
10     {
11         /// <summary>
12         /// Retorna el valor promedio de un arreglo de datos
13         /// </summary>
14         /// <param name = "input"> Arreglo de datos </param>
15         /// <returns> Promedio </returns>
16         public double Mean(double[] input)
17         {
18             //Llamado de nuestra libreria para funciones básicas
19             Basics MATLAB = new Basics();
20
21             double output;
22             double suma = MATLAB.SumList(input);
23             output = Math.Round(MATLAB.Div(suma, input.Length), 4);
24
25             return output;
26         }
27     }

```

Este método nos permite obtener el valor promedio de un conjunto de datos de entrada, implementando los métodos creados de la clase Basics, como lo es el método SumList.

Var

```

28     /// <summary>
29     /// Retorna la varianza de un arreglo de datos para una distribución normal o gaussiana
30     /// </summary>
31     /// <param name = "input"> Arreglo de datos </param>
32     /// <returns> Varianza </returns>
33     public double Var(double[] input)
34     {
35         //Llamado de nuestra libreria para funciones básicas
36         Basics MATLAB = new Basics();
37
38         double output;
39         double promedio = Mean(input);
40         double n = input.Length;
41         double suma = 0;
42
43         for (int i = 0; i <= n - 1; i++)
44         {
45             suma = MATLAB.Sum(suma, Math.Pow(MATLAB.Resta(input[i], promedio), 2));
46         }
47
48         output = Math.Round(MATLAB.Div(suma, n), 4);
49
50     }
51 }
52

```

Este método nos permite obtener la varianza de un conjunto de datos de entrada mediante su ecuación, y emplea el método Mean de esta misma clase, ya que es necesario para su cálculo.

Std

```

53     /// <summary>
54     /// Retorna la desviación estandar de un arreglo de datos para una distribución normal o gaussiana
55     /// </summary>
56     /// <param name = "input"> Arreglo de datos </param>
57     /// <returns> Desviacon Estandar </returns>
58     public double Std(double[] input)
59     {
60         double output = Math.Sqrt(Var(input));
61         output = Math.Round(output, 4);
62
63         return output;
64     }
65

```

El método Std (Standard Deviation) nos permite obtener la desviación estándar de un conjunto de datos, como vimos estos conceptos en nuestras materias de probabilidad y estadística, este valor se obtiene de sacar la raíz cuadrada de la varianza.

Normpdf

```

66     /// <summary>
67     /// Retorna un arreglo con la distribucion normal o gaussiana mediante parametros, un arreglo de datos (input1), el promedio (i
68     /// </summary>
69     /// <param name = "input1"> Arreglo de datos </param>
70     /// <param name = "input2"> Promedio de los datos </param>
71     /// <param name = "input3"> Desviación estandar de los datos </param>
72     /// <returns> Distribución normal en arreglo </returns>
73     public double[] Normpdf(double[] input1, double input2, double input3)
74     {
75         //Llamado de nuestra libreria para funciones básicas
76         Basics MATLAB = new Basics();
77
78         double exp;
79         double fx;
80         double[] output = new double[input1.Length];
81
82         for (int i = 0; i < input1.Length; i++)
83         {
84             exp = MATLAB.Div(MATLAB.Prod(-1, Math.Pow(MATLAB.Resta(input1[i], input2), 2)), MATLAB.Prod(2, Math.Pow(input3, 2)));
85             fx = MATLAB.Prod(MATLAB.Div(1, MATLAB.Prod(input3, Math.Sqrt(MATLAB.Prod(2, Math.PI)))), Math.Exp(exp));
86             output[i] = Math.Round(fx, 4);
87         }
88
89         return output;
90     }
91

```

El método Normpdf (Normal probability density function) o función de densidad de probabilidad para una distribución normal (Gaussiana), nos permite generar un arreglo con los valores de densidad de probabilidad para un conjunto de datos de entrada, para que se pueda obtener correctamente la densidad de probabilidad, también se requiere el promedio del conjunto de datos de entrada, así como de la desviación estándar de los mismos datos.

```

92  /// <summary>
93  /// Retorna el valor asociado a la distribucion normal o gaussiana mediante parametros, un datos (input1), el promedio asociado
94  /// </summary>
95  /// <param name = "input1"> Dato </param>
96  /// <param name = "input2"> Promedio del dato </param>
97  /// <param name = "input3"> Desviación estandar del dato </param>
98  /// <returns> Distribucion normal individual </returns>
99  public double Normpdf(double input1, double input2, double input3)
100 {
101     //Llamado de nuestra libreria para funciones básicas
102     Basics MATLAB = new Basics();
103
104     double exp;
105     double fx;
106     double output;
107
108     exp = MATLAB.Div(MATLAB.Prod(-1, Math.Pow(MATLAB.Resta(input1, input2), 2)), MATLAB.Prod(2, Math.Pow(input3, 2)));
109     fx = MATLAB.Prod(MATLAB.Div(1, MATLAB.Prod(input3, Math.Sqrt(MATLAB.Prod(2, Math.PI)))), Math.Exp(exp));
110     output = Math.Round(fx, 4);
111
112     return output;
113 }
114

```

La sobrecarga del método Normpdf nos genera la densidad de probabilidad de un solo dato de entrada (Likelihood), ya que, a diferencia de su otra sobrecarga, esa sobrecarga nos ayuda a generar principalmente la campana de distribución normal. Igualmente requiere del promedio y varianza del grupo de datos al que pertenece nuestro dato de entrada.

Regression

```

115  /// <summary>
116  /// Regresa una tupla que contiene como primer elemento la ordenada al origen y como segundo elemento la pendiente
117  /// </summary>
118  /// <param name="input1"> Arreglo de datos para X </param>
119  /// <param name="input2"> Arreglo de datos para Y </param>
120  /// <returns> Tupla con ordenada y pendiente </returns>
121  public Tuple<double, double> Regression(double[] input1, double[] input2)
122  {
123      //Llamado de nuestra libreria para funciones básicas
124      Basics MATLAB = new Basics();
125
126      //Número de datos
127      int n = input1.Length;
128      //Para obtener la ordenada
129      //Obtenemos la serie de x al cuadrado
130      double[] XX = MATLAB.ProdArray(input1, input1);
131      //Sumatoria de x al cuadrado
132      double sumXX = MATLAB.SumList(XX);
133      //Sumatoria de y
134      double sumY = MATLAB.SumList(input2);
135      //Obtenemos la serie de xy
136      double[] XY = MATLAB.ProdArray(input1, input2);
137      //Sumatoria de xy
138      double sumXY = MATLAB.SumList(XY);
139      //Sumatoria de x
140      double sumX = MATLAB.SumList(input1);
141      //Obtenemos la ordenada al origen
142      double output1 = ((sumXX * sumY) - (sumXY * sumX)) / ((n * sumXX) - (Math.Pow(sumX, 2)));
143      //Para obtener la pendiente
144      //Promedio de x
145      double MeanX = Mean(input1);
146
147      //Promedio de y
148      double MeanY = Mean(input2);
149      //Obteniendo la pendiente
150      double output2 = (sumXY - (n * MeanX * MeanY)) / (sumXX - ((Math.Pow(sumX, 2) / n)));
151      output1 = Math.Round(output1, 4);
152      output2 = Math.Round(output2, 4);
153
154      return new Tuple<double, double>(output1, output2);
155  }
156 }
157

```

El método Regression, genera la regresión lineal de dos conjuntos de datos de entrada (X, Y), mediante el cálculo de las sumatorias de X^2 , Y, XY, X y el cálculo de los promedios de X y Y, obtenido una pendiente y una ordenada mediante dos ecuaciones. Finalmente devuelve una tupla de dos elementos, siendo la primera la ordenada al origen, y la segunda el valor de la pendiente.

Referencias:

<https://la.mathworks.com/help/matlab/ref/diff.html>

<https://la.mathworks.com/help/matlab/ref/linspace.html?lang=en>

https://la.mathworks.com/help/matlab/ref/mean.html?s_tid=doc_ta

https://la.mathworks.com/help/matlab/ref/var.html?s_tid=doc_ta#d123e1539620

https://la.mathworks.com/help/matlab/ref/std.html?s_tid=doc_ta

<https://la.mathworks.com/help/stats/normpdf.html>

https://es.wikipedia.org/wiki/Regresi%C3%B3n_lineal



Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ingeniería Mecánica e Industrial



Laboratorio de Cómputo de Ingeniería
Mecatrónica

Temas Selectos de Programación I (1964)

Profesor: Ing. Barrón Velázquez Gersain

Grupo 3

Práctica No. 1 Programación Concurrente

Brigada 1:

- Goytia González Jorge Hadamard
- Ordaz Lucas Efraín Uriel

Semestre 2021-1

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	3/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Práctica #1

Programación concurrente

1. Seguridad en la ejecución

	Peligro o Fuente de energía	Riesgo asociado
1	Tensión alterna	Electrocución

2. Objetivos de aprendizaje

OBJETIVO GENERAL: Entender el concepto de programación concurrente y ser capaz de estructurar el código para su implementación.

OBJETIVOS ESPECÍFICOS:

- Realizar un código que considere la implementación de hilos.
- Evaluar las diferencias de código de programas secuenciales en comparación con programas distribuidos.
- Identificar el código para sincronizar el acceso de hilos a objetos en un programa.

3. Introducción

Un programa concurrente es aquél en el que un cálculo puede avanzar sin esperar a que se completen otros cálculos, ejecuta procesos en paralelo. Su desarrollo fue motivado originalmente por el deseo de generar sistemas operativos más confiables, aunque actualmente su alcance ha llegado a muchas áreas de desarrollo tecnológico por los beneficios que ofrece respecto a la programación tradicional.

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	4/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Un algoritmo es un texto, o una representación gráfica, que describe sentencias que deben ejecutarse; un proceso, por otro lado, es un "texto en acción", una entidad dinámica generada por la ejecución de un algoritmo. Un subprocesso (también conocido como hilo) es un proceso definido por un solo flujo de control (esto puede ser un programa individual, una computadora o una red) y, en la mayoría de los casos, es un componente de un proceso superior en donde se ha generado.

Pueden existir múltiples hilos dentro de un proceso, ejecutándose simultáneamente y compartiendo recursos, como la memoria del sistema, mientras que diferentes procesos podrían no compartir este tipo de recursos.

La implementación de hilos difiere entre los distintos sistemas operativos. Los sistemas con un solo procesador generalmente implementan hilos mediante un mecanismo de división de tiempo: la unidad central de procesamiento (CPU) cambia entre diferentes hilos de software. Este cambio de contexto generalmente ocurre suficientemente rápido como para que los usuarios perciban que los hilos o tareas se ejecutan en paralelo. En un sistema multiprocesador o multinúcleo, varios hilos pueden ejecutarse en paralelo, con cada procesador o núcleo ejecutando un subprocesso separado simultáneamente.

El uso de hilos es un modelo generalizado de programación y ejecución que permite que existan múltiples subprocessos en el contexto de un proceso dado. Sin embargo, en un ambiente de programación donde se han generado varios hilos trabajando sobre elementos o controles en común del proceso padre, debe controlarse el acceso a estos recursos de manera secuencial, con el fin de evitar la generación de excepciones debido a que dichos elementos pueden estar siendo usados por otros hilos en un momento dado. Regularmente los distintos lenguajes de programación proveen mecanismos para evitar este tipo de problemas.

La programación concurrente tiene, entre otras, las siguientes ventajas:

Aumento del rendimiento del programa ya que la ejecución paralela de un programa concurrente permite que la cantidad de tareas completadas en un tiempo determinado aumente proporcionalmente a la cantidad de procesadores del sistema.

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	5/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Permitir que una aplicación siga respondiendo a la interacción con el usuario pues si en un programa el subprocesso de ejecución principal se bloquea en una tarea de ejecución prolongada, la aplicación completa puede aparecer como congelada. Al mover tales tareas de larga ejecución a un subprocesso que se ejecuta simultáneamente con el proceso de ejecución principal, es posible que la aplicación siga respondiendo a las entradas del usuario mientras ejecuta tareas en segundo plano.

Al usar subprocessos, una aplicación puede servir a múltiples clientes al mismo tiempo, utilizando menos recursos de los que necesitaría al usar múltiples copias del proceso total requerido.

Muchos lenguajes de programación soportan hilos. Los lenguajes C, C++, Java, Python y .NET Framework, por mencionar algunos, exponen los subprocessos a los desarrolladores y permiten diferentes grados de usabilidad. Algunos lenguajes están diseñados primordialmente para la programación paralela, especialmente mediante la unidad de procesamiento de gráficos (GPU, por sus siglas en inglés), sin requerir concurrencia o hilos.

4. Material y equipo



Computadora

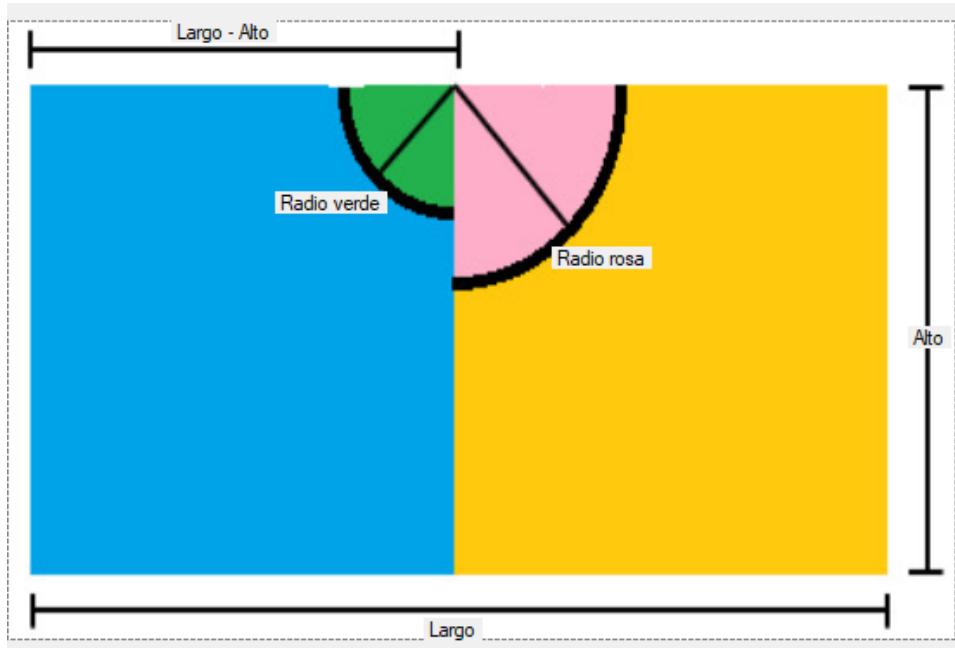
	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	6/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

5. Desarrollo

Para desarrollar esta práctica, utilizamos la palabra reservada TASK la cual en visual studio nos permite hacer diversas tareas de manera asíncrona, esto con el fin de reducir el tiempo de ejecución, sin embargo, debemos tener cuidado al momento de aplicar el TASK ya que, en ocasiones, las tareas que debe hacer al mismo tiempo son demasiadas que el programa se termina tardando más de lo que lo haría habitualmente.

Con el ejemplo visto en la clase, nos dimos una mejor idea de que entregar en la práctica

En esta práctica se realizó el cálculo del centroide de una figura irregular



El cálculo del centroide de esta figura, sigue el siguiente proceso: Calcular el centroide de cada figura y al final meter esos valores a una ecuación

$$X_C = \frac{\bar{x}A}{A}$$

$$Y_C = \frac{\bar{y}A}{A} =$$

Donde:

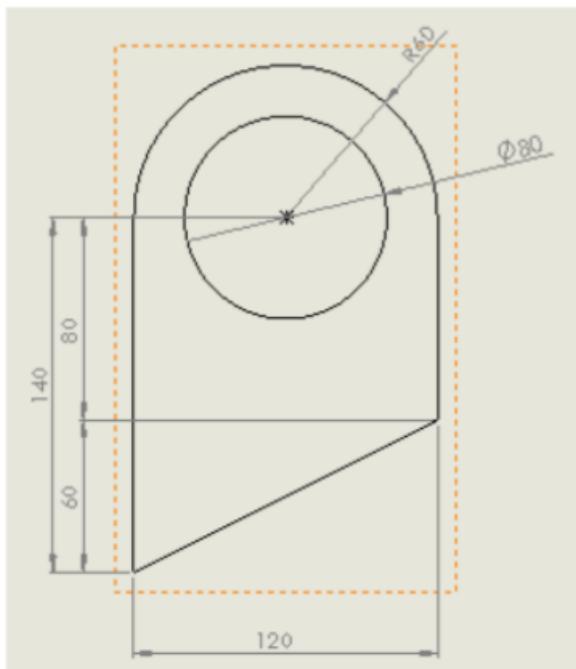
\bar{x} = el centro de la figura en el eje x que se esta estudiando

\bar{y} = el centro de la figura en el eje y que se esta estudiando

A = suma de las áreas de las respectivas figuras geométricas

Un ejemplo podría ser el siguiente

Obtener el centroide de la siguiente figura



Como observamos, esta figura se puede dividir en varias figuras, para poder estudiar una por una

Segmento	A (m^2)	x (m)	y (m)	xA (m^3)	yA (m^3)
1 rectángulo	$(80*120)=9600$	60	100	$(9600*60)=576000$	$(9600*100)=960000$
2 semicírculo	$(\pi(60)^2)/2=5654.88$	60	165.4	$(5654.88*60)=339292.8$	$(5654.88*165.4)=935317.15$
3 círculo	$(\pi(40)^2)=-5026.54$	60	140	$(-5026.54*60)=-301592.4$	$(-5026.54*140)=-703715.6$
4 triángulo	$(60*120)/2=3600$	40	20	$(3600*40)=144000$	$(3600*20)=72000$
Sumatorias	13828.34			757700.4	1263601.55

Se creó la siguiente tabla donde dividimos la imagen en 4 figuras geométricas, la cual tienen áreas distintas y se ubican los centro en x, y, además de la multiplicación de esos centros por su área

correspondiente

$$X_c = \frac{\bar{x}A}{A} = \frac{757700.4}{13828.34} = 54.8$$

$$Y_c = \frac{\bar{y}A}{A} = \frac{1263601.55}{13828.34} = 91.4$$

Con estos datos conocidos, sacamos la suma de todas las áreas, la multiplicación de xA y yA
Obteniendo nuestros valores de centros

Lo que buscamos en esta práctica, es que nos realice la suma todos los valores de área, x (centro), y (centro) y las multiplicaciones de xA y yA de manera asíncrona

Métodos simples que se utilizaron

```
//Métodos lineales/síncronos

/// <summary>
/// Suma de las areas obtenidas de las figuras
/// </summary>
/// <param name="ARect"> Area del rectangulo </param>
/// <param name="ACuad"> Area del cuadrado </param>
/// <param name="ACir1"> Area del cuarto de circulo1 </param>
/// <param name="ACir2"> Area del cuarto de circulo2 </param>
/// <returns> regresa la suma de todos los valores de entrada </returns>
3 referencias
public static double Sum(double ARect, double ACuad, double ACir1, double ACir2)
{
    double output;
    output = ARect + ACuad - ACir1 - ACir2;
    return output;
}
```

```
/// <summary>
/// Calcula el area del cuadrado
/// </summary>
/// <param name="b"> ancho de la figura </param>
/// <param name="a"> largo de la figura </param>
/// <returns></returns>
4 referencias
public static double Multiplicacion(double b, double a)
{
    double output;
    output = a * b;
    return output;
}

/// <summary>
/// Calcula el area de un cuarto de circulo
/// </summary>
/// <param name="r"> radio del circulo </param>
/// <returns> regresa el area del cuarto de circulo </returns>
1 referencia
public static double CuartCirA(double r)
{
    double output;
    output = (Math.PI * (r * r)) / 4;
    return output;
}
```

```
/// <summary>
/// Calcula el centroide del cuadrado o rectangulo, ya sea en x o
/// </summary>
/// <param name="m"> ancho de la figura </param>
/// <returns></returns>
2 referencias
public static double CcuadRec(double m)
{
    double output;
    output = m / 2;
    return output;
}

/// <summary>
/// Calcula el centroide del circulo tanto en x como en y
/// </summary>
/// <param name="r"> radio del circulo </param>
/// <returns> regresa el valor del centroide x o y </returns>
1 referencia
public static double CcuartCir(double r)
{
    double output;
    output = (4 * r) / (Math.PI * 3);
    return output;
}
```

```

    /// <summary>
    /// Calculo del Centroide total de la figura
    /// </summary>
    /// <param name="sumA"> Suma de todas las areas de las figuras </param>
    /// <param name="SumCxA"> suma de los centroides de la fiuras (puede ser en x o y) </param>
    /// <returns> regresa el valor del centroide la figura (ya sea x o y) </returns>
    2 referencias
public static double Centroide(double sumA, double SumCxA)
{
    double output;
    output = SumCxA / sumA;
    return output;
}

    /// <summary>
    /// Realiza todas las opeaciones del rectangulo (Area, calculo del centroide en 'x' y 'y',
    /// la multiplicacion del area por su centroide en 'x' y 'y'
    /// </summary>
    /// <param name="largo"> largo de la figura </param>
    /// <param name="ancho"> ancho de la figura </param>
    /// <returns> regresa el area y el valor de los centroide x y y </returns>
    1 referencia
public static Tuple<double, double, double> Rectangulo(double largo, double ancho)
{
    //Declaramos el uso de muetra libreria
    Basics MATLAB = new Basics();

    double area = MATLAB.Prod(largo, ancho);
    double Cx = MATLAB.Div(largo, 2);
    double Cy = MATLAB.Div(ancho, 2);
    double C_X = MATLAB.Prod(Cx, area);
    double C_Y = MATLAB.Prod(Cy, area);

    return new Tuple<double, double, double>(area, C_X, C_Y);
}

    /// <summary>
    /// Realiza todas las opeaciones del Cuatro de circulo (Area, calculo del centroide en 'x' y 'y',
    /// la multiplicacion del area por su centroide en 'x' y 'y'
    /// </summary>
    /// <param name="largo"> radio del cuadrado </param>
    /// <returns> regresa el area y el valor de los centroide x y y </returns>
    1 referencia
public static Tuple<double, double, double> Cuadrado(double largo)
{
    //Declaramos el uso de muetra libreria
    Basics MATLAB = new Basics();

    double area = MATLAB.Prod(largo, largo);
    double Cx = MATLAB.Div(largo, 2);
    double Cy = MATLAB.Div(largo, 2);
    double C_X = MATLAB.Prod(Cx, area);
    double C_Y = MATLAB.Prod(Cy, area);

    return new Tuple<double, double, double>(area, C_X, C_Y);
}

```

```

    /// <summary>
    /// Realiza todas las operaciones del Cuarto de circulo (Area, calculo del centroide en 'x' y 'y',
    /// la multiplicacion del area por su centroide en 'x' y 'y'
    /// </summary>
    /// <param name="radio"> radio del circulo </param>
    /// <returns> regresa el area y el valor de los centroide x y y </returns>
    1 referencia
    public static Tuple<double, double, double> Semicirculo1(double radio)
    {
        //Declaramos el uso de muetra libreria
        Basics MATLAB = new Basics();

        double area = (MATLAB.Prod(Math.PI, Math.Pow(radio, 2))) / 4;
        double C = MATLAB.Prod(4, radio) / MATLAB.Prod(3, Math.PI);
        double C_X = MATLAB.Prod(C, area);
        double C_Y = MATLAB.Prod(C, area);

        return new Tuple<double, double, double>(area, C_X, C_Y);
    }

```

```

    /// <summary>
    /// Realiza todas las operaciones del Cuarto de circulo (Area, calculo del centroide en 'x' y 'y',
    /// la multiplicacion del area por su centroide en 'x' y 'y'
    /// </summary>
    /// <param name="radio"> radio del circulo </param>
    /// <returns> regresa el area y el valor de los centroide x y y </returns>
    1 referencia
    public static Tuple<double, double, double> Semicirculo2(double radio)
    {
        //Declaramos el uso de muetra libreria
        Basics MATLAB = new Basics();

        double area = (MATLAB.Prod(Math.PI, Math.Pow(radio, 2))) / 4;
        double C = MATLAB.Prod(4, radio) / MATLAB.Prod(3, Math.PI);
        double C_X = MATLAB.Prod(C, area);
        double C_Y = MATLAB.Prod(C, area);

        return new Tuple<double, double, double>(area, C_X, C_Y);
    }
}

```

Por lo que nuestro programa nos quedó de la siguiente manera

```

    //Metodos concurrentes / asincronos
    /// <summary>
    /// Realiza todas las operaciones del cuadrado (Area, calculo del centroide en 'x' y 'y',
    /// la multiplicacion del area por su centroide en 'x' y 'y'
    /// </summary>
    /// <param name="ancho"> valor de entrada que es la resta del largo menos el alto </param>
    /// <param name="largo"> valor de entrada del largo de la figura completa </param>
    /// <returns></returns>
    2 referencias
    public static async Task<(double, double, double)> CuadradoRec(double ancho, double largo)
    {
        double ACuadrado = Multiplicacion(ancho, largo);
        double CXCuadrado = CcuadRec(ancho);
        double CYCuadrado = CcuadRec(largo);
        double AxCXCuadrado = Multiplicacion(ACuadrado, CXCuadrado);
        double AxCYCuadrado = Multiplicacion(ACuadrado, CYCuadrado);

        (double, double, double) output = (ACuadrado, AxCXCuadrado, AxCYCuadrado);
        return await Task.FromResult(output);
    }

```

Como vemos en nuestra figura irregular, está compuesta por rectángulos, por lo que podemos hacer un

método para estos dos, en el que vamos a obtener el área del cuadrado, el centro del cuadrado en x, el centro del cuadrado en y, la multiplicación de xA y Ya, y al final devolveremos una tupla con los tres valores de interés que son área, xA y yA. En este método lo declaramos como un TASK al principio y al momento de regresar debemos esperar a que acabe todos los cálculos, para terminar el método

```

1  /// <summary>
2  /// realiza todas las operaciones del circulo (Area, calculo del centroide en 'x' y 'y',
3  /// la multiplicacion del area por su centroide en 'x' y 'y'
4  /// </summary>
5  /// <param name="r"> radio del circulo </param>
6  /// <returns> Regresa tres parametros, el area del circulo, multiplicain del centroide de x y y </returns>
7  2 referencias
8  public static async Task<(double, double, double)> CuartCir(double r)
9  {
10     double AcuarCir = CuartCirA(r);
11     double CXcuartCir = CcuartCir(r);
12     double AxCuartCir = Multiplicacion(AcuarCir, CXcuartCir);
13
14     (double, double, double) output = (AcuarCir, AxCuartCir, AxCuartCir);
15     return await Task.FromResult(output);
16 }
```

El siguiente método asíncrono que realizará es para los círculos que se encuentran en la figura, como vemos, de igual forma que el anterior, este método será asíncrono, por lo que lo iniciamos con TASK y al final del método nos regresará una tupla con tres elementos. Dentro del método se realizarán los cálculos del área del cuarto círculo, el centro del cuarto de círculo en x y y, además de la multiplicación de xA y ya, con los métodos simples creados con la intención reducir código en este método además de darle más orden a nuestro programa

En nuestra parte “principal” con los métodos vistos anteriormente, tendremos lo siguiente

```

1  1 using System;
2  2 using System.Diagnostics;
3  3 using System.Threading.Tasks;
4  4 using System.Windows.Forms;
5  5 using MATLAB_Library;
6
7  6 namespace Practical1__Entregable_
8  7 {
9  8     4 referencias
10  9     public partial class Form1 : Form
11 10     {
12 11         1 referencia
13 12         public Form1()
14 13         {
15 14             InitializeComponent();
16 15             this.CenterToScreen();
17 16         }
18
19         1 referencia
20         private void BtnExit_Click(object sender, EventArgs e)
21         {
22             Application.Exit();
23         }
24
25         1 referencia
26         private async void BtnCalcular_Click(object sender, EventArgs e)
27         {
28             try
29             {
30                 //Declaracion de los relojes para el conteo de tiempo
31                 Stopwatch tiempo1 = new Stopwatch();
32                 Stopwatch tiempo2 = new Stopwatch();
33             }
34         }
35     }
36 }
```

De la línea 1-21, es código que se crea de manera automática al momento de hacer un form

Dentro d nuestro botón “BttnCalcular_Click”, va a calcular el valor del centroide de toda la figura.

Las siguientes instrucciones las meteremos dentro de un try. Empezamos dos variables que medirán el tiempo de ejecución en milisegundos

```
//iniciando el contador de timepol para la programacion concurrente
tiempol.Start();

//Asignacion de valores contenidos en los texbox a las variables
double alto = Convert.ToDouble(textBoxAlto.Text);
double largo = Convert.ToDouble(textBoxLargo.Text);
double RCirv = Convert.ToDouble(textBoxRad1.Text);
double RCirr = Convert.ToDouble(textBoxRad2.Text);

//Llamadas de las funciones asincronas mediante tuplas
(double, double, double) OpRectangulo = await CuadradoRec(largo - alto, alto);
(double, double, double) OpCuadrado = await CuadradoRec(alto, alto);
(double, double, double) OpCirculo1 = await CuartCir(RCirv);
(double, double, double) OpCirculo2 = await CuartCir(RCirr);

//Llamadas de funciones sincronas para suma de productos y suma de areas
double SumAreas = Sum(OpRectangulo.Item1, OpCuadrado.Item1, OpCirculo1.Item1, OpCirculo2.Item1);
double SumAxX = Sum(OpRectangulo.Item2, OpCuadrado.Item2, OpCirculo1.Item2, OpCirculo2.Item2);
double SumAxY = Sum(OpRectangulo.Item3, OpCuadrado.Item3, OpCirculo1.Item3, OpCirculo2.Item3);

//Llamada de funciones asincronas para el calculo de centroides
double CentroideXT = Centroide(SumAreas, SumAxX);
double CentroideYT = Centroide(SumAreas, SumAxY);

//se detiene el tiempo de programacion concurrente
tiempol.Stop();

-----
//Inicio de tiempo para porgramacion lineal
tiempo2.Start();
```

Iniciaremos el conteo de una variable de tiempo previamente declarada. Luego, los valores que recibimos en nuestros textbox los vamos a convertir a valores de tipo double

Con estos datos, llamamos a nuestros métodos, creando variables que van a guardar las tuplas que nos regresen, como vemos en la figura, tenemos dos rectángulos y dos cuartos de círculo, porque tenemos que crear cuatro variables de tipo tupla y llamar a cada método dos veces, ya que son distintas medidas

Con estos datos obtenidos, llamamos al método sum, que nos entregara la suma de las áreas de todas las figuras, de todos los valores de xA y yA, guardándolos en tres variables de tipo double. Al final llamaremos a la función “Centroide” que nos entregara el centroide en x y y de la figura total, y le estaremos mandando los datos que se obtuvieron de la suma de las áreas de todas las figuras, de todos los valores de xA y yA. Y detendrá el tiempo de ejecución de nuestra primera variable de tiempo

Ahora se va a iniciar la segunda variable de tiempo y lo que medirá, será el tiempo de ejecución del mismo programa, pero sin utilizar el TASK

```

//-->
//Inicio de tiempo para programacion lineal
tiempo2.Start();

//Declaracion de variables para almacenamiento de tuplas
var fig1 = Cuadrado(alto);
var fig2 = Rectangulo(largo - alto, alto);
var fig3 = Semicirculo1(RCirv);
var fig4 = Semicirculo2(RCirr);

//Sumas correspondientes a los valores devueltos por las tuplas
double SA = fig1.Item1 + fig2.Item1 - fig3.Item1 - fig4.Item1;
double SXxA = fig1.Item2 + fig2.Item2 - fig3.Item2 - fig4.Item2;
double SYxA = fig1.Item3 + fig2.Item3 - fig3.Item3 - fig4.Item3;

//Calculo de centroides totales
double CX = SXxA / SA;
double CY = SYxA / SA;

//Se detiene el tiempo de la programacion lineal
tiempo2.Stop();

//Mostramos resultados en el form
txtShow.Text = "Valores concurrentes: " + "\n" + "Centroide X: " + CentroideXT.ToString() +
txtShow.Show();

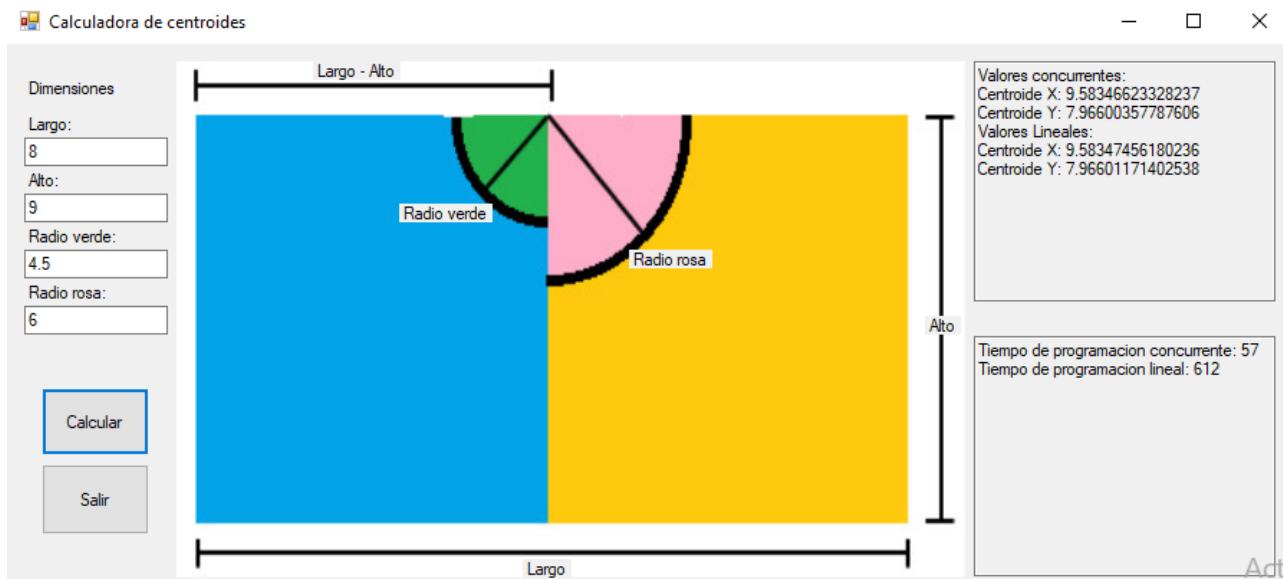
txtShow2.Text = "Tiempo de programacion concurrente: " + tiempo1.ElapsedMilliseconds.ToString() + "\n \n" + "Tiempo de programacion lineal: " + tiempo2.ElapsedMilliseconds.ToString();
txtShow2.Show();
}

catch
}

```

Como vemos, este procedimiento es mucho mas corto en cuestión de código, al final nos imprimirá el valor de los centroides y el tiempo de ejecución de nuestra variable de tiempo dos

Por lo que nuestro programa, probándolo, quedaría de la siguiente manera



Como vemos en la parte derecha, el tiempo de ejecución con TASK es mucho menor a hacerlo de manera habitual, en cierto punto de la práctica, agregamos más TASK a ciertas funciones por lo que terminaba dando un tiempo mucho mayor a la programación habitual, debemos tener mucho cuidado en ese aspecto porque si queremos hacer una tarea de manera más rápida podríamos terminar empeorando, el profesor nos comentó que esto se debe a que nuestro equipo no cuenta con muchos núcleos para poder hacer varias tareas a la vez

6. Bibliografía

- BELL, Douglas y PARR, Mike. **C# para estudiantes.** México, Pearson, 2010.
- BRINCH, Hansen. **The origin of concurrent programming.** New York, Springer, 2002.
- RAYNAL, Michel. **Concurrent Programming: Algorithms, Principles, and Foundations.** New York, Springer, 2013.
- LAAKMANN McDOWELL, Gayle. **Cracking the Coding Interview: 189 Programming Questions and Solutions.** Palo Alto, CA., CareerCup, 2016.



Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ingeniería Mecánica e Industrial



Laboratorio de Cómputo de Ingeniería
Mecatrónica

Temas Selectos de Programación I (1964)

Profesor: Ing. Barrón Velázquez Gersain

Grupo 3

Práctica No. 2
Implementación de una Interfaz de
Programación de Aplicaciones
(API)

Brigada 1:

- Goytia González Jorge Hadamard
- Ordaz Lucas Efraín Uriel

Semestre 2021-1

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	7/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Práctica #2

Implementación de una Interfaz de Programación de Aplicaciones (API)

1. Seguridad en la ejecución

	Peligro o Fuente de energía	Riesgo asociado
1	Tensión alterna	Electrocución

2. Objetivos de aprendizaje

OBJETIVO GENERAL: El alumno comprenderá el concepto de una interfaz de programación de aplicaciones (API) y lo integrará a un proyecto que aproveche algunas de las características ofrecidas por una API en particular.

OBJETIVOS ESPECÍFICOS:

- Implementar una API
- Entender las ventajas de ocupar una API respecto a la generación del código específico para atender un problema en particular.

3. Introducción

Una API es un conjunto de herramientas, programas y protocolos que se integran a un proyecto de software con la intención de aprovechar las ventajas de comunicarse con productos propios o de terceros.

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	8/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Generalmente el uso de una API permite ahorrar tiempo y costo de desarrollo debido a que las aplicaciones se ajustan en variadas ocasiones a las necesidades de los negocios e instituciones en general.

Un ejemplo de una API que se ocupa de manera muy recurrente es la de Google Maps que permite trazar rutas y estimar tiempos de traslado por parte del usuario dependiendo de la geolocalización.

Generalmente una API requiere de un registro por parte del programador para poder ocuparla y dependiendo de la licencia hasta un cobro por derecho de uso que establece los límites de uso. La manera en que se puede ocupar una API es generando una cadena de código HASH que se trata de una secuencia aleatoria de caracteres de una longitud definida y única que permite identificar al usuario programador. Si se llegase a perder dicho HASH ya no se podría acceder con el proveedor y la API deja de funcionar correctamente.

4. Material y equipo



Computadora

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	9/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

5. Desarrollo

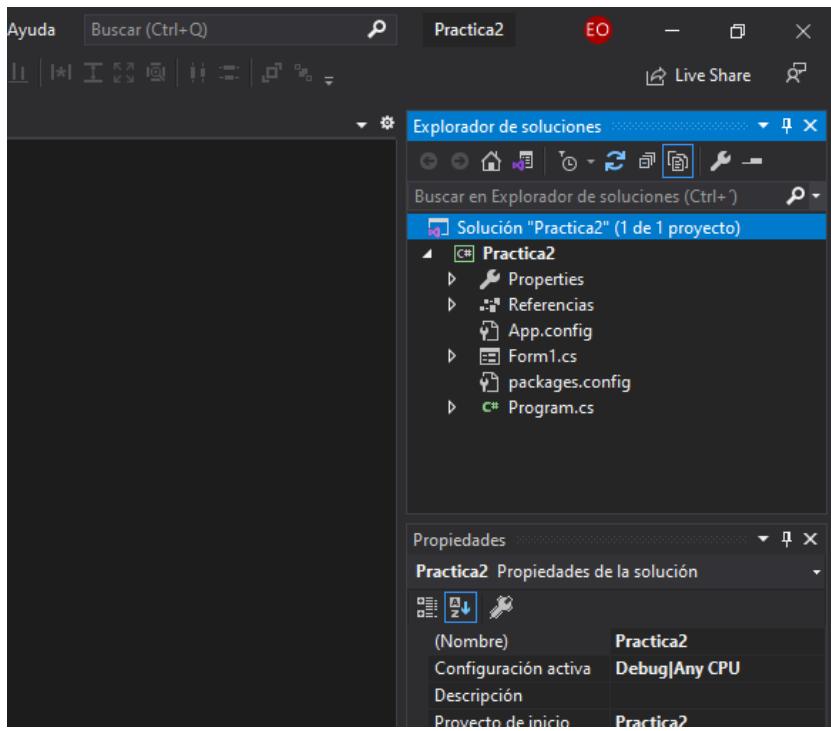
Para el desarrollo de la práctica se utilizó la API de Excel

The screenshot shows the homepage of the SpreadsheetLight website. At the top, there's a navigation bar with links like 'YouTube', 'Portal del Becario...', 'Facultad de Ingeniería...', 'EDUCAFI PLUS: Áre...', 'AVI', 'Facebook', 'fichas apa', 'EDUCAFI MS', and 'Spotify'. Below the navigation is the main header 'SpreadsheetLight' with a green circular icon containing a grid pattern. A green banner below the header contains the text 'Notificaciones de lanzamiento'. The main content area features a large image of a SpreadsheetLight book cover. To the right of the book, there's a testimonial quote from 'Josh Hill': 'Al igual que la magia, funcionó de inmediato, fuera de la caja, sin muss, sin alboroto y sin dolor de cabeza.' Below the testimonial is a link 'Activar Window'. The overall theme is green and professional.

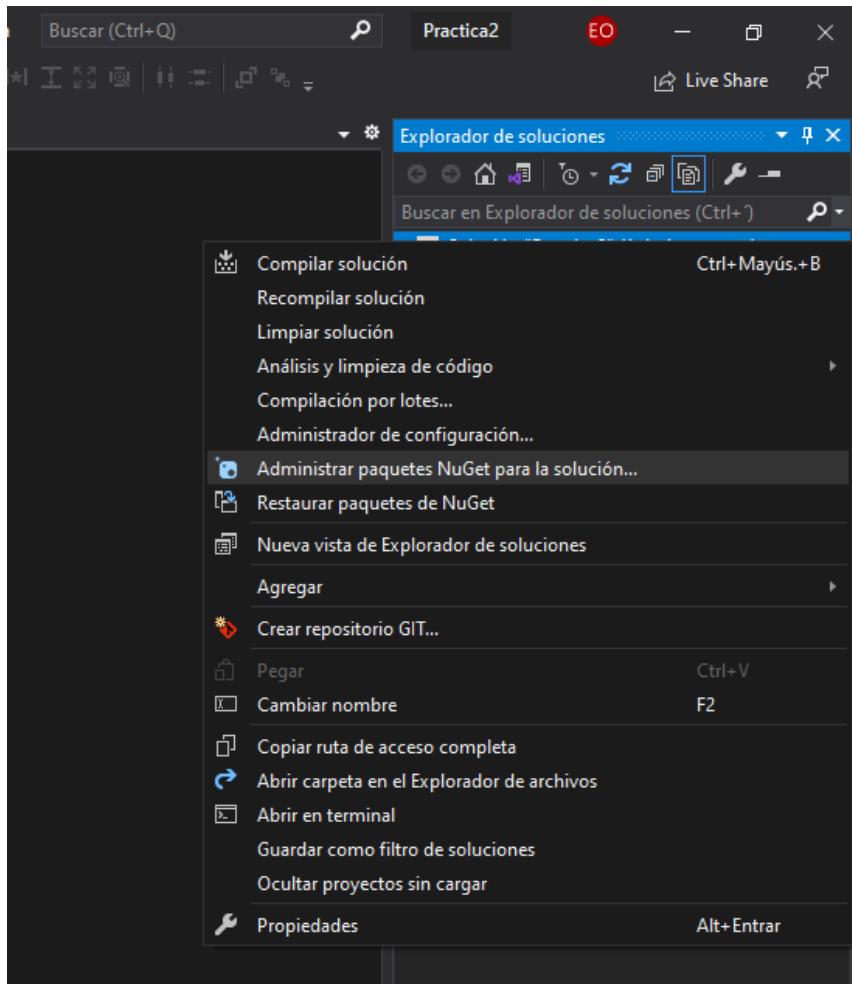
[Biblioteca de hojas de cálculo de código abierto compatible con Microsoft Excel 2007/2010/2013 y LibreOffice Calc](#)

La cual tiene aplicación en nuestro programa de visual studio

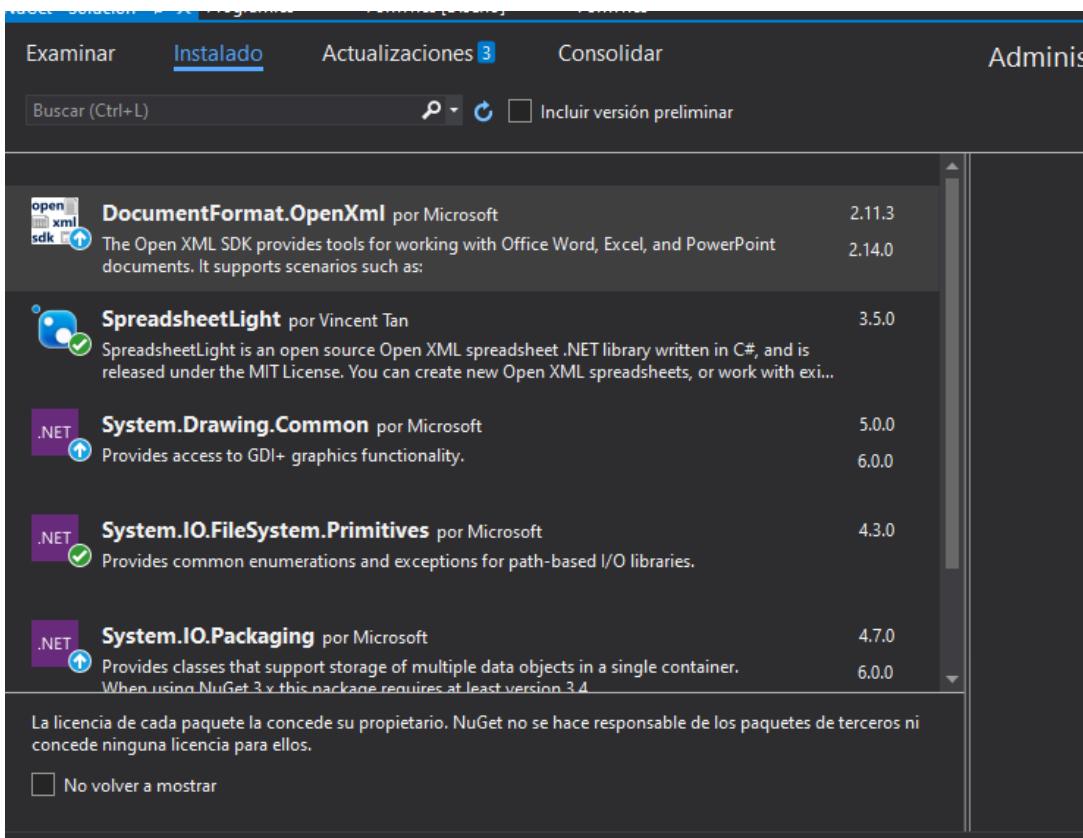
Para poder utilizar esta API debemos de ir a la parte derecha de nuestro programa de visual studio y en la parte de nuestro proyecto, damos click derecho



Y seleccionamos la opción de “administrar paquetes Nugget para la solución”



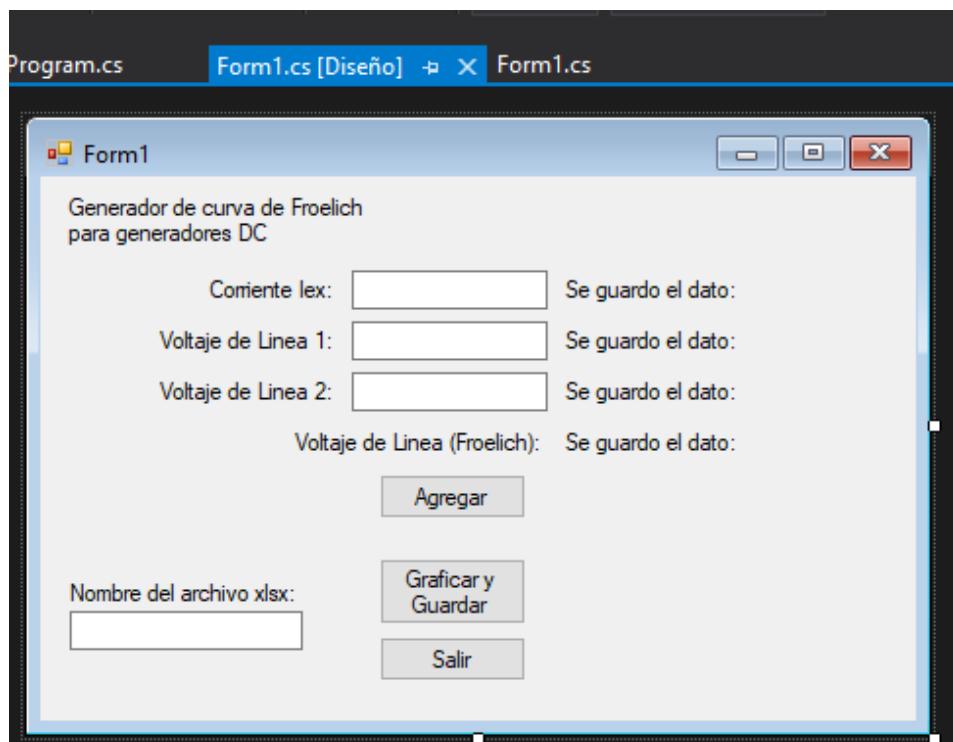
Y nos va a desplegar la siguiente parte



Y en examinar, buscamos, “DocumentFormat.OpenXml”

Y de esta forma ya podemos trabajar con la API desarrollada por una persona o grupo ajeno a Microsoft

Ahora lo que realizamos en esta práctica fue crear un documento de Excel, desde nuestro form el cual va a guardar ciertos datos de entrada, en este caso va a registrar diversas corrientes y distintos valores de voltaje de línea 1 y línea 2, y va a calcular el voltaje de Froelich



Nos basamos un poco en el siguiente ejemplo que nos brinda la pagina

<https://spreadsheetlight.com/downloads/samplecode/ChartsLine.cs>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using DocumentFormat.OpenXml;
using DocumentFormat.OpenXml.Spreadsheet;
using SpreadsheetLight;
using SpreadsheetLight.Charts;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            SLDocument sl = new SLDocument();

            sl.SetCellValue("C2", "Apple");
            sl.SetCellValue("D2", "Banana");
            sl.SetCellValue("E2", "Cherry");
            sl.SetCellValue("F2", "Durian");
            sl.SetCellValue("G2", "Elderberry");
            sl.SetCellValue("B3", "North");
            sl.SetCellValue("B4", "South");
            sl.SetCellValue("B5", "East");
            sl.SetCellValue("B6", "West");

            Random rand = new Random();
            for (int i = 3; i <= 6; ++i)
            {
                for (int j = 3; j <= 7; ++j)
                {
                    sl.SetCellValue(i, j, 9000 * rand.NextDouble() + 1000);
                }
            }

            double fChartHeight = 15.0;
            double fChartWidth = 7.5;

            SLChart chart;

            chart = sl.CreateChart("B2", "G6");
            chart.SetChartType(SLLineChartType.Line);
            chart.SetChartPosition(1, 9, 1 + fChartHeight, 9 + fChartWidth);
            sl.InsertChart(chart);

            chart = sl.CreateChart("B2", "G6");
            chart.SetChartType(SLLineChartType.StackedLineWithMarkers);
            chart.SetChartStyle(SLLineStyle.Style5);
            chart.SetChartPosition(7, 1, 7 + fChartHeight, 1 + fChartWidth);
            sl.InsertChart(chart);

            chart = sl.CreateChart("B2", "G6");
            chart.SetChartType(SLLineChartType.Line3D);
            chart.SetChartStyle(SLLineStyle.Style30);
            chart.SetChartPosition(16, 9, 16 + fChartHeight, 9 + fChartWidth);
            sl.InsertChart(chart);
```

```
    sl.SaveAs("ChartsLine.xlsx");

    Console.WriteLine("End of program");
    Console.ReadLine();
}

}
```

Este programa lo que hace es crear una hoja de cálculo en la que va a crear las celdas con títulos en las columnas y nos mostrara el ancho y el largo de cada elemento registrado en cada columna, que como se puede observar, son frutas

Por lo que nuestro programa nos quedó de la siguiente manera

```
caz Practica2.cs BttnExit_Click

using System;
using System.Windows.Forms;
using SpreadsheetLight;
using SpreadsheetLight.Charts;
using MATLAB_Library;

namespace Practica2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.CenterToScreen();

            //Creamos las cabeceras para nuestros datos
            sl.SetCellValue("C2", "Iex[A]");
            sl.SetCellValue("D2", "VL (Calculado) [V]");
            sl.SetCellValue("E2", "VL (Iex ->) [V]");
            sl.SetCellValue("F2", "VL (Iex ->) [V]");
        }

        //Hacemos el llamado de la API/libreria del MIT para el manejo de archivos en Excel
        readonly SLDocument sl = new SLDocument();
        //Nuestra libreria
        readonly Basics MATLAB = new Basics();

        private void BttnExit_Click(object sender, EventArgs e)
        {

```

Primero creamos nuestros títulos de las columnas, que serán 4. Despu s hacemos el llamado de la API por medio del comando “SLDocument”

Hasta este punto se creó una hoja en Excel de cuatro columnas con su respectivo título

```

        {
            //Cerramos la aplicacion
            Application.Exit();
        }

    1 referencia
    private void BttnGG_Click(object sender, EventArgs e)
    {
        try
        {
            string nombre = txtBoxNombre.Text;
            //Dimensiones de aspecto para el gráfico
            double fChartHeight = 15.0;
            double fChartWidth = 7;

            //Creacion del objeto grafica de excel
            SLChart chart;

            //Modificaciones de los parametros de nuestro grafico a generar
            chart = sl.CreateChart("C2", "D15");
            chart.SetChartType(SLLineChartType.StackedLineWithMarkers);
            chart.SetChartStyle(SLChartStyle.Style5);
            chart.SetChartPosition(1, 9, 1 + fChartHeight, 9 + fChartWidth);
            sl.InsertChart(chart);

            //Guardado y nombrado de nuestro Excel creado
            sl.SaveAs(nombre + ".xlsx");

            MessageBox.Show("Archivo creado y guardado con exito");
        }
        catch
        {
    
```

En nuestro boton de graficar y guardar, vamos a guardar el nombre que el ususario puede ponerle al archivo excel en un variable de tipo string

Posteriormente vamos a darle ciertas dimensiones al grafico que se nos va a mostrar, este valor puede variar depende de nosotros

Creamos la grafica con la funcion SLChart y posteriormente se le van a asignar, los datos que va a tomar el grafico, es decir toma desde el primer elemento de x (C2) hasta el ultimo valor de y (D15). Despues le indicamos que tipo de grafica sera (lineal), luego el estilo de grafica y al final lo insrtamos en nuestro programa de excel

Guardamos el archivo y mandamos un mensaje de que el archivo se guardo con exito

```

        {
    }
}

/*
//Arreglos de datos leidos en prueba de laboratorio
double[] Iex = new double[] { .48, .50, .60, .70, .80, .90, 1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6 };
double[] VL1 = new double[] { 49, 50, 57, 65, 71, 80, 87, 98, 103, 108, 115, 118, 125 };
double[] VL2 = new double[] { 52, 56, 64, 71, 78, 86, 94, 99, 105, 111, 115, 118, 125 };
*/
//Arreglos de datos leidos en prueba de laboratorio
readonly static double[] Iex = new double[13];
readonly static double[] VL1 = new double[13];
readonly static double[] VL2 = new double[13];

//Índices para el manejo de la hoja de cálculo
readonly int jColumn1 = 3;
readonly int jColumn4 = 4;
readonly int jColumn2 = 5;
readonly int jColumn3 = 6;
int iRow = 3;
int i = 0;

referencia
private void BtnAdd_Click(object sender, EventArgs e)
{

```

Las partes comentadas, fueron utilizadas para la maquetación de esta práctica, en la entrega los valores serán metidos por el usuario

Para ir llenando nuestra hoja de Excel, se realizó lo siguiente :

Se declararon tres arreglos que va a guardar el valor de la corriente, el voltaje 1 y el voltaje 2

Se declararon tres valores enteros, que nos serviría para ubicarnos en la columna en Excel
Iniciamos un contador en 0, que nos serviría para llenar todas las columnas una por una

```
try
{
    if(i == Iex.Length)
    {
        MessageBox.Show("Ya se han tomado todas las lecturas");
    }
    else
    {
        //Conversion de los datos
        double IexD = Convert.ToDouble(textBoxIex.Text);
        double VL1D = Convert.ToDouble(textBoxVL1.Text);
        double VL2D = Convert.ToDouble(textBoxVL2.Text);
        //Comenzamos a agregar los datos en los arreglos
        Iex[i] = IexD;
        VL1[i] = VL1D;
        VL2[i] = VL2D;
        //Insertamos la lectura de corriente
        sl.SetCellValue(iRow, jColumn1, Iex[i]);
        //Insertamos las lecturas de Voltaje con corriente incremental
        sl.SetCellValue(iRow, jColumn2, VL1[i]);
        //Insertamos las lecturas de Voltaje con corriente decremental
        sl.SetCellValue(iRow, jColumn3, VL2[i]);
        //Insertamos los valores calculados
        double Calculo = Froelich(IexD);
        sl.SetCellValue(iRow, jColumn4, Calculo);
        iRow++;
        i++;
        //Se muestran los datos agregados
        label10.Text = Convert.ToString(IexD);
        label11.Text = Convert.ToString(VL1D);
```

Convertimos los datos que está guardando el usuario en variables de tipo double

Las guardamos en nuestro primer elemento de nuestros arreglos previamente creados

Metemos esos datos en Excel con “SetCellValue” en la que le estamos indicando el número de la columna y la fila donde lo debe guardar

Para nuestro último valor en la columna, llamamos a nuestra función “Froelich”, el cual veremos más adelante, y de igual forma lo estaremos metiendo a nuestra hoja de Excel con el método previamente explicado.

Ahora aumentamos el valor de nuestro contador y mostraremos los datos agregados

```

        label12.Text = Convert.ToString(VL2D);
        label13.Text = Convert.ToString(Calculo);
        //Limpiamos los campos
        txtBoxIex.Clear();
        txtBoxVL1.Clear();
        txtBoxVL2.Clear();
    }

}

catch
{
}

}

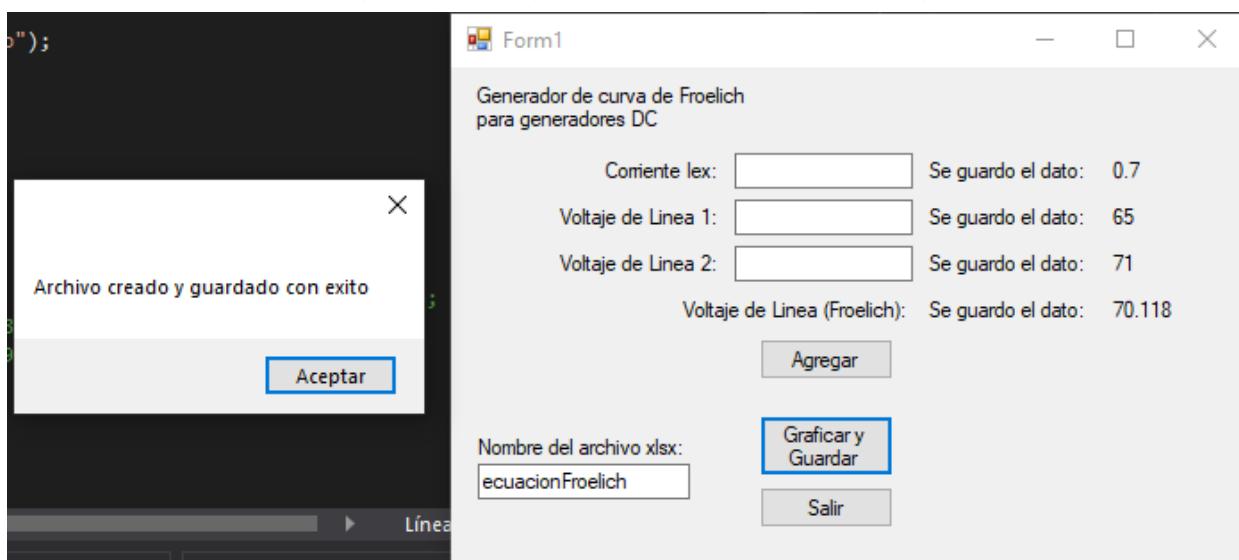
/// <summary>
/// Calculamos el voltaje de linea con la ecuacion de Froelich
/// </summary>
/// <param name="Iex"> Corriente leida </param>
/// <returns></returns>
1 referencia
private double Froelich(double Iex)
{
    double output;
    output = ((MATLAB.Prod(2.542, Iex)) / (MATLAB.Sum(2.489, Iex))) * (125.664);
    output = Math.Round(output, 3);
    return output;
}
}

```

Después de mostrar los datos, limpiamos los texbox

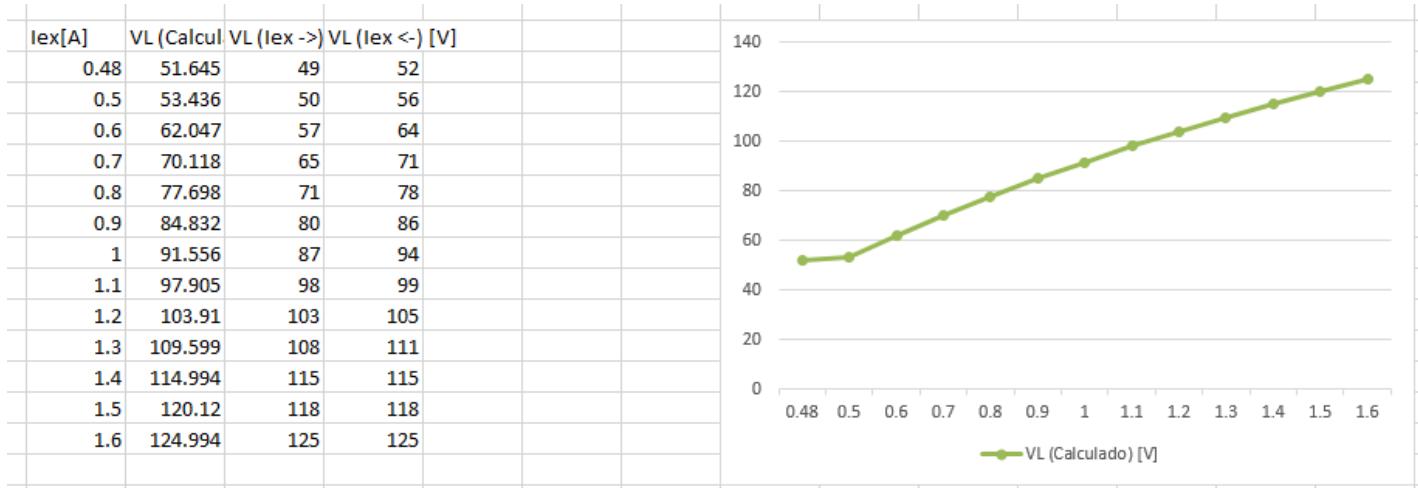
Para nuestro método “Froelich”, vamos a recibir un solo parámetro, que será la corriente
Lo que hará este método es $((2.542 * \text{corriente de entrada}) / (\text{suma de } 2.489 + \text{la corriente}))$ este resultado por (125.664) y al final redondeamos a 3 decimales y devolvemos el valor

Y al final tendremos el siguiente arreglo de datos en nuestro programa



Ejecutando nuestro programa y después de meter algunos datos, le colocamos un nombre al archivo y le dimos al botón de guardar y graficar

Para buscar nuestro archivo, nos vamos a las carpetas que se crean en el proyecto /bin/debug y tiene que aparecer el nombre del archivo que le colocamos



Y como vemos, tenemos el archivo deseado con los valores donde deben ir y su grafica correspondiente

6. Bibliografía

- JACOBSON,Daniel, BRAIL,Greg y WOODS, Dan. **APIs A Strategy Guide**.USA, O'Reilly,2012.
- HOLLANDER, Ami (Ed.). **Learning Firebase**. <https://riptutorial.com/Download.firebaseio.pdf>, 20-06-2019.



Universidad Nacional Autónoma de México



Facultad de Ingeniería

División de Ingeniería Mecánica e Industrial

Laboratorio de Cómputo de Ingeniería
Mecatrónica

Temas Selectos de Programación I (1964)

Profesor: Ing. Barrón Velázquez Gersain

Grupo 3

Práctica No. 3

Base de Datos: Manejo y conexión

Brigada 1:

- Goytia González Jorge Hadamard
- Ordaz Lucas Efraín Uriel

Semestre 2021-1

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	10/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Práctica #3

Base de Datos: Manejo y conexión

1. Seguridad en la ejecución

	Peligro o Fuente de energía	Riesgo asociado
1	Tensión alterna	Electrocución

2. Objetivos de aprendizaje

OBJETIVO GENERAL: El alumno comprenderá el manejo de bases de datos y realizará una conexión con un equipo para modificar una en particular.

OBJETIVOS ESPECÍFICOS:

- Realizar el código para las operaciones básicas en una base de datos.
- Desarrollar un código que utilice los elementos disponibles en una base de datos local y muestre algunos tipos de datos que contenga un registro solicitado.

3. Introducción

Una base de datos es un conjunto de información estructurada bajo un contexto particular que puede ser accedida desde una computadora, dispositivo móvil o cualquier equipo que requiera manejarla.

En general las bases de datos pueden clasificarse como relacionales y no relacionales.

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84	
		Versión:	01	
		Página	11/21	
		Sección ISO	8.3	
		Fecha de emisión	24 de enero de 2020	
Facultad de Ingeniería		Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada				

Las bases de datos relacionales son aquéllas que se pueden equiparar con una tabla de Excel, donde los datos son básicamente integrados a celdas y cada columna representa un tipo de dato distinto, mientras que cada fila distingue a un registro.

Por ejemplo: Una tienda de mascotas donde se almacenan los datos de cada animal como nombre, especie, fecha de nacimiento y vacunas. La información por cada mascota la podemos almacenar en una tabla como se muestra a continuación:

Nombre	Especie	Fecha de nacimiento	Vacunas
Fígaro	Gato	14/noviembre/2017	Parásitos, Viral
Cofi	Perro	18/Enero/2016	Viral, Triple

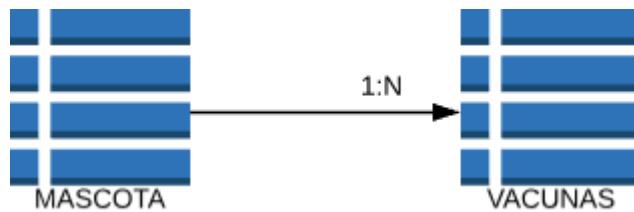
De la tabla anterior se puede observar que los datos que se insertan en Vacunas pueden ser múltiples por lo que en las bases relationales se hace la separación en más tablas y se ocupa un identificador (ID) para reconocer la mascota de la que se trata.

ID	Nombre	Especie	Fecha de nacimiento
100	Fígaro	Gato	14/noviembre/2017
200	Cofi	Perro	18/Enero/2016

ID_Vacunas	Vacunas
10	Parásitos
20	Viral
30	Triple

Para poder relacionar ambas tablas se acostumbra a utilizar un esquema que nos permita visualizar el comportamiento de ambas a la vez:

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	12/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			



Para manipular una base de datos relacional se ocupa, generalmente, lenguaje SQL que contiene instrucciones definidas para su manipulación como puede ser: eliminar tablas, ingresar datos a la tabla, eliminar datos de la tabla y hacer cambios en los datos de la tabla.

Las bases de datos no relacionales son generalmente representadas en un esquema de árbol lineal donde el ID de cada objeto es el que nos permite distinguirlo de los demás. En este caso, a diferencia del otro tipo de bases de datos, todos los objetos son independientes y no se pueden agrupar de acuerdo con características en particular, lo que generalmente se conoce como clusters.

4. Material y equipo



Computadora

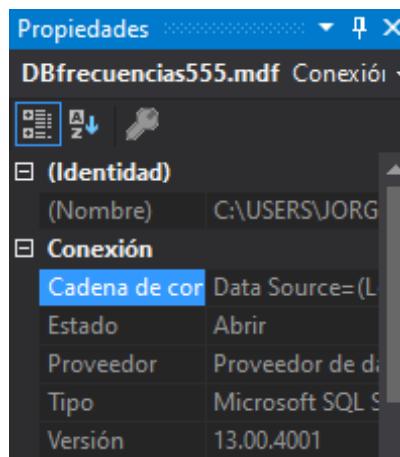
	Manual de prácticas de Temas Selectos de Programación I y II	
	Código:	MADO-84
	Versión:	01
	Página	13/21
	Sección ISO	8.3
	Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica	
La impresión de este documento es una copia no controlada		

5. Desarrollo

Para desarrollar esta práctica buscamos documentación acerca de SQLServer, principalmente la sintaxis de las instrucciones que maneja y el como realizar adecuadamente las consultas a nuestras bases de datos.

Como bien comprendimos durante la indagación del tema, no podemos realizar de manera adecuada el manejo de la base de datos, sin antes tener el acceso a la misma, por lo que se recomienda utilizar el Wizard de Visual Studio para agregar la base de datos al proyecto de forma adecuada.

Como punto de partida, se requiere de una cadena de conexión, que se nos da en la ventana de propiedades al seleccionar nuestra BD local para el proyecto.



Dicha cadena de conexión debe colocarse en el archivo “App.config”, asignándose un nombre a la conexión para su uso tal como se muestra, esta dirección siempre debe ser diferente si se abre el proyecto en un equipo diferente.

```

App.config  Form1.cs [Diseño]  Form1.cs 
 1  <?xml version="1.0" encoding="utf-8" ?>
 2  <configuration>
 3   <configSections>
 4   </configSections>
 5   <connectionStrings>
 6     <add name="conexion" connectionString="Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\jorge\Documents\Visual Studio 2019\Projects\Practica1\Practica1\bin\Debug\DBfrecuencias555.mdf;Integrated Security=True;Connect Timeout=30;" providerName="System.Data.SqlClient" />
 7   </connectionStrings>
 8   <startup>
 9     <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
10   </startup>
11 </configuration>
12 
```

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	13/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Una vez que se realiza esto, podemos trabajar con la base de datos local, para esta práctica se emplearon diversas instrucciones de SQL Server, siendo los comandos de “INSERT INTO” (insertar en), “CREATE TABLE” (crear tabla), “DROP TABLE” (eliminar tabla) y “TRUNCATE TABLE” (limpiar tabla) de la variedad que existen, tanto para la base en sí misma, como para las tablas. Teniendo los siguientes métodos:

```

58  /// <summary>
59  /// Método para probar la conexión con nuestra base de datos
60  /// </summary>
61  public void Conexion()
62  {
63      try
64      {
65          //Instancia de la conexión para la base de datos
66          //NOTA: La cadena de conexión debe cambiarse en App.config caso de que se ejecute en otro equipo
67          //La cadena de conexión "conexion" debe ser la nos brinda la ventana de propiedades de la base de datos
68          using (SqlConnection conexion = new SqlConnection(ConfigurationManager.ConnectionStrings["conexion"].ConnectionString))
69          {
70              //Se abre la conexión
71              conexion.Open();
72              //Mensaje de conexión exitosa
73              MessageBox.Show("Prueba de conexión realizada con éxito");
74              //Se cierra la conexión
75              conexion.Close();
76          }
77      }
78      catch (Exception excepcion)
79      {
80          //Mensaje de error en caso de que algún problema nos impida la conexión o un comando mal ejecutado
81          MessageBox.Show(excepcion.Message);
82      }
83  }

```

Antes que nada, comprobamos que realmente estemos conectados a nuestra base de datos mediante este método llamado conexión.

Para el resto de métodos como crear tabla, borrar tabla, limpiar tabla, hacemos uso de lo que son SqlDataReader, que nos permite iniciar y terminar las consultas (Querys), cualquier instrucción que no nosotros deseemos realizar en nuestra base de datos, debe de usarse la clase SqlCommand, ya que con ella, definimos que nuestra instrucción a realizar mediante la consulta, que en nuestro caso empleamos un comando de tipo texto, es decir, toda la instrucción será mediante una cadena, e interpretada por SqlCommand, para que esta se lleve a cabo a través de la consulta que se encuentra abierta.

Una vez que se termina la instrucción debemos cerrar el comando y la consulta mediante **Consulta.dispose() y Comando.Dispose()**, para evitar los mensajes de error, además de que llegue a presentarse alguno, es más fácil de identificar de donde proviene.

	Manual de prácticas de Temas Selectos de Programación I y II	Código: MADO-84
Facultad de Ingeniería		Versión: 01
		Página 13/21
		Sección ISO 8.3
		Fecha de emisión 24 de enero de 2020

Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica
--

La impresión de este documento es una copia no controlada

```

85  /// <summary>
86  /// Método para crear nuestra tabla en la base de datos local del proyecto
87  /// </summary>
88  public void CrearTabla()
89  {
90      try
91      {
92          //Instancia de la conexion para la base de datos
93          //NOTA: La cadena de conexion debe cambiarse en App.config caso de que se ejecute en otro equipo
94          //La cadena de conexion "conexion" debe ser la nos brinda la ventana de propiedades de la base de datos
95          using (var conexion = new SqlConnection(ConfigurationManager.ConnectionStrings["conexion"].ConnectionString))
96          {
97              //Abrimos la conexion
98              conexion.Open();
99              //Iniciamos una consulta (Query)
100             SqlDataReader consulta1;
101             //Iniciamos una instrucción para SQL Server mediante la clase SqlCommand
102             using (var comando = new SqlCommand())
103             {
104                 comando.Connection = conexion;
105                 comando.CommandText = "CREATE TABLE Tabla_Frecuencias (" +
106                     "ID int identity(1,1) primary key NOT NULL," +
107                     "Resistor_A float NOT NULL," +
108                     "Resistor_B float NOT NULL," +
109                     "Capacitor float NOT NULL," +
110                     "Frecuencia float NOT NULL )";
111                 comando.CommandType = CommandType.Text;
112                 consulta1 = comando.ExecuteReader();
113                 //Una vez ejecutado el comando, terminamos la consulta y el comando
114                 comando.Dispose();
115                 consulta1.Dispose();
116                 //Declararemos un data adapter, para llenar el dataset correspondiente a nuestra tabla en la base de datos
117                 using (SqlDataAdapter dataAdapter1 = new SqlDataAdapter())
118                 {
119                     //Iniciamos otra consulta
120                     SqlDataReader consulta2;
121                     dataAdapter1.SelectCommand = comando;
122                     comando.Connection = conexion;
123                     comando.CommandText = "SELECT * FROM Tabla_Frecuencias";
124                     comando.CommandType = CommandType.Text;
125                     consulta2 = comando.ExecuteReader();
126                     //Utilizamos nuestro SqlCommand "comando", para una nueva instrucción
127                     //Una vez ejecutado el comando, terminamos la consulta y el comando
128                     comando.Dispose();
129                     consulta2.Dispose();
130                     //Iniciamos nuestro dataset para ser rellenado con nuestro dataAdapter, que proporciona la información de nuevo
131                     DBfrecuencias555DataSet dataSet1 = new DBfrecuencias555DataSet();
132                     dataAdapter1.Fill(dataSet1, "Tabla_Frecuencias");
133                     dataGridView1.DataSource = dataSet1.Tables["Tabla_Frecuencias"];
134                 }
135                 //Cerramos la conexión
136                 conexion.Close();
137             }
138         }
139         MessageBox.Show("Tabla creada con éxito");
140     }
141     catch (Exception excepcion)
142     {
143         //Mensaje de error en caso de que algún problema nos impida la conexión o un comando mal ejecutado
144         MessageBox.Show(excepcion.Message);
145     }
146 }
```



Manual de prácticas de Temas Selectos de Programación I y II

Código:	MADO-84
Versión:	01
Página	13/21
Sección ISO	8.3
Fecha de emisión	24 de enero de 2020

Facultad de Ingeniería

Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica

La impresión de este documento es una copia no controlada

```
148     /// <summary>
149     /// Insertar nuevo elemento en la tabla
150     /// </summary>
151     public void Insertar(string a, string b, string c)
152     {
153         try
154         {
155             //Instancia de la conexion para la base de datos
156             //NOTA: La cadena de conexion debe cambiarse en App.config caso de que se ejecute en otro equipo
157             //La cadena de conexion "conexion" debe ser la nos brinda la ventana de propiedades de la base de datos
158             using (var conexion = new SqlConnection(ConfigurationManager.ConnectionStrings["conexion"].ConnectionString))
159             {
160
161                 double frec = PulseFrecuene(Convert.ToDouble(a), Convert.ToDouble(b), Convert.ToDouble(c));
162                 //Abrimos la conexion
163                 conexion.Open();
164                 //Iniciamos una consulta
165                 SqlDataReader consulta1;
166                 //Iniciamos una instruccion para SQL Server mediante la clase SqlCommand
167                 using (var comando = new SqlCommand())
168                 {
169                     comando.Connection = conexion;
170                     comando.CommandText = "INSERT INTO Tabla_Frecuencias (Resistor_A, Resistor_B, Capacitor, Frecuencia)" +
171                         "VALUES (" + a + ',' + b + ',' + c + ',' + frec + ')';
172                     comando.CommandType = CommandType.Text;
173                     consulta1 = comando.ExecuteReader();
174                     //Una vez ejecutado el comando, terminamos la consulta y el comando
175                     comando.Dispose();
176                     consulta1.Dispose();
177                     //Declaramos un data adapter, para llenar el dataset correspondiente a nuestra tabla en la base de datos
178                     using (SqlDataAdapter dataAdapter1 = new SqlDataAdapter())
179                     {
180                         //Iniciamos otra consulta
181                         SqlDataReader consulta2;
182                         dataAdapter1.SelectCommand = comando;
183                         comando.Connection = conexion;
184                         comando.CommandText = "SELECT * FROM Tabla_Frecuencias";
185                         comando.CommandType = CommandType.Text;
186                         consulta2 = comando.ExecuteReader();
187                         //Utilizamos nuestro SqlCommando "comando", para una nueva instrucion
188                         //Una vez ejecutado el comando, terminamos la consulta y el comando
189                         comando.Dispose();
190                         consulta2.Dispose();
191                         //Iniciamos nuestro dataset para ser rellenoado con nuestro dataAdapter, que proporciona la informacion de nu
192                         DBfrecuencias555DataSet dataSet1 = new DBfrecuencias555DataSet();
193                         dataAdapter1.Fill(dataSet1, "Tabla_Frecuencias");
194                         dataGridView1.DataSource = dataSet1.Tables["Tabla_Frecuencias"];
195                     }
196                     //Cerramos la conexion
197                     conexion.Close();
198                 }
199             }
200             MessageBox.Show("Datos agregados correctamente");
201         }
202         catch (Exception excepcion)
203         {
204             //Mensaje de error en caso de que algun problema nos impida la conexion o un comando mal ejecutado
205             MessageBox.Show(excepcion.Message);
206         }
207     }
```

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	13/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

```

236 //Declararemos un data adapter, para llenar el dataset correspondiente a nuestra tabla en la base de datos
237 using (SqlDataAdapter dataAdapter1 = new SqlDataAdapter())
238 {
239     //Iniciamos otra consulta
240     SqlCommand comando;
241     dataAdapter1.SelectCommand = comando;
242     comando.Connection = conexion;
243     comando.CommandText = "SELECT * FROM Tabla_Frecuencias";
244     comando.CommandType = CommandType.Text;
245     consulta2 = comando.ExecuteReader();
246     //Utilizamos nuestro SQLCommando "comando", para una nueva instrucion
247     //Una vez ejecutado el comando, terminamos la consulta y el comando
248     comando.Dispose();
249     consulta2.Dispose();
250     //Iniciamos nuestro dataset para ser rellenado con nuestro dataAdapter, que proporciona la informacion de nue
251     DBfrecuencias555DataSet dataSet1 = new DBfrecuencias555DataSet();
252     dataAdapter1.Fill(dataSet1, "Tabla_Frecuencias");
253     dataGridView1.DataSource = dataSet1.Tables["Tabla_Frecuencias"];
254 }
255     //Cerramos la conexion
256     conexion.Close();
257 }
258 }
259 MessageBox.Show("La tabla ahora está vacía");
260 }
261 catch (Exception excepcion)
262 {
263     //Mensaje de error en caso de que algún problema nos impida la conexión o un comando mal ejecutado
264     MessageBox.Show(excepcion.Message);
265 }
266 }
267 /// <summary>
268 /// Elimina completamente la tabla de la base de datos
269 /// </summary>
270 public void EliminarTabla()
271 {
272     try
273     {
274         //Instancia de la conexión para la base de datos
275         //NOTA: La cadena de conexión debe cambiarse en App.config caso de que se ejecute en otro equipo
276         //La cadena de conexión "conexion" debe ser la que nos brinda la ventana de propiedades de la base de datos
277         using (var conexion = new SqlConnection(ConfigurationManager.ConnectionStrings["conexion"].ConnectionString))
278         {
279             //Abrimos la conexión
280             conexion.Open();
281             //Iniciamos una consulta
282             SqlDataReader consulta1;
283             //Iniciamos una instrucción para SQL Server mediante la clase SqlCommand
284             using (var comando = new SqlCommand())
285             {
286                 comando.Connection = conexion;
287                 comando.CommandText = "DROP TABLE Tabla_Frecuencias";
288                 comando.CommandType = CommandType.Text;
289                 consulta1 = comando.ExecuteReader();
290                 //Una vez ejecutado el comando, terminamos la consulta y el comando
291                 comando.Dispose();
292                 consulta1.Dispose();
293                 //Cerramos la conexión
294                 conexion.Close();
295             }
296             //Dado que la tabla ya no existe, deja de mostrarse información en el dataGridView
297             dataGridView1.DataSource = null;
298         }
299         MessageBox.Show("La tabla se ha eliminado de la base de datos");
300     }
301     catch (Exception excepcion)
302     {
303         //Mensaje de error en caso de que algún problema nos impida la conexión o un comando mal ejecutado
304         MessageBox.Show(excepcion.Message);
305     }
306 }
307 }
```



Manual de prácticas de Temas Selectos de Programación I y II	Código: MADO-84 Versión: 01 Página 13/21 Sección ISO 8.3 Fecha de emisión 24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica
La impresión de este documento es una copia no controlada	

```

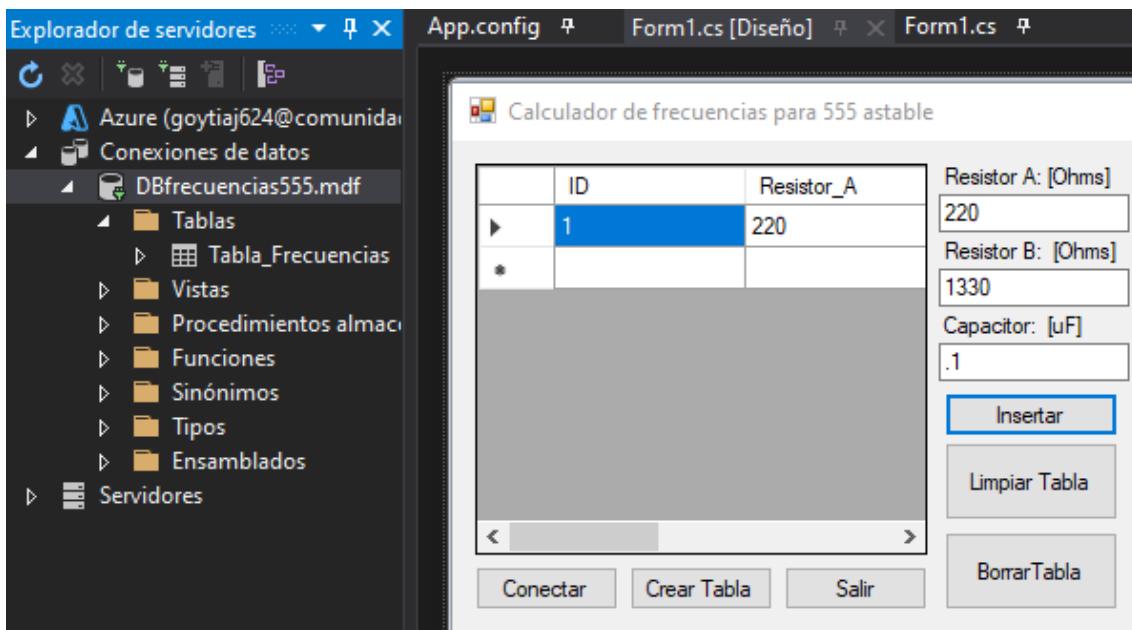
309  /// <summary>
310  /// Obtiene la frecuencia para la configuración de un CI 555 en configuración astable
311  /// </summary>
312  /// <param name="resA"> Resistor A </param>
313  /// <param name="resB"> Resistor B </param>
314  /// <param name="cap"> Capacitor </param>
315  /// <returns> Frecuencia </returns>
316  public double PulseFrecuene(double resA, double resB, double cap)
317  {
318    double frecuencia;
319    frecuencia = 1.44 / ((resA + (2 * resB)) * (cap * Math.Pow(10, -6)));
320    frecuencia = Math.Round(frecuencia, 4);
321    return frecuencia;
322  }

```

Dado que se nos solicito realizar la practica con una aplicación ingenieril, decidimos el hacer nuestra base de datos para almacenar los valores de diseño para circuitos 555 en configuración astable, solo requerimos ingresar los valores nominales de los componentes y el método PulseFrecuense se encargará de obtener la frecuencia de trabajo para esos valores, y los guardara en la tabla con el nombre que se le asigna en el método CrearTabla.

Finalmente tenemos la siguiente distribución para nuestro formulario de Windows Forms, que emplea un datagridview, permitiéndonos ver, los elementos que contiene la tabla de nuestra base de datos ya sea que existan o este vacía.

Para comprobar que las instrucciones funcionan correctamente, podemos usar el Explorador de Servidores durante la ejecución para confirmar la instrucción.



	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	13/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Aunque no se pueda ver completamente la tabla, podemos darnos cuenta de que se agregaron correctamente los datos, ya que, desde un inicio, no existía ninguna tabla de datos.

6. Bibliografía

- BEIGHLEY, Lynn. **Head First SQL**.USA, O'Reilly 2007.
- NEVADO, Victoria. **Introducción a las bases de datos**. Madrid, Vision Libros, 2010.
- CHARLIE, Brooks. **Enterprise NoSQL for Dummies**. USA, John Wiley&Sons, 2014.
- <https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient?view=dotnet-plat-ext-5.0>
- <https://www.w3schools.com/sql/default.asp>



Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ingeniería Mecánica e Industrial



Laboratorio de Cómputo de Ingeniería
Mecatrónica

Temas Selectos de Programación I (1964)

Profesor: Ing. Barrón Velázquez Gersain

Grupo 3

Practica #4

Interfaz Hardware Software

Brigada 1:

- Goytia González Jorge Hadamard
- Ordaz Lucas Efraín Uriel

Semestre 2021-1

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	14/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Práctica #4

Interfaz Software Hardware

1. Seguridad en la ejecución

	Peligro o Fuente de energía	Riesgo asociado
1	Tensión alterna	Electrocución

2. Objetivos de aprendizaje

OBJETIVO GENERAL: Entender los protocolos existentes para comunicar un programa (software) con elementos físicos (hardware) que le sean de utilidad.

OBJETIVOS ESPECÍFICOS:

- Entender qué es la comunicación serial.
- Realizar el código para la comunicación entre un programa y un dispositivo externo.

3. Introducción

La interfaz software-hardware es de amplio uso en la vida académica y profesional de las personas involucradas en la industria de la programación. Una de las formas más recurridas para la comunicación entre software y hardware es a través de microcontroladores o microcomputadoras con la intención de tener entradas y salidas de propósito general, tanto digitales como analógicas y a partir de ellos acceder a información de sensores o generar respuestas en actuadores con base en la ejecución de un programa.

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	15/21
		Sección ISO	8.3
	Fecha de emisión		
Facultad de Ingeniería		Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica	
La impresión de este documento es una copia no controlada			

La manera más sencilla de comunicar una computadora con un microcontrolador es mediante el uso de la comunicación serial. La comunicación serial es un proceso por el cual se puede enviar un bit a la vez, o palabras de bits; entre sus ventajas está la reducción de cables con respecto a un proceso en paralelo, disminuyendo así la posibilidad de introducir ruido de línea en un proceso.

Entre los protocolos más conocidos de comunicación serial se encuentra el USB (Universal Serial Bus, sin embargo, existen más protocolos de comunicación como los que se mencionan a continuación:

- Ethernet
- I²C
- RS-232
- RS-485
- SPI

Un protocolo es un conjunto de reglas y normas que están obligadas a cumplir todos los sistemas con la finalidad de estandarizar la comunicación entre ellos. Sin ese conjunto de reglas, la comunicación requeriría condiciones particulares cada vez que se deseara establecer y eso complicaría la interacción entre desarrolladores. Algunas de las características que norman los protocolos son:

- Detección de la conexión física
- Handshaking
- Cómo iniciar y finalizar un mensaje.
- Procedimientos para dar formato a un mensaje
- Detección de pérdida de información

En el caso específico del uso de microcontroladores, la comunicación serial que se utiliza sigue el protocolo RS-232, por ello se requiere un complemento que permita convertir ese protocolo en USB, para conectarse directamente con la computadora en donde se ejecute el software que interesa interactúe con el hardware.



	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	16/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Figura 1. Proceso de conversión USB a RS232.

En la figura 1 se observan los pines usados en el protocolo USB, D+ y D-, ambos se utilizan para enviar y recibir información, pero nunca al mismo tiempo, es decir, en algún momento, envían información y en otro reciben; a lo anterior se le conoce como comunicación Half Dúplex.

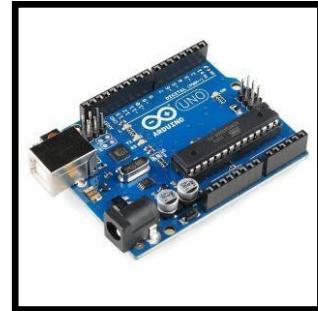
En el caso de los pines entre el convertidor y el microcontrolador se encuentran RX (recepción de datos) y TX (transmisión de datos). Cada pin desempeña una función diferente; el pin TX sólo envía y el RX sólo recibe, de tal forma que al comunicarse con otro dispositivo se realiza una conexión cruzada y la dirección de la información es en un sólo sentido tal cual como se muestra en las flechas.

En el caso de algunas tarjetas de desarrollo, el convertidor USB a RS-232 ya está integrado, por lo que no es necesario conectar un adaptador adicional reduciendo así la cantidad de elementos a utilizar.

4. Material y equipo



Computadora



Microcontrolador o Tarjeta de desarrollo con convertidor USB a RS-232

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	17/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

5. Desarrollo

Para esta práctica, se decidió, utilizar un servomotor, que con ayuda de arduino y visual studio, se hará un programa, que, desde nuestro programa de c#, se pueda manipular un servomotor, en el que se modificará el ángulo de este

¿Qué es un servomotor?

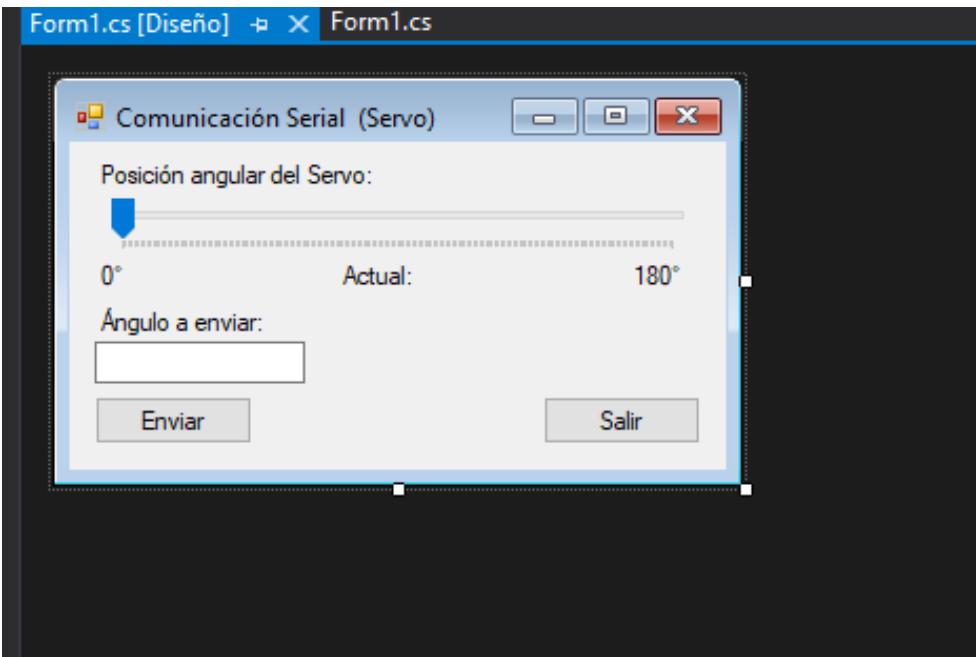
Los servomotores son parte de un sistema de circuito cerrado y se componen de varias partes, a saber, un circuito de control, servomotor, eje, potenciómetro, engranajes de accionamiento, amplificador y un codificador o resolutor.

Es un dispositivo eléctrico autónomo que gira partes de una máquina con alta eficiencia y con gran precisión. El eje de salida de este motor se puede mover a un ángulo, posición y velocidad particulares que un motor normal no tiene, utiliza un motor normal y lo acopla con un sensor para retroalimentación posicional.

El controlador es la parte más importante del servomotor diseñado y utilizado específicamente para este propósito.



Con este dispositivo, se realizó lo siguiente en visual studio



En nuestra form, desplegara la siguiente ventana, la cual va a tener un scroll, con el que podremos configurar la posición del servomotor en un rango de 0° a 180°. Pero si no queremos esa opción, en la parte de abajo, en un textbox, podemos meter el valor del ángulo deseado y enviárselo al arduino con el botón enviar

```

using System;
using System.Windows.Forms;
using System.IO.Ports;

namespace Practica4
{
    public partial class Form1 : Form
    {
        //Creamos instancia para el puerto serial
        static SerialPort Arduino;

        //referencia
        public Form1()
        {
            InitializeComponent();
            this.CenterToScreen();
            try
            {

                //Declaramos el puerto serial
                Arduino = new SerialPort
                {
                    PortName = "COM3", //Se sustituye según el puerto donde se conecte el microcontrolador
                    BaudRate = 9600
                };
                //Abrimos la conexión al puerto serial
                Arduino.Open();
                //Vinculamos el evento de cerrar formulario
                this.FormClosing += Form1_FormClosing;
            }
        }
    }
}

```

Para trabajar con arduino, agregaremos a la biblioteca (en la parte superior del programa) “Sistem.IO.Ports”. una vez declarada esta biblioteca, haremos lo siguiente

Creamos la instancia para nuestro puerto serial, la cual vamos a llamar simplemente “Arduino”, dentro

del método fom que se crea por default, vamos a trabajar gran parte de nuestro código
Declaramos el puerto serial, y le indicamos que va a tomar la entrada “COM3” e iniciara la comunicación en 9600 que es la cantidad de baudios que maneja el puerto en serie, después abrimos la conexión y en caso de que cerremos el programa se cerrara la conexión con un método que veremos más adelante

```
        catch (Exception excepcion)
    {
        //Mensaje en caso de alguna excepcion
        MessageBox.Show(excepcion.Message);
    }
}

1 referencia
private void BtnExit_Click(object sender, EventArgs e)
{
    //Se cierra el programa
    Application.Exit();
}

1 referencia
private void TrackBar1_Scroll(object sender, EventArgs e)
{
    try
    {
        //Si la comunicación serial está abierta
        if (Arduino.IsOpen)
        {
            //Número de grados en trackbar
            int grados = trackBar1.Value;
            //Se envia el valor del ángulo a arduino
            Arduino.WriteLine(Convert.ToString(grados) + "\n");
            //Notificación de cambio del trackbar al valor actual
        }
    }
}
```

En caso de que no se pueda conectar el puerto serial nos arrojara una excepción (en catch)

En nuestro botón de salir, simplemente cierra el programa

Para la parte del scroll, se manejó lo siguiente. Primero el valor que recibimos del scroll lo convertimos a entero y por medio de la función “write”, le mandaremos el número que recibimos del scroll, pero convertido a cadena además de mandarle un salto de línea que nos sirve para indicarle a arduino que lo que le mandamos hasta ese momento, finalizó.

```

        //Mostramos debajo del trackbar el valor actual
        label4.Text = "Actual: " + Convert.ToString(grados) + "°";
    }
}
catch (Exception excepcion)
{
    //Mensaje en caso de alguna excepcion
    MessageBox.Show(excepcion.Message);
}
}

1 referencia
private void BtnEnvio_Click(object sender, EventArgs e)
{
    try
    {
        //Si la comunicación serial está abierta
        if (Arduino.IsOpen)
        {
            //Número de grados en textBox
            string grados = txtAngulo.Text;
            //Se envia el valor del ángulo a arduino
            Arduino.WriteLine(grados + "\n");
            //Mostramos debajo del trackbar el valor actual (enviado)
            label4.Text = "Actual: " + grados + "°";
            //Movemos el trackBar hasta el valor enviado
            trackBar1.Value = Convert.ToInt32(grados);
        }
    }
}

```

Después, solamente con un label que ya está agregado, le mandaremos el valor registrado en el scroll y lo mostraremos

Para nuestro botón de enviar el numero escrito desde el teclado, se realizó lo siguiente.

De igual forma y como buena práctica, meteremos la siguiente instrucción en un try. Vamos a validar que nuestra conexión exista, después, el dato que nosotros metimos, lo guardaremos en una variable de tipo string y la enviaremos al arduino con “write” y posteriormente, le vamos a mandar al label, el valor escrito en el textbox

```
        //Movemos el trackBar hasta el valor enviado
        trackBar1.Value = Convert.ToInt32(grados);
    }
}
catch (Exception excepcion)
{
    //Mensaje en caso de alguna excepcion
    MessageBox.Show(excepcion.Message);
}
}

1 referencia
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    try
    {
        //Si la comunicacion serial esta abierta
        if (Arduino.IsOpen)
        {
            Arduino.Close(); //Se cierra la comunicacion serial
        }
    catch (Exception excepcion)
    {
        //Mensaje en caso de alguna excepcion
        MessageBox.Show(excepcion.Message);
    }
}
}
```

Moveremos el scroll a la parte que hemos enviado en el textbox, y se maneja un cath en caso de algún problema

Para nuestra acción de “FormClosing” vamos a validar que nuestro arduino esté conectado, en caso de que si, lo vamos a cerrar, y manejaremos una excepción por buena práctica de programación

Ahora en nuestro arduino tenemos los siguiente

```

//Agregamos la libreria para un manejo más eficiente del servo
#include <Servo.h>
//Se declara nuestro servo a utilizar
Servo servol;
//Variables a utilizar
char c; //caracter para cortar el mensaje
String info; //donde guardamos la información recibida por Visual Studio
int grados = 0; //donde guardaremos los grados

void setup()
{
    Serial.begin (9600);
    servol.attach(9); //Indica en donde se envia la señal de control
}
void loop()
{
    Info_Serial(); //Nuestro método para obtener los grados
    servol.write(grados); //Le indicamos al servo su posición en grados
}
//Método para recibir la información de Visual Studio

```

Para arduino, utilizaremos la biblioteca de “Servo.h”, declaramos el servo a utilizar (“Servo1”) y tres variables, una de tipo char, otra de tipo string y una de tipo entero. Para el void stup, solo inciaremos la comunicación del arduino en 9600 y le indicaremos al programa que nuestro servo está conectado al pin 9 de nuestro arduino. Para void loop, delaramos un método, que veremos más adelante, lo que realice el método, vamos a mandarle a nuestro servo la información que obtiene “grados”

```

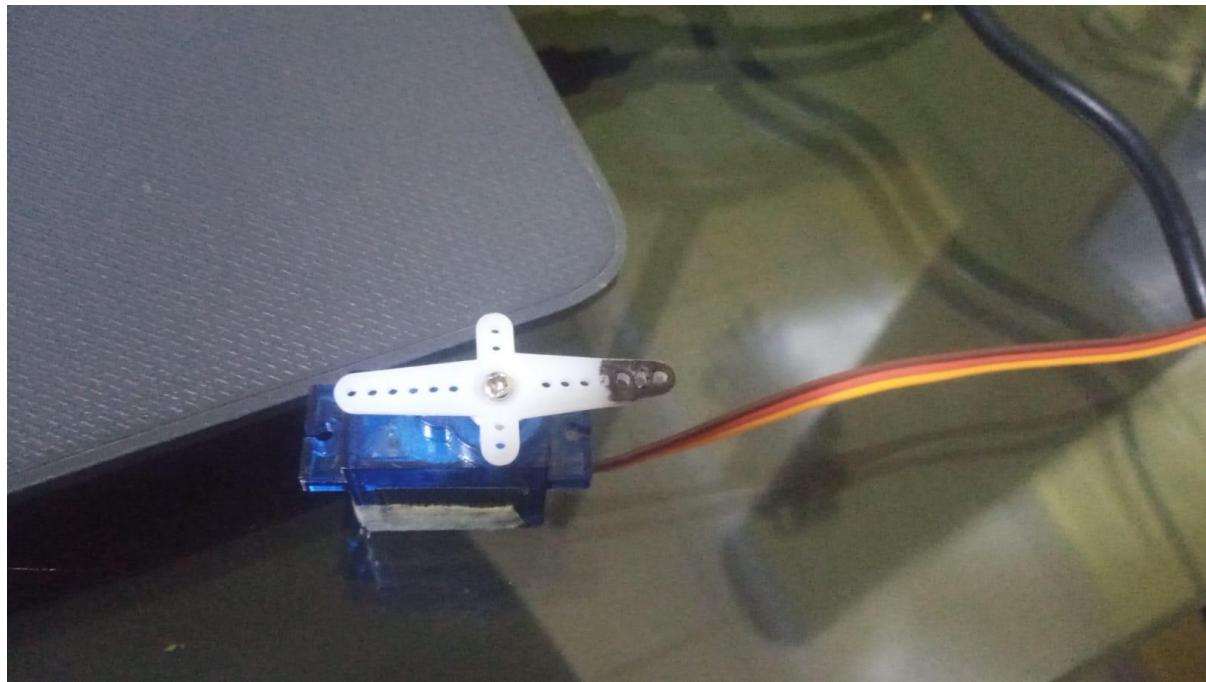
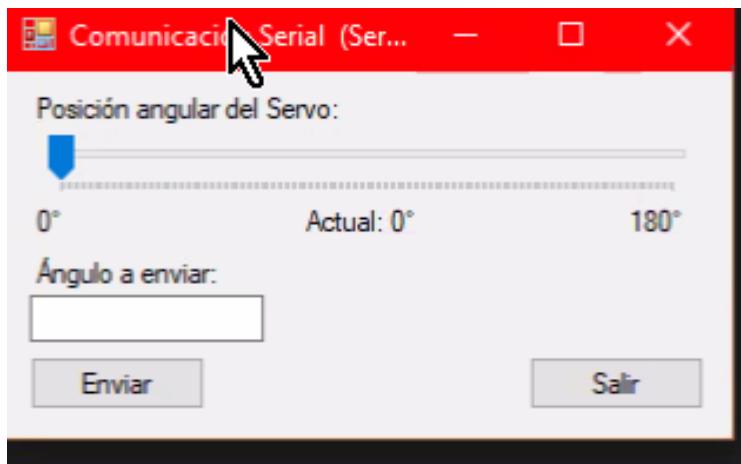
void Info_Serial()
{
    //Si queremos recibir mas de un byte de informacion
    while (Serial.available() > 0)
    {
        c = Serial.read(); //Recibimos la información mediante puerto serial
        if(c == '\n')
        {
            break; //Si se detecta el salto de linea, el mensaje se a terminado
        }
        else
        {
            info = info + c; //Guardamos la información de puerto serial
        }
    }
    //Cuando c tiene asignado el salto de linea
    if(c == '\n')
    {
        grados = info.toInt(); //Si se detecta el salto de linea, guardamos el valor del angulo
        c = 0; //Asignamos a c el caracter 0, puesto que se detecto el fin del mensaje
        info = ""; // Se limpia la cadena de informacion
    }
}

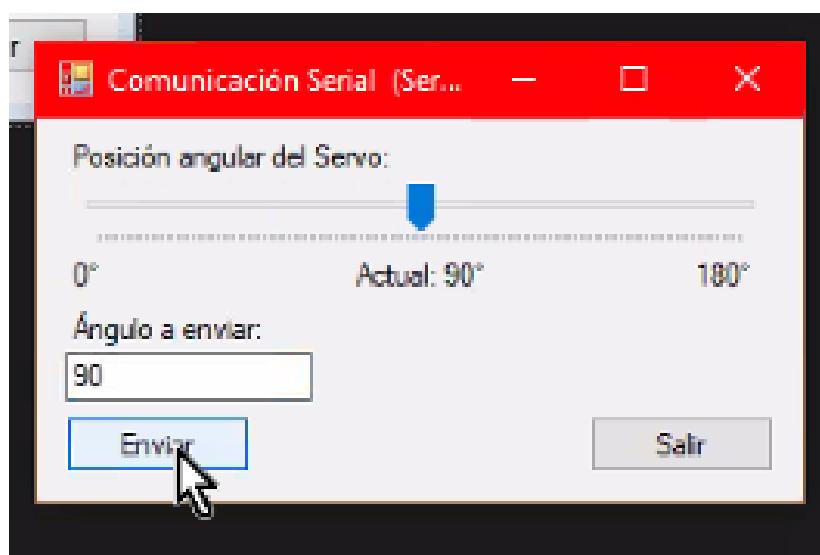
```

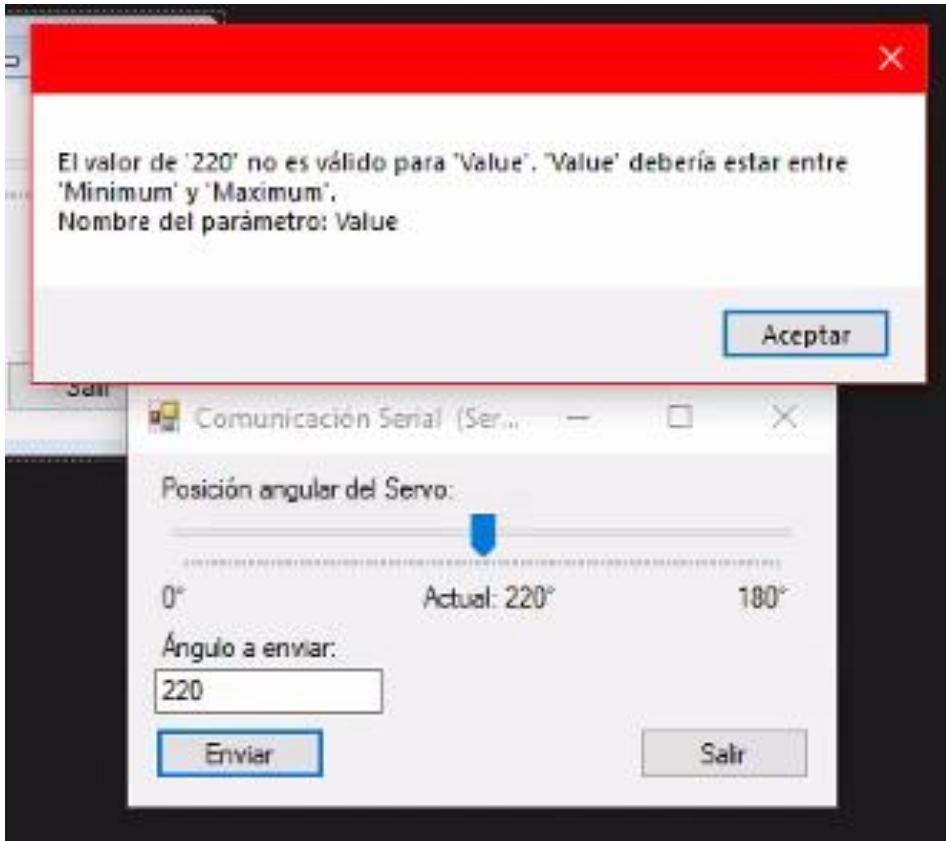
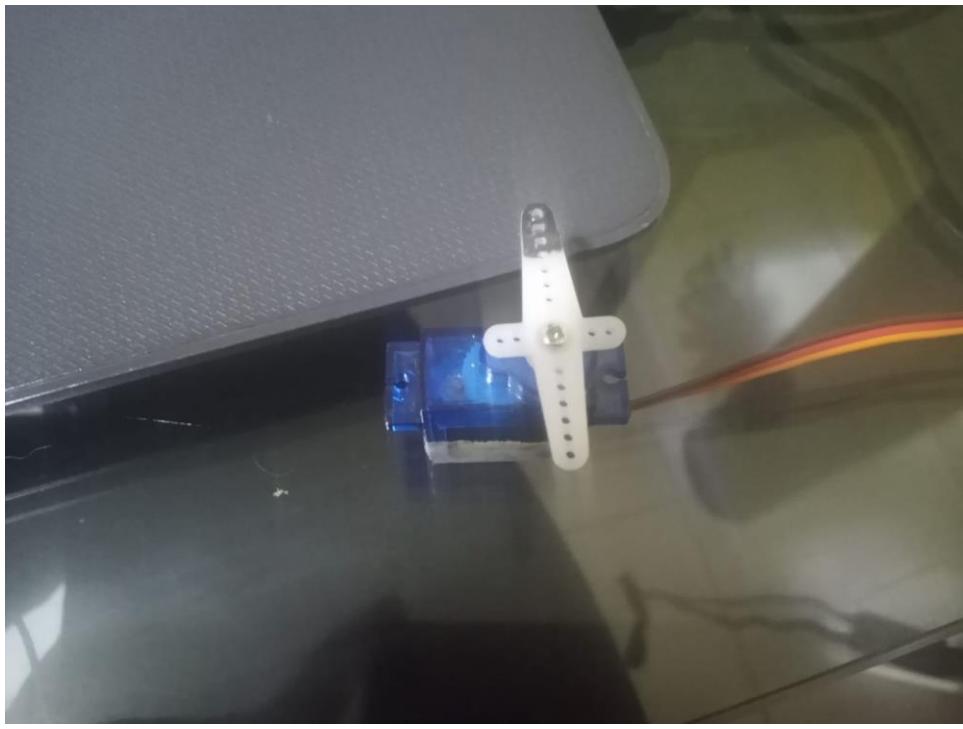
Nuestro método, lo empezaremos con un while ya que va recibir más de in byte de información, lo queharemos después, es que lo que se reciba del programa de visual (Serial.read) lo vamos a guardar en la variable char y procederemos a comparar, si lo que recibimos es un salto de línea, saldrá del ciclo while, en caso contrario, va a guardar ese carácter y lo ira almacenando en la misma variable de tipo string. Cuando salga del ciclo, volveremos a preguntar si lo último que recibió la variable char fue un salto de línea (\n), vamos a guardar en nuestra variable entera, el valor que almacenamos en nuestro string no

sin antes convertir la variable de string a int, la variable char y string la volveremos a iniciar en cero

Con esto ya tenemos la comunicación completa con nuestro arduino y el programa de visual studio







Aquí vemos que al meter un valor más grande al que puede llegar, el programa se va al valor máximo que es 180

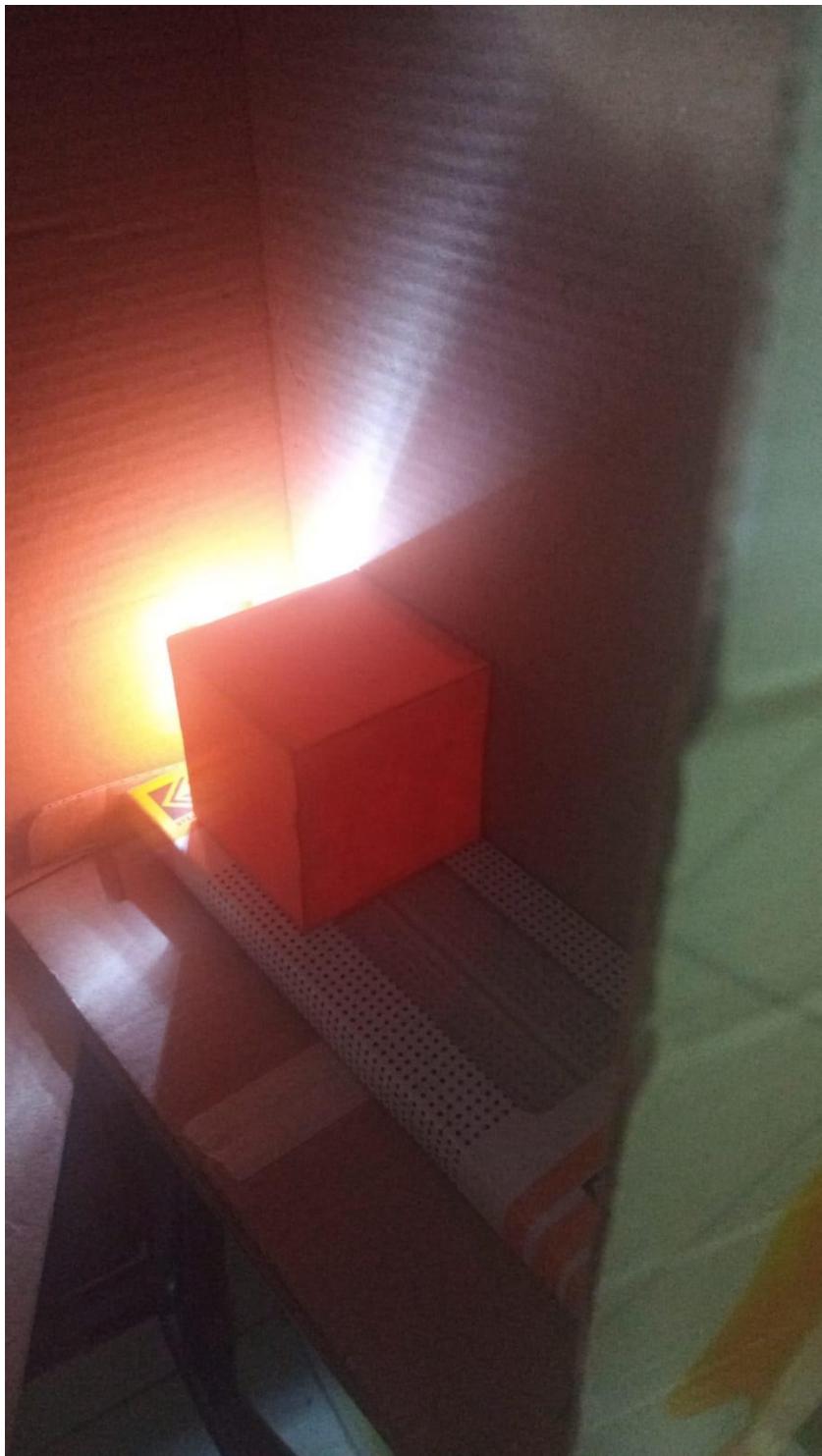


Se anexo otro proyecto a la solución para en esta ocasión recibir datos de arduino y poder mostrarlos desde visual studio

Ahora con un sensor RGB vamos a recibir el color de una caja de color Naranja, la cual va ir variando sus datos durante la conexión



Aquí al momento de hacer la conexión, recibimos los datos de forma automatica



Por lo que en nuestro programa de visual studio tenemos lo siguiente

```

using System;
using System.IO.Ports;
using System.Threading;
using System.Windows.Forms;

namespace Practica4_Recibir_
{
    //referencias
    public partial class Form1 : Form
    {
        //Variables globales
        string MensajeSerial;
        //Creamos instancia para el puerto serial
        readonly SerialPort Arduino;
        bool IsClosed = false;

        //referencia
        public Form1()
        {
            InitializeComponent();
            this.CenterToScreen();

            try
            {
                //Declaramos el puerto serial
                Arduino = new SerialPort
                {
                    PortName = "COM3", //Se sustituye según el puerto donde se conecte el microcontrolador
                    BaudRate = 9600
                };
                //Vinculamos el evento de cerrar formulario
                this.FormClosing += Form1_FormClosing;
            }
            catch (Exception excepcion)
            {
                //Mensaje en caso de alguna excepcion
                MessageBox.Show(excepcion.Message);
            }
        }

        //referencia
        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            try
            {
                IsClosed = true;
                //Si la comunicacion serial esta abierta
                if (Arduino.IsOpen)
                {
                    Arduino.Close(); //Se cierra la comunicacion serial
                }
            }
            catch (Exception excepcion)
            {
                //Mensaje en caso de alguna excepcion
                MessageBox.Show(excepcion.Message);
            }
        }
    }
}

```

```

    i referencia
    private void BttnConectar_Click(object sender, EventArgs e)
    {
        try
        {
            Arduino.PortName = "COM3";
            Arduino.BaudRate = 9600;
            //Abrimos la conexion serial
            Arduino.Open();
            btnConectar.Enabled = false;
            btnDesconectar.Enabled = true;
        }
        //Mensaje en caso de alguna excepcion
        catch (Exception error)
        {
            MessageBox.Show(error.Message);
        }
    }

    i referencia
    private void BttnDesconectar_Click(object sender, EventArgs e)
    {
        try
        {
            //Cerramos la conexion serial
            Arduino.Close();
            btnConectar.Enabled = true;
            btnDesconectar.Enabled = false;
        }
        //Mensaje en caso de alguna excepcion
        catch (Exception error)
        {
            MessageBox.Show(error.Message);
        }
    }

    i referencia
    private void BttnExit_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    i referencia
    private void EscucharSerial()
    {
        //Mientras la comunicacion serial no este cerrada
        while (!IsClosed)
        {
            try
            {
                //Leemos la comunicacion serial de arduino
                MensajeSerial = Arduino.ReadLine();
                //Separamos los valores por comas y los guardamos en un arreglo
                string[] valores = MensajeSerial.Split(',');
                //Indicamos el valor final de la cadena
                string[] valor = valores[2].Split('\r');
            }
        }
    }

```

```

        if(valores.Length == 3)
        {
            //ejecutamos otro delegado para poder procesar la informacion al momento de recibirla
            this.BeginInvoke(new EventHandler(delegate
            {
                if (MensajeSerial != null)
                {
                    //Mostramos los valores separados
                    textBoxR.Text = valores[0];
                    textBoxG.Text = valores[1];
                    textBoxB.Text = valores[2];
                }
            }));
        }
    }
    catch
    {
    }
}
}

private void Form1_Load(object sender, EventArgs e)
{
    //Traemos la informacion del arduino al hilo principal del programa
    Thread Hilo = new Thread(EscucharSerial);
    Hilo.Start();
}
}
}

```

Y en arduino se realizó el siguiente código

```

// Sensor de Color
//
// Arduino TCS230 SENSOR COLOR
// 8      OUT
// 7      S3
// 6      S2
// 5      S1
// 4      S0
// 5V     VCC
// GND   GND
//
#define S0 4
#define S1 5
#define S2 6
#define S3 7
#define sensorSalida 8
double Rojo_Frec = 0;
double Verde_Frec = 0;
double Azul_Frec = 0;

void setup() {
    pinMode(S0, OUTPUT);
    pinMode(S1, OUTPUT);
    pinMode(S2, OUTPUT);
    pinMode(S3, OUTPUT);
    pinMode(sensorSalida, INPUT);

    // Se configura la escala de Frecuencia en 20%
    digitalWrite(S0,HIGH);
    digitalWrite(S1,LOW);
    Serial.begin(9600);
}

```

Para nuestro código de arduino, definimos 5 variables de conexión para arduino, después tres variables

de tipo double, que van a guardar la escala de color R, G y B, todas las vamos a iniciar en cero En nuestro void seup las conexiones en arduino las declaramos como variables de salida y nuestra variable de entrada será la última variable de conexión (sensorSalida), después configuraremos la escala de la frecuencia en 20

```
void setup() {
    // Se configura la escala de Frecuencia en 20
    digitalWrite(S0,HIGH);
    digitalWrite(S1,LOW);
    Serial.begin(9600);
}

void loop() {
    // Configura el filtro ROJO para tomar lectura
    digitalWrite(S2,LOW);
    digitalWrite(S3,LOW);
    delay(100);
    Rojo_Frec = pulseIn(sensorSalida, LOW);
    delay(100);

    // Configura el filtro VERDE para tomar lectura
    digitalWrite(S2,HIGH);
    digitalWrite(S3,HIGH);
    delay(100);
    Verde_Frec = pulseIn(sensorSalida, LOW);
    delay(100);

    // Configura el filtro AZUL para tomar lectura
    digitalWrite(S2,LOW);
    digitalWrite(S3,HIGH);
    delay(100);
    Azul_Frec = pulseIn(sensorSalida, LOW);
    delay(100);

    if(Rojo_Frec<400){
        Serial.println(String(Rojo_Frec)+"."+String(Verde_Frec)+"."+String(Azul_Frec));
        delay(3000);
    }
}
```

Aquí configuramos cada color, Rojo, Verde y Azul en una frecuencia determinada, el arduino se esperará cierto tiempo para poder visualizar el valor que registro hace un momento y luego hará nuevamente la lectura del color. Después solo le asignaremos a nuestras variables de tipo double.

Aquí lo que hace es tomar la lectura de cada color uno en uno, no toma las tres lecturas al mismo tiempo

6. Bibliografía

- LAAKMANN McDOWELL, Gayle. **Cracking the Coding Interview: 189 Programming Questions and Solutions**. Palo Alto, CA., CareerCup, 2016.
- BELL, Douglas y PARR, Mike. C# para estudiantes. México, Pearson, 2010



Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ingeniería Mecánica e Industrial



Laboratorio de Cómputo de Ingeniería
Mecatrónica

Temas Selectos de Programación I (1964)

Profesor: Ing. Barrón Velázquez Gersain

Grupo 3

Práctica No. 5 Servicios Web

Brigada 1:

- Goytia González Jorge Hadamard
- Ordaz Lucas Efraín Uriel

Semestre 2021-1

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	18/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

Práctica #5

Servicios web

1. Seguridad en la ejecución

	Peligro o Fuente de energía	Riesgo asociado
1	Tensión alterna	Electrocución

2. Objetivos de aprendizaje

OBJETIVO GENERAL: El alumno comprenderá el concepto de servicio web, así como su utilidad e implementará uno.

OBJETIVOS ESPECÍFICOS:

- Generar el código que permita utilizar protocolos de servicios web.
- Identificar los elementos para lograr la compatibilidad entre un programa desarrollado y un servicio web.

3. Introducción

Las aplicaciones web han sufrido una evolución análoga a la que ya padecieron las aplicaciones de escritorio que utilizan los recursos propios de cada sistema operativo para construir su interfaz de usuario. Inicialmente, estas aplicaciones se ejecutaban en una única máquina, que era además la máquina donde se almacenaban los datos requeridos, procesados y entregados.

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	19/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

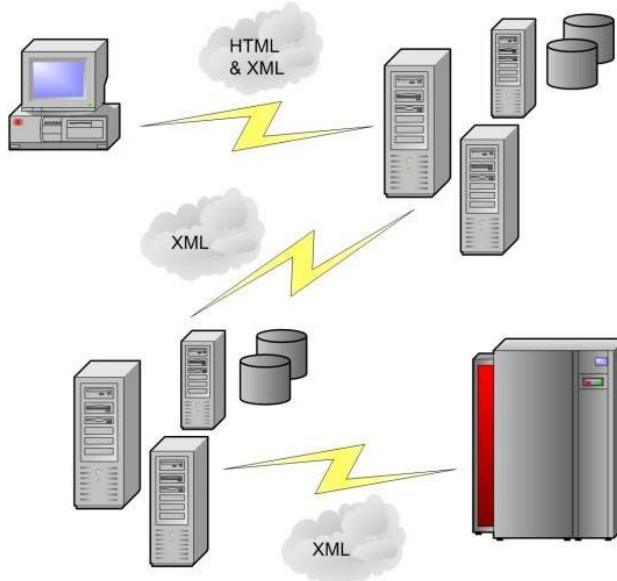


Figura 1. Representación gráfica de la arquitectura de un servicio web.

Un servicio web (web service) es un conjunto de técnicas, protocolos y estándares que sirven para el intercambio de datos entre aplicaciones a través de internet. Al ser un servicio apoyado en HTTP y un envío de datos, el servicio web puede funcionar con diferentes lenguajes de programación.

Esta inter-operatividad e intercambio de mensajes entre aplicaciones a través de la red, están típicamente basados en XML sobre HTTP juntamente con otros estándares, lo que lo hace ideal para internet.

Entre algunos de los estándares utilizados para realizar servicios web están:

- Web Services Protocol Stack: conjunto de servicios y protocolos de los servicios web.
- XML (Extensible Markup Language): formato estándar para los datos que se vayan a intercambiar.
- WSDL (Web Services Description Language): es el lenguaje de la interfaz pública para los servicios web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios web.

	Manual de prácticas de Temas Selectos de Programación I y II	Código:	MADO-84
		Versión:	01
		Página	20/21
		Sección ISO	8.3
		Fecha de emisión	24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica		
La impresión de este documento es una copia no controlada			

La complejidad de un servicio web se ve muy afectada al tener que crear autenticación de las máquinas que acceden a su servicio, es por ello por lo que la configuración y aprendizaje de los protocolos es muy importante para evitar un mal uso o tener vulnerabilidades en el sistema. Por las razones anteriores, hoy día se ofrece una variedad de proveedores de servicios de almacenamiento y accesibilidad a través de la nube que simplifican las tareas de los desarrolladores; de tal forma que con sólo subir el programa que se desea tener como servicio web, los proveedores se encargan del cifrado, seguridad y acceso al mismo, disminuyendo la incidencia de ataques informáticos y promoviendo un correcto ecosistema de servicios a través de internet. Ejemplos de tales proveedores son: Google Cloud Platform, Amazon Web Services, y Azure.

4. Material y equipo



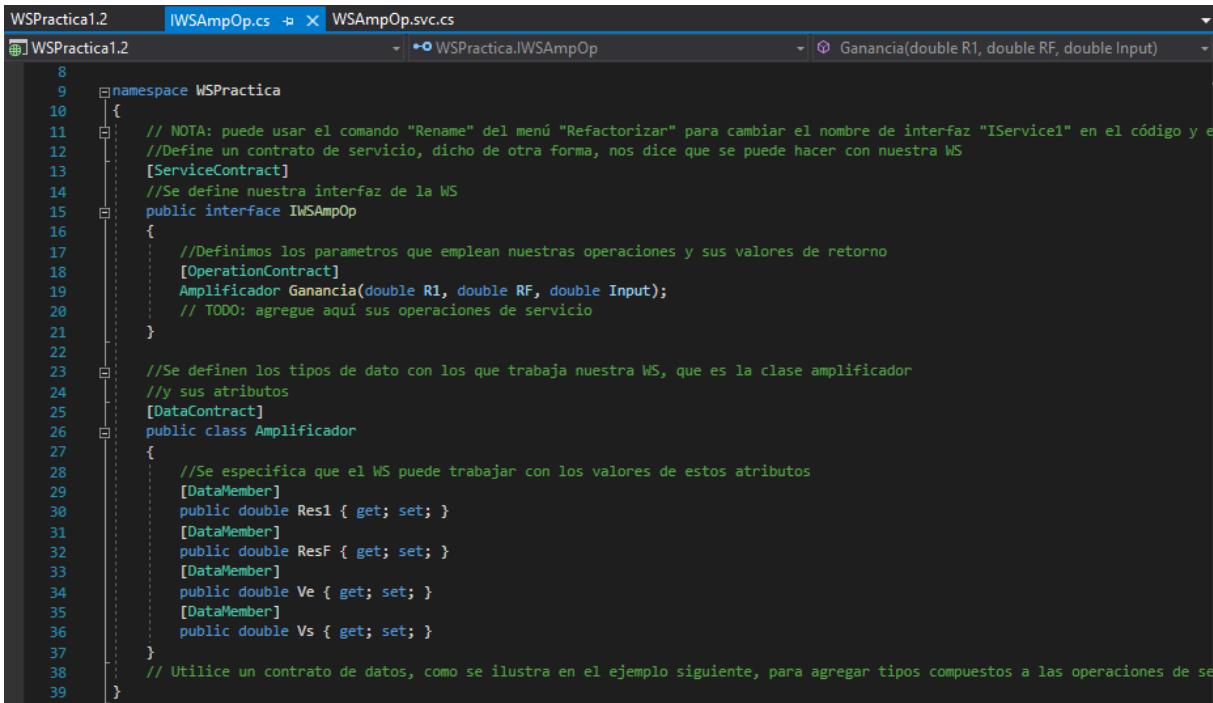
Computadora

	Manual de prácticas de Temas Selectos de Programación I y II	Código: MADO-84 Versión: 01 Página: 21/21 Sección ISO: 8.3 Fecha de emisión: 24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica	
La impresión de este documento es una copia no controlada		

5. Desarrollo

Para esta practica realizamos un servicio web simple mediante WCF que sus siglas son *Windows Communication Foundation* que es un framework para la creación de aplicaciones orientadas a servicios y que fue desarrollada principalmente para crear servicios en SOAP.

Para la creación de nuestro servicio web, se necesitó la creación de un proyecto del tipo de *Aplicación de Servicios WCF*, una vez que se crea el proyecto, nos genera dos archivos, siendo los que mostramos a continuación.



```

WSPractica1.2 IWSAmpOp.cs & X WSAmpOp.svc.cs
WSPRACTICA.2                                     • WSPRACTICA.IWSAmpOp                                     • Ganancia(double R1, double RF, double Input)
8
9     namespace WSPRACTICA
10    {
11        // NOTA: puede usar el comando "Rename" del menú "Refactorizar" para cambiar el nombre de interfaz "IService1" en el código y en el contrato de servicio.
12        // Define un contrato de servicio, dicho de otra forma, nos dice que se puede hacer con nuestra WS
13        [ServiceContract]
14        // Se define nuestra interfaz de la WS
15        public interface IWSAmpOp
16        {
17            // Definimos los parámetros que emplean nuestras operaciones y sus valores de retorno
18            [OperationContract]
19            Amplificador Ganancia(double R1, double RF, double Input);
20            // TODO: agregue aquí sus operaciones de servicio
21        }
22
23        // Se definen los tipos de dato con los que trabaja nuestra WS, que es la clase amplificador
24        // y sus atributos
25        [DataContract]
26        public class Amplificador
27        {
28            // Se especifica que el WS puede trabajar con los valores de estos atributos
29            [DataMember]
30            public double Res1 { get; set; }
31            [DataMember]
32            public double ResF { get; set; }
33            [DataMember]
34            public double Ve { get; set; }
35            [DataMember]
36            public double Vs { get; set; }
37        }
38        // Utilice un contrato de datos, como se ilustra en el ejemplo siguiente, para agregar tipos compuestos a las operaciones de servicio
39    }

```

Para el archivo de interfaz, se define todo lo que puede hacer nuestro servicio web, tal como dice **[ServiceContract]**, lo que se encuentre a continuación, ósea nuestra interfaz, es lo que ofrece nuestra aplicación de servicio web, la línea **[OperationContract]** nos permite definir que métodos tenemos a nuestra disposición con este servicio. Para lo que corresponde a **[DataContract]** nos permite indicar la estructura de los datos que emplea y **[DataMember]** nos resalta si podemos trabajar con ciertos datos que pertenezcan a un **DataContract**, de lo contrario, el cliente no tiene acceso a enviar o recibir información a no ser que se encuentre etiquetado como **DataMember**.

	Manual de prácticas de Temas Selectos de Programación I y II	Código: MADO-84 Versión: 01 Página 21/21 Sección ISO 8.3 Fecha de emisión 24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica	
La impresión de este documento es una copia no controlada		

```

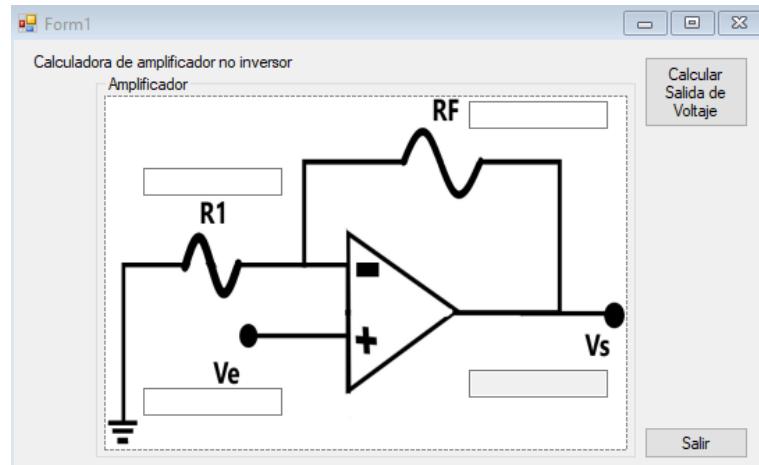
WSPractica1.2      IWSAmpOp.cs      WSAmpOp.svc.cs + X
WSPractica1.2      ↗ WSPractica.WSAmpOp      ↗ Ganancia(double R1, double RF, double Input)
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.ServiceModel;
6  using System.ServiceModel.Web;
7  using System.Text;
8
9  namespace WSPractica
10 {
11     // NOTA: puede usar el comando "Rename" del menú "Refactorizar" para cambiar el nombre de clase "Service1" en el código, en svc
12     // NOTE: para iniciar el Cliente de prueba WCF para probar este servicio, seleccione Service1.svc o Service1.svc.cs en el Explorador de soluciones y pulse F5
13     //Creamos la clase AmpOp en la WS que hereda de la interfaz IWSAmpOp
14     public class WSAmpOp : IWSAmpOp
15     {
16         //Este es el método que heredamos por la interfaz, que corresponde a nuestra clase amplificador
17         public Amplificador Ganancia(double R1, double RF, double Input)
18         {
19             double output = ((RF/R1)+1)*Input;
20             return new Amplificador() { Res1 = R1, ResF = RF, Ve = Input, Vs = output };
21         }
22     }
23 }
24

```

Finalmente, para poder trabajar con nuestro servicio web, se define una clase, que hereda de nuestra interfaz, ya en esta parte, se define el método que se tiene referenciado al heredar en nuestra clase.

La aplicación de este servicio web es el calcular el voltaje de salida mediante la ganancia en un amplificador operacional en configuración no inversora, por lo que se necesita del valor de sus resistores y el voltaje de entrada que se va a amplificar.

Por último, ya que se tiene definida nuestro servicio web (que no se encuentra en la red, sino que se está consumiendo de manera local), se desarrolló un pequeño proyecto denominado Cliente, siendo esta su interfaz gráfica en Forms.



	Manual de prácticas de Temas Selectos de Programación I y II	Código: MADO-84
Facultad de Ingeniería		Versión: 01
		Página 21/21
		Sección ISO 8.3
		Fecha de emisión 24 de enero de 2020
	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica	
	La impresión de este documento es una copia no controlada	

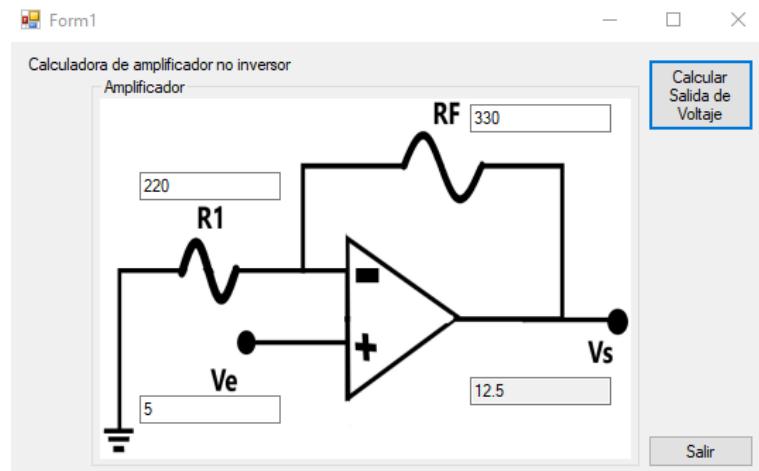
Y respectivamente, el código del formulario empleado para trabajar la práctica.

```

21  private void BtnExit_Click(object sender, EventArgs e)
22  {
23      //Cerramos el programa
24      Application.Exit();
25  }
26
27  private void BtnCalcular_Click(object sender, EventArgs e)
28  {
29      //Conversion de datos de los textBox
30      double R1 = Convert.ToDouble(textBoxR1.Text);
31      double RF = Convert.ToDouble(textBoxRF.Text);
32      double Ve = Convert.ToDouble(textBoxVe.Text);
33      try
34      {
35          //Creamos la instancia de la WS creada
36          using (WSAmpOp.WSAmpOpClient cliente = new WSAmpOp.WSAmpOpClient())
37          {
38              //Usamos un var, ya que el amplificador es una clase dentro de nuestra WS
39              var AmpOp = cliente.Ganancia(R1, RF, Ve);
40              //Mostramos el voltaje de salida que se le asigna al amplificador en un textbox
41              textBoxVs.Text = Convert.ToString(AmpOp.Vs);
42          }
43      }
44      catch (Exception excepcion)
45      {
46          MessageBox.Show(excepcion.Message);
47      }
48  }

```

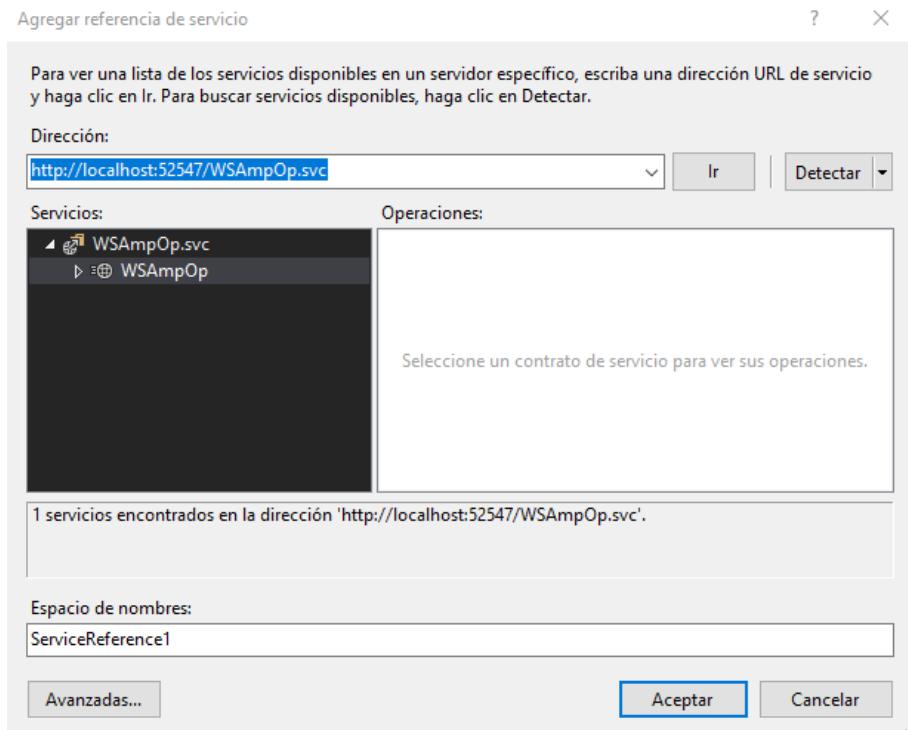
Dado que la funcionalidad del programa se encuentra en el servicio web, las líneas de código resultaron ser mínimas, ya que solo creamos la instancia del servicio web y enviamos los parámetros que necesita para su operación, finalmente mostramos el resultado en nuestro formulario con el resto de los datos de entrada.



Para poder consumir nuestro servicio web de manera correcta, debemos de compilar el proyecto de servicio web aparte, una vez compilado, en el proyecto de cliente se

	Manual de prácticas de Temas Selectos de Programación I y II	Código: MADO-84 Versión: 01 Página 21/21 Sección ISO 8.3 Fecha de emisión 24 de enero de 2020
Facultad de Ingeniería	Área/Departamento: Laboratorio de Cómputo de Ingeniería Mecatrónica	
La impresión de este documento es una copia no controlada		

debe de agregar la “Referencia de Servicios” en el explorador de soluciones.



6. Bibliografía

- MARTIN, Robert C. **Código limpio: Manual de estilo para el desarrollo ágil de software**. España, Anaya Multimedia, 2012
- CEBALLOS SIERRA, Francisco Javier. **Microsoft C#. Curso de programación**. México, Alfaomega, 2007
- DEITEL, Harvey y Deitel, PAUL. **C# Cómo programar**. España, Pearson, 2007
- LÓPEZ ROMÁN, Leobardo. **Metodología de la programación orientada a objetos**. México, Alfaomega, 2007.
- <https://docs.microsoft.com/es-es/dotnet/framework/wcf/whats-wcf>