FACULTAD DE INGENIERIA
UNIVERSIDAD DE BUENOS AIRES

UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA

95.13 Métodos Matemáticos y Numéricos

TRABAJO PRÁCTICO

APROBACIÓN DEL TRABAJO PRÁCTICO:

.....

AÑO 2020 – VERANO

ALUMNOS:

NOMBRE Y PADRÓN: *Julián Campos 97001*
Marco Chacin 99749
Ignacio Kairuz 99933

INTRODUCCIÓN

En el siguiente trabajo práctico se utilizan métodos de resoluciones numéricas aprendidas en clases, una para ecuaciones lineales y otra para ecuaciones no lineales, y a través de ellas poder resolver un problema cercano al campo de trabajo como ingenieros.

OBJETIVOS

El objetivo del trabajo práctico es obtener el valor de una resistencia para que la corriente que pase por esa rama del circuito sea la deseada, utilizando los métodos de resolución numérica de Gauss-Seidel y Secante, y programándolos desde Visual Basic usando Excell.

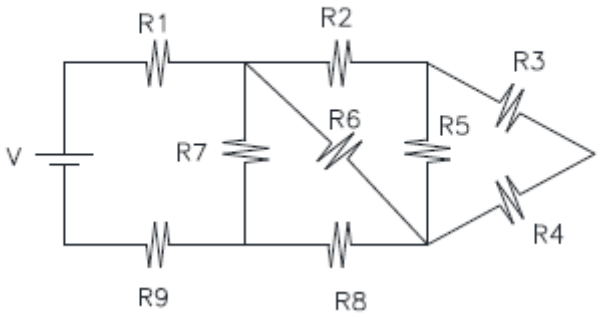
RESUMEN DEL TRABAJO

Con la herramienta de programación Visual Basic en Excel, se programaron los códigos de los métodos numéricos de Gauss-Seidel y Secante, el primero usado para sistemas de ecuaciones lineales (que en este caso fueron dadas por las ecuaciones lineales del circuito eléctrico sobre el cual se trabaja), y el segundo para la resolución de ecuaciones no lineales, el cual no necesita de una función para trabajar ya que depende de los resultados anteriores de la misma, estos resultados se obtuvieron de realizar primero el Gauss-Seidel, y usamos este método para hallar el resultado más exacto del problema. Con ayuda de los datos proporcionados por el circuito eléctrico y a través de la convergencia de los métodos, se pudo hallar el valor deseado de la resistencia R8 (con un cierto margen de error y cierto número de iteraciones) para que la corriente i2 sea 0,75, resolviendo así un problema físico-matemático con métodos de resolución numérica.

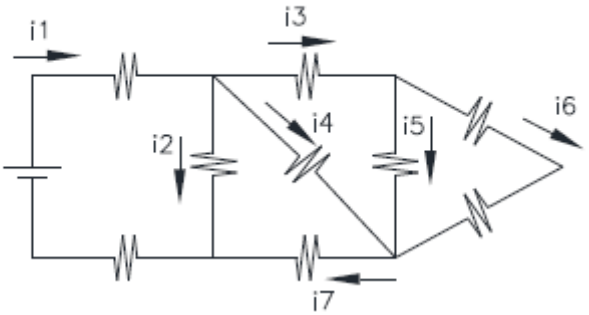
DESARROLLO DEL TRABAJO

Para la realización del trabajo lo primero que se proporcionó el esquema del circuito eléctrico para poder hallar las ecuaciones lineales del circuito a resolver.

Resistencias y fuentes del circuito:



Corrientes:



Esquema 1.

Partiendo del circuito eléctrico, se plantearon las ecuaciones de los nodos y así se pudo obtener un sistema de ecuaciones lineales vinculando una tensión con las resistencias y las corrientes. Para resolver dicho sistema se utilizó el método de Gauss-Seidel, tomando en cuenta sus detalles como componer una matriz diagonal dominante, para que así el método pudiera converger. Los datos proporcionados para realizar las ecuaciones fueron los siguientes:

R1	6
R2	3
R3	5
R4	4
R5	5
R6	3
R7	6
R8	2
R9	4
V	20

Tabla 1

Estos datos fueron puestos en una en Excell para poder asi tomar de la misma todos los valores necesarios al momento de la programacion.

		Matriz de Resolucion de Cicuito Electrico					b	Vector de corrientes inicial
10	6	0	0	0	0	0	20	1
0	6	0	-3	0	0	-2	0	2
0	0	1	0	-1	-1	0	0	1
0	0	-1	-1	0	0	1	0	2
0	0	0	3	-8	-3	0	0	1
0	0	0	0	5	-9	0	0	1
-1	1	0	0	0	0	1	0	1

Imagen 1

El metodo de Gauss-Seidel es un método iterativo, lo que significa que se parte de una aproximación inicial y se repite el proceso hasta llegar a una solución con un margen de error tan pequeño como se quiera. Cumple con la siguiente ecuacion:

$$x_i^{(k)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) / a_{ij}$$

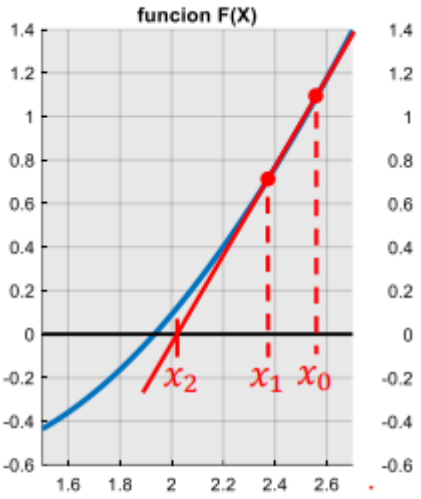
Imagen 2

Se partio de esta ecuacion para poder escribirla en lenguaje de programacion y realizar las primeras iteraciones con los datos de la matriz y asi hallar los primeros “valores semilla” para ser usados despues por el metodo de la secante. El metodo de la Secante es un metodo de convergencia y precision, el cual se usa para refinar las soluciones de ecuaciones no lineales, que en este caso nos ayudo a encontrar el valor deseado de R8. Para realizar la progracion del metodo de Gauss-Seidel se escribio de la siguiente manera:

```
i = 1
cond_corte = 1

While cond_corte > tolerancia

f(1, i + 1) = (b(1) - A(1, 2) * f(2, i)) / A(1, 1)
f(2, i + 1) = (b(2) - A(2, 4) * f(4, i) + Rc(1) * f(7, i)) / A(2, 2)
f(3, i + 1) = (b(3) - A(3, 5) * f(5, i) - A(3, 6) * f(6, i)) / A(3, 3)
f(4, i + 1) = (b(4) - A(4, 3) * f(3, i + 1) - A(4, 7) * f(7, i)) / A(4, 4)
f(5, i + 1) = (b(5) - A(5, 4) * f(4, i + 1) - A(5, 6) * f(6, i)) / A(5, 5)
f(6, i + 1) = (b(6) - A(6, 5) * f(5, i + 1)) / A(6, 6)
f(7, i + 1) = (b(7) - A(7, 1) * f(1, i + 1) - A(7, 2) * f(2, i + 1)) / A(7, 7)

cond_corte = (((f(1, i + 1) - f(1, i)) ^ 2 + (f(2, i + 1) - f(2, i)) ^ 2 + (f(3, i + 1) - f(3, i)) ^ 2 + (f(4, i + 1)

i = i + 1

Wend

h(1) = f(2, i) - Range("b1")

Range("b4") = Rc(1)
Range("a4") = ("Valor semilla 1")
Range("c4") = f(2, i)
```

El metodo de la secante es una variación del método de Newton-Raphson donde en vez de calcular la derivada de la función en el punto de estudio, se aproxima la pendiente a la recta que une la función evaluada en el punto de estudio y en el punto de la iteración anterior. Este metodo es mucho mas facil de programar que el de Newton-Raphson ya que no incluye la derivada, y justo en nuestro caso, donde no conociamos la funcion (definimos una nosotros), nos facilito los calculos ya que usamos dos "valores semilla" para iniciar con la iteraciones. Este metodo no siempre converge, se necesita estar en un rango de valores de la funcion que esten cercanos a la raiz, y por dicho motivo realizamos primero el metodo de Gauss-Seidel y asi hallamos dichos valores. Tambien es orden de convergencia es mayor a otros metodos como el de punto fijo, por lo que se puede hallar la solucion buscada con menor numero de iteraciones.

SECANTE



$$x_{k+1} = x_k - \frac{f(x_k)}{\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}}$$

sin *control*

$\lambda \sim 0.5$

$p \sim 1.6$

Ya con todos los datos y las ecuaciones escritas en lenguaje de programacion se busco la manera de combinar los metodos para asi hallar el valor de R8 que hiciera que la corrientes i2 fuera de 0.75 A. Lo primero fue asignar los valores a variables de programacion, y estos disponerlos en las ecuaciones de los métodos, y a traves de medios iterativos de programacion como *For* y *While* se hicieron trabajar los métodos. En primer lugar se realizaron las iteraciones de dos Gaus-Seidel con valores de R8 supuestos por nosotros sabiendo que estaban cercanos a la solucion, y asi hallar los primeros valores para usar la secante, el mismo proceso se repitio hasta que la respuesta se aproximara lo mas posible con

un error menor a 0,0001

Se hallaba asi un valor distinto pero cercano de R8, y tambien un valor de i2. Se definió una función para que con los valores hallados de i2, se pudiera ir hallando otros valores de R8, a través del método de la secante hasta que convergiera para hallar la corriente deseada, en este caso i2, a cierto valor de R8 que es lo que se buscaba. Se estableció que el error debía ser menor a 0,0001 en este método, al igual que los valores dados que Gauss-Seidel. Se definió una función "f(x)" en la que "x" es el valor de "R8" y "f(x)" la corriente deseada, en este caso i2, para obtener el valor deseado de "f(x)" se define otra función g(x) como f(x) menos el valor deseado de i2, con el objetivo de obtener las raíces de la nueva función. El valor de las raíces de la nueva función será el valor de "i2" tal que "f(R8)" es igual al valor deseado. Para programar el método de la secante se escribió de la siguiente manera:

```
i = 1
C = 1

While C > tolerancia

Rc(i + 2) = Rc(i + 1) - ((h(i + 1) * (Rc(i + 1) - Rc(i))) / (h(i + 1) - h(i)))


cond_corte = 1
j = 1
```

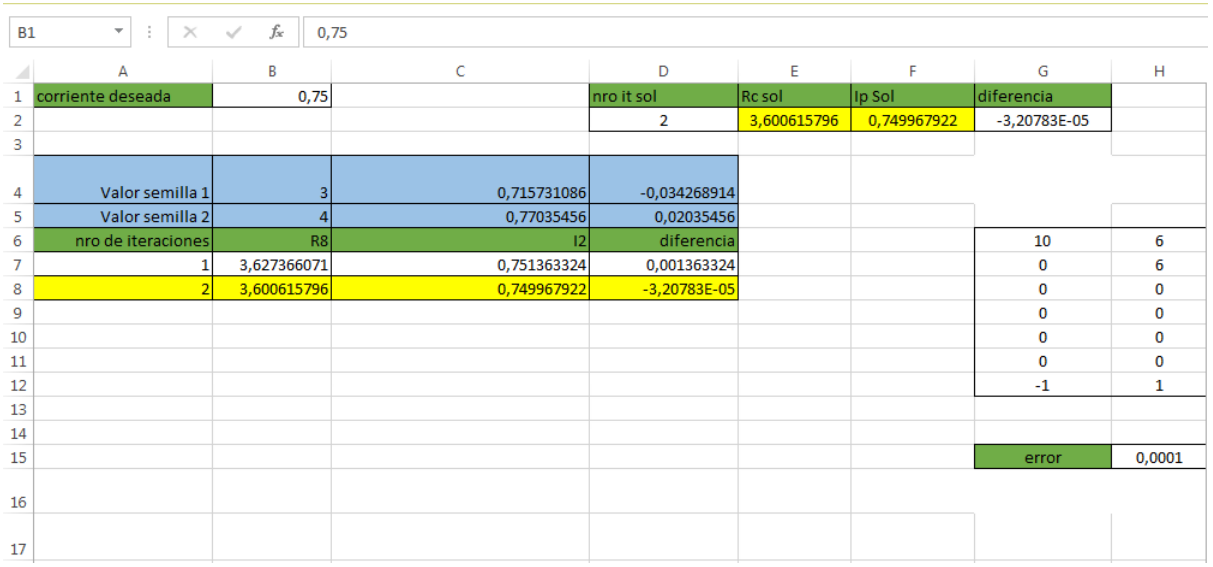
Por último se diseñó una planilla en Excel donde se pudieran ver todos los valores proporcionados y obtenidos a través de la iteración del código, y a través de un botón, se pudiera iniciar dicho proceso para encontrar el valor de la resistencia R8.

CONCLUSION

Los métodos numéricos aprendidos en clases pueden ser muy útiles para la resolución de problemas donde se necesiten resolver grandes sistemas de ecuaciones lineales. Siempre y cuando se cumplan las condiciones de los métodos (para que puedan converger y funcionar), los mismos darán una solución bastante aproximada de la solución verdadera que se busca; también dependerá de la cota de error que se maneje. En el caso de Gauss-Seidel se tiene que tener siempre en cuenta que la matriz sea "diagonal dominante" y se usó debido a que el otro método, que es el de Jacobi, no converge de manera tan rápida como este. Para el método de la secante vimos que su escritura en programación no es tan complicada y tanto su convergencia y aproximación son bastantes rápidas y exactas. A través de las herramientas de programación como lo son el *For* y *While* se pudieron hacer iterar los métodos hasta hallar las soluciones con la precisión deseada, que para nosotros era un error menor a 0.0001. Despues de primero realizar las iteraciones con el método de Gauss-Seidel, con dos iteraciones del método de la secante se pudo hallar el valor de R8 aproximado que fue de

3,600615796, con un valor para i2 de 0,7499679217, los dos valores con un error menor al ya menciado.

ANEXO 1 (Corrida del Programa)



B1 | 0,75

	A	B	C	D	E	F	G	H
1	corriente deseada	0,75		nro it sol	Rc sol	Ip Sol	diferencia	
2				2	3,600615796	0,749967922	-3,20783E-05	
3								
4	Valor semilla 1	3	0,715731086	-0,034268914				
5	Valor semilla 2	4	0,77035456	0,02035456				
6	nro de iteraciones	R8	I2	diferencia			10	6
7	1	3,627366071	0,751363324	0,001363324			0	6
8	2	3,600615796	0,749967922	-3,20783E-05			0	0
9							0	0
10							0	0
11							0	0
12							-1	1
13								
14								
15							error	0,0001
16								
17								

ANEXO 2 (Código en texto)

```
Sub CommandButton1_Click()


Dim A(1 To 7, 1 To 7) As Double
Dim b(1 To 7) As Double
Dim f(1 To 7, 1 To 1000) As Double
Dim Rc(1000) As Double
Dim h(1000) As Double
Dim tolerancia As Double



For t = 1 To 7
b(t) = Range("o6").Cells(t)
Next t



For o = 1 To 7
For p = 1 To 7
A(p, o) = Range("g6").Cells(p, o)
Next p
Next o
```

```
For t = 1 To 7
f(t, 1) = Range("p6").Cells(t)
Next t



tolerancia = Range("h15")

Rc(1) = -A(2, 7)

Rc(2) = -Range("m14")




i = 1
cond_corte = 1

While cond_corte > tolerancia

f(1, i + 1) = (b(1) - A(1, 2) * f(2, i)) / A(1, 1)
f(2, i + 1) = (b(2) - A(2, 4) * f(4, i) + Rc(1) * f(7, i)) / A(2, 2)
f(3, i + 1) = (b(3) - A(3, 5) * f(5, i) - A(3, 6) * f(6, i)) / A(3, 3)
f(4, i + 1) = (b(4) - A(4, 3) * f(3, i + 1) - A(4, 7) * f(7, i)) / A(4, 4)
f(5, i + 1) = (b(5) - A(5, 4) * f(4, i + 1) - A(5, 6) * f(6, i)) / A(5, 5)
f(6, i + 1) = (b(6) - A(6, 5) * f(5, i + 1)) / A(6, 6)
f(7, i + 1) = (b(7) - A(7, 1) * f(1, i + 1) - A(7, 2) * f(2, i + 1)) / A(7, 7)

cond_corte = (((f(1, i + 1) - f(1, i)) ^ 2 + (f(2, i + 1) - f(2, i)) ^ 2 + (f(3, i + 1) - f(3, i)) ^ 2 +
(f(4, i + 1) - f(4, i)) ^ 2 + (f(5, i + 1) - f(5, i)) ^ 2 + (f(6, i + 1) - f(6, i)) ^ 2 + (f(7, i + 1) - f(7,
i)) ^ 2) ^ 0.5)

i = i + 1

Wend

h(1) = f(2, i) - Range("b1")

Range("b4") = Rc(1)
Range("a4") = ("Valor semilla 1")
Range("c4") = f(2, i)




i = 1
cond_corte = 1
```

```
While cond_corte > tolerancia

f(1, i + 1) = (b(1) - A(1, 2) * f(2, i)) / A(1, 1)
f(2, i + 1) = (b(2) - A(2, 4) * f(4, i) + Rc(2) * f(7, i)) / A(2, 2)
f(3, i + 1) = (b(3) - A(3, 5) * f(5, i) - A(3, 6) * f(6, i)) / A(3, 3)
f(4, i + 1) = (b(4) - A(4, 3) * f(3, i + 1) - A(4, 7) * f(7, i)) / A(4, 4)
f(5, i + 1) = (b(5) - A(5, 4) * f(4, i + 1) - A(5, 6) * f(6, i)) / A(5, 5)
f(6, i + 1) = (b(6) - A(6, 5) * f(5, i + 1)) / A(6, 6)
f(7, i + 1) = (b(7) - A(7, 1) * f(1, i + 1) - A(7, 2) * f(2, i + 1)) / A(7, 7)

cond_corte = (((f(1, i + 1) - f(1, i)) ^ 2 + (f(2, i + 1) - f(2, i)) ^ 2 + (f(3, i + 1) - f(3, i)) ^ 2 +
(f(4, i + 1) - f(4, i)) ^ 2 + (f(5, i + 1) - f(5, i)) ^ 2 + (f(6, i + 1) - f(6, i)) ^ 2 + (f(7, i + 1) - f(7,
i)) ^ 2) ^ 0.5)


i = i + 1
Wend


h(2) = f(2, i) - Range("b1")

Range("b5") = Rc(2)
Range("a5") = ("Valor semilla 2")
Range("c5") = f(2, i)


i = 1
C = 1

While C > tolerancia

Rc(i + 2) = Rc(i + 1) - ((h(i + 1) * (Rc(i + 1) - Rc(i))) / (h(i + 1) - h(i)))


cond_corte = 1
j = 1

While cond_corte > tolerancia

f(1, j + 1) = (b(1) - A(1, 2) * f(2, j)) / A(1, 1)
f(2, j + 1) = (b(2) - A(2, 4) * f(4, j) + Rc(i + 2) * f(7, j)) / A(2, 2)
f(3, j + 1) = (b(3) - A(3, 5) * f(5, j) - A(3, 6) * f(6, j)) / A(3, 3)
f(4, j + 1) = (b(4) - A(4, 3) * f(3, j + 1) - A(4, 7) * f(7, j)) / A(4, 4)
f(5, j + 1) = (b(5) - A(5, 4) * f(4, j + 1) - A(5, 6) * f(6, j)) / A(5, 5)
f(6, j + 1) = (b(6) - A(6, 5) * f(5, j + 1)) / A(6, 6)
f(7, j + 1) = (b(7) - A(7, 1) * f(1, j + 1) - A(7, 2) * f(2, j + 1)) / A(7, 7)
```

```
cond_corte = (((f(1, j + 1) - f(1, j)) ^ 2 + (f(2, j + 1) - f(2, j)) ^ 2 + (f(3, j + 1) - f(3, j)) ^ 2 +
(f(4, j + 1) - f(4, j)) ^ 2 + (f(5, j + 1) - f(5, j)) ^ 2 + (f(6, j + 1) - f(6, j)) ^ 2 + (f(7, j + 1) - f(7,
j)) ^ 2) ^ 0.5)

j = j + 1
Wend


Range("e2") = Rc(i + 2)
Range("d2") = i
Range("f2") = f(2, j)



Range("b7").Cells(i) = Rc(i + 2)
Range("a7").Cells(i) = i
Range("c7").Cells(i) = f(2, j)


i = i + 1

h(i + 1) = f(2, j) - Range("b1")


C = ((f(2, j) - Range("b1")) ^ 2) ^ 0.5

Wend


End Sub
```