

En búsqueda de la capacitancia del Hilbertrón, un viaje por el método de relajación con ansiedad (diferencias finitas centradas de cuarto orden).

Azpeitia Arias, Ángel Alejandro.
alejandroazp@ciencias.unam.mx

Pérez Flores, Julio Alfonso.
julio_perez@ciencias.unam.mx

Pérez Portillo, Sonia María.
perezportillos@ciencias.unam.mx

Facultad de Ciencias, Universidad Nacional Autónoma de México.
01 Diciembre, 2023.

Resumen.

Se caracterizo la capacitancia de un condensador coplanar con electrodos basados en seudo curva de Hilbert de segundo orden (Hilbertrón), mediante el método de relajación modificado usando diferencias finitas centradas de cuarto orden, con tolerancia de 2.22×10^{-7} , además se expone el comportamiento del campo eléctrico, el potencial y la carga.

Abstract: The capacitance of a coplanar capacitor with electrodes based on a second-order Hilbert pseudo-curve (called the Hilbertrón) was characterized using the modified relaxation method employing fourth-order centered finite differences, with a tolerance of 2.22×10^{-7} . Furthermore, the behavior of the electric field, potential, and charge is presented.

Keywords: Space-filling Curve based Electrodes, Coplanar Capacitor.

1. Introducción.

1.1. State of Art.

Un condensador coplanar es un componente electrónico conformado por dos electrodos ubicados en el mismo plano cubiertos por un medio dieléctrico, es importante destacar que la capacitancia de este arreglo tiene atribuciones derivadas de un campo eléctrico uniforme entre el grosor de los electrodos y otro no uniforme que se dispersa alrededor del espacio en los extremos de los electrodos, conocido como efecto fringe [1] [2]. Cundo un material externo atraviesa este ultimo campo eléctrico, la capacitancia asociada es modificada, por lo que esta configuración de electrodos es útil para caracterizar cambios en constantes dieléctricas, por el cambio del material o sus propiedades, alguno de los ejemplos donde esta aplicación es útil es en las técnicas de evaluación no destructivas [3] [4], la medición de humedad relativa en suelos [5][6], y en la detección de fugas de humedad en edificaciones [7].

Abdollahi-Mamoudan et al.[8] muestra que los factores determinantes en la obtención de un valor de capacitancia óptimo son la geometría, el espacio entre placas y la frecuencia, cabe destacar que mientras mas área de electrodos tenga frontera con la otra placa, y la separación entre estas sea mas chica, la capacitancia aumenta. Una forma de poder acreditar las condiciones anteriores es mediante el uso de curvas de llenado del espacio [9] de grosor controlado que separan los electrodos. Debido a que la capacitancia se obtiene mediante el desarrollo de la ecuación de Laplace o Poisson, la geometría vuelve a jugar un papel fundamental en la obtención de la capacitancia de forma analítica. Algunas geometrías triviales como el de dos tiras coplanares han sido solucionadas mediante el uso de integrales de Cauchy [10], algunas geometrías más complejas como un arreglo de anillos han sido solucionadas mediante el uso de la transformada de Hankel [7], no obstante; Parker [11], Naghed y Wolff, [12] y Campbell [13] han mostrado que este problema es soluble mediante méto-

do de diferencias finitas.

Es por esto que este trabajo se propone la caracterización de la capacitancia de una geometría basada en una pseudo-curva de Hilbert de segundo orden, mediante el método de relajación [14], modificándolo de tal forma que se utilices diferencias finitas centradas de cuarto orden. Esto con la finalidad de obtener resoluciones a la ecuación de Laplace y al campo eléctrico de geometrías complejas, así como realizar una discusión de la viabilidad de este método para la obtención teórica de condensadores para la medición de humedad relativa en suelos.

El condensador coplanar con electrodos basados en una pseudo curva de Hilbert analizado en este trabajo fue realizado en el contexto del desarrollo del proyecto "El Ecomático 3000", en donde se hace uso este condensador en una sonda para medir humedad del suelo, a esta implementación se le llamó el Hilbertrón.

1.2. Marco Teórico.

Para obtener la capacitancia, se necesita calcular la carga en el dispositivo. Se sabe que las cargas eléctricas generan campos eléctricos y magnéticos, que cumplen las cuatro ecuaciones de Maxwell, una de ellas es la ley de Gauss.

$$\nabla \cdot \vec{E} = \frac{\rho_{total}}{\epsilon_0} \quad (1)$$

Además, para el caso electrostático (las cargas no se mueven), se cumple la igualdad $\vec{E} = -\nabla\phi$, remplazando el campo eléctrico en la ley de Gauss se obtiene ec. 2

$$\nabla^2\phi = \frac{-\rho_{total}}{\epsilon_0} \quad (2)$$

Y en una región del espacio donde no hay carga eléctrica, se llega a la ecuación que es nombrada como ecuación de Laplace ec. 3

$$\nabla^2\phi = 0 \quad (3)$$

$$\phi(x, y) \approx \frac{16\phi(x-1, y) + 16\phi(x, y-1) + 16\phi(x+1, y) + 16\phi(x, y+1) - \phi(x-2, y) - \phi(x, y-2) - \phi(x+2, y) - \phi(x, y+2)}{60} \quad (4)$$

Ecuación 4: Aproximación del Laplaciano mediante diferencias finitas centradas de cuarto orden.

$$-\frac{\rho(x, y)}{\epsilon_0} \approx \frac{16\phi(x-1, y) + 16\phi(x, y-1) + 16\phi(x+1, y) + 16\phi(x, y+1) - \phi(x-2, y) - \phi(x, y-2) - \phi(x+2, y) - \phi(x, y+2) - 60\phi(x, y)}{12h^2} \quad (5)$$

Ecuación 5: Aproximación de la ecuación de Poisson, mediante diferencias finitas centradas de cuarto orden.

Se sabe que por el teorema de existencia y unicidad, que la solución a una ecuación diferencial es única, por lo tanto si encontramos una solución que cumpla las condiciones de frontera esta es la única solución al problema.

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial x^2} \quad (6)$$

La ecuación de Laplace puede ser aproximada por el método de relajación, que consiste en aproximar las segundas derivadas parciales por método de diferencias centrales de cuarto orden, como se muestra en la ecuación **ec. 8** y **ec. 7**, análogamente para y . En la ecuación **ec. 6** se sustituyen las derivadas centradas, por **ec. 3** se iguala a 0, despejando $\phi(x, y)$, se encuentra que el valor de cada celda del mallado se approxima con el promedio de sus cuatro vecinos además haciendo $h = 1$ se obtiene **ec. 4**. Como consecuencia de la extensión de esta expresión y del hecho de que el método de relajación utilizado en electrodinámica solo utiliza diferencias finitas de segundo orden, se donomina a este procedimiento en el actual trabajo como método de relajación con ansiedad.

$$\frac{d^2\phi}{dx^2} \approx \frac{-\phi_{i+2} + 16\phi_{i+1} - 30\phi_i + 16\phi_{i-1} - \phi_{i-2}}{12h^2} \quad (7)$$

$$\frac{d^2\phi}{dy^2} \approx \frac{-\phi_{j+2} + 16\phi_{j+1} - 30\phi_j + 16\phi_{j-1} - \phi_{j-2}}{12h^2} \quad (8)$$

donde $i+k = x_i + kh$, $j+k = x_j + kh$, con x_i, x_j puntos iniciales

Para calcular el campo eléctrico (por completez), se usa el método de diferencias centrales en cada componente del campo eléctrico partiendo de $\vec{E} = -\nabla\phi$

$$\vec{E} \approx \left(-\frac{\phi(x+h, y) - \phi(x-h, y)}{2h}, -\frac{\phi(x, y+h) - \phi(x, y-h)}{2h} \right) \quad (9)$$

Como ya se tiene calculado el potencial en toda el área y sabiendo que la ecuación **ec. 2** se cumple en toda región del espacio, nuevamente podemos aplicar diferencias finitas y llegar a la expresión **ec. 5**

La capacitancia se define como $C = \frac{Q}{\Delta\phi}$, para calcular la carga total. Donde Q es la carga positiva, pues si la aproximación es buena, tendremos la misma cantidad de carga positiva y negativa.

$$Q = \int \rho(x, y) dx dy \approx \sum_i \sum_j \rho(i, j) \quad (10)$$

2. Método.

Un condensador coplanar bloque pensado para poder construir condensadores de mayor capacitancia mediante un arreglo lineal de n de estos bloques, al cual denominamos el Hilbertrón, fue diseñado bajo la siguiente premisa; en un rectángulo de **2046 x 2430 μm** se trazó una pseudo-curva de Hilbert de segundo orden, con **127 μm** de ancho, a la parte superior desconexa se le asignó potencial positivo además se conectan los tres conjuntos mediante una tira de **192 μm** , la parte conexa se le asignó potencial neutro, y solo por completez se le agrega una tira de **192 μm** de ancho, y del largo del rectángulo, la **fig. 1**, muestra un diagrama del capacitor y sus dimensiones.

Para la implementación descrita en la **ec. 4**, se empleo un programa escrito en Rust el cual genera un mallado de **2046 x 2430**, con un arreglo matricial, donde cada entrada de la matriz representa un área de **1 μm x 1 μm** . Las regiones de los electrodos fueron llenadas con el valor del potencial descrito en la construcción, en las regiones en color blanco **fig. 1 a)** se asigno un potencial de **1.75 V**, para reducir el tiempo de convergencia. Se aplicaron **5000** iteraciones sobre la operación **ec. 4** en la región en blanco, para respetar las condiciones de frontera (bordes de los electrodos), e igualmente observar como se comporta el potencial. Con este potencial se calcula el campo eléctrico en toda el área con la ecuación **ec.9**, finalmente con ayuda de **ec. 5** y **ec. 10**, se muestra la distribución de carga y la carga total. Finalmente la matriz de potencial discreto, la matriz con los componentes E_x, E_y y la matriz de carga discreto, fueron exportadas en archivos **.txt** y se generaron gráficas en Python con la librería **Matplotlib**, los códigos se pueden apreciar en el **apéndice A**.

3. Resultados y Discusión.

Se realizaron **1000** iteraciones con una tolerancia **2.22×10^{-7}** . Al calcular la matriz de potencial con el método de relajación sobre la región blanca se obtiene la gráfica **fig. 2 a)**. Donde se aprecia que el potencial tiene un cambio suave.

Se aprecia mejor el cambio del potencial en la gráfica **fig. 2 b)** donde es claro que siguen el contorno de los electrodos, con suavización en las esquinas, donde hay efectos de borde.

Las gráficas del campo eléctrico **fig. 3 a)** y **b)** muestran líneas perpendiculares a las líneas equipotenciales, resultado

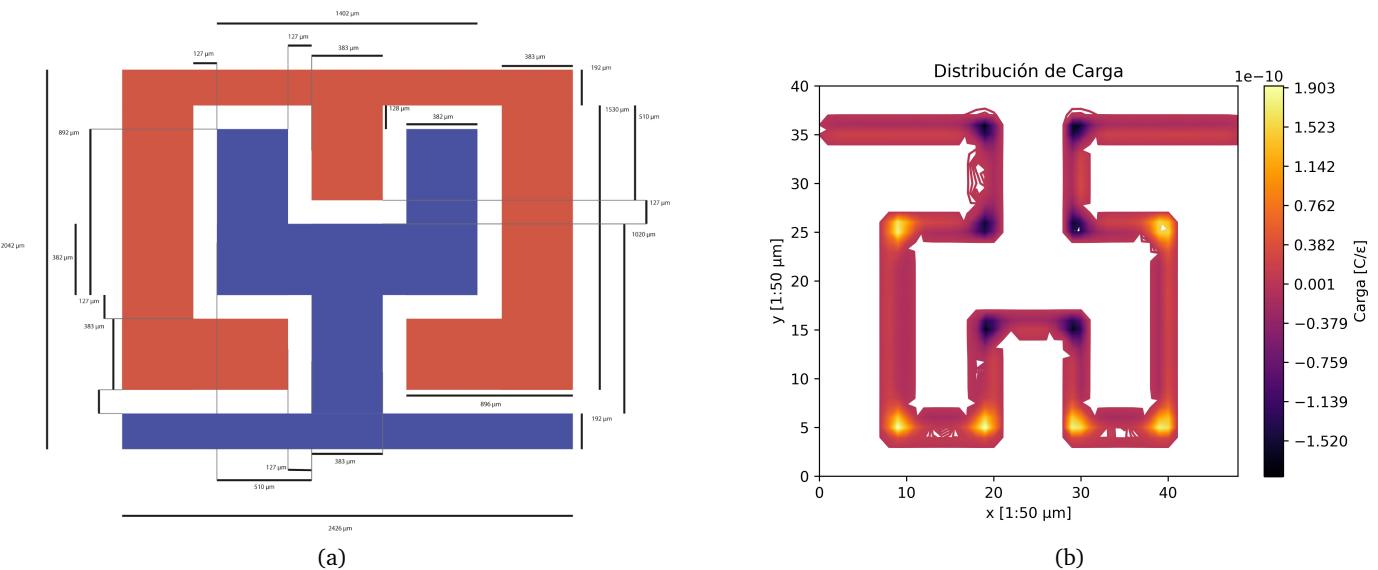


Figura 1: (a): Construcción del Hilbertrón. (b): Densidad de carga calculada y a escala , debido al exceso de elementos con carga neutra, estos fueron retirarlos de la gráfica para una mejor visualización del resultado, no obstante las regiones en color uva tienden a $0 \frac{C}{\epsilon}$.

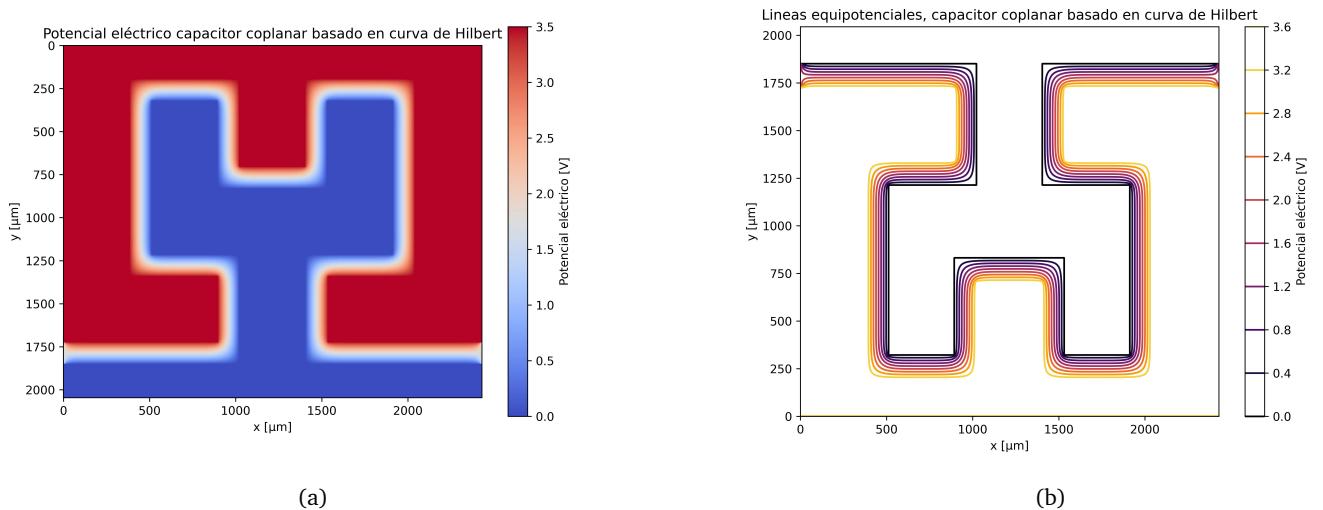


Figura 2: (a): Potencial eléctrico obtenido mediante la solución de la ecuación de Laplace con diferencias finitas de cuarto orden. (b): Lineas equipotenciales del potencial calculado.

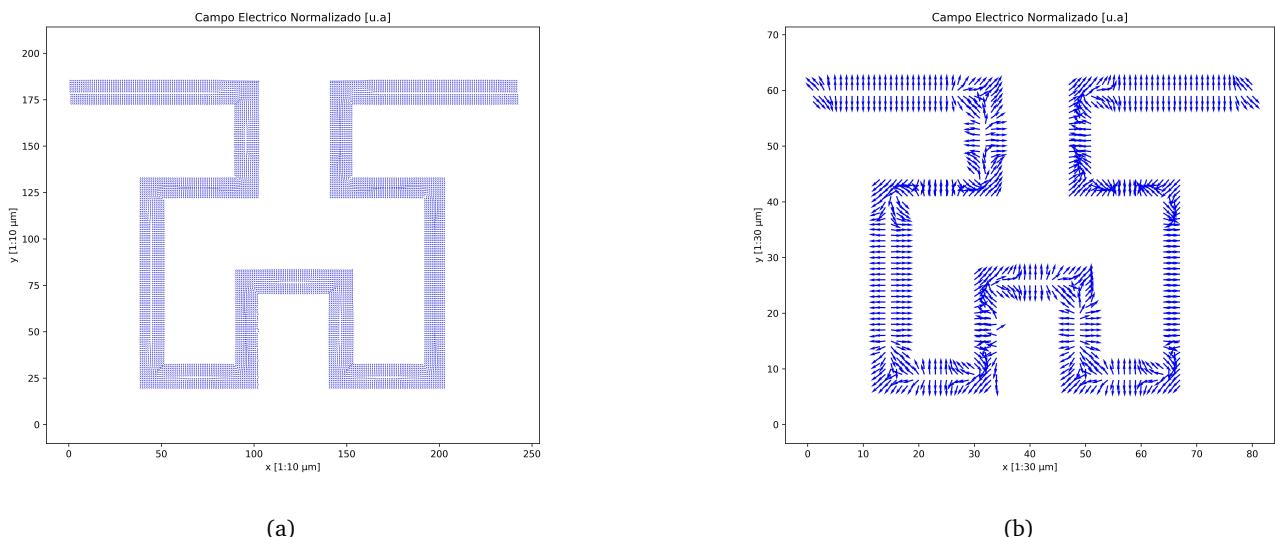


Figura 3: (a): Campo eléctrico calculado mediante el gradiente de la matriz de potencial discreto. El campo se aprecia normalizado (flechas de magnitud 1), con fines de mostrar la dirección (Escala: 1:10 μm) (b): El mismo campo eléctrico, bajo las misma condiciones con un escalamiento de 1:30 μm

esperado con base a la relación potencial - campo eléctrico, van del electrodo con potencial positivo al negativo, lo que indica la presencia de carga positiva y negativa, pues las líneas de campo nacen en las positivas y mueren en las negativas.

Finalmente al calcular la densidad de carga se obtiene la gráfica **fig. 1 b)**, donde podemos apreciar que la carga positiva se acumula en el electrodo con voltaje positivo y negativa en el electrodo con voltaje 0. Al sumar la carga positiva en se obtiene $Q = 638.2516$ pesos y la diferencia de voltaje es **3.5 V** con lo que la capacitancia es:

$$C = \frac{638.2516\epsilon}{3.5V}$$

Donde ϵ , es la constante dielectrica del material.

Si bien el método presenta resultados acorde a los principios de la electrodinámica, cabe destacar que estos presentan distorsión por efecto de borde, visibles en las **fig. 2 b)** y **fig. 3 b)**, producto de las condiciones de frontera y de la construcción del método. Estas afectaciones se ven principalmente en la convergencia del método, puesto que para las primeras iteraciones se genera una falsa convergencia en la iteración 643, existiendo demasiadas regiones con el potencial especulativo inicial, no obstante; las distorsiones de bordes aumentan en el mallado con base a las iteraciones tienden al infinito, por esta razón un numero de iteraciones fue impuesto en vez de aplicar una condición de tolerancia para terminar las iteraciones. Así como, en la carga acumulada, en primera instancia este problema se quería abordar mediante la discontinuidad del campo eléctrico, debido a que en las esquinas aparecen estas distorsiones por efecto de borde también se presenta una distorsión en la carga acumulada. Para futuras referencias este efecto puede ser reducido redondeando los bordes del condensador, y aumentando la escala de la matriz.

Otro inconveniente para poder establecer resultados fidedignos, es la falta de datos experimentales para realizar comparaciones en la capacitancia. Debido a que la capacitancia varia en función de la constante dielectrica del material, para realizar una caracterización

de la capacitancia fiable, un condensador coplanar debes ser construido con una capa de grosor homogénea de un material con función dieléctrica bien estudiada, a temperaturas controladas y frecuencias bajas. Desafortunadamente no fue posible acceder a un método para depositar películas de grosor homogéneo en el condensador.

En consecuencia al párrafo anterior establecer una frontera en el plano Z tampoco no fue posible, la falta de estas aportaciones modifica el valor de la capacitancia, sin embargo; debido a que no existe nada que contenga al campo eléctrico en la componente z, una de las condiciones de frontera para resolver el problema seria de tipo Neumann $\left. \frac{\partial \phi}{\partial z} \right|_h = 0$, donde h es la distancia de las placas al fin del material dieléctrico, por lo que de considerar estas fronteras el código debe ser modificado para poder usar condiciones de frontera Neumann y no solo Dirichlet como las usadas en este acercamiento.

4. Conclusiones.

Las gráficas y resultados coinciden con lo esperado según los principios de la electrostática, además los avances realizados en este estudio muestran que el método es útil para la resolución de la ecuación de Laplace en electrostática, a la par que permite realizar un acercamiento didáctico de las interacciones del campo eléctrico y el potencial en geometrías complejas, no obstante para asegurarnos de la veracidad de los resultados se tendría que hacer la comparación con los resultados experimentales.

Agradecimientos.

Agradecemos a **Reyes Coronado Alejandro**, **Viveros Armas Eduardo Enrique** (*departamento de Física, Facultad de Ciencias*), por la revisión del método y el ajuste en las condiciones de frontera; así como a **López Aparicio Jehú** (*laboratorio de electrónica, departamento de Física, Facultad de Ciencias*), por iniciar los trámites para tener acceso a equipo de Spin Coating, con la finalidad de realizar comparaciones experimentales en un futuro cercano.

Apéndice A: Código fuente del método en RUST y gráfadora en Python.

Método de diferencias finitas centras de cuarto en orden en Rust

```
1 /*La idea general de código es que resuelve la ecuación de Laplace utilizando diferencias
2 centradas de cuarto orden, para ello se genera una matriz de 2046x430 donde cada índice
3 representa 1 um y se rellena con un valor arbitrario (en este caso la mitad del potencial
4 más grande)
5
6 https://www.dam.brown.edu/people/alcyew/handouts/numdiff.pdf
7
8 Posteriormente se proponen las fronteras de tipo Dirichlet mutando los valores de algunas zonas
9 de la matriz con respecto al potencial de nuestros electrodos 5 y 0 Volts y a la geometría de
10 nuestros electrodos (ver PDF).
11
12 Finalmente se busca el conjunto de pares ordenados donde los valores con los que se relleno la
13 matriz en un principio no fueron mutados. Esto es importante porque, debido a que el potencial
14 de nuestra frontera no debe ser mutado por la diferencia finita entonces solo iteramos en esa
15 colección de índices y con eso ahorraremos tiempo por cada paso que hace
16
17 En otro proceso aparte exporta los datos de la matriz cuando ya se terminaron de hacer 10000
18 iteraciones en un archivo txt con escritura paralela (para no demorarse tanto tiempo)
19 para poder graficar en python porque no sabemos hacerlo en Rust.*/
20
21 use rayon::prelude::*; //librería de escritura paralela
22 use std::fs::File; //elemento de la librería estándar de Rust que accesa rutas del ordenador
23 use std::io::{self, Write}; //método escribir sobre objeto de la librería estándar de Rust
24
25 //número máximo de iteraciones
26 const MAX_ITER: usize = 5000;
27
28 //Declaración de nuestras funciones
29 //-----
```



```
30
31 fn copiar_matriz_a_vector(matriz: &Vec<Vec<f32>>, indices: &Vec<(usize, usize)>) -> Vec<f32> {
32     let mut resultado = Vec::new();
33
34     for &(fila, columna) in indices {
35         if let Some(fila_matriz) = matriz.get(fila) {
36             if let Some(valor) = fila_matriz.get(columna) {
37                 resultado.push(*valor);
38             } else {
39                 panic!("índices de columna fuera de rango en la fila {}", fila);
39             }
34         } else {
35             panic!("índice de fila fuera de rango: {}", fila);
36         }
37     }
38
39     resultado
37 }
38
39 fn restar_vectores(a: &Vec<f32>, b: &Vec<f32>) -> Vec<f32> {
40     if a.len() != b.len() {
41         panic!("Los vectores deben tener la misma longitud para realizar la resta.");
42     }
43
44     let mut resultado = Vec::with_capacity(a.len());
45
46     for i in 0..a.len() {
47         resultado.push(a[i] - b[i]);
48     }
49
50     resultado
```

```

61 }
62
63 fn obtener_promedio(arr:&Vec<f32>) -> Option<f32> {
64     let longitud = arr.len();
65
66     if longitud == 0 {
67         None // No se puede calcular el promedio de un array vaco
68     } else {
69         let suma: f32 = arr.iter().sum();
70         Some(suma / (longitud as f32))
71     }
72 }
73
74 /* Funcin para crear una matriz de m filas por n columnas con todos sus elementos valiendo
   1.75.
Este constructor usa la estructura vector de vectores */
75
76 fn crear_matriz(m: usize, n: usize, fill: f32) -> Vec<Vec<f32>> {
77     let mut matriz = Vec::with_capacity(m);
78
79     for _ in 0..m {
80         let fila: Vec<f32> = vec![fill; n];
81         matriz.push(fila);
82     }
83
84     matriz
85 }
86
87
88 /*En esta funcion lo que se hace es realizar dos iteraciones sobre una matriz mutables y
   obtener sus
respectivos indices i,j si el valor del elemento matriz[i][j] es igual a un valor buscado, nos
   regresa
un vector con una coleccin de pares ordenados.*/
89
90 fn obtener_valores(matriz: &mut Vec<Vec<f32>>, valor_buscado: f32) -> Vec<(usize, usize)> {
91     let mut indices_a_cambiar = Vec::new();
92
93     for (i, fila) in matriz.iter_mut().enumerate() {
94         for (j, valor) in fila.iter_mut().enumerate() {
95             if *valor == valor_buscado {
96                 // Guardar los ndices
97                 indices_a_cambiar.push((i, j));
98             }
99         }
100    }
101
102    indices_a_cambiar
103 }
104
105 /*
106 Esta funcin, llamada write_matrix_to_file_parallel, toma una matriz mutable de nmeros de punto
   flotante (f32)
107 y un camino de archivo (file_path), y escribe el contenido de la matriz en un archivo, mediante
   escritura paralela.
108 */
109
110 fn write_matrix_to_file_parallel(matrix: &mut Vec<Vec<f32>>, file_path: &str) -> io::Result<()>
111 {
112     let formatted_rows: Vec<String> = matrix
113         .par_iter_mut()
114         .map(|row| {
115             // Formatear la fila como una cadena separada por tabulaciones
116             row.iter_mut()
117             .map(|num| num.to_string())
118             .collect::<Vec<String>>()
119             .join("\t")
120         })

```

```

121     })
122     .collect();
123
124     let mut file = File::create(file_path)?;
125
126     for row_str in formatted_rows {
127         // Escribir la fila en el archivo seguido de un salto de linea
128         writeln!(file, "{}", row_str)?;
129     }
130
131     Ok(())
132 }
133 // -----
134
135 //Ejecucion Principal.
136 // -----
137 fn main() {
138     // Número de filas y columnas
139
140     /*Cabe aclarar que se agregaron cuatro puntos mas para satisfacer las condiciones iniciales
141      del
142      metodo de diferencias finitas centradas de cuarto orden y no perder informacion, i.e i<=2, e
143      i<= n-2, de tal forma que el termino i+2 corresponde al 0 de nuestro sistema real */
144     let m = 2046;
145     let n = 2430;
146
147     // Crear la matriz
148
149     let mut phi: Vec<Vec<f32>> = crear_matriz(m, n, 1.75);
150
151     /* En ambos casos (potencial positivo y potencial 0), lo que se hace es iterar sobre las
152      filas primero
153      y despues sobre las columnas. si el indice de las columnas se encuentra en una de las
154      colecciones de
155      puntos mencionadas (las cuales corresponden geometricamente a los electrodos del capacitor
156      coplanar),
157      entonces modifica el valor de la matriz al del potencial deseado.*/
158
159     //potencial positivo
160     for i in 2..m - 2 {
161         if (0..=194).contains(&i) {
162             for j in 0..n {
163                 phi[i][j] = 3.5;
164             }
165         }
166
167         if (194..=704).contains(&i) {
168             for range in [(0..385), (1021..1404), (2040..n)] {
169                 for j in range {
170                     phi[i][j] = 3.5;
171                 }
172             }
173         }
174
175         if (704..=1340).contains(&i) {
176             for range in [(0..385), (2040..n)] {
177                 for j in range {
178                     phi[i][j] = 3.5;
179                 }
180             }
181         }
182     }
183 }
```

```

179     if (1340..=1723).contains(&i) {
180         for range in [(0..898), (1535..n)] {
181             for j in range {
182                 phi[i][j] = 3.5;
183             }
184         }
185     }
186 }
187
188 //potencial 0
189 for i in 0..m {
190     if (322..=832).contains(&i) {
191         for range in [(512..894), (1531..1913)] {
192             for j in range {
193                 phi[i][j] = 0.0;
194             }
195         }
196     }
197
198     if (832..=1214).contains(&i) {
199         for j in 512..1913 {
200             phi[i][j] = 0.0;
201         }
202     }
203
204     if (1214..=1851).contains(&i) {
205         for j in 1022..1405 {
206             phi[i][j] = 0.0;
207         }
208     }
209
210     if i >= 1851 {
211         for j in 0..n {
212             phi[i][j] = 0.0;
213         }
214     }
215 }
216
217 //obtener indices a iterar
218 let indices = obtener_valores(&mut phi, 1.75);
219
220 //iteraciones del metodo
221 for _k in 1..MAX_ITER {
222     /* Como la colección de parejas ordenadas que obtuvimos en la sección anterior no es
223      accesible para operar en este ciclo las clonamos (supongo que hay una forma más óptima
224      de pasárselas al scope) pero se itera sobre esa colección de puntos
225
226      Sabemos que en esa colección de puntos hay puntos que no cumplen las condiciones
227      iniciales
228      si i<=2, e i<= n-2, entonces forazmos a que solo itere sobre los puntos que si cumplen
229      esa
230      condición y evalúe el nuevo valor de phi (chechar foto del desglose de la solución de la
231      ecuación de Laplace en la foto) */
232
233     let phiant=copiar_matriz_a_vector(&phi, &indices);
234     for (i, j) in &indices {
235         // Verificar que los índices estén dentro del rango de la matriz
236         if *i < phi.len() && *j < phi[0].len() {
237             if *i >= 2 && *j >= 2 && *i + 2 < phi.len() && *j + 2 < phi[0].len() {
238                 phi[*i][*j] = (16.0 * phi[*i - 1][*j]
239                             + 16.0 * phi[*i][*j - 1]
240                             + 16.0 * phi[*i + 1][*j]
241                             + 16.0 * phi[*i][*j + 1]
242                             - 1.0 * phi[*i - 2][*j]
243                             - 1.0 * phi[*i][*j - 2]
244                             - 1.0 * phi[*i + 2][*j]

```

```

243             - 1.0 * phi[*i][*j + 2])
244             / 60.0;
245         }
246     }
247 }
248
249     let tol = match obtener_promedio(&restar_vectores(&copiar_matriz_a_vector(&phi, &
250 indices), &phant)) {
251     Some(mean) => mean.abs(),
252     None => {
253         eprintln!("Error calculating mean. Exiting.");
254         return;
255     }
256 };
257
258     println!("Iteracion n: {}, tol: {}.", _k, tol);
259 }
260
261 /*hacemos las matrices de las componentes en x y Y del campo electrico de nuevo de la
262 matriz de potencial hay
263 cuatro puntos mas para satisfacer las condiciones iniciales del metodo de diferencias
264 finitas centradas de cuarto
265 orden pero no nos importan y los quitamos de las matrices resultantes
266 */
267
268 let mut ex: Vec<Vec<f32>> = crear_matriz(m, n, 0.0); // Componente x del campo electrico
269 let mut ey: Vec<Vec<f32>> = crear_matriz(m, n, 0.0); // Componente y del campo electrico
270
271 /*Iteramos sobre las matrices Ey Ex y Phi para aplicar el operador numerico gradiente el
272 cual es diferencia finita
273 centrada de cuarto orden para los elementos de x dejando a y fija y la diferencia finita
274 centrada de cuarto orden para
275 los elementos de y dejando a x fija.
276
277 los indices ip e ij estan correjidos para poder acceder phi y sus condiciones iniciales*/
278
279 let len_i = phi.len();
280 let len_j = phi[0].len();
281
282 for (i, j) in &indices {
283     if *i >= 2 && *j >= 2 && *i + 2 < len_i && *j + 2 < len_j {
284         ey[*i][*j] = -1.0
285         * (-1.0 * phi[*i + 2][*j] + 16.0 * phi[*i + 1][*j] - 30.0 * phi[*i][*j]
286             + 16.0 * phi[*i - 1][*j]
287             - 1.0 * phi[*i - 2][*j])
288         / 12.0;
289         ex[*i][*j] = -1.0
290         * (-1.0 * phi[*i][*j + 2] + 16.0 * phi[*i][*j + 1] - 30.0 * phi[*i][*j]
291             + 16.0 * phi[*i][*j - 1]
292             - 1.0 * phi[*i][*j - 2])
293         / 12.0;
294     }
295 }
296
297 /*Lo que hace este segmento de cdigo es realizar una iteracion sobre todas las areas
298 del potencial calculado electrico aplica el laplaciano a cada elemento y discrepa si es
299 positivo o negativo para hacer una suma ponderada de la carga, posteriormente el valor
300 del laplaciano se guarda en una matriz de distribucion de carga discreta. */
301
302 let mut qp: f32=0.0;
303 let mut qn : f32=0.0;
304 let mut phip: Vec<Vec<f32>> = crear_matriz(m, n, 0.0);
305
306 for i in 1..m{
307     for j in 1..n{

```

```

304     if i >= 2 && j >= 2 && i + 2 < phi.len() && j + 2 < phi[0].len() {
305         let res = (16.0 * phi[i - 1][j]
306                     + 16.0 * phi[i][j - 1]
307                     + 16.0 * phi[i + 1][j]
308                     + 16.0 * phi[i][j + 1]
309                     - 60.0 * phi[i][j]
310                     - 1.0 * phi[i - 2][j]
311                     - 1.0 * phi[i][j - 2]
312                     - 1.0 * phi[i + 2][j]
313                     - 1.0 * phi[i][j + 2])
314             /12.0;
315
316         phip[i][j]=res;
317
318         if res>0.0{
319             qp=qp+res;
320         }
321         else if res<0.0 {
322             qn=qn+res;
323         }
324     }
325 }
326
327
328 //imprimir los valores ponderados de la carga
329 let Q: Vec<(f32, f32)> = vec![(qp, qn)];
330 println!("Carga Positiva, Carga negativa [Pesos]: {:{}}", Q);
331
332 // Llamar a la funcin para escribir en el archivo de manera paralela y checar si no hay
333 // error
334 if let Err(e) = write_matrix_to_file_parallel(&mut phi, "output.txt") {
335     eprintln!("Error al escribir en el archivo: {}", e);
336 } else {
337     println!("Datos escritos exitosamente en el archivo.");
338 }
339
340 if let Err(e) = write_matrix_to_file_parallel(&mut ex, "ex.txt") {
341     eprintln!("Error al escribir en el archivo: {}", e);
342 } else {
343     println!("Datos escritos exitosamente en el archivo.");
344 }
345
346 if let Err(e) = write_matrix_to_file_parallel(&mut ey, "ey.txt") {
347     eprintln!("Error al escribir en el archivo: {}", e);
348 } else {
349     println!("Datos escritos exitosamente en el archivo.");
350 }
351
352 if let Err(e) = write_matrix_to_file_parallel(&mut phip, "phip.txt") {
353     eprintln!("Error al escribir en el archivo: {}", e);
354 } else {
355     println!("Datos escritos exitosamente en el archivo.");
356 }
357 }
```

Graficadora en Python

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from pathlib import Path
4
5 def normalize_matrix(matrix):
6     matrix = np.array(matrix, dtype=float)
7     magnitudes = np.linalg.norm(matrix, axis=1) # Calcular magnitudes a lo largo de las filas
8
9     normalized_matrix = matrix / magnitudes[:, np.newaxis] # Divide cada fila por su magnitud
10    return 100 * normalized_matrix
11
12 def archivo_texto_a_matriz(ruta_archivo):
13     # Abrir el archivo de texto
14     path = Path(__file__).parent / ruta_archivo
15     with path.open('r') as archivo:
16         # Leer lneas del archivo
17         lineas = archivo.readlines()
18
19         # Procesar las lneas y crear una matriz
20         matriz = [list(map(float, linea.strip().split())) for linea in lineas]
21
22     # Convertir la lista de listas a un array de NumPy
23     matriz = np.array(matriz)
24
25     return matriz
26
27 #Leer el ouput, almacenar phi y convertirlo en una matriz.
28 ruta_archivo = 'output.txt'
29 phi = archivo_texto_a_matriz(ruta_archivo)
30
31
32 print("Matriz:")
33 print(phi)
34
35 #se usan las matrices phi, Ex y Ey para generar las visualizaciones
36
37 #Graficar potencial
38 plt.figure(figsize=(8, 6))
39 #plt.contourf(phi, 100, cmap='inferno')
40 plt.contour(phi, 8, cmap='inferno')
41 plt.colorbar(label='Potencial elctrico [V]')
42 #plt.quiver(-Ex,-Ey,scale=5)
43 plt.title('Lineas equipotenciales, capacitor coplanar basado en curva de Hilbert')
44 plt.xlabel('x [m]')
45 plt.ylabel('y [m]')
46 plt.savefig('pothileq.jpg', dpi=600)
47 plt.show()
48
49 plt.figure(figsize=(8, 6))
50 #plt.contourf(rho, 100, cmap='inferno')
51 #plt.contour(rho, 8, cmap='inferno')
52 plt.imshow(phi, cmap='coolwarm', interpolation='nearest')
53 plt.colorbar(label='Potencial elctrico [V]')
54 #plt.quiver(-Ex,-Ey,scale=5)
55 plt.title('Potencial elctrico capacitor coplanar basado en curva de Hilbert')
56 plt.xlabel('x [m]')
57 plt.ylabel('y [m]')
58 plt.savefig('pothilheat.jpg', dpi=600)
59 plt.show()
60
61
62 file_path_x = Path(__file__).parent / 'ex.txt'
63 file_path_y = Path(__file__).parent / 'ey.txt'
64
65 Ex = np.loadtxt(file_path_x)
```

```

66 Ey = np.loadtxt(file_path_y)
67
68 Ex_sub = Ex[:,::10,::10]
69 Ey_sub = Ey[:,::10,::10]
70
71
72 magnitude = np.sqrt(Ex_sub**2 + Ey_sub**2)
73
74 Exn = Ex_sub/magnitude
75 Eyn = Ey_sub/magnitude
76
77 # Crear una cuadricula de coordenadas
78 x = np.arange(0, Exn.shape[1])
79 y = np.arange(0, Eyn.shape[0])
80
81 # Crear una malla de coordenadas
82 X, Y = np.meshgrid(x, y)
83
84 # Graficar el campo vectorial
85 plt.figure(figsize=(10, 8))
86 plt.quiver(x, y, Exn, Eyn, angles='xy', scale_units='xy', scale=1, color='blue', headwidth=2.5)
87 plt.axis("scaled")
88 plt.title('Campo Electrico Normalizado [u.a]')
89 plt.xlabel('x [1:10 m]')
90 plt.ylabel('y [1:10 m]')
91 plt.savefig('campovectorialfull.jpg', dpi=600)
92 plt.show()
93
94 Ex_sub = Ex[:,::30,::30]
95 Ey_sub = Ey[:,::30,::30]
96
97
98 magnitude = np.sqrt(Ex_sub**2 + Ey_sub**2)
99
100 Exn = Ex_sub/magnitude
101 Eyn = Ey_sub/magnitude
102
103 # Crear una cuadricula de coordenadas
104 x = np.arange(0, Exn.shape[1])
105 y = np.arange(0, Eyn.shape[0])
106
107 # Crear una malla de coordenadas
108 X, Y = np.meshgrid(x, y)
109
110 # Graficar el campo vectorial
111 plt.figure(figsize=(10, 8))
112 plt.quiver(x, y, Exn, Eyn, angles='xy', scale_units='xy', scale=0.5, color='blue', headwidth
113 =2.5)
114 plt.axis("scaled")
115 plt.title('Campo Electrico Normalizado [u.a]')
116 plt.xlabel('x [1:30 m]')
117 plt.ylabel('y [1:30 m]')
118 plt.savefig('campovectorial.jpg', dpi=600)
119 plt.show()
120
121 #Graficar densidad de carga
122 rutacarga = 'phip.txt'
123 Q = archivo_texto_a_matriz(rutacarga)
124 Qn = Q[:,::50,::50]*(1/1e+6) #factor de escala para usar unidades del S.I
125 nonzero_indices = np.nonzero(Qn)
126
127 # Extraer los valores distintos de cero y sus indices correspondientes
128 nonzero_values = Qn[nonzero_indices]
129
130 # Crear una malla para el grafico de contornos

```

```

131 x, y = np.meshgrid(range(Qn.shape[1]), range(Qn.shape[0]))
132
133 unique_nonzero_values = np.unique(nonzero_values)# Crear el gráfico de contorno
134 plt.contour(x, y, Qn, levels=np.linspace(unique_nonzero_values.min(), unique_nonzero_values.max()
135   (), len(unique_nonzero_values)), cmap='inferno')
136 plt.title('Distribución de Carga')
137 plt.xlabel('x [1:50 m]')
138 plt.ylabel('y [1:50 m]')
139 plt.colorbar(label='Carga [C/]')
140 plt.savefig('carga.jpg', dpi=600)
141 plt.show()

```

Referencias.

- [1] M.-H. Bao, «Electrostatic driving and capacitive sensing,» en *Micro Mechanical Transducers - Pressure Sensors, Accelerometers and Gyroscopes*, Elsevier, 2000, págs. 139-198.
- [2] H. Eren y L. Sandor, «Fringe-effect capacitive proximity sensors for tamper proof enclosures,» en *2005 Sensors for Industry Conference*, IEEE, 2005.
- [3] M. Mwelango, T. Zhu, K. Wen et al., «Coplanar capacitive sensors and their applications in non-destructive evaluation: a review,» en, *Nondestruct. Test. Eval.*, vol. 38, n.º 5, págs. 861-905, 2023.
- [4] R. I. Haque, M. Lubej y D. Briand, «Design and printing of a coplanar capacitive proximity sensor to detect the gap between dielectric foils edges,» en, *Sens. Actuators A Phys.*, vol. 337, n.º 113424, pág. 113 424, 2022.
- [5] X. Deng, L. Yang, Z. Fu et al., «A calibration-free capacitive moisture detection method for multiple soil environments,» en, *Measurement (Lond.)*, vol. 173, n.º 108599, pág. 108 599, 2021.
- [6] S. G. Surya, S. Yuvaraja, E. Varrla, M. S. Baghini, V. S. Palaparthi y K. N. Salama, «An in-field integrated capacitive sensor for rapid detection and quantification of soil moisture,» en, *Sens. Actuators B Chem.*, vol. 321, n.º 128542, pág. 128 542, 2020.
- [7] J. Guo, P. Hu y J. Tan, «Analysis of a segmented annular coplanar capacitive tilt sensor with increased sensitivity,» en, *Sensors (Basel)*, vol. 16, n.º 1, pág. 133, 2016.
- [8] F. Abdollahi-Mamoudan, S. Savard, C. Ibarra-Castanedo, T. Fillete y X. Malague, «Influence of different design parameters on a coplanar capacitive sensor performance,» en, *NDT E Int.*, vol. 126, n.º 102588, pág. 102 588, 2022.
- [9] A. R. Butz, «Convergence with Hilbert's space filling curve,» *Journal of Computer and System Sciences*, vol. 3, n.º 2, págs. 128-146, 1969, ISSN: 0022-0000. DOI: [https://doi.org/10.1016/S0022-0000\(69\)80010-3](https://doi.org/10.1016/S0022-0000(69)80010-3). dirección: <https://www.sciencedirect.com/science/article/pii/S0022000069800103>.
- [10] F. R. Zypman, «Mathematical expression for the capacitance of coplanar strips,» en, *J. Electrostat.*, vol. 101, n.º 103371, pág. 103 371, 2019.
- [11] G. W. Parker, «What is the Capacitance of Parallel Plates?» en, *Comput. Phys.*, vol. 5, n.º 5, págs. 534-540, 1991.
- [12] M. Naghed e I. Wolff, «Equivalent capacitances of coplanar waveguide discontinuities and interdigitated capacitors using a three-dimensional finite difference method,» *IEEE Trans. Microw. Theory Tech.*, vol. 38, n.º 12, págs. 1808-1815, 1990.
- [13] J. B. Campbell, «Finite difference techniques for ring capacitors,» en, *J. Eng. Math.*, vol. 9, n.º 1, págs. 21-28, 1975.
- [14] D. G. Robertson, *Relaxation methods for partial differential equations: Applications to electrostatics*, <http://faculty.otterbein.edu/DRobertson/compsci/em-stud.pdf>, Accessed: 2023-12-1.