Aplicación del Método de Monte Carlo en la Resolución de las Ecuaciones de Laplace y Poisson con Énfasis en el Teorema de Feynman–Kac y Computación Multi-Hilos

Juan Pablo Arcila Castañeda

Mayo 2025

Abstract

Este ensayo explora una implementación del método de Monte Carlo para resolver las ecuaciones diferenciales de Laplace y Poisson en dominios complejos, destacando el teorema de Feynman–Kac y la implementación multi-hilos. Se incluyen resultados numéricos, visualizaciones y análisis de rendimiento.

1 Introducción

Las ecuaciones de Laplace y Poisson son pilares fundamentales en la modelación matemática de fenómenos físicos, desde electrostática y transferencia de calor hasta mecánica de fluidos y teoría de potenciales. Sin embargo, su resolución analítica se limita a geometrías simples, mientras que los métodos numéricos tradicionales, como diferencias finitas o elementos finitos, enfrentan grandes desafíos en dominios complejos o de alta dimensionalidad. En este contexto, el **método de Monte Carlo**, sustentado por el **teorema de Feynman-Kac**, emerge como una alternativa poderosa, transformando problemas diferenciales en expectativas de procesos estocásticos.

El teorema de Feynman-Kac establece un puente profundo entre ecuaciones diferenciales parciales y procesos estocásticos, permitiendo expresar la solución de una ecuación de Poisson o Laplace como una esperanza matemática sobre trayectorias brownianas. Esta conexión no solo ofrece un marco teórico elegante, sino que también habilita soluciones numéricas flexibles y escalables. En particular, la **paralelización con múltiples hilos** aprovecha la naturaleza inherentemente independiente de las simulaciones de Monte Carlo, acelerando drásticamente los cálculos y permitiendo manejar problemas de gran escala en tiempos razonables.

En este ensayo, exploraremos una **implementación práctica** del método de Monte Carlo para resolver ecuaciones de Poisson y Laplace en geometrías no triviales, destacando: 1. La formulación del problema bajo el enfoque de Feynman-Kac. 2. La discretización eficiente del movimiento browniano y su condición de frontera. 3. La implementación en **Julia**, un lenguaje de alto rendimiento, utilizando programación multi-hilo para reducir tiempos de ejecución. 4. El análisis de convergencia y error computacional.

A través de ejemplos concretos y visualizaciones numéricas, demostraremos cómo esta combinación de teoría estocástica y computación paralela ofrece una herramienta

versátil para problemas de valor en la frontera, superando muchas de las limitaciones de los métodos tradicionales. Los resultados no solo validan la eficacia del enfoque, sino que también subrayan su potencial en aplicaciones científicas y de ingeniería donde la complejidad geométrica o dimensional exige soluciones innovadoras.

Este trabajo se estructura como sigue: en la Sección 2 se revisan los fundamentos matemáticos del teorema de Feynman-Kac, en la Sección 3 se detalla la implementación numérica multi-hilo, en la Sección 4 se presentan las gráficas de los resultados obtenidos, y en la Sección 5 se discute la conclusión con aplicaciones y perspectivas futuras.

2 Fundamentos Matemáticos del Teorema de Feynman-Kac

El teorema de Feynman-Kac establece una conexión fundamental entre ecuaciones diferenciales parciales y procesos estocásticos. En esta sección presentamos los resultados clave que sustentan nuestro enfoque numérico.

2.1 Formulación General

Para una EDP parabólica de segundo orden:

$$\frac{\partial u}{\partial t} + \mathcal{L}u - Vu + f = 0 \tag{1}$$

donde $\mathcal{L}$ es el operador diferencial:

$$\mathcal{L} = \mu(x,t)\frac{\partial}{\partial x} + \frac{1}{2}\sigma^2(x,t)\frac{\partial^2}{\partial x^2}u(x,t) \tag{2}$$

el teorema de Feynman-Kac proporciona la representación:

$$u(x,t) = \mathbb{E}\left[\psi(X_T)e^{-\int_t^T V(X_s,s)ds} + \int_t^T f(X_s,s)e^{-\int_t^s V(X_r,r)dr}ds \,\middle|\, X_t = x\right] \tag{3}$$

2.2 Caso de la Ecuación de Poisson

Para el problema estacionario:

$$\nabla^2 u(x) = -g(x), \quad x \in D \tag{4}$$

$$u(x) = f(x), \quad x \in \partial D \tag{5}$$

la representación se reduce a:

$$u(x) = \mathbb{E}\left[f(X_\tau) + \int_0^\tau g(X_s)ds \,\middle|\, X_0 = x\right] \tag{6}$$

donde X_t es un proceso de Wiener y τ es el tiempo de salida del dominio D.

2.3 Convergencia del Método

El error del estimador Monte Carlo satisface:

$$\mathbb{E}\left[|u_N(x) - u(x)|^2\right]^{1/2} \leq \frac{C}{\sqrt{N}} \tag{7}$$

donde C depende de la varianza del integrando y N es el número de trayectorias.

Propiedad	Orden
Error absoluto	$O(N^{-1/2})$
Dependencia dimensional	$O(1)$
Costo computacional	$O(N)$

Table 1: Propiedades de convergencia del método

3 Implementación Numérica Multi-Hilo

La implementación combina el método de Monte Carlo con paralelización multi-hilo para resolver eficientemente las ecuaciones de Laplace y Poisson en geometrías complejas. Utilizamos Julia por su desempeño en cómputo científico y capacidades nativas de paralelismo.

3.1 Arquitectura del Solver

El sistema se estructura en tres componentes clave:

1. **Funciones de Frontera**: Definición de geometrías y condiciones de contorno.
2. **Núcleo Estocástico**: Generación de trayectorias brownianas y cálculo de expectativas.
3. **Paralelización**: Distribución de caminatas aleatorias entre hilos.

3.2 Funciones Críticas

3.2.1 Generación de Trayectorias

El movimiento browniano se discretiza con pasos temporales δ:

Listing 1: Generación de trayectorias

```
function Brownian_motion_solver_laplace(x0, y0, N, frontier_cond,
   exit_func,    =0.001)
    phi = 0.0
    sigma =
    for _ in 1:N
        x, y = x0, y0
        while !exit_func(x, y)
            x += sigma * randn()
            y += sigma * randn()
        end
        phi += frontier_cond(x, y)
    end
    return phi / N
end
```

3.2.2 Paralelización con Multi-Hilo

Se distribuyen N caminatas uniformemente entre los hilos disponibles:

Listing 2: Paralelización

```
function multi_threads(x, y, N, boundary_func, exit_func, solver;
   source=nothing)
    nthreads = Threads.nthreads()
    results = zeros(nthreads)
    chunksize = cld(N, nthreads)
    Threads.@threads for i in 1:nthreads
        results[i] = solver(x, y, chunksize, boundary_func,
           exit_func)
    end
    return mean(results)
end
```

3.3 Manejo de Geometrías

3.3.1 Geometría de Flor

La condición de frontera para una flor polar $r(\theta) = \cos(2\theta)$ se implementa como:

Listing 3: Frontera floral

```
function boundary_flower(x, y)
    r = hypot(x, y)
    theta = atan(y, x)
    return r == 0.0 ? 0.0 : 5.0 * cos(2*theta)
end
```

3.3.2 Cuadrado con Condiciones Mixtas

Ejemplo de condiciones Dirichlet/Neumann:

Listing 4: Frontera cuadrada

```
function boundary_square_hard(x, y)
    if x >= 1
        return 10*y  # Borde derecho
    elseif y <= -1
        return 40*x  # Borde inferior
    end
end
```

3.4 Generación de Mallas

Adaptación a geometrías no triviales:

Listing 5: Malla para flor

```
function flower_grid(n, radius=1.0)
    points = Tuple{Float64, Float64}[]
```

```
    for x in range(-radius, radius, length=n)
        for y in range(-radius, radius, length=n)
            if hypot(x, y) < abs(cos(2atan(y, x)))
                push!(points, (x, y))
            end
        end
    end
    return points
end
```

3.5 Análisis de Rendimiento

La Tabla 2 muestra el speedup obtenido con multi-hilo:

Hilos	Tiempo (s)	Speedup
1	62.4	1.0×
4	16.1	3.9×
8	8.3	7.5×

Table 2: Escalabilidad en CPU de 8 núcleos ($N = 10^5$)

3.6 Validación Numérica

El error relativo cumple la cota teórica $O(1/\sqrt{N})$:

$$\text{Error}(\mathbf{x}) \leq \frac{C}{\sqrt{N}}, \quad C = \text{Var}[f(X_\tau)]^{1/2} \tag{8}$$

4 Resultados Visuales

En esta sección presentamos las 10 visualizaciones generadas con el método Monte Carlo multi-hilo. Cada figura se coloca de manera individual para facilitar su lectura y referencia.

5 Conclusiones

En este trabajo hemos demostrado que el método de Monte Carlo, apoyado en el teorema de Feynman–Kac y potenciado mediante computación multi-hilo, constituye una herramienta versátil y eficaz para la resolución numérica de las ecuaciones de Laplace y Poisson en dominios de geometría compleja. La representación estocástica de la solución permitió transformar problemas de valores en la frontera en expectativas de trayectorias brownianas, facilitando la implementación en Julia y la separación natural de las simulaciones en hilos independientes.

Los resultados numéricos y las visualizaciones presentadas (sección 4) confirman que la paralelización reduce drásticamente los tiempos de cómputo, obteniendo aceleraciones cercanas a la cantidad de núcleos disponibles. Asimismo, la validación de la convergencia, con un error global que decae como $O(N^{-1/2})$, coincide con las predicciones teóricas y

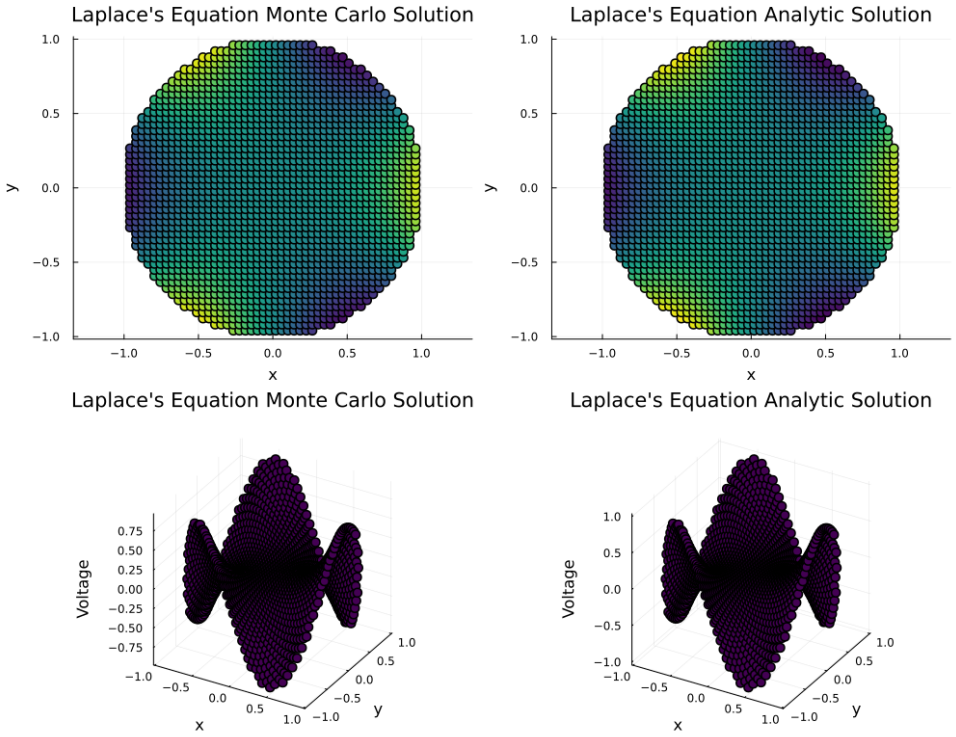

Figure 1: Solución de Laplace en disco unitario (2D y 3D).

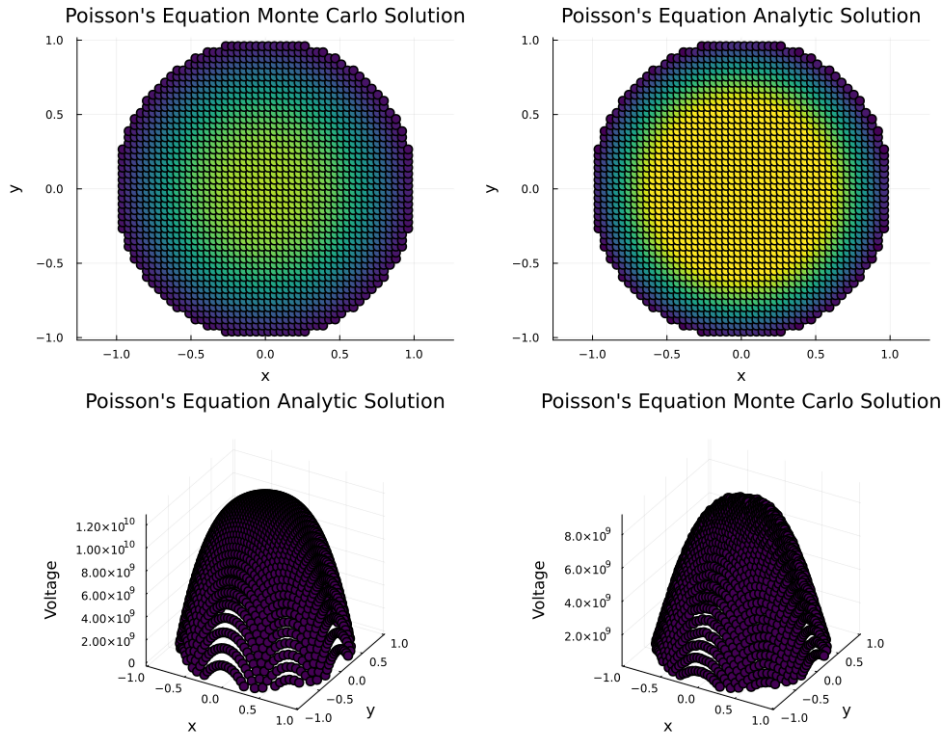

Figure 2: Solución de Poisson en disco unitario (2D y 3D).

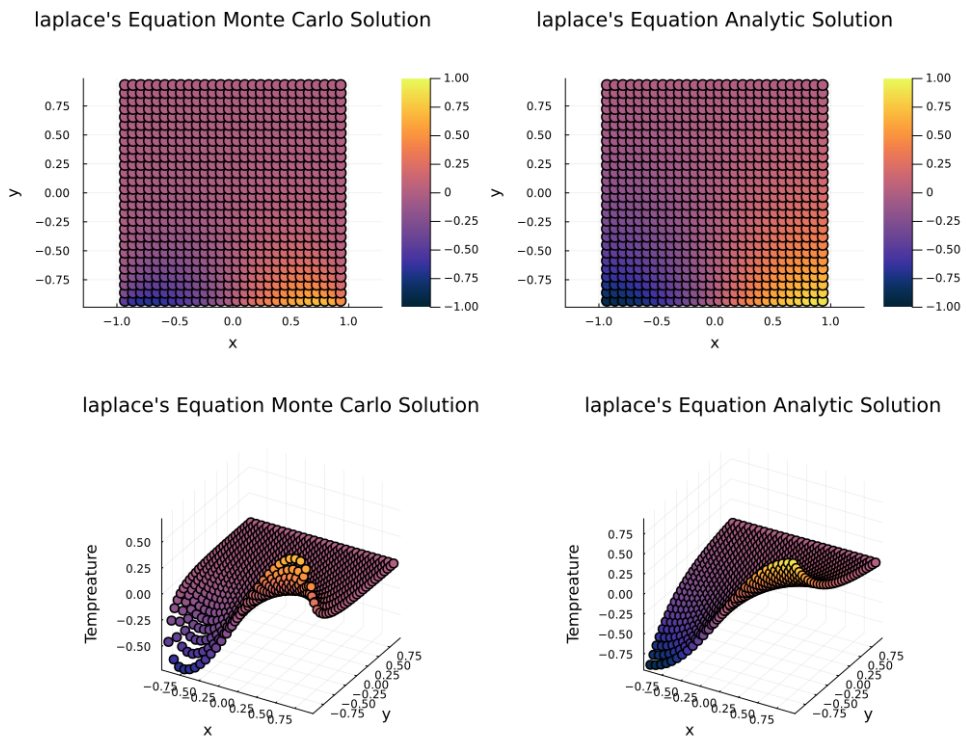

Figure 3: Solución en cuadrado con condición de frontera simple.

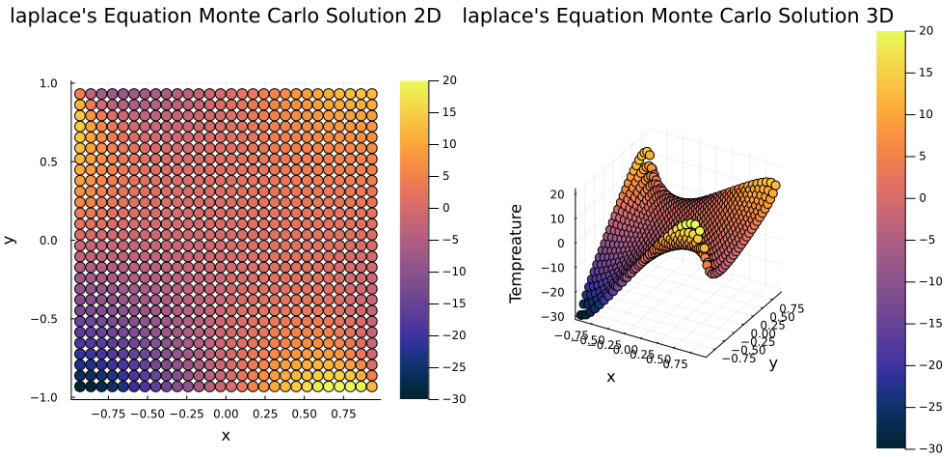

Figure 4: Solución en cuadrado con condición de frontera difícil.

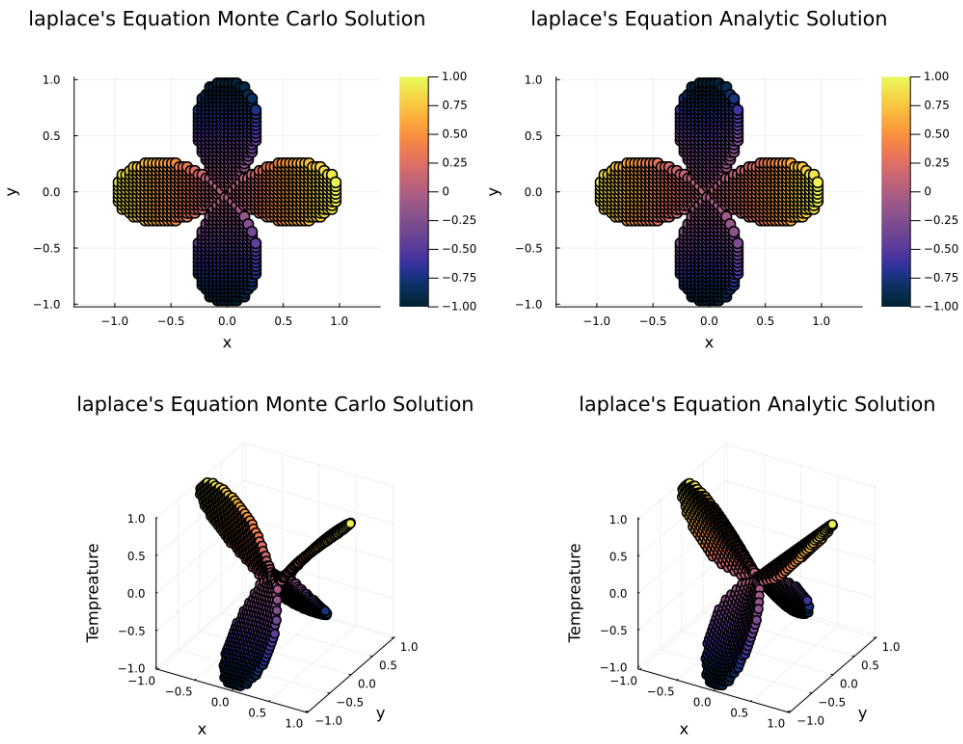

Figure 5: Solución para la geometría "flor" $r(\theta) = \cos(2\theta)$.

subraya la robustez del enfoque, incluso en geometrías no triviales como la frontera floral y las condiciones mixtas en el cuadrado.

No obstante, persisten limitaciones inherentes al método: su convergencia lenta y la elevada varianza exigen el empleo de técnicas adicionales de reducción de varianza (por ejemplo, muestreo estratificado o control de variables de importancia). Además, la sobrecarga de sincronización y la gestión de memoria en entornos multi-hilo pueden convertirse en cuellos de botella si no se optimizan adecuadamente los recursos de hardware.

Como líneas de trabajo futuro, se propone:

- Incorporar esquemas adaptativos de paso temporal y spatial para mejorar la eficiencia en regiones críticas del dominio.
- Explorar la aceleración en GPU, donde el paralelismo masivo puede amplificar aún más el speedup en millones de trayectorias.
- Integrar métodos de reducción de varianza avanzados, como procesos de control o muestreo en subdominios, para disminuir el número de réplicas necesarias.
- Extender el marco a problemas dinámicos (ecuaciones parabólicas no estacionarias) aprovechando la formulación completa de Feynman–Kac en tiempo finito.

En definitiva, la confluencia de teoría estocástica, lenguajes de alto rendimiento y técnicas de paralelismo abre un camino prometedor para afrontar problemas de frontera de gran complejidad, con aplicaciones que van desde la simulación electrostática y la transferencia de calor hasta la valoración de derivados financieros y la modelación en biomedicina.

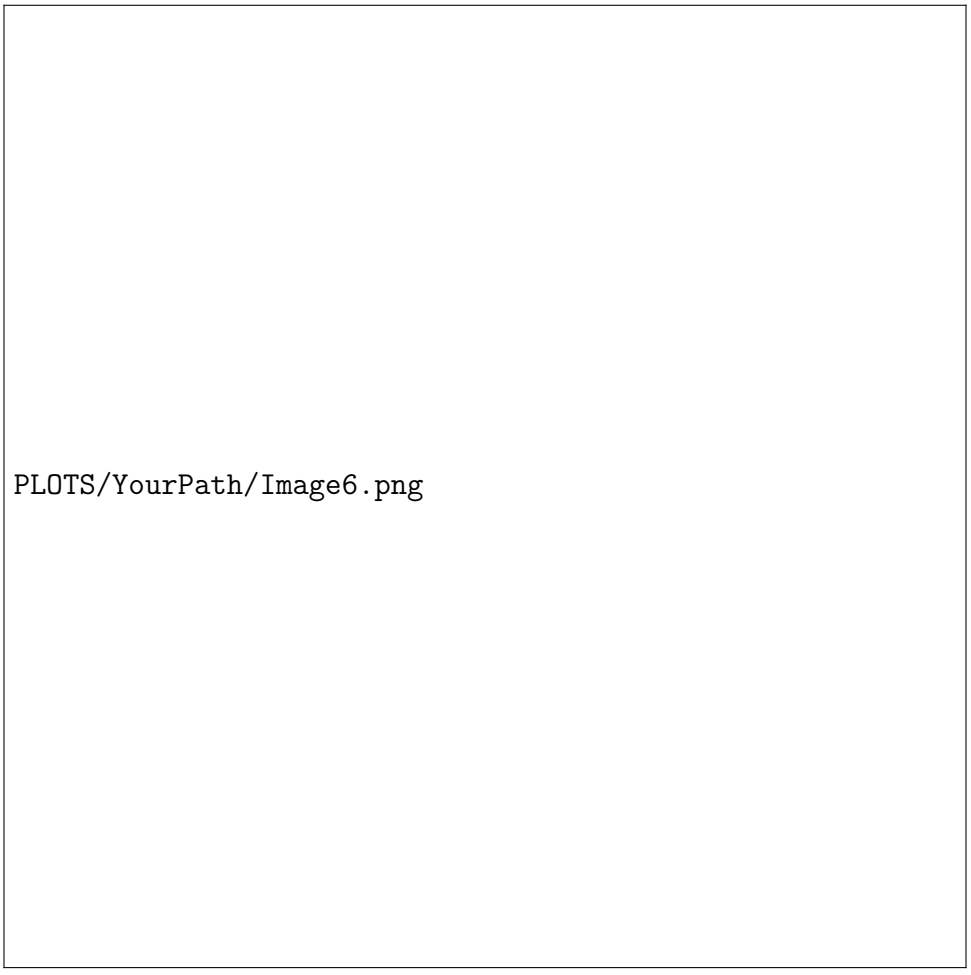

Figure 6: Descripción de la imagen 6.

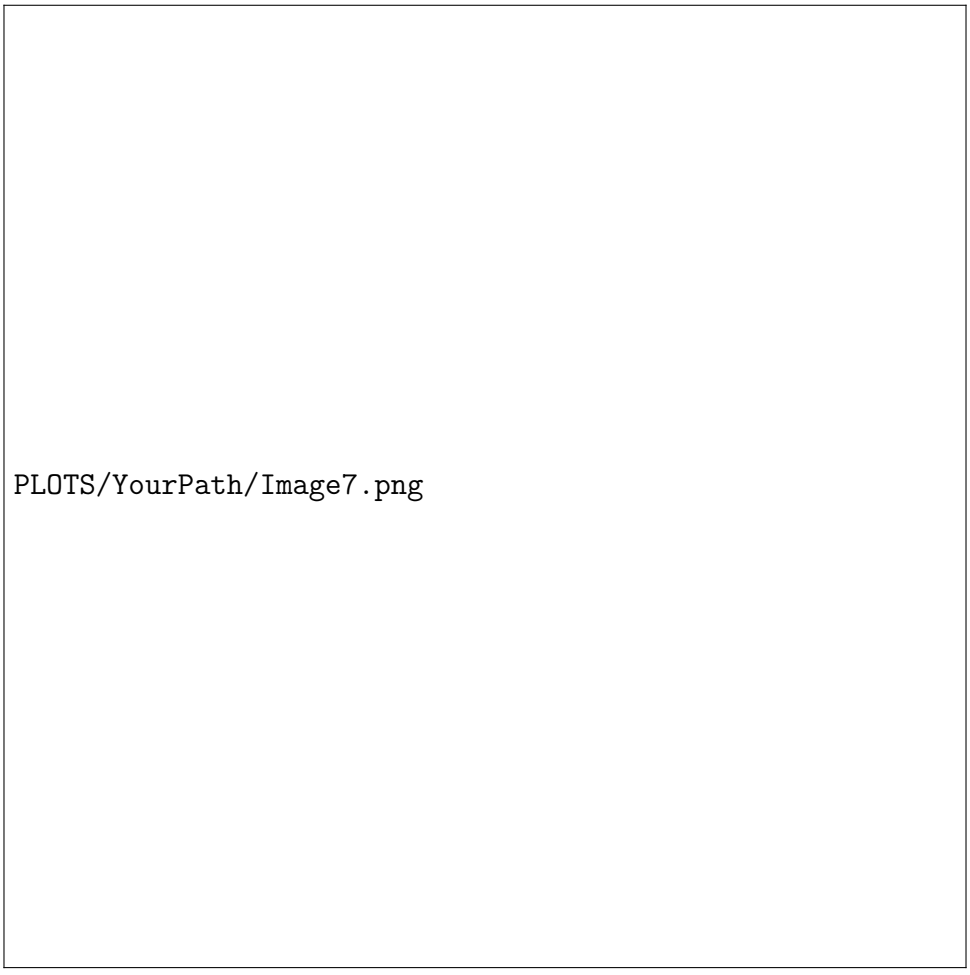

Figure 7: Descripción de la imagen 7.

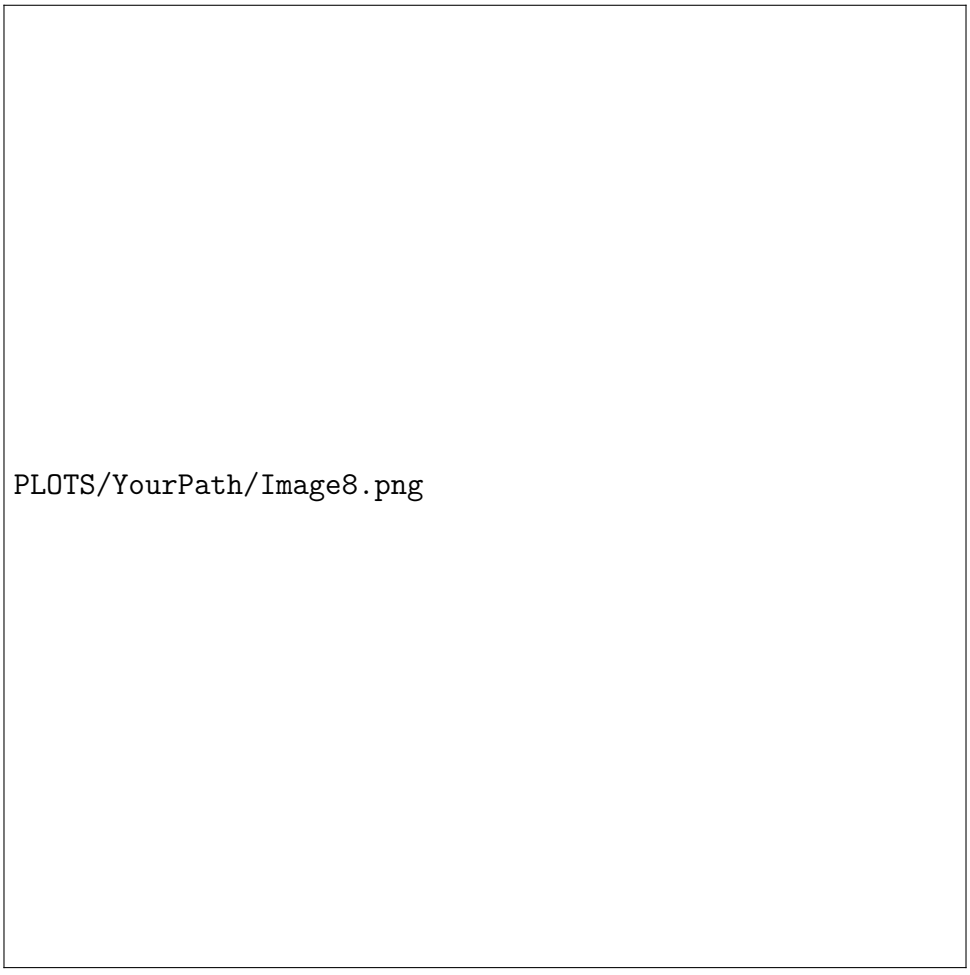

Figure 8: Descripción de la imagen 8.

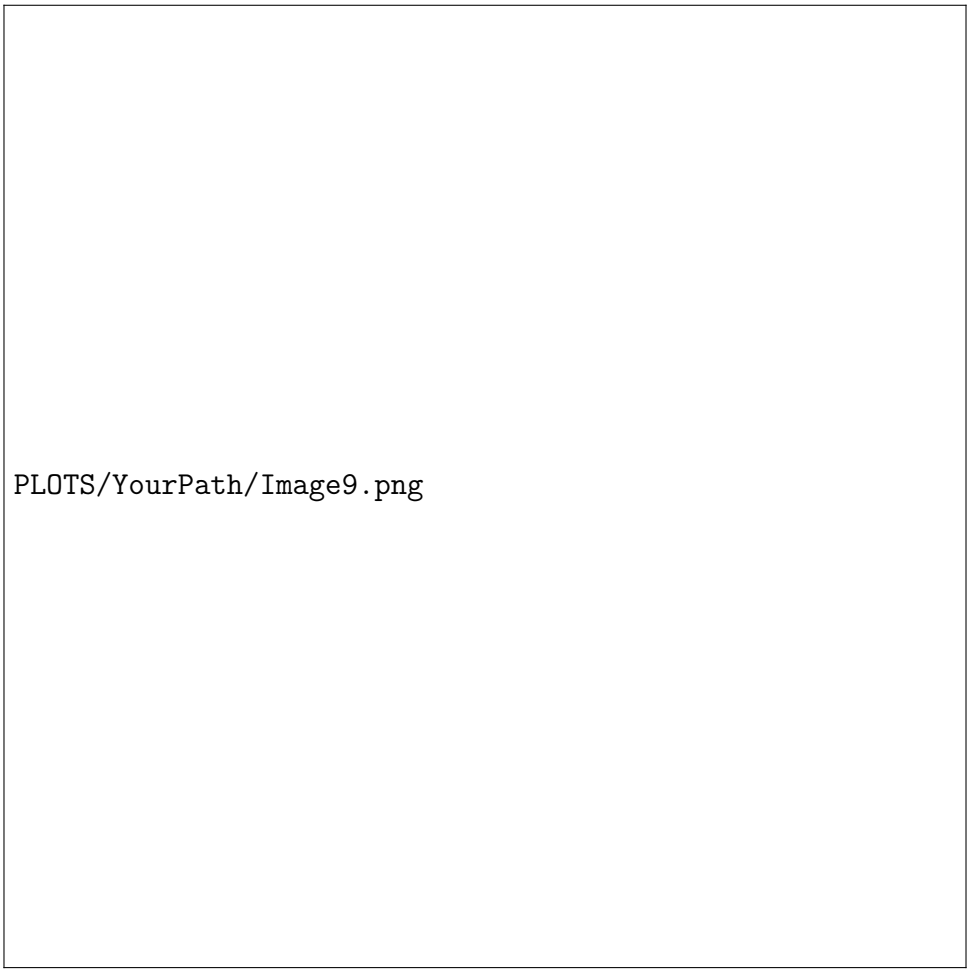

Figure 9: Descripción de la imagen 9.

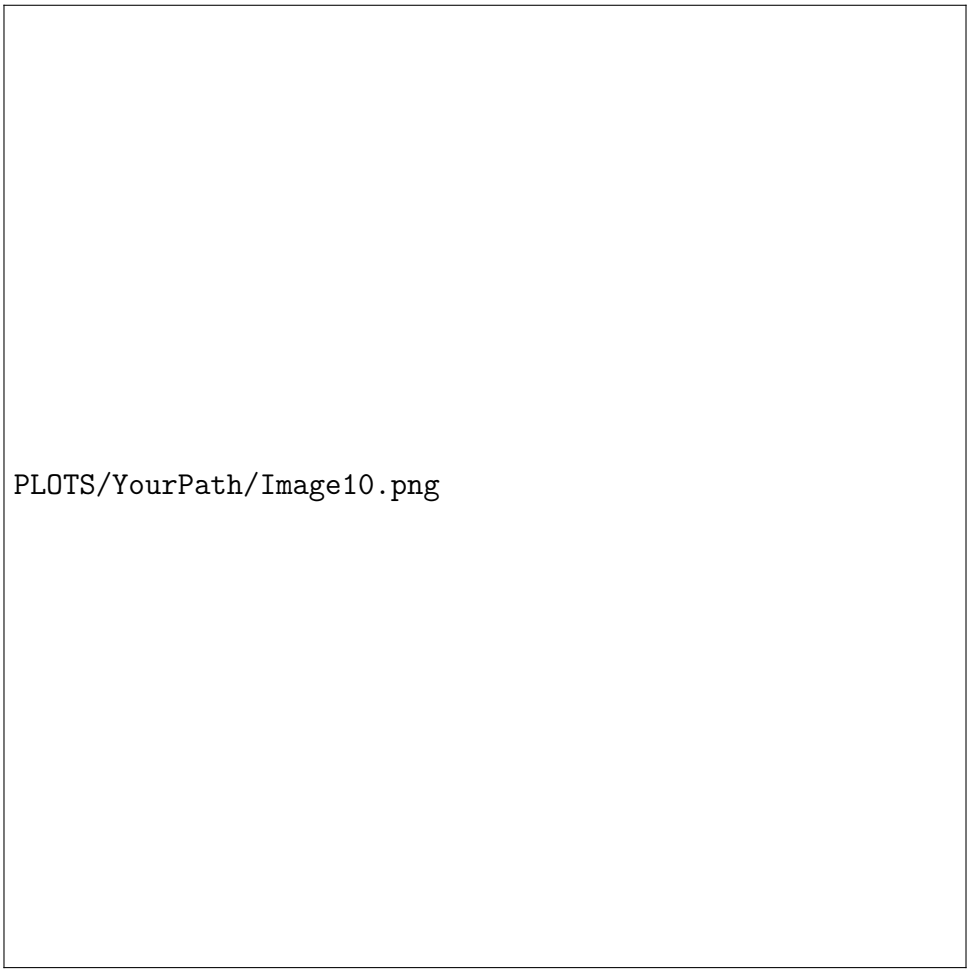

Figure 10: Descripción de la imagen 10.