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

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

Pérez Flores, Julio Alfonso.
julio_perez@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 una pseudo curva de Hilbert de segundo orden (nombrado el 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, en este arreglo la capacitancia 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]**. Cuando 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 debido al cambio del material o sus propiedades, alguno de los ejemplos donde esta aplicación es útil son 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 que afectan a la capacitancia son la geometría, el espacio entre placas y la frecuencia de carga, mientras mas área de un electrodo tenga frontera con el electrodo opuesto 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 (Space-filling curve) **[9]** de grosor controlado que separen 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, como son 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 método 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 utilice 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 la capacitancia en condensadores utilizados en 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 de 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.

$$\nabla \cdot \vec{E} = \frac{\rho_{total}}{\epsilon_0}. \tag{1}$$

Ecuación 1: Ley de Gauss en su forma diferencial, donde $\vec{E}$, es el campo electrico y ρ la densidad de carga

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**

$$\nabla^2\phi = \frac{-\rho_{total}}{\epsilon_0}. \tag{2}$$

Ecuación 2: Ecuación de poisson, donde ϕ, es el potencial eléctrico.

$$\phi(x,y) \approx \frac{16\phi_{i-1,\,j} + 16\phi_{i,\,j-1} + 16\phi_{i+1,\,j} + 16\phi_{i,\,j+1} - \phi_{i-2,\,j} - \phi_{i,\,j-2} - \phi_{i+2,\,j} - \phi_{i,\,j+2}}{60}. \quad (3)$$

Ecuación 3: Aproximación del Laplaciano mediante diferencias finitas centradas de cuarto orden. Donde $\phi_{i+k,j+k}$, significa $\phi(x_i+hk, y_j+hk)$, con x_i, x_j puntos iniciales y h la distancia entre dos puntos de la partición

$$-\frac{\rho(x,y)}{\epsilon_0} \approx \frac{16\phi_{i-1,\,j} + 16\phi_{i,\,j-1} + 16\phi_{i+1,\,j} + 16\phi i,\ j+1 - \phi_{i-2,\,j} - \phi_{i,\,j-2} - \phi_{i+2,\,y} - \phi_{i,\,j+2} - 60\phi(i,j)}{12h^2}. \quad (4)$$

Ecuación 4: Aproximación de la ecuación de Poisson, mediante diferencias finitas centradas de cuarto orden. Donde $\phi_{i+k,\,j+k}$, significa $\phi(x_i+hk,\ y_j+hk)$, con x_i, x_j puntos iniciales y h la distancia entre dos puntos de la partición

Y en una región del espacio donde no hay carga eléctrica, se llega a la ecuación que es nombrada como ecuación de Laplace **ec. 5**

$$\nabla^2\phi = 0. \quad (5)$$

Ecuación 5: Ecuación de Laplace.

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

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. 7** y análogamente, la **ec. 8** para y.

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

Ecuación 6: Operador Laplaciano en coordenadas cartesianas, donde ϕ es una función escalar

En la ecuación **ec. 6** se sustituyen las segundas derivadas por las expresiones descritas (**ecs: 7, 8**), por la **ec. 5** se igual a 0, despejando $\phi(x,y)$, se obtiene que el valor de cada celda del mallado se aproxima con el promedio de sus cuatro vecinos (**ec. 3**). 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 denomina a este procedimiento en este documento como método de relajación con ansiedad.

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

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

Ecuaciones 7, 8: Aproximación por diferencias finitas centradas de cuarto orden de las segundas derivadas de rho en x y, y, donde $\phi_{i+k,\,j+k}$, significa $\phi(x_i+hk,\ y_j+hk)$, con x_i, x_j puntos iniciales y h la distancia entre dos puntos de la partición

Para calcular el campo eléctrico, se utiliza el método de diferencias centrales de cuarto orden en cada componente tomando en cuenta que $\vec{E} = -\nabla\phi$

$$\vec{E} \approx \frac{-\phi_{i+2,\,j} + \phi_{i+1,\,j} - \phi_{i-1,\,j} + \phi_{i-2,\,j}}{12h}\hat{e}_i + , \quad (9)$$
$$\frac{-\phi_{i,\,j+2} + \phi_{i,\,j+1} - \phi_{i,\,j-1} + \phi_{i,\,j-2}}{12h}\hat{e}_j$$

Ecuación 9: aproximación del campo eléctrico por diferencias finitas centradas de cuarto orden, donde $\phi_{i+k,\,j+k}$, significa $\phi(x_i+hk,\ y_j+hk)$, con x_i, x_j puntos iniciales y h la distancia entre dos puntos de la partición.

Como se ha 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. 4**

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.

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

Ecuación 10: aproximación de la carga total en la geometría del condensador.

2. Método.

Un condensador coplanar bloque pensado para poder construir condensadores de mayor capacitancia mediante un arreglo lineal y en paralelo de n de estos bloques, al cual denominamos el Hilbertrón, fue diseñado baja la siguiente premisa; en un rectángulo de **2046 x 2430 μm** se trazó una pseudo-curva de Hilbert de segundo orden, con **127μm** de ancho, a la parte superior disconexa se le asignó potencial positivo (3.5 V) además se conectan los tres conjuntos mediante una tira de **192μm**, la parte conexa se le asignó potencial neutro (0 V), y por completez se le agrega una tira de **192μm** de ancho, y del largo del rectángulo, la **fig. 1**, muestra un diagrama del condensador y sus dimensiones.

Para la implementación descrita en la **ec. 3**, se empleo un programa escrito en Rust el cual genera un mallado de **2046 × 2430**, con un arreglo matricial, donde cada entrada de la matriz representa un área de **1μm × 1μm**, de tal forma que $h = 1$.

Las regiones de los electrodos fueron llenadas con el valor del potencial descrito en la construcción, en las regiones en color blanco **fig. ?? a)** se asigno un potencial de **1.75 V**, para reducir el tiempo de convergencia. Se aplicaron **5000** iteraciones sobre la operación **ec. 3** en la región en blanco, para respetar las condiciones de frontera (bordes de los electrodos). Con este potencial se calcula el campo eléctrico en toda el área con la ecuación **ec.9**, finalmente con ayuda de **ec. 4** y **ec. 10**, se muestra la distribución de carga y la carga total. Finalmente la matriz de potencial discreto, la matriz con los componentes $\mathbf{E_x}$, $\mathbf{E_y}$ 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 **5000** iteraciones con una tolerancia $\mathbf{2.22 \times 10^{-7}}$. 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.

La gráfica del campo eléctrico **fig. 3 a)** muestra lineas perpendiculares a las lineas equipotenciales, resultado esperado con base a la relación potencial - campo eléctrico, estann van del electrodo con potencial positivo al neutro, 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. 3 b)**, donde podemos apreciar que la carga positiva se acumula en el electrodo con voltejea positivo y negativa en el electrodo con voltaje 0. Al sumar la carga positiva en se obtiene $Q = 0.000638$ pesos y la diferencia de voltaje es **3.5 V** con lo que la capacitancia es:

$$C = \frac{0.000638\varepsilon}{3.5V}$$

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 a)**, 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 borden 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.

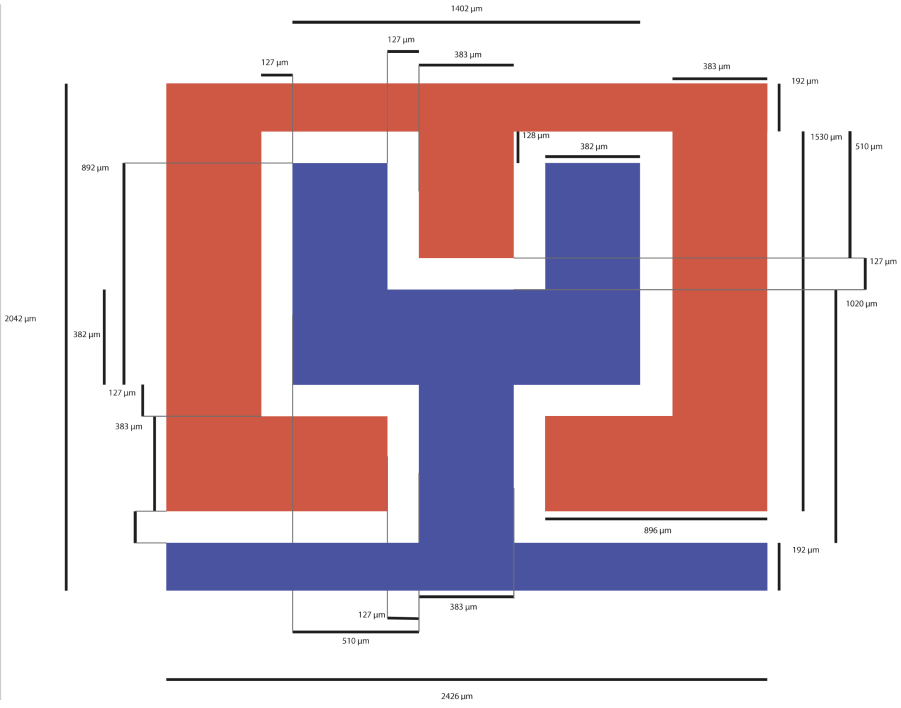
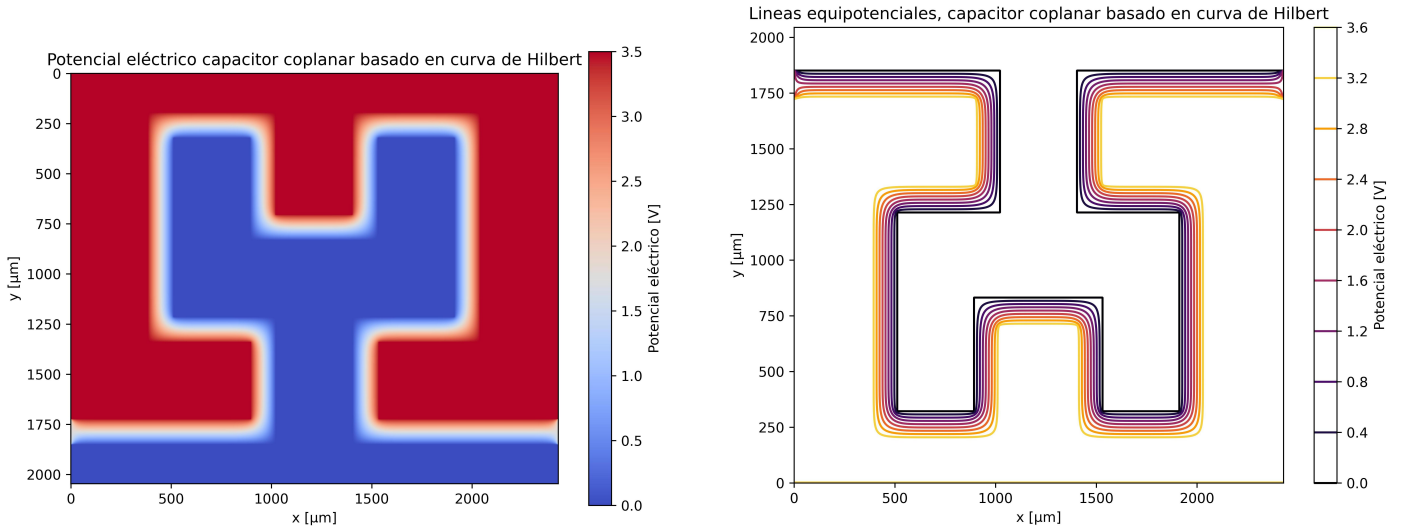
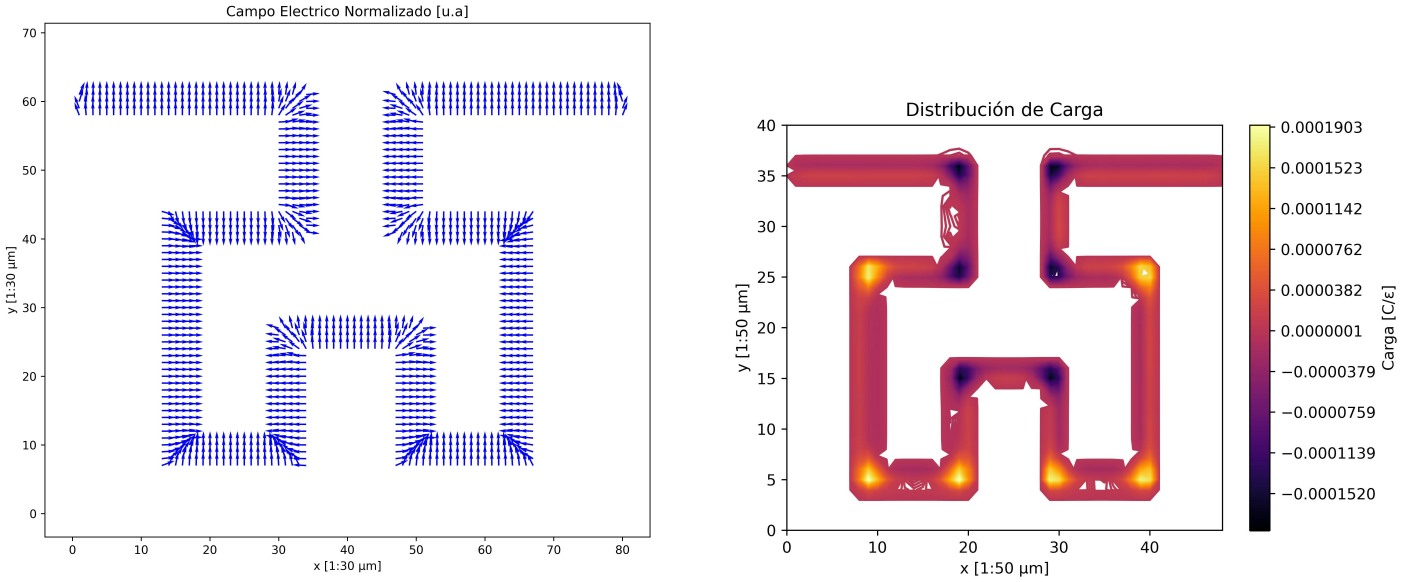


Figura 1: Geometría y dimensiones del Hilbertrón



(a) (b)

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



(a) (b)

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: 1:30 μm) (b): Densidad de carga calculada y en escala 1: 50 μm entre la constante de permitividad electrica del material, donde las regiones en color uva tienden a 0 $\frac{C}{\epsilon}$.

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 dieléctrica 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 tramites 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 gráficadora en Python.

Método de diferencias finitas centras de cuarto en orden en Rust

```rust
/*La idea general de codigo es que resuelve la ecuacin de Laplace utilizando  diferencias
centradas de cuarto orden, para ello se genera una matriz de 2046x430 donde cada indice
representa 1 um y se rellena con un valor arbitrario (en este caso la mitad del potencial
ms grande)

https://www.dam.brown.edu/people/alcyew/handouts/numdiff.pdf

Posteriormente se proponen las fronteras de tipo Dirichlet mutando los valores de algunas zonas
de la matriz con respecto al potencial de nuestro electrodos 5 y 0 Volts y a la geometria de
nuestros electrodos (ver PDF).

Finalmente se busca el conjunto de pares ordenados donde los valores con los que se relleno la
matriz en un principio no fueron mutados. Esto es importante porque, debido a que el potencial
de nuestra frontera no debe ser mutado por la diferencia finita entonces solo iteramos en esa
coleccin de indices y con eso ahorramos tiempo por cada paso que hace

En  otro proceso aparte exporta los datos de la matriz cuando ya se terminaron de hacer 10000
iteraciones en un archivo txt con escritura paralela (para no demorarse tanto tiempo)
para poder graficar en python porque no sabemos hacerlo en Rust.*/

use rayon::prelude::*; //libreria de escritura paralela
use std::fs::File; //elemento de la libreria estandar de Rust que accesa rutas del ordenado
use std::io::{self, Write}; //metodo escribir sobre objeto de la libreria estandar de Rust

//numero maximo de iteraciones
const MAX_ITER: usize = 5000;

//Declaracin de nuestras funciones
//
    ----------------------------------------------------------------------------------------------------


fn copiar_matriz_a_vector(matriz: &Vec<Vec<f32>>, indices: &Vec<(usize, usize)>) -> Vec<f32> {
    let mut resultado = Vec::new();

    for &(fila, columna) in indices {
        if let Some(fila_matriz) = matriz.get(fila) {
            if let Some(valor) = fila_matriz.get(columna) {
                resultado.push(*valor);
            } else {
                panic!("ndices de columna fuera de rango en la fila {}", fila);
            }
        } else {
            panic!("ndice de fila fuera de rango: {}", fila);
        }
    }

```

```
    resultado
}

fn restar_vectores(a: &Vec<f32>, b: &Vec<f32>) -> Vec<f32> {
    if a.len() != b.len() {
        panic!("Los vectores deben tener la misma longitud para realizar la resta.");
    }

    let mut resultado = Vec::with_capacity(a.len());

    for i in 0..a.len() {
        resultado.push(a[i] - b[i]);
    }

    resultado
}

fn obtener_promedio(arr:&Vec<f32>) -> Option<f32> {
    let longitud = arr.len();

    if longitud == 0 {
        None // No se puede calcular el promedio de un array vaco
    } else {
        let suma: f32 = arr.iter().sum();
        Some(suma / (longitud as f32))
    }
}

/*  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 */

fn crear_matriz(m: usize, n: usize, fill: f32) -> Vec<Vec<f32>> {
    let mut matriz = Vec::with_capacity(m);

    for _ in 0..m {
        let fila: Vec<f32> = vec![fill; n];
        matriz.push(fila);
    }

    matriz
}

/*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.*/

fn obtener_valores(matriz: &mut Vec<Vec<f32>>, valor_buscado: f32) -> Vec<(usize, usize)> {
    let mut indices_a_cambiar = Vec::new();

    for (i, fila) in matriz.iter_mut().enumerate() {
        for (j, valor) in fila.iter_mut().enumerate() {
            if *valor == valor_buscado {
                // Guardar los ndices
                indices_a_cambiar.push((i, j));
            }
        }
    }

    indices_a_cambiar
}

/*
Esta funcin, llamada write_matrix_to_file_parallel, toma una matriz mutable de nmeros de punto
```

```
    flotante (f32)
y un camino de archivo (file_path), y escribe el contenido de la matriz en un archivo, mediante escritura paralela.
*/

fn write_matrix_to_file_parallel(matrix: &mut Vec<Vec<f32>>, file_path: &str) -> io::Result<()> {
    let formatted_rows: Vec<String> = matrix
        .par_iter_mut()
        .map(|row| {
            // Formatear la fila como una cadena separada por tabulaciones
            row.iter_mut()
                .map(|num| num.to_string())
                .collect::<Vec<String>>()
                .join("\t")
        })
        .collect();

    let mut file = File::create(file_path)?;

    for row_str in formatted_rows {
        // Escribir la fila en el archivo seguido de un salto de lnea
        writeln!(file, "{}", row_str)?;
    }

    Ok(())
}
//
    ------------------------------------------------------------------------------------------------------


//Ejecucin Principal.
//
    ------------------------------------------------------------------------------------------------------

fn main() {
    // Nmero de filas y columnas

    /*Cabe aclarar que se agregaron cuatro puntos mas para satisfacer las condiciones iniciales del
    metodo de diferencias finitas centradas de cuarto orden y no perder informacin, i,e i<=2, e i<= n-2, de tal forma que el termino i+2 corresponde al 0 de nuestro sistema real */
    let m = 2046;
    let n = 2430;

    // Crear la matriz

    let mut phi: Vec<Vec<f32>> = crear_matriz(m, n, 1.75);

    /* En ambos casos (potencial positivo y potencial 0),lo que se hace es iterar sobre las filas primero
    y despues sobre las columnas. si el indice de las columnas se encuentra en una de las colecciones de
    puntos mencionadas (las cuales corresponden geometricamente a los electrodos del capacitor coplanar),
    entonces modifica el valor de la matriz al del potencial deseado.*/

    //potencial positivo
    for i in 2..m - 2 {
        if (0..=194).contains(&i) {
            for j in 0..n {
                phi[i][j] = 3.5;
            }
        }

        if (194..=704).contains(&i) {
```

```
            for range in [(0..385), (1021..1404), (2040..n)] {
                for j in range {
                    phi[i][j] = 3.5;
                }
            }
        }

        if (704..=1340).contains(&i) {
            for range in [(0..385), (2040..n)] {
                for j in range {
                    phi[i][j] = 3.5;
                }
            }
        }

        if (1340..=1723).contains(&i) {
            for range in [(0..898), (1535..n)] {
                for j in range {
                    phi[i][j] = 3.5;
                }
            }
        }
    }

    //potencial 0
    for i in 0..m {
        if (322..=832).contains(&i) {
            for range in [(512..894), (1531..1913)] {
                for j in range {
                    phi[i][j] = 0.0;
                }
            }
        }

        if (832..=1214).contains(&i) {
            for j in 512..1913 {
                phi[i][j] = 0.0;
            }
        }

        if (1214..=1851).contains(&i) {
            for j in 1022..1405 {
                phi[i][j] = 0.0;
            }
        }

        if i >= 1851 {
            for j in 0..n {
                phi[i][j] = 0.0;
            }
        }
    }

    //obtener indices a iterar
    let indices = obtener_valores(&mut phi, 1.75);

    //iteraciones del metodo
   for _k in 1..MAX_ITER {
        /* Como la coleccion de parejas ordenadas que obtuvimos en la seccin anterior no es
        accesible para operar en este ciclo las clonamos (supongo que hay una forma mas optima
        de pasarlas al scope) pero se itera sobre esa coleccin de puntos

        Sabemos que en esa coleccin de puntos hay puntos que no cumplen las condiciones
    iniciales
        ie i<=2, e i<= n-2, entonces forazmos a que solo itere sobre los puntos que si cumplen
    esa
```

```
        condicin y evalue el nuevo valor de phi (checar foto del desgloze de la solucin de la
        ecuacin de laplace en la foto))*/

        let phiant=copiar_matriz_a_vector(&phi, &indices);
        for (i, j) in &indices {
            // Verificar que los ndices estn dentro del rango de la matriz
            if *i < phi.len() && *j < phi[0].len() {
                if *i >= 2 && *j >= 2 && *i + 2 < phi.len() && *j + 2 < phi[0].len() {
                    phi[*i][*j] = (16.0 * phi[*i - 1][*j]
                        + 16.0 * phi[*i][*j - 1]
                        + 16.0 * phi[*i + 1][*j]
                        + 16.0 * phi[*i][*j + 1]
                        - 1.0 * phi[*i - 2][*j]
                        - 1.0 * phi[*i][*j - 2]
                        - 1.0 * phi[*i + 2][*j]
                        - 1.0 * phi[*i][*j + 2])
                        / 60.0;
                }
            }
        }

         let tol = match obtener_promedio(&restar_vectores(&copiar_matriz_a_vector(&phi, &
    indices), &phiant)) {
            Some(mean) => mean.abs(),
            None => {
                eprintln!("Error calculating mean. Exiting.");
                return;
            }
        };

        println!("Iteracin n: {}, tol: {}.", _k,tol);

    }

    /*hacemos las matrices de las componentes en x y Y del campo electrico de nuevo de la
    matriz de potencial hay
    cuatro puntos mas para satisfacer las condiciones iniciales del  metodo de diferencias
    finitas centradas de cuarto
    orden pero no nos importan y los quitamos de las matrices resultantes
    */

    let mut ex: Vec<Vec<f32>> = crear_matriz(m, n, 0.0); // Componente x del campo elctrico
    let mut ey: Vec<Vec<f32>> = crear_matriz(m, n, 0.0); // Componente y del campo elctrico

    /*Iteramos sobre las matrices Ey Ex y Phi para aplicar el operador numerico gradiente el
    cual es diferencia finita
    centrada de cuarto orden para los elementos de x dejando a y fija y la diferencia finita
    centrada de cuarto orden para
    los elementos de y dejando a x fija.

    los indices ip e ij estan correjidos para poder acceder phi y sus condiciones iniciales*/

    let len_i = phi.len();
    let len_j = phi[0].len();

    for (i, j) in &indices {
        if *i >= 2 && *j >= 2 && *i + 2 < len_i && *j + 2 < len_j {
            ey[*i][*j] = -1.0
                * (-1.0 * phi[*i + 2][*j] + 8.0 * phi[*i + 1][*j] - 8.0 * phi[*i - 1][*j] + phi
    [*i - 2][*j])
                / 12.0;
            ex[*i][*j] = -1.0
                * (-1.0 * phi[*i][*j + 2] + 8.0 * phi[*i][*j + 1] - 8.0 * phi[*i][*j - 1] + phi
    [*i][*j - 2])
                / 12.0;
        }
```

```
    }

    /*Lo que hace este segmento de cdigo es realizar una iteracin sobre todas las areas
    del potencial calculado elctrico aplica el laplaciano a cada elemento y discrepa si es
    positivo o negativo para hacer una suma ponderada de la carga, posteriormente el valor
    del laplaciano se guarda en una matriz de distribucin de carga discreta. */

    let mut qp: f32=0.0;
    let mut qn : f32=0.0;
    let mut rho: Vec<Vec<f32>> = crear_matriz(m, n, 0.0);

    for i in 1..m{
        for j in 1..n{
            if i >= 2 && j >= 2 && i + 2 < phi.len() && j + 2 < phi[0].len() {
                let res = (1.0/1e+6)*((16.0 * phi[i - 1][j]
                    + 16.0 * phi[i][j - 1]
                    + 16.0 * phi[i + 1][j]
                    + 16.0 * phi[i][j + 1]
                    -60.0 * phi[i][j]
                    - 1.0 * phi[i - 2][j]
                    - 1.0 * phi[i][j - 2]
                    - 1.0 * phi[i + 2][j]
                    - 1.0 * phi[i][j + 2])
                    /12.0); //# se agrego factor de escala para usar unidades del S.I

                rho[i][j]=res;

                if res>0.0{
                    qp=qp+res;
                }
                else if res<0.0 {
                    qn=qn+res;
                }
            }
        }
    }

    //imprimir los valores ponderados de la carga
    let Q: Vec<(f32, f32)> = vec![(qp, qn)];
    println!("Carga Positiva, Carga negativa [Pesos]: {:?}", Q);

    // Llamar a la funcin para escribir en el archivo de manera paralela y checar si no hay
    error
    if let Err(e) = write_matrix_to_file_parallel(&mut phi, "Datos usados en el articulo/output
    .txt") {
        eprintln!("Error al escribir en el archivo: {}", e);
    } else {
        println!("Datos escritos exitosamente en el archivo.");
    }

    if let Err(e) = write_matrix_to_file_parallel(&mut ex, "Datos usados en el articulo/ex.txt"
    ) {
        eprintln!("Error al escribir en el archivo: {}", e);
    } else {
        println!("Datos escritos exitosamente en el archivo.");
    }

    if let Err(e) = write_matrix_to_file_parallel(&mut ey, "Datos usados en el articulo/ey.txt"
    ) {
        eprintln!("Error al escribir en el archivo: {}", e);
    } else {
        println!("Datos escritos exitosamente en el archivo.");
    }

    if let Err(e) = write_matrix_to_file_parallel(&mut rho, "Datos usados en el articulo/rho.
    txt") {
```

```
        eprintln!("Error al escribir en el archivo: {}", e);
    } else {
        println!("Datos escritos exitosamente en el archivo.");
    }

}
```

Graficadora en Pyhton

```
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path

def normalize_matrix(matrix):
    matrix = np.array(matrix, dtype=float)
    magnitudes = np.linalg.norm(matrix, axis=1)  # Calcular magnitudes a lo largo de las filas

    normalized_matrix = matrix / magnitudes[:, np.newaxis]  # Divide cada fila por su magnitud
    return 100 * normalized_matrix

def archivo_texto_a_matriz(ruta_archivo):
    # Abrir el archivo de texto
    path = Path(__file__).parent / ruta_archivo
    with path.open('r') as archivo:
        # Leer lneas del archivo
        lineas = archivo.readlines()

        # Procesar las lneas y crear una matriz
        matriz = [list(map(float, linea.strip().split( ))) for linea in lineas]

        # Convertir la lista de listas a un array de NumPy
        matriz = np.array(matriz)

    return matriz

#Leer el ouput, almacenar phi y convertirlo en una matriz.
ruta_archivo = 'Datos usados en el articulo/output.txt'
phi = archivo_texto_a_matriz(ruta_archivo)


print("Matriz:")
print(phi)

#se usan las matrices phi, Ex y Ey para generar las visualizaciones

#Graficar potencial
plt.figure(figsize=(8, 6))
#plt.contourf(phi, 100, cmap='inferno')
plt.contour(phi, 8, cmap='inferno')
plt.colorbar(label='Potencial elctrico [V]')
#plt.quiver(-Ex,-Ey,scale=5)
plt.title('Lineas equipotenciales, capacitor coplanar basado en curva de Hilbert')
plt.xlabel('x [m]')
plt.ylabel('y [m]')
plt.savefig('graficas/pothileq.jpg', dpi=600)
plt.show()

plt.figure(figsize=(8, 6))
#plt.contourf(rho, 100, cmap='inferno')
#plt.contour(rho, 8, cmap='inferno')
plt.imshow(phi, cmap='coolwarm', interpolation='nearest')
plt.colorbar(label='Potencial elctrico [V]')
#plt.quiver(-Ex,-Ey,scale=5)
plt.title('Potencial elctrico capacitor coplanar basado en curva de Hilbert')
plt.xlabel('x [m]')
plt.ylabel('y [m]')
plt.savefig('graficas/pothilheat.jpg', dpi=600)
```

```
plt.show()


file_path_x = Path(__file__).parent / 'Datos usados en el articulo/ex.txt'
file_path_y = Path(__file__).parent / 'Datos usados en el articulo/ey.txt'

Ex = np.loadtxt(file_path_x)
Ey = np.loadtxt(file_path_y)

Ex_sub = Ex[::10,::10]
Ey_sub = Ey[::10,::10]


magnitude = np.sqrt(Ex_sub**2 + Ey_sub**2)

Exn = Ex_sub/magnitude
Eyn = Ey_sub/magnitude

# Crear una cuadrcula de coordenadas
x = np.arange(0, Exn.shape[1])
y = np.arange(0, Eyn.shape[0])

# Crear una malla de coordenadas
X, Y = np.meshgrid(x, y)

# Graficar el campo vectorial
plt.figure(figsize=(10, 8))
plt.quiver(x, y, Exn, Eyn, angles='xy', scale_units='xy', scale=1, color='blue', headwidth=2.5)
plt.axis("scaled")
plt.title('Campo Electrico Normalizado [u.a]')
plt.xlabel('x [1:10 m]')
plt.ylabel('y [1:10 m]')
plt.savefig('graficas/campovectorialfull.jpg', dpi=600)
plt.show()

Ex_sub = Ex[::30,::30]
Ey_sub = Ey[::30,::30]


magnitude = np.sqrt(Ex_sub**2 + Ey_sub**2)

Exn = Ex_sub/magnitude
Eyn = Ey_sub/magnitude

# Crear una cuadrcula de coordenadas
x = np.arange(0, Exn.shape[1])
y = np.arange(0, Eyn.shape[0])

# Crear una malla de coordenadas
X, Y = np.meshgrid(x, y)

# Graficar el campo vectorial
plt.figure(figsize=(10, 8))
plt.quiver(x, y, Exn, Eyn, angles='xy', scale_units='xy', scale=0.5, color='blue', headwidth
    =2.5)
plt.axis("scaled")
plt.title('Campo Electrico Normalizado [u.a]')
plt.xlabel('x [1:30 m]')
plt.ylabel('y [1:30 m]')
plt.savefig('graficas/campovectorial.jpg', dpi=600)
plt.show()

#Graficar densidad de carga
rutacarga = 'Datos usados en el articulo/phip.txt'
Q = archivo_texto_a_matriz(rutacarga)
Qn = Q[::50,::50]
```

```

nonzero_indices = np.nonzero(Qn)

# Extraer los valores distintos de cero y sus ndices correspondientes
nonzero_values = Qn[nonzero_indices]

# Crear una malla para el grfico de contornos
x, y = np.meshgrid(range(Qn.shape[1]), range(Qn.shape[0]))

unique_nonzero_values = np.unique(nonzero_values)# Crear el grfico de contorno
plt.contour(x, y, Qn, levels=np.linspace(unique_nonzero_values.min(), unique_nonzero_values.max
    (), len(unique_nonzero_values)), cmap='inferno')
plt.title('Distribucin de Carga')
plt.xlabel('x [1:50 m]')
plt.ylabel('y [1:50 m]')
plt.colorbar(label='Carga [C/]')
plt.savefig('graficas/carga.jpg', dpi=600)
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. Palaparthy 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. Filleter y X. Maldague, «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. 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.