

UNIVERSIDAD TÉCNICA
FEDERICO SANTA MARÍA

Departamento de Matemática
MAT468
Segundo Semestre 2023

Certamen 1

Simulación estocástica

Nombre: Diego Alejandro Astaburuaga Corveleyn
Rol: 202010018-7

Valparaíso, Octubre de 2023

Índice

1. Preliminares	2
2. Pregunta 1.	3
2.1. Planteamiento del problema	3
2.2. Esquema de simulación.	3
2.3. Tiempos de simulación con respecto a la dimensión p	4
2.4. Comprobación de que el método funciona.	9
2.4.1. Verificación del promedio.	10
2.4.2. Verificación de la varianza de la matriz.	13
2.5. Aspectos técnicos.	15
2.6. Información que puede ser relevante.	15
2.7. Simulación de matrices con valores propios acotados arbitrariamente.	16
2.8. Extensión a k fraccionario.	16
2.8.1. Modelamiento.	16
2.8.2. Tiempos de realización respecto a p	17
2.8.3. Comprobación de que el método funciona.	18
2.9. Conclusión.	20
3. Pregunta 2.	21
3.1. Planteamiento del problema.	21
3.2. Inciso a)	21
3.3. Inciso b)	22
3.4. Inciso c)	22
4. Pregunta 3.	26
4.1. Planteamiento del problema.	26
4.2. Simular la secuencia W_k	26
4.3. Simulando funciones en $L^2([0, 1])$	30
4.3.1. Simular funciones periódicas y pares en $L^2([-1, 1])$	30
4.3.2. Simulando funciones estrictamente crecientes en $L^2([0, 1])$	32
4.3.3. Simular funciones positivas que integren 1 en $L^2([-1, 1])$	35
4.3.4. Comentarios finales.	36
5. Pregunta 4.	37
5.1. Planteamiento del problema.	37
5.2. Supuestos del modelamiento.	39
5.3. Entendiendo una simulación.	39
5.4. Múltiples simulaciones.	44
5.5. Múltiples equipos.	47
5.6. Conclusión.	49

1. Preliminares

A continuación se presenta la entrega del certamen 1 del ramo simulación estocástica. Las imágenes y códigos mostrados son de propia autoría con fuerte apoyo de herramientas computacionales y documentos a libre disposición en internet, información encontrada en el mismo curso y otros.

Se pueden encontrar todos los códigos utilizados en mi repositorio de GitHub (<- cuyo enlace está aquí). En caso de no poder utilizar el enlace anterior, la url es la siguiente:

<https://github.com/Darkrayyss/Certamen-1-MAT468-Diego-Astaburuaga>

Se remarca que todos los códigos y comentarios se encuentran en los archivos pertinentes a cada pregunta, por ejemplo:

Pregunta1.ipynb

2. Pregunta 1.

En esta parte se encuentra todo lo pertinente a la pregunta 1 del certamen 1 de simulación estocástica, para más detalles sobre los códigos, puede dirigirse a la entrega del documento o bien al **repositorio Github**, luego a la carpeta **Entrega Certamen** y al documento **Pregunta1.ipynb**.

pd: En relación a esta pregunta, algunas imágenes y desarrollos se encuentran exclusivamente en la carpeta de códigos de prueba, en específico en el archivo **Pregunta1.ipynb**.

2.1. Planteamiento del problema.

El problema consiste en simular matrices simétricas y definidas positivas desde la función de densidad de probabilidad:

$$f(X|k, V) = \frac{\det(X)^{(k-p-1)/2}}{2^{kp/2} \Gamma_p(k/2) \det(V)^{k/2}} \exp\left(-\frac{1}{2} \text{Tr}(V^{-1}X)\right) \quad (1)$$

que depende de un valor k número real (primeramente asumido entero) tal que $k > p - 1$ y V una matriz simétrica y definida positiva de dimensión $p \times p$.

Investigando sobre esto, se tiene que esta función de densidad de probabilidad es la correspondiente a la distribución Wishart de dimensión p , k grados de libertad y matriz de covarianza $\Sigma = V$, desde ahora simplemente Σ . Esto se puede verificar desde las siguientes fuentes:

- Lecture 2. The Wishart distribution (University of Pittsburgh.)
- Wishart Distributions and Inverse-Wishart Sampling (Washington University in St. Louis.)
- MAT-269: Distribución Wishart (Curso análisis multivariado, Felipe Osorio.)

Desde estas definiciones, para $X = \sum_{i=1}^k Z_i Z_i^T$ con Z_1, \dots, Z_k variables aleatorias independientes desde una distribución $N_p(0, \Sigma)$, se dice que sigue una distribución de Wishart $W_p(k, \Sigma)$.

2.2. Esquema de simulación.

La base de la simulación será utilizar variables aleatorias desde una distribución normal univariada de media 0 y varianza igual a 1. Para esto se propone lo siguiente, suponiendo que se cuenta con la matriz Σ simétrica y definida positiva:

1. Simule $Y_i^{(1)}, \dots, Y_i^{(p)}$ de forma independiente desde una distribución $N(0, 1)$ para $i = 1, \dots, k$.
2. Escriba $Y_i = (Y_i^{(1)}, \dots, Y_i^{(p)})^T$ para $i = 1, \dots, k$.
3. Calcule la factorización de Cholesky de Σ dada por $\Sigma = BB^T$.
4. Obtenga $Z_i = BY_i$ para $i = 1, \dots, k$.
5. Construya $X = \sum_{i=1}^k Z_i Z_i^T$.

Se afirma que el esquema de simulación es consistente con lo que se busca simular dado que se tienen los siguientes hechos:

- Si $Y_1, \dots, Y_p \sim N(0, 1)$, entonces $Y = (Y_1, \dots, Y_p)^T \sim N_p(0, I_{p \times p})$.
- Si $Y \sim N_p(0, I_{p \times p})$ y $\Sigma = BB^T$, entonces $BY \sim N_p(0, \Sigma)$ (Análisis de regresión, Osorio).
- Si Σ es definida positiva, entonces X mediante la construcción anterior será definida positiva con probabilidad 1 (Análisis multivariado, Osorio).

Además se nota que los cálculos anteriores pueden ser simplificados mediante utilizar:

$$X = \sum_{i=1}^k Z_i Z_i^T = \sum_{i=1}^k BY_i Y_i^T B^T = B \left(\sum_{i=1}^k Y_i Y_i^T \right) B^T \quad (2)$$

Finalmente como se requiere conocer previamente la matriz Σ para empezar a simular, esta se genera mediante el siguiente algoritmo:

1. Dada la dimensión p , genere B : una matriz aleatoria $p \times p$.
2. Calcule $\Sigma = BB^T$.
3. Verifique que Σ tenga factorización Cholesky, en caso contrario, vuelva al punto inicial.

Donde esta construcción nos asegura que Σ sea simétrica y definida positiva.

Es importante recalcar que este método permite simular la Wishart pero para parámetros k enteros, al final de la pregunta se aborda como utilizar todo lo anterior para k fraccionario.

2.3. Tiempos de simulación con respecto a la dimensión p .

Para entender este aspecto debemos considerar primero distintas probabilidades entre la elección de la dimensión p y los grados de libertad k , teniendo en cuenta la relación $k \geq p$. Con esto en mente, para distintos valores de k , se obtuvo una estimación de los tiempos de computo en promedio, para varios valores de p , lo cual se puede ver resumido en el siguiente gráfico:

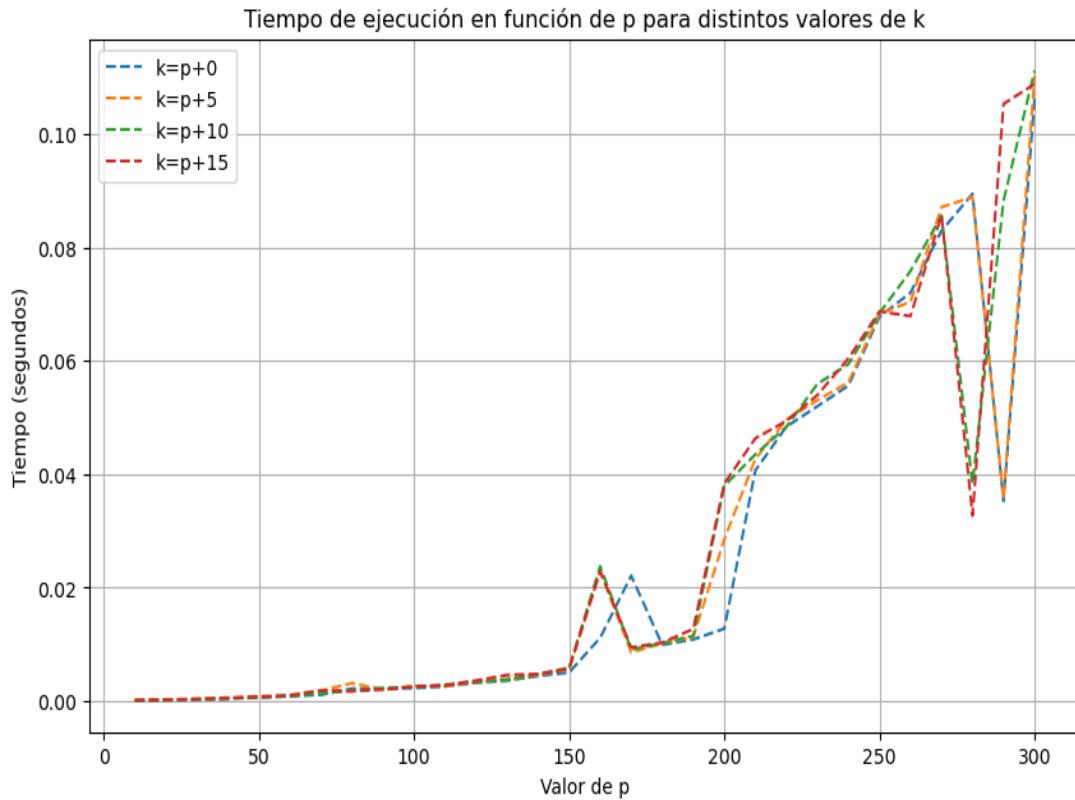


Figura 1: Tiempo de ejecución promedio en función de p para distintos valores de k .

observando que se tiene lo esperado, que para dimensiones mayores el tiempo de computo aumenta, por otro lado, se nota que este tiempo de computo no se ve afectado por la cantidad de grados de libertad, al menos no significativamente. Se menciona que esto es lo esperado, dado que aumentar la dimensión aumenta el tamaño de las matrices involucradas de forma cuadratica, mientras que aumentar los grados de libertad sólo involucra hacer una operación matricial extra.

Durante la simulación, para obtener el promedio también se simularon distintas matrices Σ para cada iteración lo cual no afectó a la medición del tiempo de simulación dado que demora relativamente poco en relación a la simulación de la matriz desde la Wishart:

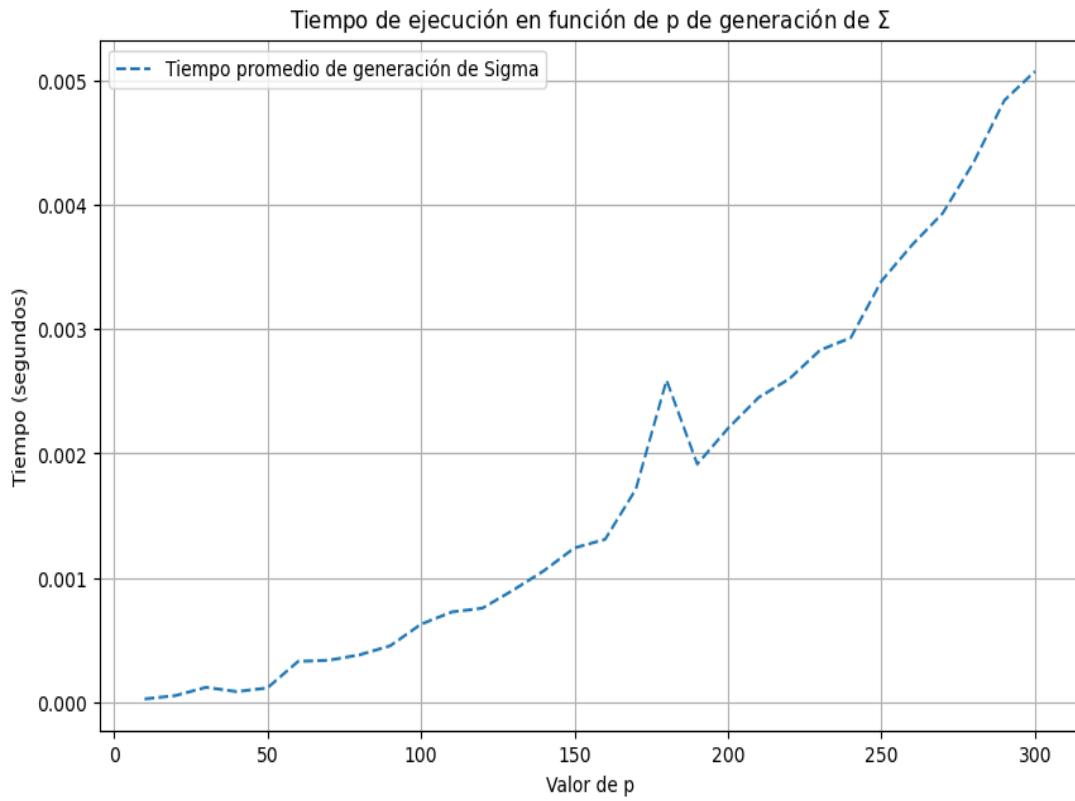


Figura 2: Tiempo de simulación promedio en función de p de la matriz Σ .

Dado que entendemos que el tiempo de simulación tiene un comportamiento similar para distintos k , ajustemos un modelo exponencial y un modelo cuadrático a los tiempos de computo promedio para $k = p$ con el fin de entender su comportamiento:

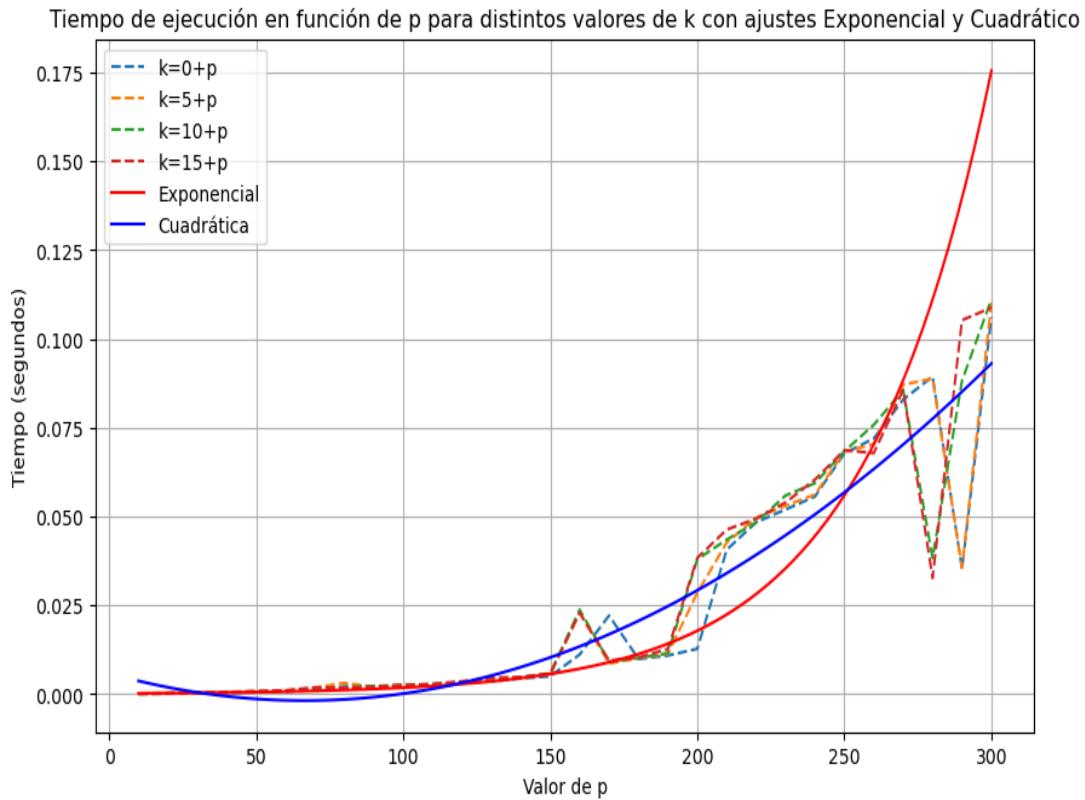


Figura 3: Ajuste exponencial y cuadrático a los tiempos de computo para distintos k .

donde el ajuste parece ser más cuadrático. Con el fin de reducir la varianza en la estimación del promedio del tiempo de computo para cada p , se realiza nuevamente el experimento de simulación pero considerando más iteraciones para el promedio, luego repitiendo el procedimiento de ajustar modelos se tiene que claramente el tiempo de computo crece de forma cuadrática en función de la dimensión pedida.

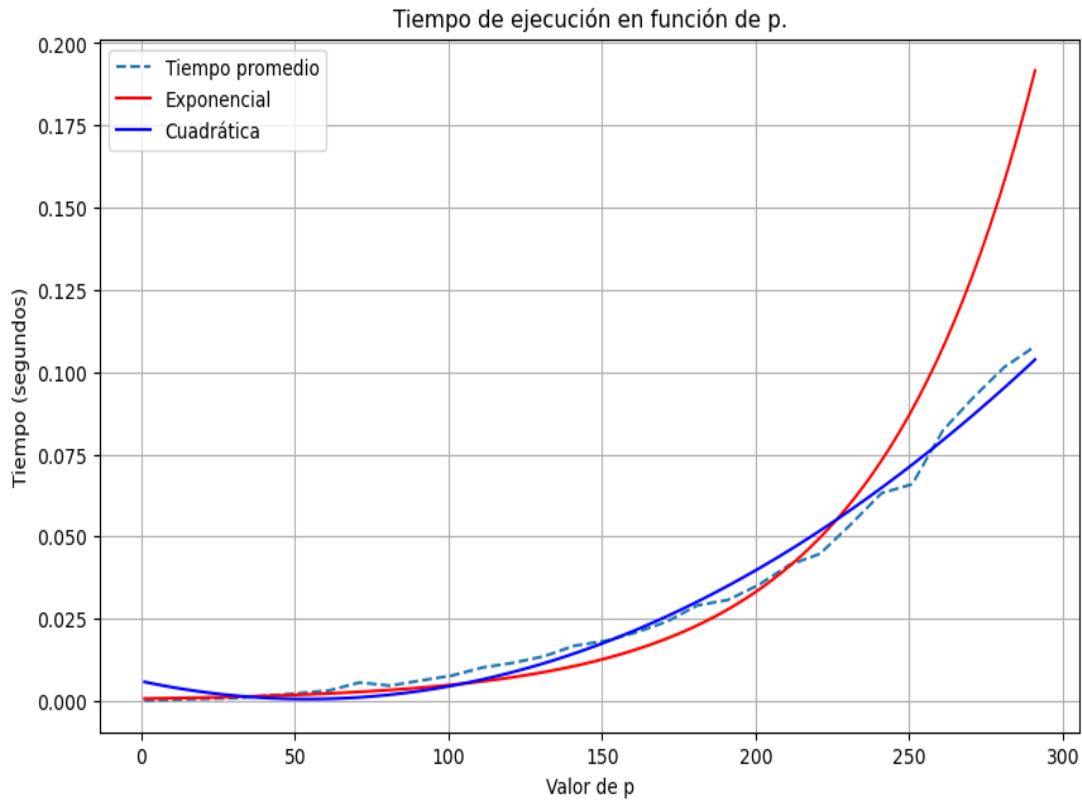


Figura 4: Ajuste exponencial y cuadrático a los tiempos de computo para $k = p$.

Finalmente se menciona que para $p = 10$ fijo, se tiene que el tiempo de computo crece de forma lineal en función de k .

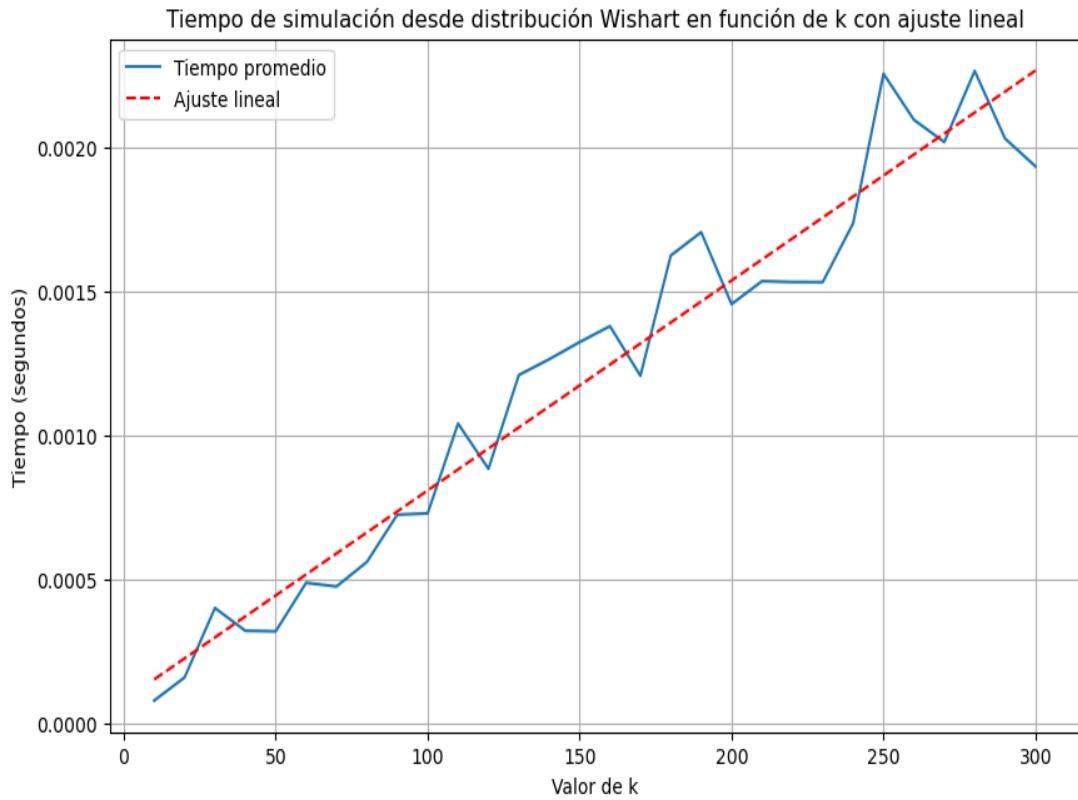


Figura 5: Ajuste lineal para los tiempos de computo para $p = 10$.

2.4. Comprobación de que el método funciona.

Para esta sección, utilizaremos el hecho de que conocemos teóricamente los momentos de la distribución que queremos simular, más específicos estos son los dados por las observaciones de la evaluación, donde si $X \sim W_p(k, \Sigma)$:

1. $E[X] = k\Sigma$.
2. Escribiendo $X = (X_{ij})$ y $\Sigma = (\sigma_{ij})$ para $1 \leq i, j \leq p$. Se tiene que $Var[X_{ij}] = k(\sigma_{ij}^2 + \sigma_{ii}\sigma_{jj})$.
3. Se verifica que $X_{ii} \sim \sigma_{ii}\chi_k^2$.

donde se obviará esta tercera observación dado que es claro por la construcción del método que la diagonal de X distribuye como una χ_k^2 ponderada, esto debido a que

$$X = B \left(\sum_{i=1}^k Y_i Y_i^T \right) B^T$$

y

$$\left(\sum_{i=1}^k Y_i Y_i^T \right)_{jj} = \sum_{i=1}^k (Y_i^{(j)})^2$$

distribuyen χ_k^2 , luego se obtiene la observación considerando que $\Sigma = BB^T$.

2.4.1. Verificación del promedio.

Considere el estimador insesgado de $E[X] = k\Sigma$ como

$$T(\underline{X}) = \bar{X}$$

Luego para medir la distancia entre la matriz de estimación y la matriz del valor teórico, utilizamos la norma de Frobenius dada por

$$\|A\|_F := (\text{tr}(A^T A))^{1/2}$$

Veamos como se comporta la distancia de Frobenius entre el estimador de la media y el valor real para una cantidad fija de iteraciones y distintos valores de k y p adecuados:

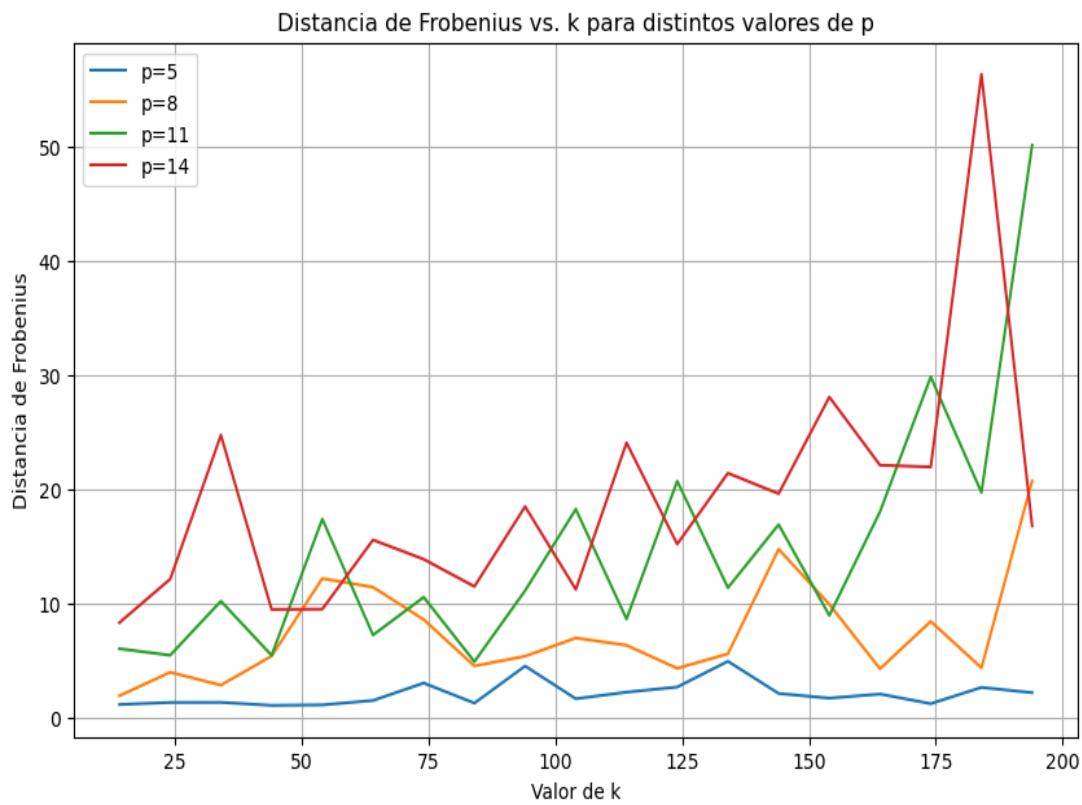


Figura 6: Distancia de Frobenias para n fijo y k, p variables.

de donde se observa que a medida que aumenta el valor de k , pareciera que la estimación de la media fuera menos precisa, además, como es de esperarse la estimación se ve afectada a medida que aumenta la dimensión p por lo que es de esperarse que se requieran más iteraciones para converger al valor esperado.

Realizando un estudio de simulación más exhaustivo acerca de la convergencia al valor esperado que puede verse en el código, se deduce que la estimación es correcta pero es muy sensible, en término de número de iteraciones, a la dimensión p . Para reafirmar visualmente esta idea, se presentan 3 experimentos de simulación para cada $p \in \{5, 8, 11, 14, 17\}$ con la finalidad de ilustrar como disminuye la distancia entre el promedio y el valor esperado a medida que aumenta el número de iteraciones.

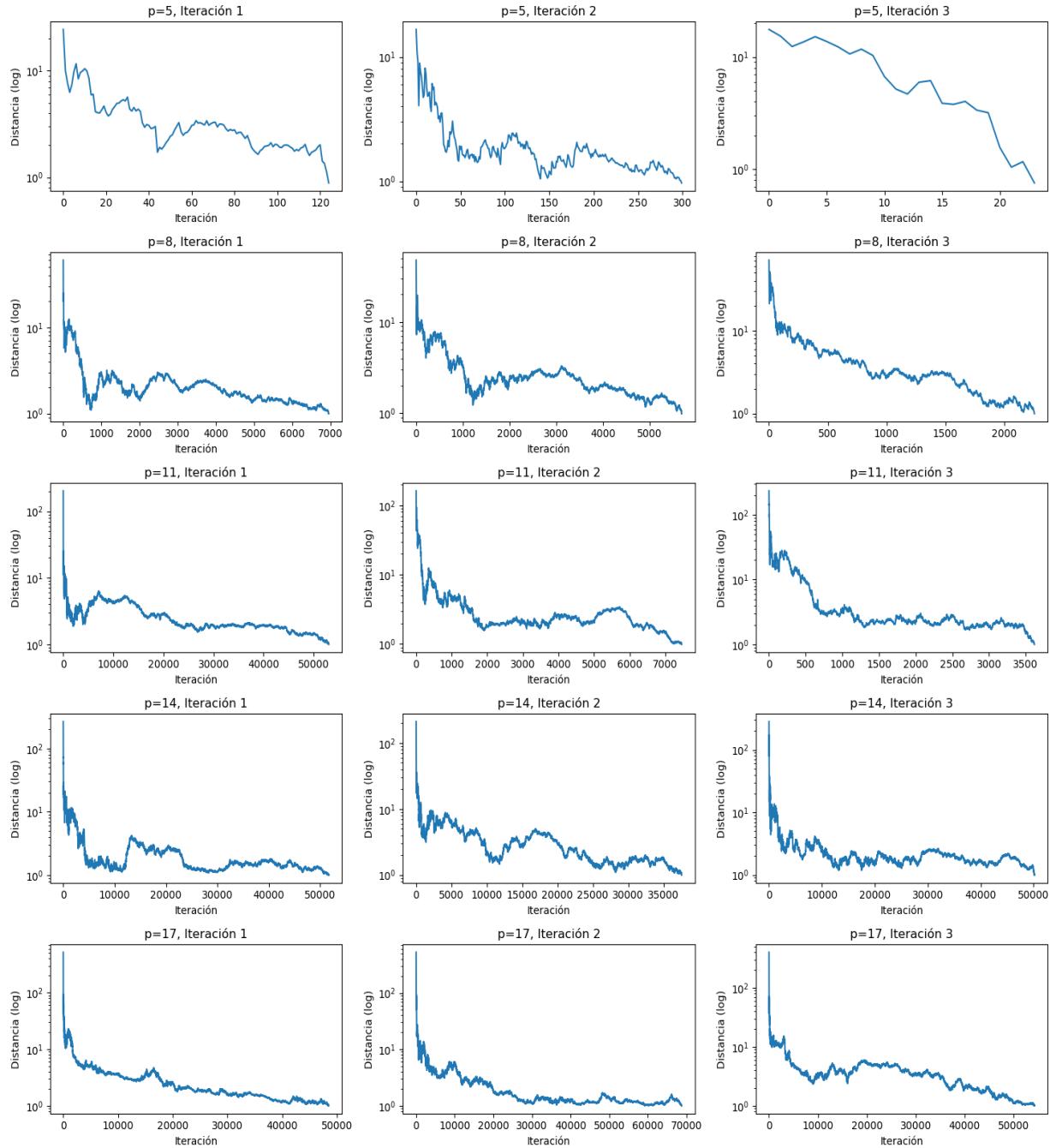


Figura 7: Ejemplificación de la convergencia del promedio al valor esperado.

para finalizar este punto, se menciona que la cantidad de iteraciones necesaria para alcanzar cierta tolerancia fija, tiene alta varianza, es decir, la cantidad de iteraciones necesaria cambia de forma significativa.

2.4.2. Verificación de la varianza de la matriz.

Para este punto se debe entender que la varianza muestral de una muestra desde $W_p(k, \Sigma)$, significa la matriz donde cada entrada tiene la varianza para esa misma entrada. Similar a lo realizado para el valor esperado, veamos la distancia de Frobenius entre la matriz de varianza teórica y la matriz de varianza muestral para distintos valores de k y p :

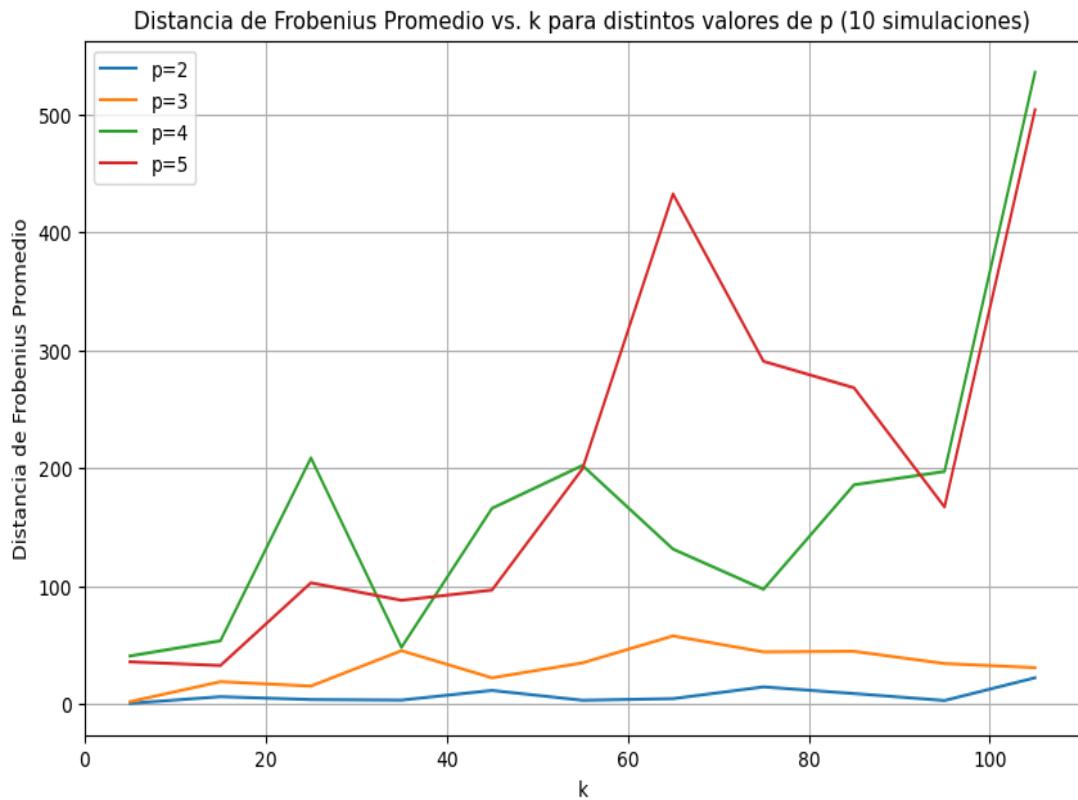


Figura 8: Distancia Matrices de varianza teórica y muestral para distintos p, k .

observando que; a medida que crece el valor de k posiblemente se requieren de más muestras para acercarse a la varianza teórica y que, para mayores valores de p , es crecimiento en la distancia aumenta significativamente.

Nuevamente, simulando para distintos valores de p , se puede mostrar experimentalmente como la varianza muestral se acerca a la varianza teórica con la cantidad suficiente de iteraciones:

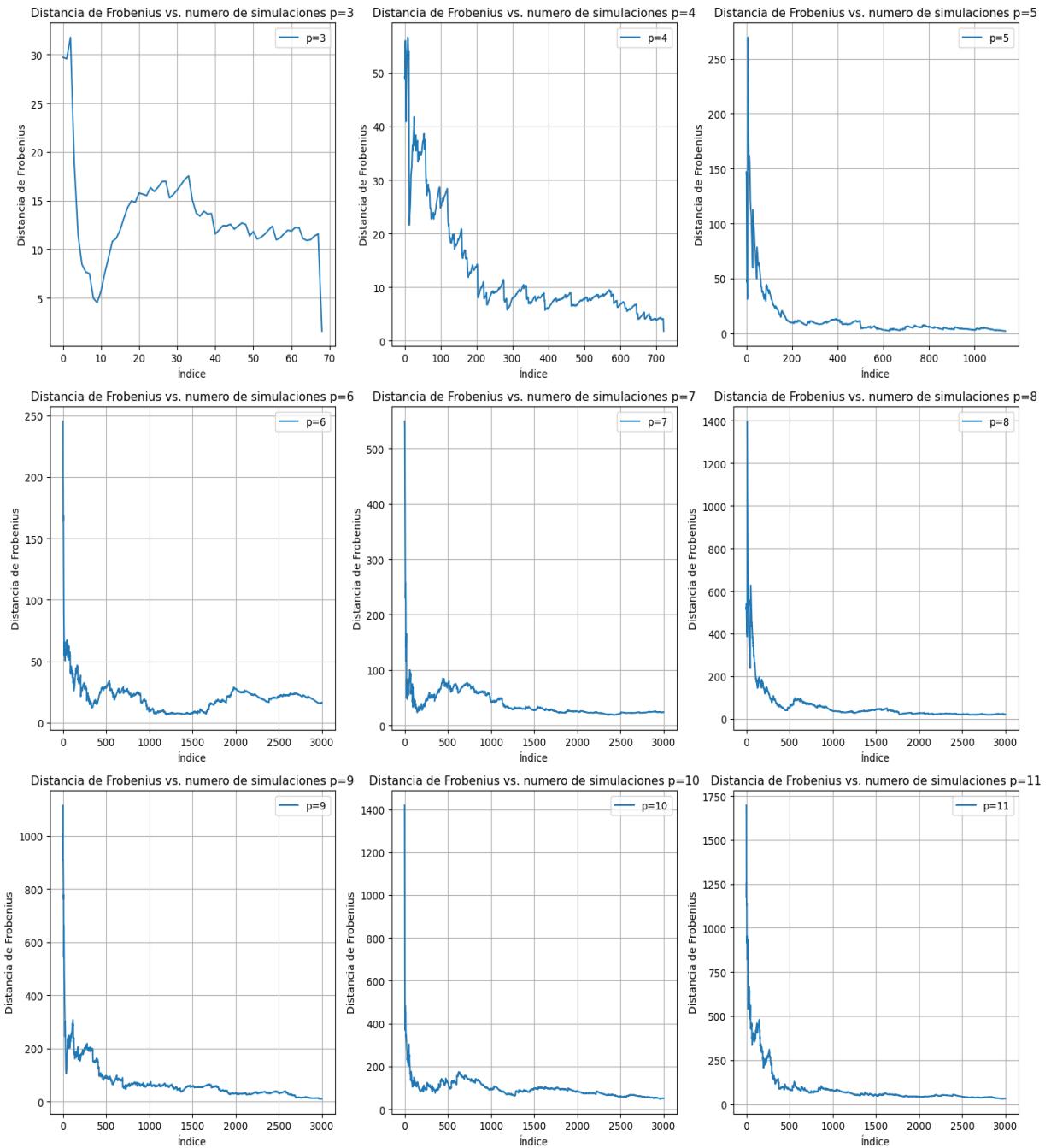


Figura 9: Ejemplificación de la convergencia de la varianza muestral a la varianza teórica.

Finalmente se ilustra a modo de ejemplo que una distancia de Frobenius "alta" (mayor a 1) no indica realmente que las matrices sean muy distintas y por lo tanto, podemos afirmar que efectivamente el método simula matrices desde la distribución objetivo, al menos, bajo el criterio de los momentos

que se está utilizando.

Varianza teórica:

$$\begin{bmatrix} 6,65 & 8,49 & 2,00 & 13,70 \\ 8,49 & 11,68 & 2,63 & 17,86 \\ 2,00 & 2,63 & 0,64 & 4,28 \\ 13,70 & 17,86 & 4,28 & 28,99 \end{bmatrix}$$

Varianza calculada:

$$\begin{bmatrix} 6,32 & 8,09 & 1,90 & 13,02 \\ 8,09 & 11,23 & 2,51 & 17,02 \\ 1,90 & 2,51 & 0,61 & 4,06 \\ 13,02 & 17,02 & 4,06 & 27,51 \end{bmatrix}$$

Distancia entre ambas: 2,308126783446759

2.5. Aspectos técnicos.

Sobre el hardware utilizado para hacer las simulaciones y trabajo, se dejan las indicaciones esta sección:

- Procesador 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz.
- RAM instalada 16,0 GB (15,7 GB usable).
- Tipo de sistema Sistema operativo de 64 bits, procesador basado en x64.

2.6. Información que puede ser relevante.

Para el trabajo y formación del esquema de simulación se intentó recurrir primeramente a las técnicas vistas en el curso pero estas fueron descartadas por diversos motivos, principalmente se escogió la actual dado que fue pertinente realizar una investigación sobre la distribución objetivo para intentar encontrar métodos más directos de simulación, el cual fue conseguido.

Sin perjuicio de lo anterior, se realizarán algunos comentarios de interés:

- Se descartó a priori utilizar el algoritmo de Metropolis-Hastings usual debido a que se prefirió buscar otros métodos antes de empezar a programarlo y proponer distribuciones instrumentales.
- Fue tentativa la opción de utilizar el algoritmo de Langevin ajustado con Metropolis debido a que calcular $\nabla \log f(x)$ no parece ser teóricamente difícil dado que el denominador no tiene influencia debido a la derivación y que existen propiedades de valor matemático que relacionan el operador gradiente con la traza y el determinante de las matrices.
- Finalmente algoritmos tipos Gibbs Sampler fueron desechados debido a que no fue sencillo reconocer las densidades condicionales y, en caso de conocerlas, al ser una matriz $p \times p$, la cantidad de pasos necesarios para generar una simulación eran del orden de p^2 siendo computacionalmente inviable.

2.7. Simulación de matrices con valores propios acotados arbitrariamente.

Explicitando más la sección, se busca adaptar los algoritmos ya creados para poder realizar simulaciones pero únicamente para matrices cuyos valores propios se encuentre en el intervalo $(0, \lambda_{max}]$, donde 0 no puede ser un valor propio dado a la condición de que las matrices son definidas positivas.

Con esto en mente, sea μ el valor propio más alto de X matriz generada mediante el método anterior, para dicha matriz, existe también v vector propio tal que

$$Av = \mu v$$

Se afirma que la transformación $A \rightarrow A' = \frac{\lambda_{max}}{\mu} A$ genera matrices cuyos valores propios están acotados por λ_{max} , en efecto, sean μ', v' valor y vector propio de A' :

$$A'v' = \mu'v \implies Av' = \frac{\mu\mu'}{\lambda_{max}}v'$$

por definición de μ se tiene que

$$\mu'/\lambda_{max} \leq 1$$

probando que A' tiene sus valores propios acotados por λ_{max} .

2.8. Extensión a k fraccionario.

2.8.1. Modelamiento.

Esta parte se sustenta totalmente de los análisis anteriores y sirve como una extensión del método, casi agregada como anexo debido a la complejidad ya presente en el actual trabajo.

Hasta el momento se tiene la capacidad de simular para k entero tal que $k > p - 1$, para poder considera k fraccionario, se utilizará el método de aceptación rechazo. Para esto:

Sea g la distribución instrumental a utilizar dada por:

$$g(X) = \det(X)^{(k'-p-1)/2} \exp\left(-\frac{1}{2}Tr(\Sigma^{-1}X)\right)$$

con k' el entero mayor más cercano a k , luego se desea simular desde:

$$h(X) = \det(X)^{(k-p-1)/2} \exp\left(-\frac{1}{2}Tr(\Sigma^{-1}X)\right)$$

donde es necesario entender el cuociente h/g :

$$\frac{h(X)}{g(X)} = \det(X)^{(k-k')/2}$$

en donde estamos omitiendo las constantes normalizadoras de cada densidad.

Mediante simulación exhaustiva se puede calcular una cota para el determinante de X dado que conocemos previamente k, p, Σ que se verifique para una gran cantidad de casos, por ejemplo, realizar 10.000 simulaciones de X desde la instrumental, se puede calcular el determinante elevado a la potencia descrita y obtener una cota como el mayor valor de todos ellos. Denotando dicha cota como M , tendríamos que

$$\frac{h(X)}{Mg(X)} \leq 1$$

lo que da pie al algoritmo de aceptación-rechazo.

Se añade el comentario de, para simular en este nuevo método para k entero, el algoritmo no requiere pasar por el paso de aceptación-rechazo y utiliza la función creada con anterioridad.

2.8.2. Tiempos de realización respecto a p .

Ya entendemos bien como generar valores de una distribución Wishart con k grados de libertad enteros, ahora, para simular para k grados de libertad fraccionarios, se requiere primero calcular la cota M , la cual tiene un tiempo de computo de $n_{samples} \cdot C(p, k)$ donde $n_{samples}$ es el número de simulaciones usadas para calcular la cota y $C(p, k)$ es el tiempo de computo para simular una Wishart con k' entero mayor más cercano a k de dimensión p .

Por lo tanto se afirma sin experimentación que los tiempos de computo de calcular la cota M crecen de forma cuadrática respecto a p y de forma lineal respecto a k .

Con lo anterior, es claro que, para entender los tiempos de realización de una simulación desde $W_p(k, \Sigma)$ para k fraccionario, basta con entender la taza de aceptación-rechazo.

Realizando un experimento de simulación para

```
# Parámetros
p = 5
k = p + 1.652
tol = 0.1
```

y Σ distinto en cada iteración, se obtiene que el ratio de aceptación - rechazo tiene un promedio de 1.246 de rechazos por cada falla aproximadamente, es decir, por cada 4 matrices aceptadas, se tuvieron que rechazar 5. Se deja como trabajo a futuro entender la relación entre la tasa de falla promedio y la dimensión p , los grados de libertad k y la selección de la matriz Σ . De momento se omite su relación con la dimensión p debido a que no se dispone de tiempo para realizar tales simulaciones. Se comenta de manera informal que durante la elaboración de las pruebas, se observó que el ratio de aceptación-rechazo si dependía de la matriz.

2.8.3. Comprobación de que el método funciona.

En virtud del tiempo, se mostrarán resultados sencillos que ilustren que el promedio sigue convergiendo en cierto sentido a la esperanza teórica dada, para luego hacer un trabajo análogo sobre la varianza.

Utilizando $k = p - 0.523 > p - 1$ escogido de forma arbitraria, se obtienen las siguientes simulaciones para distintos p :

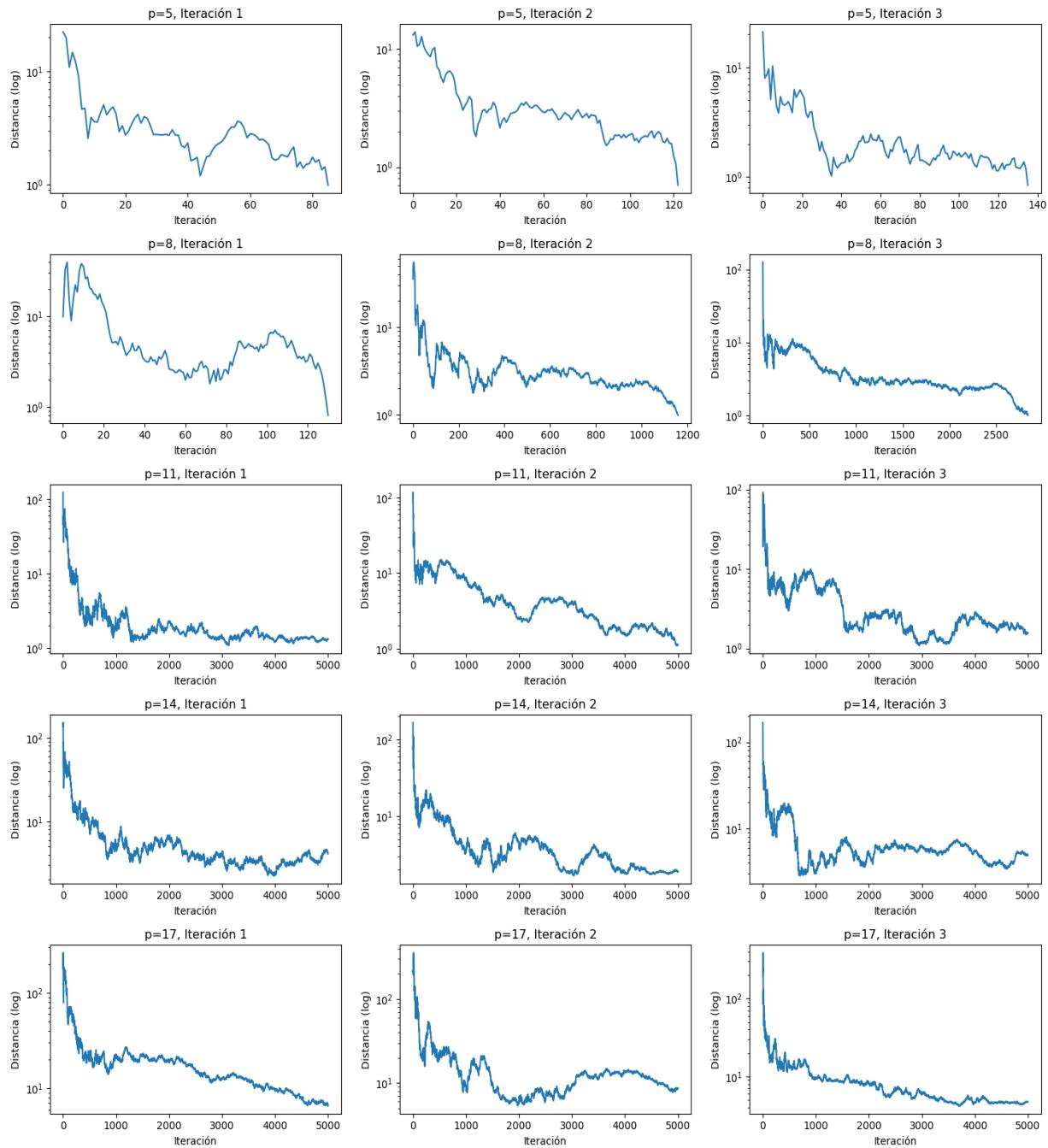


Figura 10: Distancia del promedio respecto a esperanza para distintos p .

Claramente para el valor $k = p - 0,523 > p - 1$ el método parece tener más problemas para converger. Del gráfico anterior, se debe notar que la noción de convergencia con esta norma depende de cierta manera de la dimensión que se esté trabajando, donde para las dimensiones altas, podríamos afirmar

que el hecho de que la distancia con la esperanza teórica sea menos de 10 da una noción de convergencia adecuada. Luego se realizó el mismo experimento para $k = p + 0.523$ pero no se obtuvieron conclusiones nuevas.

Sobre el cálculo de la varianza, se espera que las conclusiones realizadas para el caso k entero sean hereradas, estas no se realizarán ahora mismo pero quedan como opción a comprobar. Pese a lo anterior, se calculo para

```
# Definir los parámetros
p = 4 # Dimensión de la matriz
k = 3.542 # Parámetro k
Sigma = generate_covariance_matrix(p)

M = find_M(p, k, Sigma)
```

fijos y se obtuvo el siguiente resultado **Varianza teórica:**

$$\begin{bmatrix} 15,73 & 17,64 & 14,00 & 8,94 \\ 17,64 & 39,75 & 21,87 & 12,98 \\ 14,00 & 21,87 & 19,52 & 9,25 \\ 8,94 & 12,98 & 9,25 & 5,68 \end{bmatrix}$$

Varianza calculada:

$$\begin{bmatrix} 15,68 & 17,86 & 14,06 & 8,96 \\ 17,86 & 40,72 & 22,39 & 13,19 \\ 14,06 & 22,39 & 20,08 & 9,37 \\ 8,96 & 13,19 & 9,37 & 5,72 \end{bmatrix}$$

Distancia entre ambas: 1,4312749285053852

de donde se puede intuir que el método se verifica en este sentido.

2.9. Conclusión.

Se estudió de forma completa y transversal el trabajo de simular desde una función de densidad de probabilidad entregada, mediante la evaluación de las distintas técnicas de simulación y las características experimentales de la técnica escogida, entendiendo los tiempos de computo y verificando que la simulación es correcta.

Se recalca la importancia de conocer los tiempos de computo y como se comportan para distintos parámetros y escenarios ya que podrían permitir evitar entrar en trabajos computacionalmente inviables mediante estudiar casos más controlados y viendo la tendencia de los tiempos de computo.

3. Pregunta 2.

En esta parte se encuentra todo lo pertinente a la pregunta 2 del certamen 1 de simulación estocástica, para más detalles sobre los códigos, puede dirigirse a la entrega del documento o bien al **repositorio Github**, luego a la carpeta **Entrega Certamen** y al documento **Pregunta2.ipynb**.

3.1. Planteamiento del problema.

En esta pregunta se exploraran elementos básicos de la simulación perfecta, para más detalle, se debe revisar el certamen 1 del ramo que posee las definiciones y comentarios pertinentes.

3.2. Inciso a)

En esta sección se probará que $Algo1(\alpha)$ es un algoritmo de simulación perfecta.

Demostración. Claramente $Algo1(\alpha)$ es un esquema recursivo probabilístico por definición. Note que la probabilidad de que el algoritmo entre en recursión es $p = 1/2$, por lo tanto la probabilidad de que entre en n recursiones es p^n , por ende se puede afirmar que la probabilidad de que algoritmo entre en infinitas recursiones es 0 dado que $p^n \rightarrow 0$, por lo tanto el algoritmo termina en tiempo finito con probabilidad 1.

De otra forma, defina Y_n el evento en el cual el algoritmo termina en n recursiones o menos, también podemos definir el evento X_n como la probabilidad de que el algoritmo termina en la recursión n con X_0 el evento en el cual el algoritmo termina sin ninguna recursión, por ende:

$$P(Y_n) = P\left(\bigcup_{i=0}^n X_i\right) = \sum_{i=0}^n \frac{1}{2^{i+1}}$$

utilizando elementos de la teoría de probabilidades se prueba que el algoritmo termina en tiempo finito con probabilidad 1 ya que

$$\sum_{i=0}^{\infty} \frac{1}{2^{i+1}} = 1$$

Desde la definición de algoritmo de simulación perfecta, note que podemos utilizar $\pi_\alpha = U[\{1, 2, 3, 4, 5\}]$ para probar lo pedido, en efecto, defina X la variable aleatoria que el retorno del algoritmo, para α una familia parámetrica arbitraria, por probabilidad condicional tenemos que

$$P(X = i) = P(X_0 = i) + (1 - P(X_0 = i))P(X = i), \quad i \in \{1, 2, 3, 4, 5\}$$

donde $X_0 \sim U[\{1, \dots, 10\}]$, que se entiende como; la probabilidad de que el algoritmo retorne i es igual a la probabilidad de que lo retorne sin recursión, más la probabilidad que lo retorne luego de una recursión, por lo tanto

$$P(X = i) = \frac{1}{10} + \frac{1}{2}P(X = i)$$

por ende despejando $P(X = i) = 1/5$ para cada $i = 1, 2, 3, 4, 5$ lo cual quiere decir que $X \sim \pi_\alpha$ probando lo pedido. \square

3.3. Inciso b)

En esta sección se demostrará que $Algo2(\alpha)$ es un algoritmo de simulación perfecta que simula la distribución uniforme usando el teorema fundamental de la simulación perfecta.

Demostración. Nuevamente es claro que $Algo2(\alpha)$ es un esquema recursivo probabilístico para todo α .

Probemos que el algoritmo termina siempre en una cantidad finita de pasos con probabilidad 1, para esto note que si $\alpha_0 \leq 5$, el resultado es directo, luego para $\alpha_0 > 5$, sea $\beta \sim U[\{1, 2, \dots, \max\{5, \alpha_0\}\}]$

$$P(\text{El algoritmo entra en recursión}) = \frac{\alpha_0 - 5}{\alpha_0} = 1 - \frac{5}{\alpha_0}$$

luego suponiendo que entra el recursión con parámetro β , se tiene que

$$P(\text{El algoritmo sigue en recursión}) = \frac{\beta - \min\{5, \beta\}}{\beta} < \frac{\beta - 5}{\beta} = 1 - \frac{5}{\beta} \leq 1 - \frac{5}{\alpha_0}$$

de donde podemos obtener que

$$P(\text{El algoritmo esté en } n \text{ recursiones}) \leq \left(1 - \frac{1}{\alpha_0}\right)^n$$

entonces haciendo tender n a infinito tenemos que el algoritmo termina en tiempo finito con probabilidad 1.

Probemos que el algoritmo es localmente correcto con respecto a $\{\pi_\alpha : \alpha \in \mathcal{P}\}$ donde π_α es la uniforme discreta para $\{1, \dots, 5\}$ para todo α . Si $\alpha \leq 5$ el resultado es directo, ahora para $\alpha > 5$, sea X el retorno del algoritmo, repitiendo el formulamiento del inciso anterior y utilizando la propiedad del oráculo:

$$P(X = i) = \frac{1}{\alpha} + \left(\frac{\alpha - 5}{\alpha}\right) \frac{1}{5} = \frac{1}{5}$$

y por lo tanto el algoritmo es localmente correcto, por lo tanto por el teorema fundamental de la simulación perfecta tenemos que $Algo2(\alpha)$ es un algoritmo de simulación perfecta. \square

3.4. Inciso c)

Siendo breve con esta sección, se utilizará un ejemplo dado en clases, donde se buscará simular una distribución normal (estándar por simplicidad) desde una distribución de Laplace. Si g es la distribución de Laplace, entonces debemos hallar M tal que $h/(M \cdot g) < 1$, para esto consideramos distintos valores de x en el intervalo $[-1, 1]$ y escojemos M que maximice el ratio entre g y h , el cual será utilizado para obtener la desigualdad necesaria en estos métodos.

Simulando para $n = 10.000$ ejemplos, se cuenta la cantidad de simulaciones totales (contando rechazos) del algoritmo de aceptación rechazo, de la misma forma se cuenta la cantidad de simulaciones

del Algo3. También se almacena el tiempo total de computo de ambos algoritmos frente a la misma tarea y se obtienen los siguientes gráficos e indicadores:

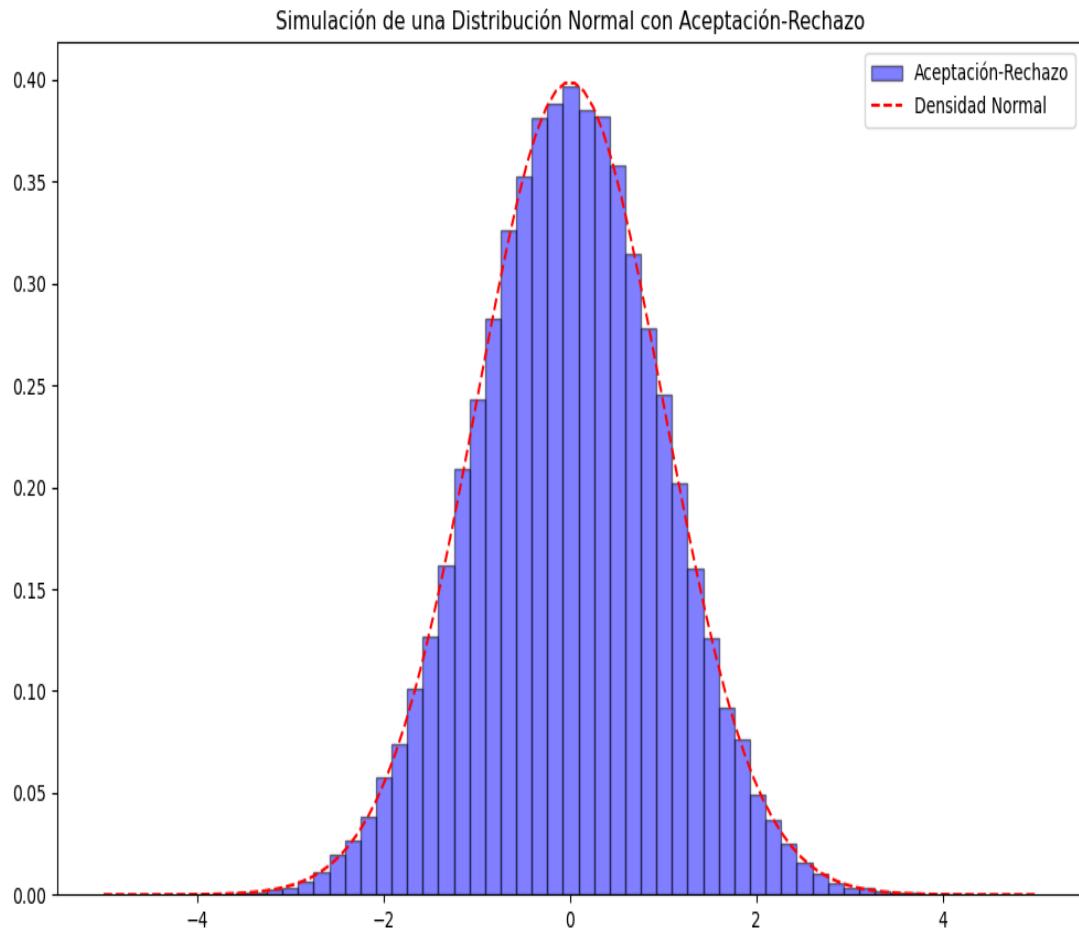


Figura 11: Histograma para aceptación rechazo.

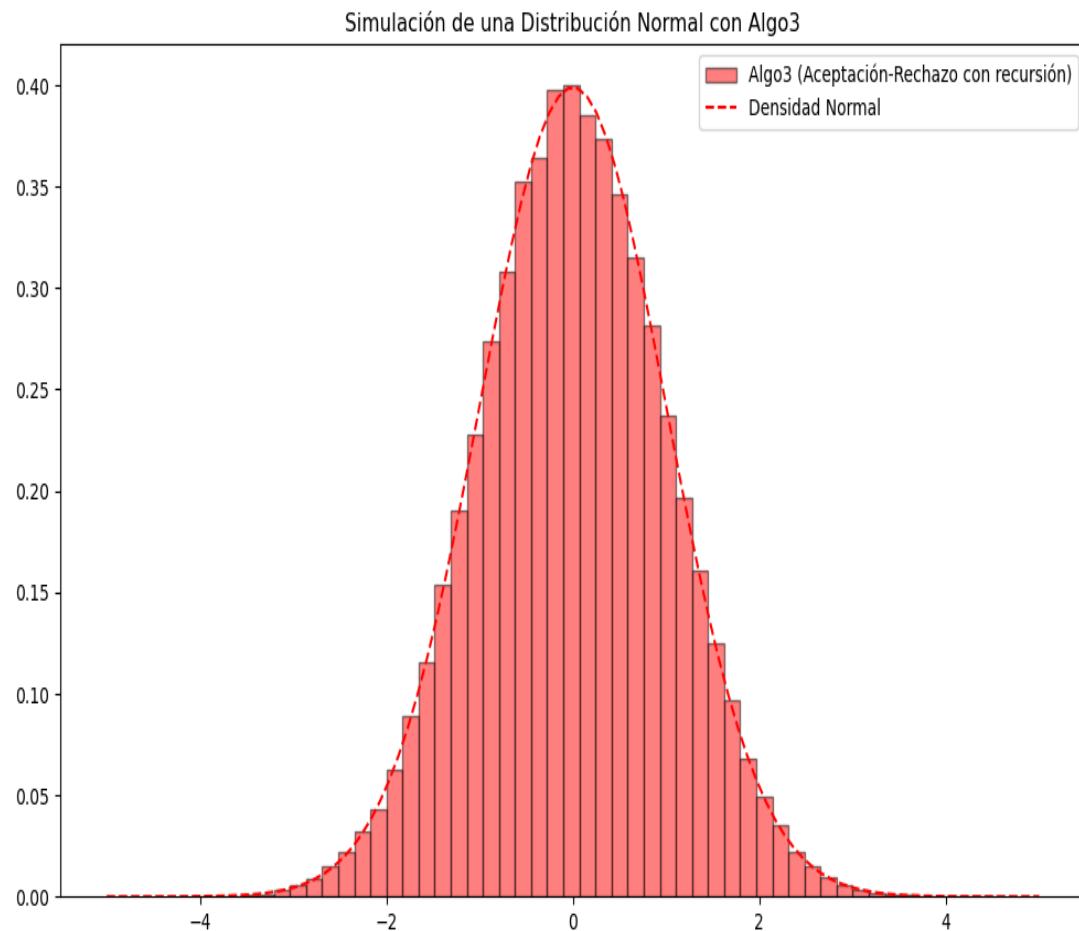


Figura 12: Histograma para algo3.

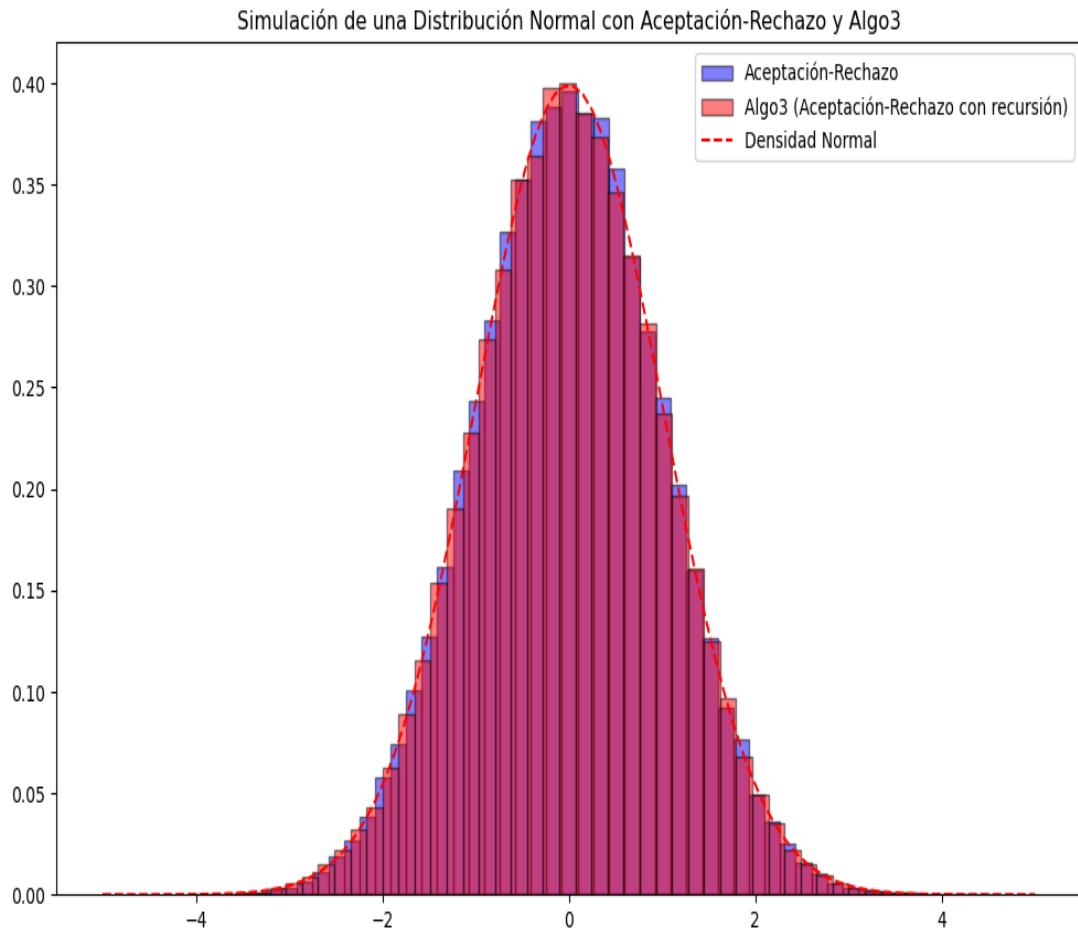


Figura 13: Histogramas en conjunto.

Cuadro 1: Comparación de Algoritmos

Algoritmo	Tiempo (s)	Iteraciones	Ejemplos Simulados	Ratio de Aceptación
Aceptación-Rechazo	15.7754	131569.0	100000	0.760057
Algo3	15.7515	131832.0	100000	0.758541

de donde para al menos este experimento no se puede concluir de manera clara si un algoritmo es mejor que otro en el sentido de cantidad de rechazos o tiempo de computo. Además se muestra que claramente los histogramas generados para ambos casos se ajustan a una distribución normal (con densidad en linea punteada roja).

Pero se menciona que para al menos este experimento, se ve que Algo3 demora menos tiempo respecto a aceptación-rechazo usual.

4. Pregunta 3.

En esta parte se encuentra todo lo pertinente a la pregunta 3 del certamen 1 de simulación estocástica, para más detalles sobre los códigos, puede dirigirse a la entrega del documento o bien al **repositorio Github**, luego a la carpeta **Entrega Certamen** y al documento **Pregunta3.ipynb**.

4.1. Planteamiento del problema.

El objetivo de esta pregunta será simular funciones en el espacio $L^2(\Omega)$ para Ω algún dominio, en específico, consideraremos $\Omega = [0, 1]$. Recuerde que toda función $f \in L^2(\Omega)$ se puede escribir mediante una secuencia de bases ortogonales $\{\phi_i\}_{i=0}^{\infty}$ y una secuencia de números reales $\{\kappa_i\}_{i=0}^{\infty}$ como

$$f(t) := f(t, \{\kappa_i\}_{i=0}^{\infty}) = \sum_{i=0}^{\infty} \kappa_i \phi_i(t)$$

Considere una secuencia de variables aleatorias independientes $\{S_k\}_{k=0}^{\infty}$ donde S_k proviene de una distribución $Beta(1 - \alpha, \theta + \alpha k)$ para parámetros α, θ adecuados. Se puede demostrar que la sucesión construida como $W_k = S_k \prod_{i=0}^{k-1} (1 - S_i)$ satisface que $W_k > 0$ para todo k y que además $\sum_{k=0}^{\infty} W_k = 1$.

4.2. Simular la secuencia W_k .

Note que en la definición de $Beta(1 - \alpha, \theta + \alpha k)$ existe la condición de que los parámetros de la distribución *Beta* deben ser no negativos para todo $k \in \mathbb{N}$, por lo que, a priori, $\alpha < 1$, notando que lo anterior se debe cumplir para todo k para un θ fijo, entonces es necesario pedir la restricción $\alpha > 0$. De forma equivalente, para $k = 0$, se deduce que $\theta > 0$. Por lo tanto se afirma que $\alpha \in (0, 1)$ y $\theta > 0$.

Por otro lado, para poder simular las funciones se requiere de:

- Tener un mecanismo para simular pesos (claramente finitos) $\{\kappa_i\}_{i=0}^{\infty}$.
- Una base ortogonal de funciones desde donde simular.
- Hallar un n adecuado que trunque la suma, es decir

$$f(t) \approx \sum_{i=0}^n \kappa_i \varphi_i(t)$$

En base a lo mostrado anteriormente, utilizaremos la secuencia W_k dado que la suma de sus elementos siempre es menor o igual 1, lo que permite que, para una base ortogonal uniformemente acotada, se generen funciones acotadas que sean tratables computacionalmente y permitan realizar transformaciones que se verán más adelante.

Por lo tanto una condición razonable es considerar n suficientemente grande tal que

$$\left| \sum_{k=0}^n W_k - 1 \right| < \text{tolerancia}$$

para cierta tolerancia para la mayoría de las simulaciones, donde claramente dicho n dependerá de los parámetros. Experimentando para distintos largos de cadena W_k , distintos parámetros α, θ y tolerancias razonables, se presentan distintas simulaciones de secuencias para $tol = 0,05$ y $n = 1.000$:

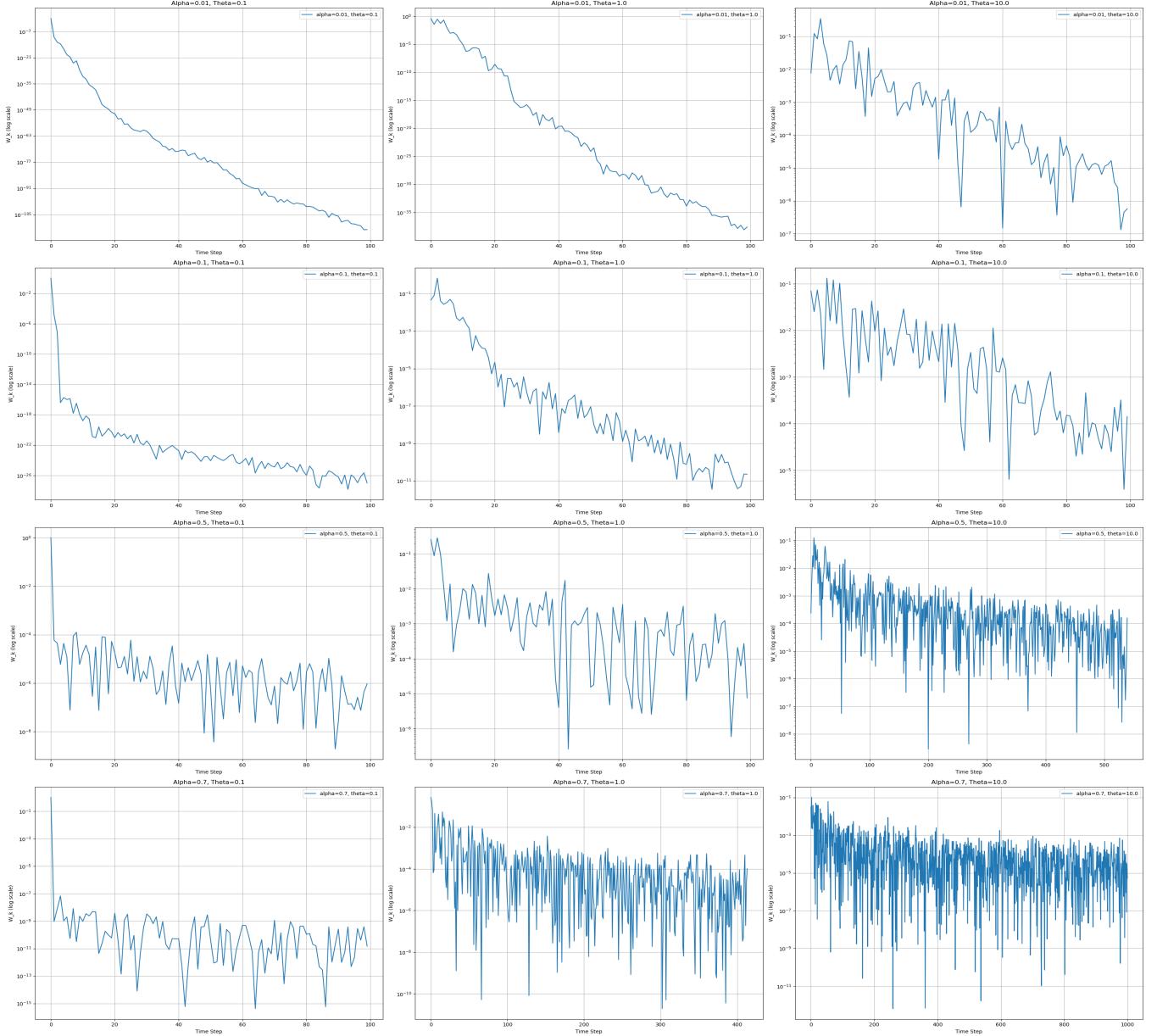


Figura 14: Secuencia W_k para distintos parámetros.

donde aquellas sucesiones con largo menor a n se deben a que alcanzan la tolerancia pedida antes de alcanzar el largo máximo, mientras que aquellas que no, no podemos asegurar dicha convergencia. Usando esto y mediante otros experimentos exhaustivos se decidió usar $\alpha \in (0.35, 0.65)$ y $\theta \in (0.5, 5)$

para necesitar de a lo más 1.000 elementos para tener una convergencia razonable de la sucesión de sumas parciales.

Se observa que: a medida que se consideran valores de θ mayores, la convergencia es más lenta pero se consideran valores más uniformes en su totalidad y por ende mayor importancia de cada función de la base en la simulación (este experimento se realizó de forma más precisa pero se omitió), mientras que para valores de α cada vez más pequeños, se tiene un decaimiento de la sucesión a valores prácticamente 0 y por ende más peso a las primeras funciones de la base.

Para cerrar este punto, se muestran simulaciones de posibles sucesiones W_k que se considerarán como pesos para los pares de α, θ seleccionados

$$\alpha \in \{0.35, 0.5, 0.65\}, \quad \theta = \{0.5, 1, 5\}$$

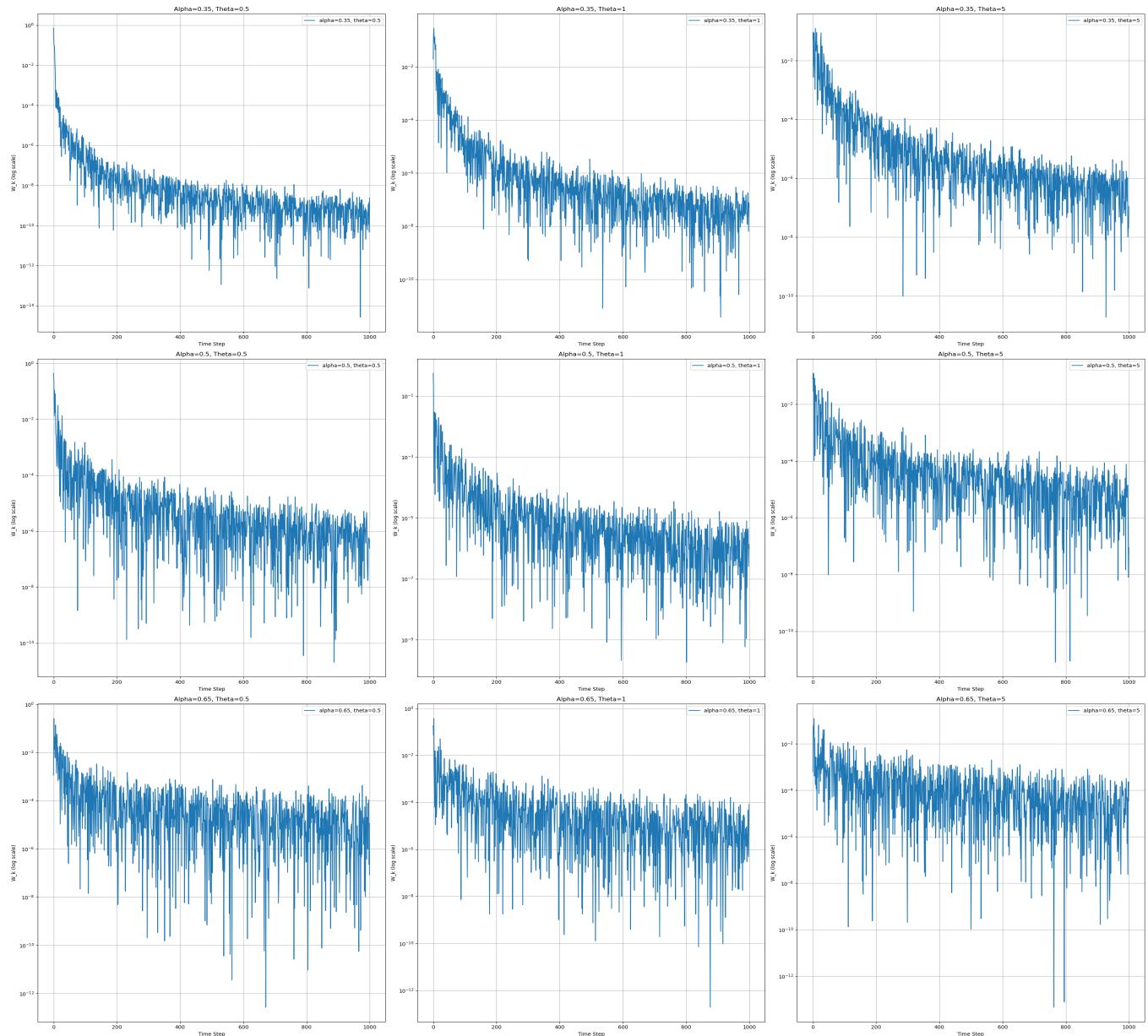


Figura 15: Secuencias W_k para parámetros escogidos.

Donde se recuerda que estos fueron escogidos para poder simular más adelante de forma eficiente y para distintos comportamientos.

4.3. Simulando funciones en $L^2([0, 1])$.

Se tiene que una base ortonormal de $L^2([0, L])$ está dada por

$$S = \left\{ \sqrt{\frac{2}{L}} \cos\left(\frac{k\pi}{L}x\right) \right\}_{k=0}^{\infty}$$

4.3.1. Simular funciones periódicas y pares en $L^2([-1, 1])$

Considera la base ortogonal de $L^2([-1, 1])$:

$$S' = \{\cos(k\pi x)\}_{k=0}^{\infty}$$

desde la paridad de la función coseno, tenemos que cualquier función generada con dicha base será par y de hecho, la base sigue siendo ortogonal en $[-1, 1]$ por la paridad, basta con separar toda la integral en $[-1, 1]$ como la suma de dos integrales entre $[0, 1]$ y $[-1, 0]$ y utilizar que dicha base es ortogonal en $L^2([0, 1])$.

Denote $\varphi_k = \cos(k\pi x)$, según análisis anteriores, si se escribe

$$f(x) = \sum_{k=0}^n w_k \varphi_k(x)$$

para $n = 100$, se debería tener una buena aproximación para los parámetros en los intervalos presentados antes.

Para entender la función a simular, se debe tener en cuenta que hay que caracterizarla por sus evaluaciones en ciertos puntos (generalmente equi-espaciados en $[-1, 1]$) x_0, \dots, x_N para N adecuado. De esta forma se puede escribir

$$f(x) \cong \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_N) \end{pmatrix} = \begin{pmatrix} \varphi_0(x_0) & \dots & \varphi_n(x_0) \\ \vdots & \ddots & \vdots \\ \varphi_0(x_N) & \dots & \varphi_n(x_N) \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix}$$

donde $W_n = (w_0, \dots, w_n)$. Por lo tanto se pueden generar funciones mediante únicamente conocer W_k (de longitud adecuada) y x los puntos a evaluar la función. Veamos unas cuantas funciones para los parámetros α y θ establecidos:

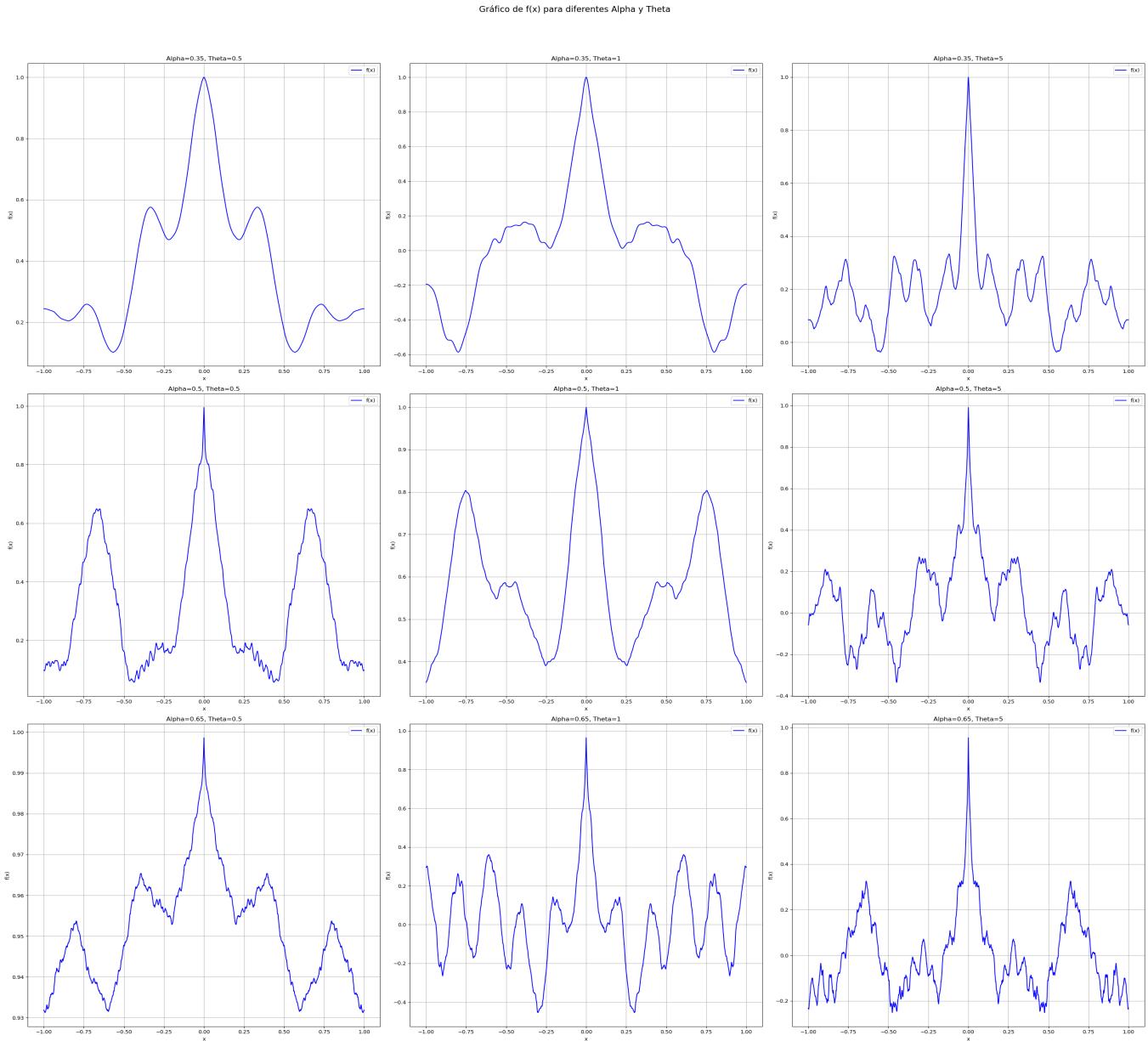


Figura 16: Funciones f para parámetros escogidos.

del análisis anterior para W_k , el resultado se puede interpretar de la siguiente manera: El parámetro α modela en cierta forma cuanto peso tendrán las funciones de la base para indices más altos (que tanto se van a 0), donde para $\alpha = 0.35$ tienen poca influencia. Por otro lado, sabemos que al aumentar el valor de θ , la variabilidad de la sucesión W_k está más controlada y por ende se pueden ver perturbaciones en la función más claras.

note que para $\theta = 0.5$ se tiene que la forma generada de la función viene dada por las primeras

funciones de la base, mientras que las otras 'señales' tienen menos importancia (alto decaimiento de W_k), luego al hacer crecer el valor de θ , se tiene que se le da más valor al resto de señales aumentando la variabilidad de la forma.

4.3.2. Simulando funciones estrictamente crecientes en $L^2([0, 1])$.

En esta sección se utiliza la misma base ortogonal y mecanismos para generar funciones $f(x)$, restringiéndolas al intervalo $[0, 1]$.

Suponga se tiene la función f simulada mediante el método anterior, como esta es continua, entonces $\exp(f)$ sigue siendo una función continua y por lo tanto la función

$$h(x) = \int_0^x \exp(f(t)) dt$$

es continua y está en $L^2([0, 1])$, además como su derivada es estrictamente positiva, entonces la función h es estrictamente creciente.

Para aligerar la notación, considere $g = \exp(f)$. Para simular h , se debe considerar lo siguiente: Para realizar este cálculo, si buscamos tener

$$h(x) \cong \begin{pmatrix} h(x_0) \\ \vdots \\ h(x_N) \end{pmatrix}$$

entonces podemos realizar un procedimiento basado en lo siguiente:

$$h(x_{i+1}) = \int_0^{x_{i+1}} g(x) dx = \int_{x_i}^{x_{i+1}} g(x) dx + \int_0^{x_i} g(x) dx = \int_{x_i}^{x_{i+1}} g(x) dx + h(x_i)$$

por lo tanto de manera recursiva podemos considerar:

$$h(x_i) = \sum_{j=0}^i \int_{x_{j-1}}^{x_j} g(x) dx$$

para $x_{-1} = 0$.

Notese que debemos realizar una aproximación de cada integral en los intervalos $[0, x_0], \dots, [x_{i-1}, x_i], \dots, [x_{N-1}, x_N]$ para esto se usará la cuadratura de Gauss.

En resumen, el esquema es el siguiente:

- Sea N la cantidad de puntos que se desean para la función y p la cantidad de nodos a utilizar en cada intervalo.
- Denotar $x^{(i)} = [x_0^{(i)}, \dots, x_p^{(i)}, \dots, x_{p-1}^{(i)}]$ los nodos de la cuadratura para el intervalo i -ésimo, para $i = 0, \dots, N$.

1. Escribir $x = [x^{(0)}, \dots, x^{(N)}]$.
2. Simular f utilizando W_k con $p * (N - 1)$ elementos y x .
3. Calcular $\exp(f)$.
4. Aproximar $h(x_0)$ mediante la cuadratura.
5. Aproximar la integral sobre $[x_i, x_{i+1}]$ usando la cuadratura.
6. Utilizar $h(x_i)$ ya calculado para calcular $h(x_{i+1})$ mediante la formula presentada antes.
7. Repetir 5. hasta terminar.

El esquema anterior fue optimizado en el algoritmo.

Se iteró sobre los intervalos y se hicieron los cálculos dentro de cada iteración aprovechando los cálculos anteriores y actualizando los valores a conveniencia hasta generar el vector que representa a h .

En este caso tenemos un nuevo parámetro p que indica la cantidad de nodos que se utilizan para calcular la integral sobre cada intervalo, utilizando $p = 50$ se obtienen las siguientes simulaciones:

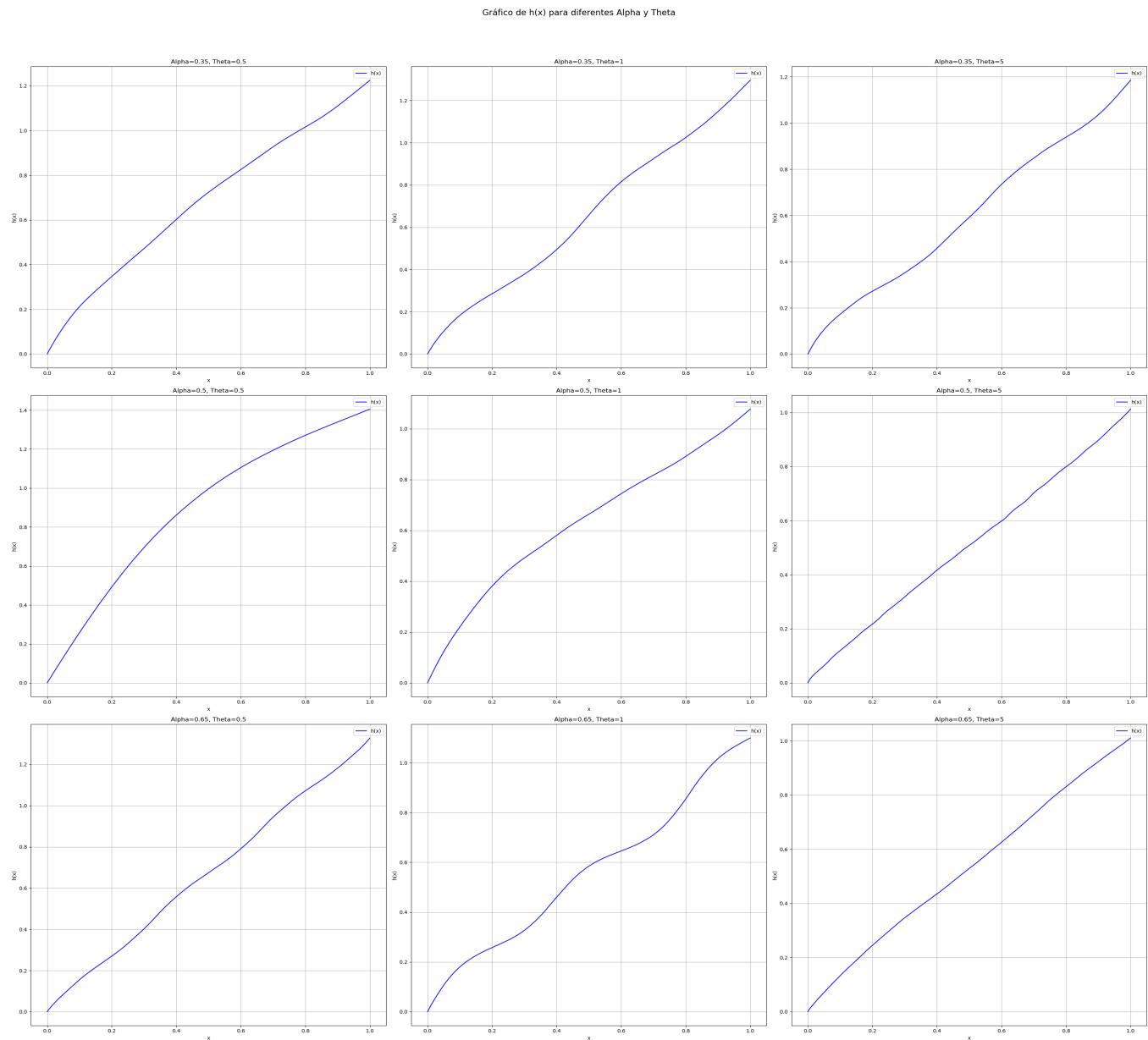


Figura 17: Funciones h para parámetros escogidos.

por inspección tiene sentido conjeturar que las funciones mostradas integran 1. Se repiten observaciones anteriores sobre α y θ , se remarca que las funciones para α alto parecen tener un comportamiento ruidoso dado que las señales (funciones de la base) de indices más altos tienen poco peso (pero significativo) en la función total, además, se vuelve a afirmar el caso extremo de α y θ bajos, donde solamente parece importar la suma de las primeras funciones mientras que el resto son basicamente 0.

4.3.3. Simular funciones positivas que integren 1 en $L^2([-1, 1])$

Dadas funciones $f(x)$, por argumentos anteriores sabemos que $\exp(f(x))$ es positiva, luego basta usar

$$g(x) = \frac{\exp(f(x))}{\int_0^1 \exp(f(x)) dx}$$

para que g sea una función positiva que integre 1. Para generar más variedad a los experimentos, Se podría la transformación $x \rightarrow x^2$ para simular, es decir:

$$g(x) = \frac{f(x)^2}{\int_0^1 f(x)^2 dx}$$

El esquema en este caso es mucho más sencillo:

1. Simular f mediante los algoritmos anteriores.
2. Elevarla al cuadrado.
3. Aproximar su integral con cuadratura de Gauss.
4. Dividir por el valor aproximado de la integral.

Se deja como comentario que todas las integrales pudieron ser calculadas con otros métodos vistos en el curso en caso de estimarse necesario.

En específico, aprovechando que las funciones $f(x)$ son acotadas inferiormente por -1, entonces $g(x) = f(x) + 1$ es positiva, luego basta con estimar su integral y dividirla por ese valor para obtener las siguientes simulaciones utilizando $p = 10.000$ nodos en total:

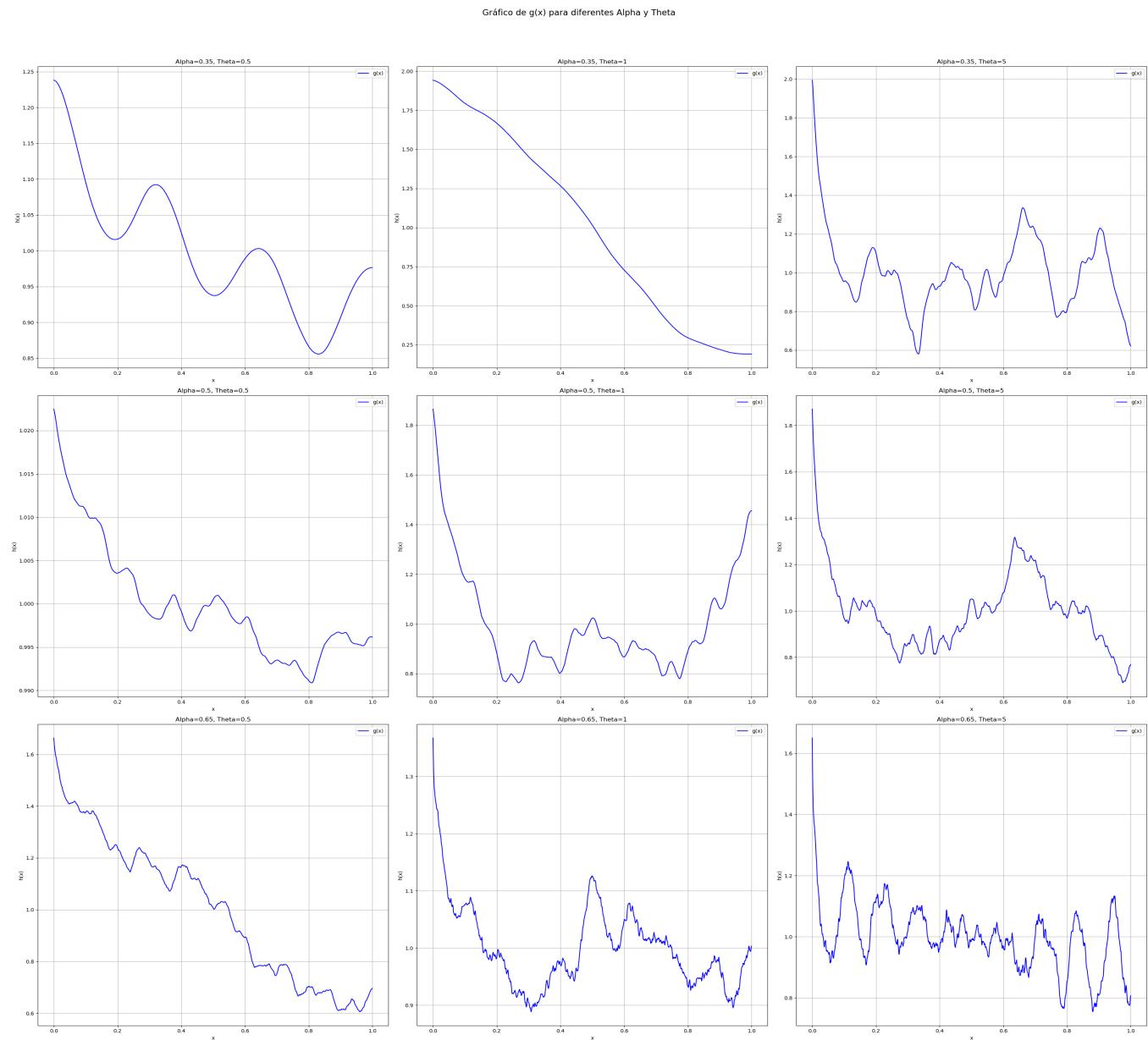


Figura 18: Funciones g para parámetros escogidos.

4.3.4. Comentarios finales.

Se debe notar que la simulación de los últimos dos puntos utilizamos una transformación para generarlas, lo que implica que el espacio en el cual viven las simulaciones sea sesgado a la transformación que se utiliza.

5. Pregunta 4.

En esta parte se encuentra todo lo pertinente a la pregunta 4 del certamen 1 de simulación estocástica, para más detalles sobre los códigos, puede dirigirse a la entrega del documento o bien al **repositorio Github**, luego a la carpeta **Entrega Certamen** y al documento **Pregunta4.ipynb**.

5.1. Planteamiento del problema.

El objetivo del trabajo será estudiar, vía simulaciones, el comportamiento de un interruptor de potencia de alto voltaje. Este interruptor está compuesto por 5 componentes que trabajan en paralelo y que reciben daño cuando hay un cortocircuito, donde luego de recibir cierta cantidad de daño las componentes quedan inoperativas, además, se dice que el interruptor queda inoperativo, cuando sus 5 componentes lo están.

Dado esto, buscaremos entender como afecta la distribución de los corto circuito a la duración de la vida útil del interruptor (antes de quedar inoperativo), para esto considere los siguientes supuestos entregados:

1. El daño es aleatorio y se divide entre el número de componentes que recibe daño.
2. Las componentes menos dañadas son más propensas a recibir daño.
3. En el momento que ocurre una falla, el número de componentes que recibe el daño es aleatorio.
4. El daño recibido es aleatorio y se modela por la variable aleatoria $Y = 10X$ donde $X \sim \text{Beta}(\alpha, \beta)$.
5. Debido a que el cortocircuito ocurre en condiciones aleatorias, se considera que $\alpha \sim U[0.75, 1.5]$ y $\beta \sim U[2.75, 3.25]$.

Modelamiento del problema.

Se deja claro que se considerará que cada componente tiene 100 puntos de vida.

Por otro lado, desde los supuestos no queda claro como determinar el número de componentes reciben daño, sólo se indica que la cantidad de estas lo es y que el daño Y calculado se reparte entre ellas. Utilizando el hecho de que las componentes menos dañadas son más propensas a recibir daño, se propone el siguiente esquema:

1. Ocurre un cortocircuito.
2. Para cada componente C_i para $i = 1, \dots, 5$, se define

$$p_i = \frac{\text{puntos de vida de } C_i}{\text{puntos de vida del sistema.}}$$

3. Para cada componente C_i , se simula desde $Ber(p_i)$, si el resultado es 1 la componente será dañada, en otro caso no recibe daño. Si en este procedimiento ninguna componente resulta dañada, se establece que las 5 recibirán daño.
4. Se obtienen α y β desde las distribuciones uniformes correspondientes y se simula X desde $Beta(\alpha, \beta)$ para calcular $Y = 10X$.
5. Si k es el número de componentes dañadas, se le restan Y/k puntos de vida a cada componente dañada. El daño en exceso producto de la muerte de una componente se pierde.

sobre este esquema se realiza el siguiente punteo de comentarios:

- Inicialmente $p_i = 0,2$ para $i = 1, \dots, 5$ y se tiene que la probabilidad de que la componente C_i se dañe es proporcional a la cantidad de vida que tenga en relación al sistema, es decir, sigue el **supuesto 2**.
- Este modelamiento tiene el problema de que inicialmente la probabilidad de que todas las componentes reciban la misma cantidad de daño es 30 % lo que genera una nuevas probabilidades iguales.
- Pese a lo anterior, esto no sugiere un mayor problema, dado que en un desarrollo paralelo se intentó reducir esta probabilidad mediante modelar una transformación adecuada $p_i^{(1)} = g(p_i)$ para incrementar los valores promedio de las probabilidades, pero los resultados no se vieron afectados significativamente (Ver [códigos de prueba](#) en el repositorio Github).
- Dado este esquema, se cumple el **supuesto 1**.

En forma de modelo gráfico se puede entender el modelo de la siguiente manera:

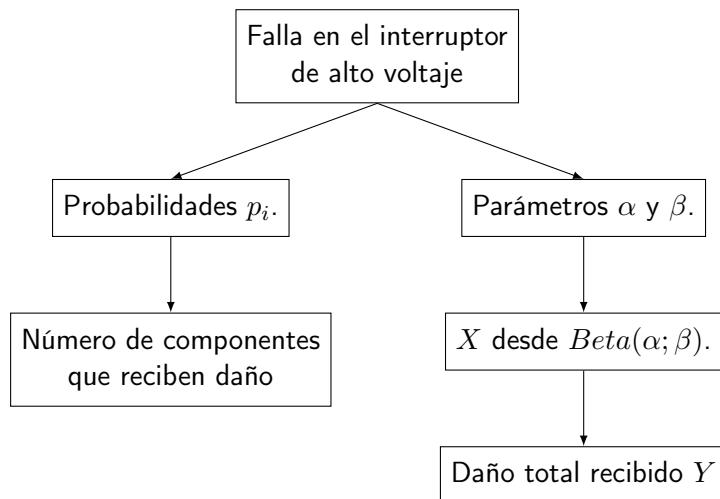


Figura 19: Modelo gráfico de la falla en el interruptor de alto voltaje.

5.2. Supuestos del modelamiento.

En esta sección se suma lo comentado anteriormente sobre como determinar el número de componentes dañadas y como interpretar el **supuesto 2**, lo que ya fue mencionado.

Lo que concierne a este punto, es el hecho de que se requiere modelar la distribución de los cortocircuito, desde ahora llamados tiempos de falla. Para proponer modelos sobre los tiempos de falla, se utilizará el hecho de que si consideramos λ : el número de fallas por unidad de tiempo, una constante, podemos modelar esta variable aleatoria mediante un proceso Poisson de parámetro λ , esto es $P(\lambda)$, de donde se tiene entonces que los tiempos entre fallas se distribuyen $\exp(-\lambda)$ siguiendo la función de densidad:

$$f(x; \lambda) = \lambda \exp(-\lambda x)$$

con media $1/\lambda$ y varianza $1/\lambda^2$. Por lo tanto obtenemos una característica importante, para λ "grandes" tenemos alta frecuencia de fallos mientras que para λ "chicos" tenemos una baja frecuencia de fallos.

Notese que se podrían suponer modelos más generales para la distribución de los tiempos de falla como la distribución de Weibull que permiten captar comportamientos un poco más complejos pero que se escapan del objetivo principal de esta pregunta, en la cual basta con considerar el modelo exponencial.

5.3. Entendiendo una simulación.

Se recalca se tiene que "una simulación" corresponde al proceso completo de crear un interruptor de potencia con sus 5 componentes y generar los cortocircuito siguiendo la distribución propuesta y dañando las componentes según el esquema presentado, donde la simulación concluye cuando las 5 componentes quedan inoperativas. Dentro del código se encuentran mayores detalles de como se realizó esto.

Para empezar, se debe comprender como es la distribución que modela los daños, la cual es una $Beta(\alpha, \beta)$ para $\alpha \sim U[0.75, 1.5]$ y $\beta \sim U[2.75, 3.25]$, donde sus posibles formas pueden ser descritas en el siguiente gráfico:

PDF de la Distribución Beta para Diferentes Valores de Alpha y Beta

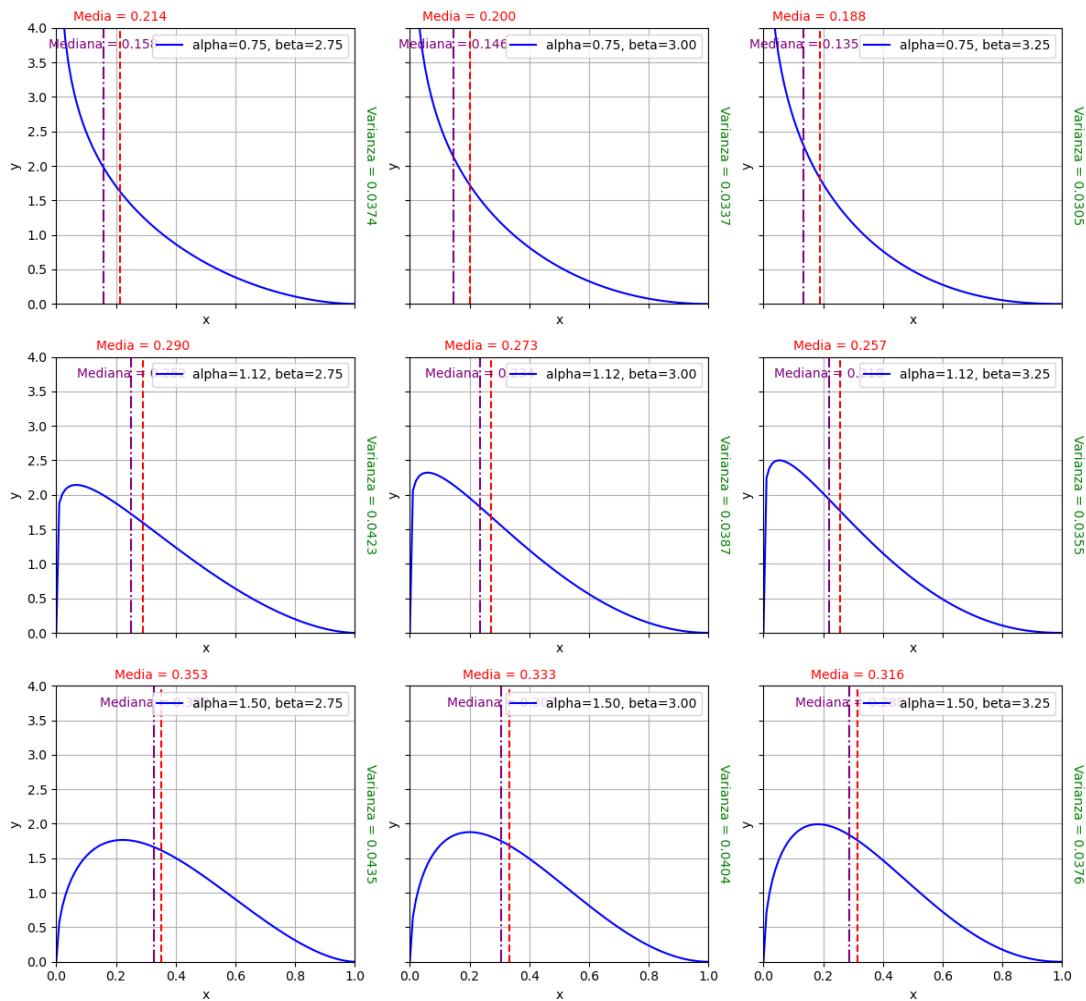


Figura 20: PDF de la distribución Beta(alpha, beta).

de lo mostrado lo más relevante de momento es que la distribución cambia significativamente su forma según el parámetro α y que la media aumenta a medida que este aumenta y disminuye cuando β lo hace, al menos desde los gráficos, dado que si consideramos que su función en términos de los parámetros de la media es:

$$E(X) = \frac{\alpha}{\alpha + \beta}$$

la relación con α es un poco más compleja. Con fines de la pregunta, podemos comentar que el daño promedio que se recibe con cada corto circuito está en una vecindad de [1.88, 3.53] puntos de daño.

A modo de ejemplo, se realiza una simulación para $\lambda = 0.1$ la cual termina luego de 208 cortocircuitos

cuya información se puede resumir en el siguiente gráfico:

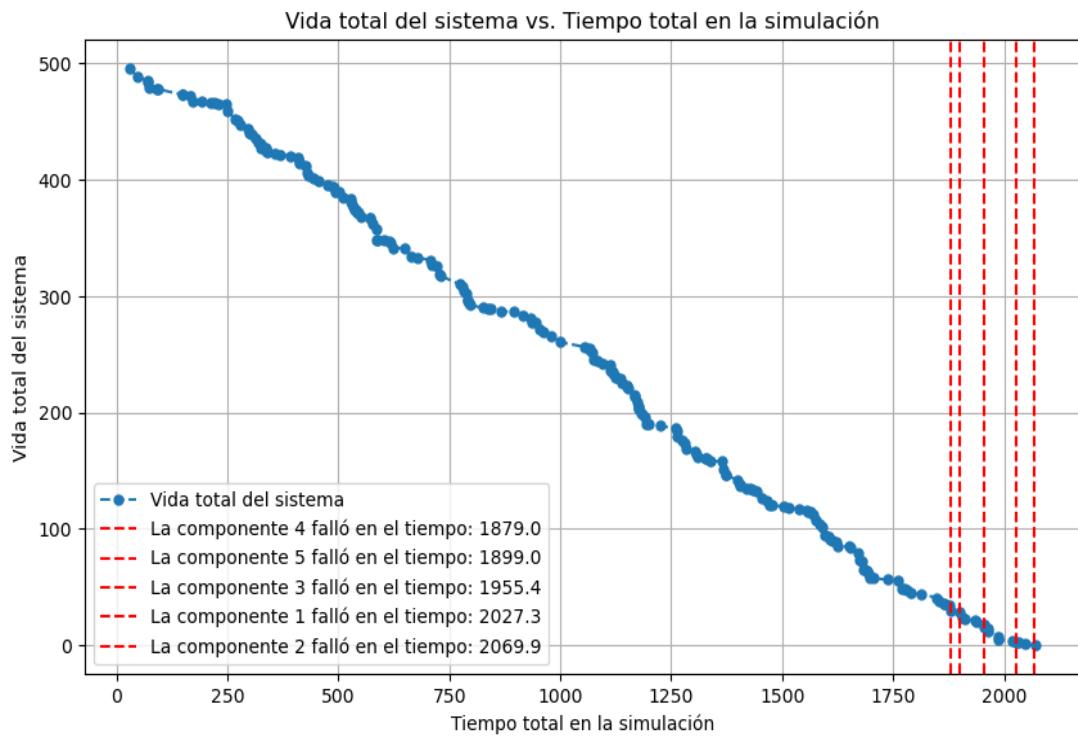


Figura 21: Vida total del sistema vs. tiempo total en la simulación.

donde cada punto corresponde a un cortocircuito y la vida mostrada es aquella luego de ser descontado el daño. Se observa que todas las componentes fallan en tiempos relativamente parecidos, lo cual es de esperarse dado que es una consecuencia del supuesto de que las componentes menos dañadas son más propensas a recibir daño.

Además se puede notar que, en términos generales, se puede ajustar una tendencia lineal, cuya pendiente puede corresponder al daño promedio que se realiza:

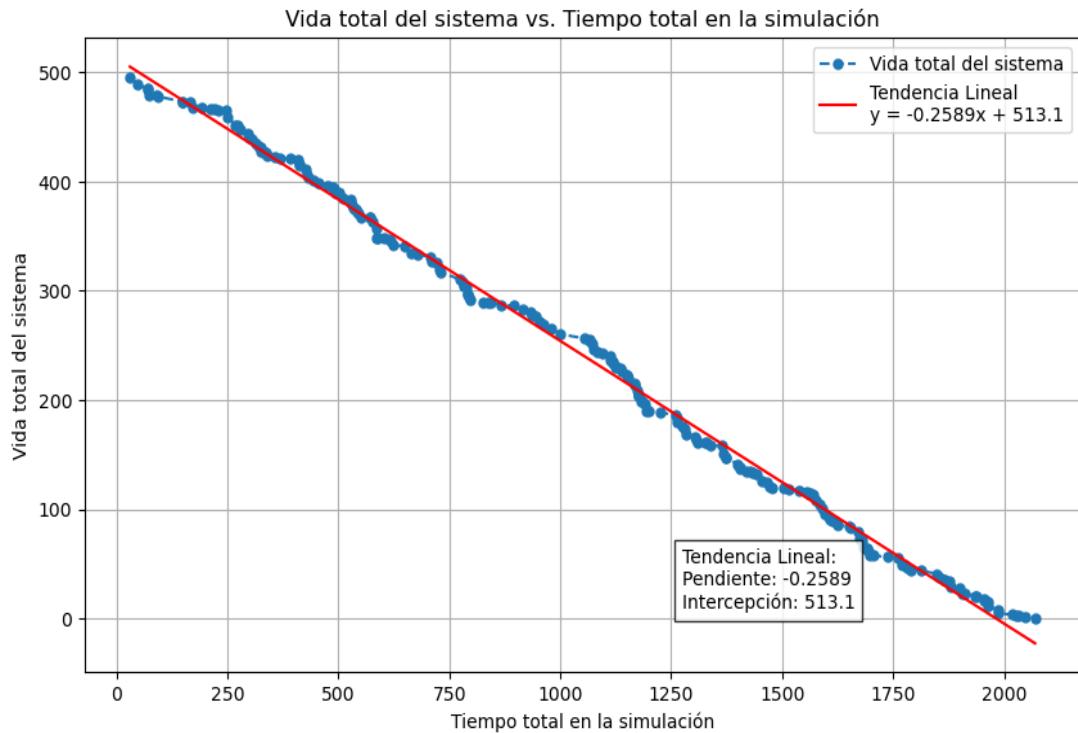


Figura 22: Vida total del sistema vs. tiempo total en la simulación con tendencia lineal.

donde en efecto, la pendiente pertenece al intervalo conjeturado anteriormente. Para reforzar esta idea, podemos graficar la distribución de los daños simulados:

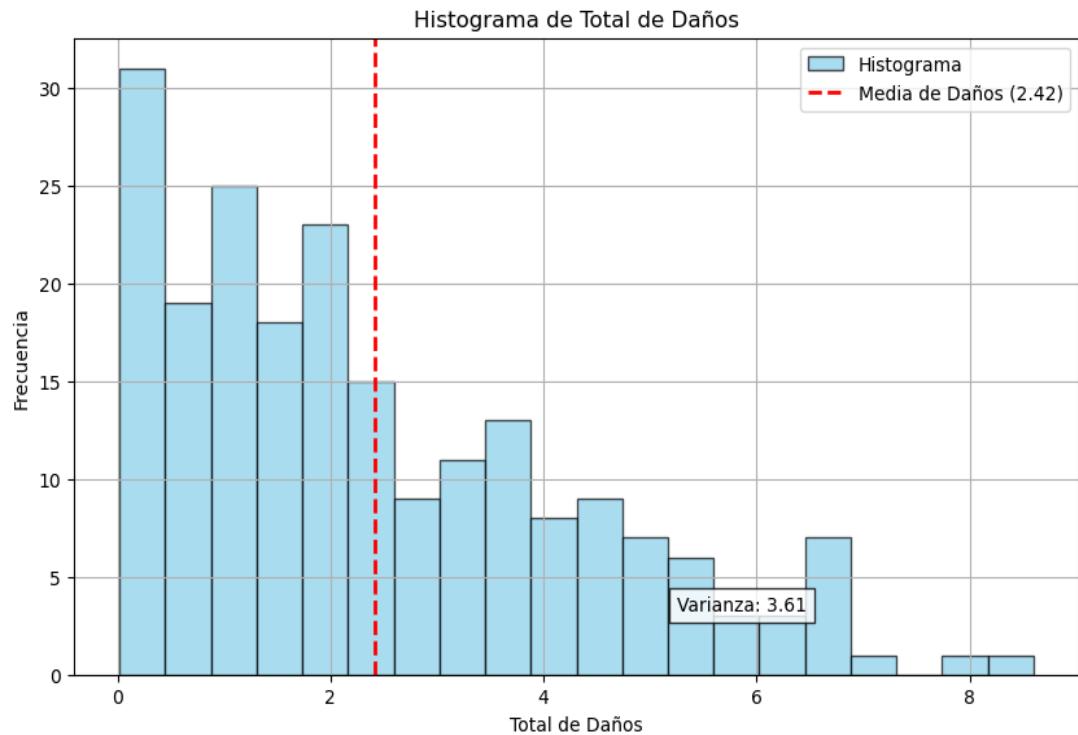


Figura 23: Histograma de daños.

y notar que de cierta forma recuerda a las distribuciones *Beta* mostradas en un inicio y que cuya media 2.42 es parecida a la pendiente de la recta anterior y efectivamente es un valor que se esperaba desde las conjeturas anteriores.

Por completitud, se mostrará el histograma de los tiempos de falla junto con su densidad teórica para verificar la correcta simulación:

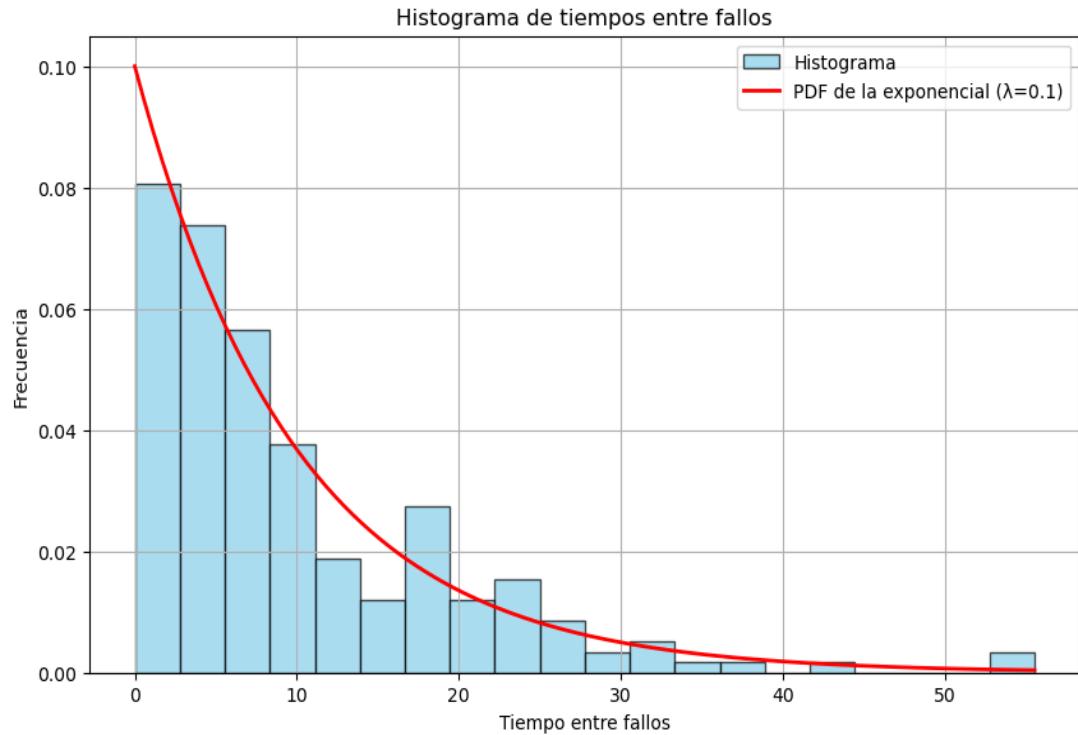


Figura 24: Histograma de tiempo entre fallos con comparación.

5.4. Múltiples simulaciones.

El objetivo de esta sección es presentar propiedades en promedio que tiene este modelamiento así como la dependencia ante el parámetro λ .

Se propone principalmente que existe una relación inversamente proporcional entre el tiempo esperado de falla del sistema $E(T(\lambda))$ (5 componentes inoperativas) y el número de fallas en promedio por unidad de tiempo λ (cortocircuitos), esto es:

$$E(T(\lambda)) = \frac{a}{\lambda^b}$$

para a, b algunas constantes, aplicando logaritmo en ambos miembros, tenemos el modelo lineal

$$\log(E(T(\lambda))) = \log(a) - b \log(\lambda)$$

veamos, mediante simulación, que $b = 1$ y $a = E[X]$ donde X es el número de fallos totales hasta que la simulación termina.

Para esto considere distintos λ 's dados por 9 valores en una partición en escala logarítmica del espacio paramétrico de muestra $[0.01, 10]$. Luego para cada λ , se realizan $n = 1.000$ simulaciones con el fin de obtener un promedio del tiempo de falla del sistema (`Mean_Time`), la varianza de los tiempos

de falla (**Variance_Time**) y promedio del total de fallos que ocurren antes de que el sistema quede inoperativo (**Mean_Total_Failures**), los resultados pueden ser vistos en la siguiente tabla:

Cuadro 2: Estadísticos con $n = 1000$ iteraciones.

Lambda	Mean_Time	Variance_Time	Mean_Total_Failures
0.010	18741.297	3007450.079	187.312
0.024	7885.696	504940.266	186.856
0.056	3337.728	93598.790	187.167
0.133	1395.111	17177.239	186.615
0.316	590.154	2951.456	187.107
0.750	250.392	561.766	187.131
1.778	105.105	87.401	186.539
4.217	44.282	15.882	187.028
10.000	18.683	2.823	187.196

Notese que el valor de fallos en promedio es constante respecto a λ , más aún, se verifica experimentalmente que

$$E(T(\lambda)) \cdot \lambda = a \approx 187$$

claro, suponiendo correcta esta relación funcional supuesta al inicio de la sección que, de hecho, se puede corroborar mediante esto. Se recalca entonces que, suponiendo verdadera esta relación, mediante estimar el parámetro λ en un experimento real, según este resultado podríamos estimar el tiempo esperado de falla mediante calcular $187/\hat{\lambda}$. Se puede observar que para valores bajos de λ , la varianza crece de manera rápida, mediante un cálculo sencillo, se puede obtener la desviación estándar y notar que esta corresponde a un poco menos 10 % del valor del promedio, siendo estable en este sentido:

Cuadro 3: Estadísticos con $n = 1000$ iteraciones extendido.

Lambda	Mean_Time	Mean_Total_Failures	Variance_Time	Desv_Time	Desv/Mean
0.010	18741.297	187.312	3007450.079	1734.200	0.093
0.024	7885.696	186.856	504940.266	710.591	0.090
0.056	3337.728	187.167	93598.790	305.939	0.092
0.133	1395.111	186.615	17177.239	131.062	0.094
0.316	590.154	187.107	2951.456	54.327	0.092
0.750	250.392	187.131	561.766	23.702	0.095
1.778	105.105	186.539	87.401	9.349	0.089
4.217	44.282	187.028	15.882	3.985	0.090
10.000	18.683	187.196	2.823	1.680	0.090

En virtud de la longitud de la pregunta se omite un análisis sobre la varianza que se encuentra en el código, dado esto, se afirma que la varianza crece en forma proporcional al inverso del cuadrado del λ (como es de esperarse de la exponencial).

Sobre los tiempos de falla, notese que salvo la componente aleatoria añadida por la distribución *Beta*, se tiene que los tiempo de falla, al menos en promedio, son en si mismos un promedio de $a = 187$ exponenciales independientes y por ende, es de esperarse que los tiempos de falla se comporten como una distribución normal al menos en forma asintótica por el teorema del límite central, mostrando de esta forma que en términos generales, la cantidad aleatoria del daño no tiene mayores implicancias en este sentido:

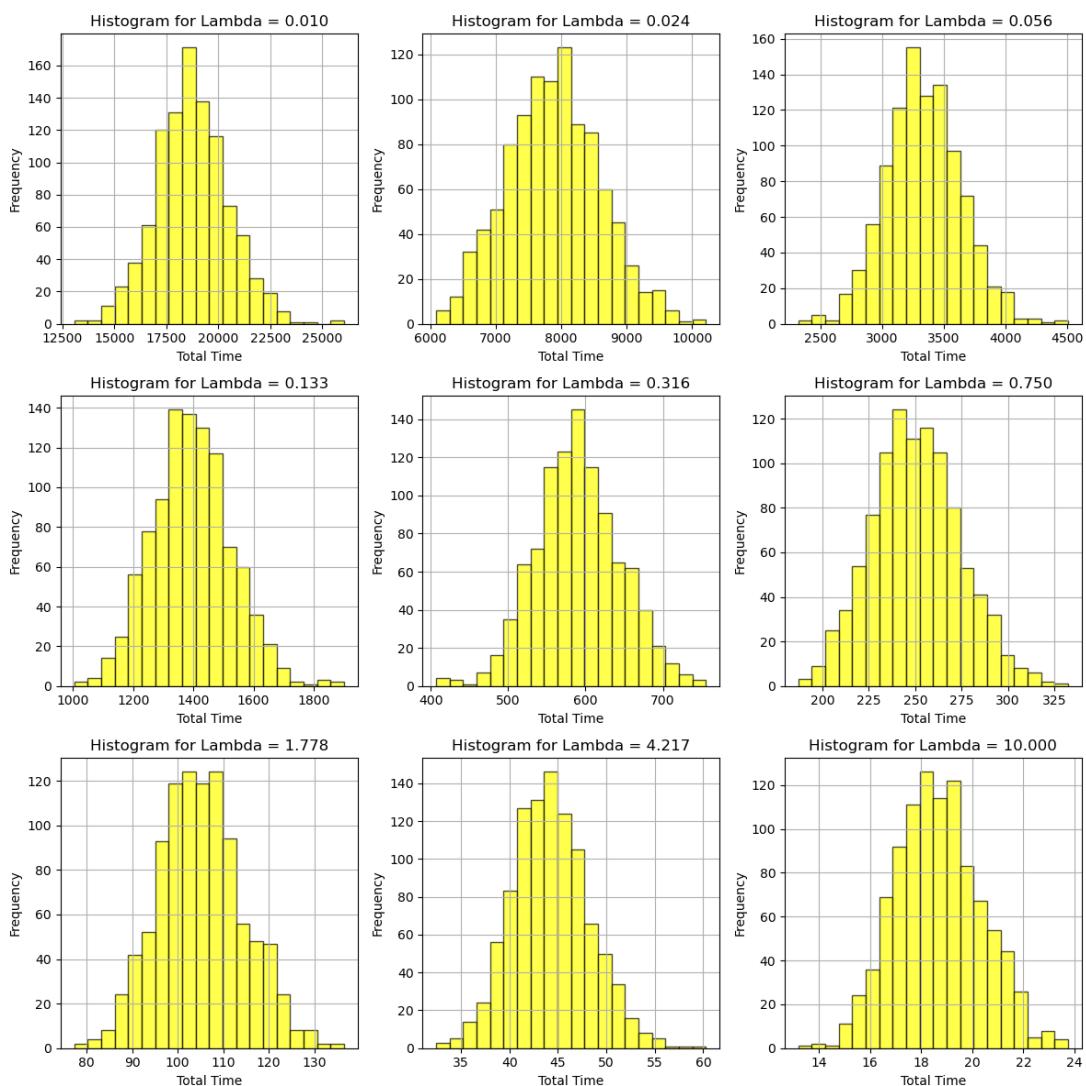


Figura 25: Histograma de tiempo de fallos para distintos λ .

Finalmente para terminar esta sección, utilicemos la escala logarítmica para hacer un ajuste lineal de los tiempos de falla promedio vs. los distintos valores de λ y corrobaremos que existe una relación funcional de la forma

$$\log(E(T(\lambda))) = \log(a) - b\lambda$$

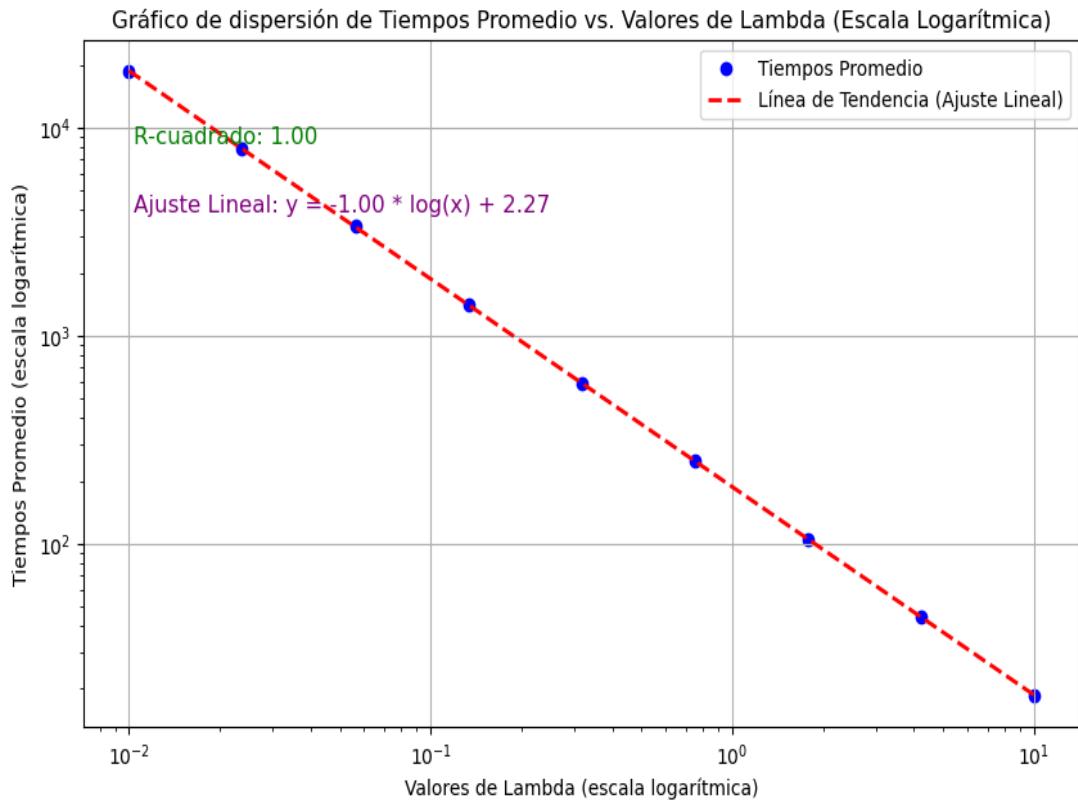


Figura 26: Ajuste lineal en escala logarítmica.

de donde volvemos a corroborar que $b = 1$ y $\log(a) = 2.27$ y por ende $a = 10^{2.27} \approx 186$.

5.5. Múltiples equipos.

Suponga se tienen E_1, \dots, E_n equipos distintos que funcionan en paralelo, diremos que dicho sistema queda inoperativo cuando cada equipo queda inoperativo, el objetivo de esta sección será estudiar como se comporta este nuevo tiempo de falla con respecto al de un equipo por si sólo.

Para esto defina T_i para $i = 1, \dots, n$ la variable aleatoria el tiempo de fallo del equipo E_i (cuando las 5 componentes de dicho equipo quedan inoperativas), luego la variable de interés para esta sección es

$$T = \max\{T_1, \dots, T_n\}$$

donde asumiremos que T_1, \dots, T_n son mutuamente independientes (funcionan en paralelo). En términos generales el comportamiento es similar a lo visto con un único equipo, salvo que, en promedio, este sistema en su conjunto tiende a fallar mucho después que el sistema con un único equipo para todo valor de λ , además, la varianza de los tiempos de falla es menor para múltiples equipos, esto se puede ver en los siguientes gráficos:

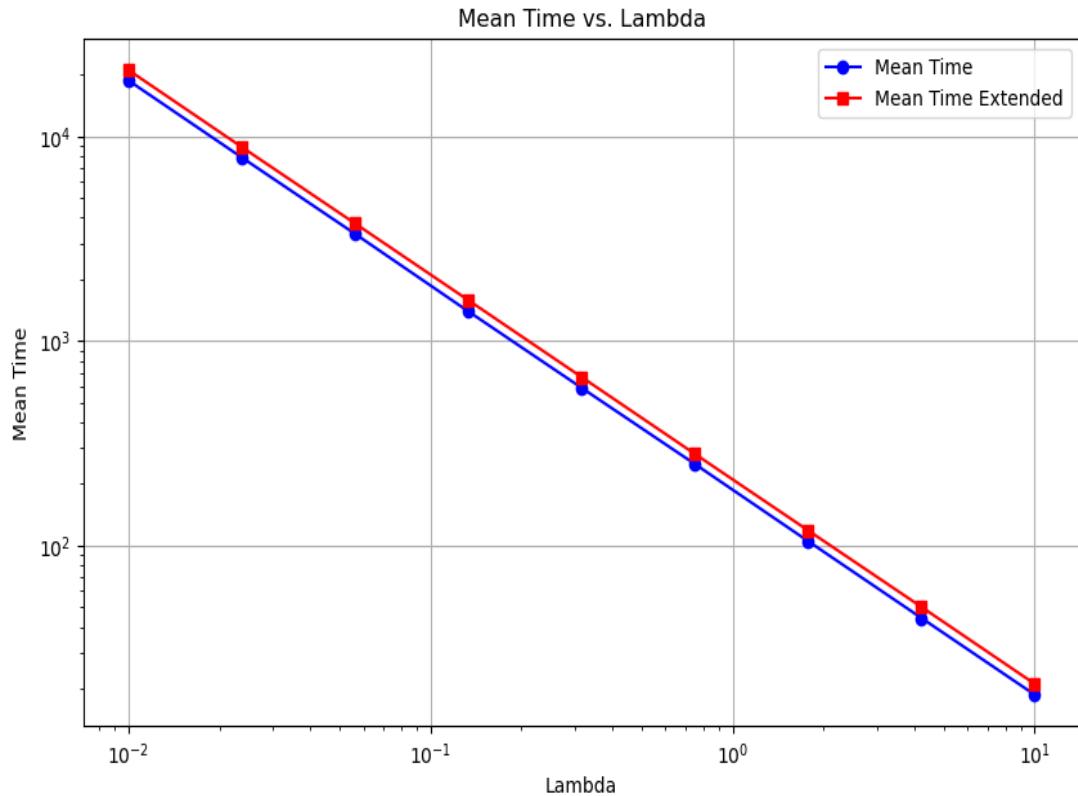


Figura 27: Comparación de media en función de λ .

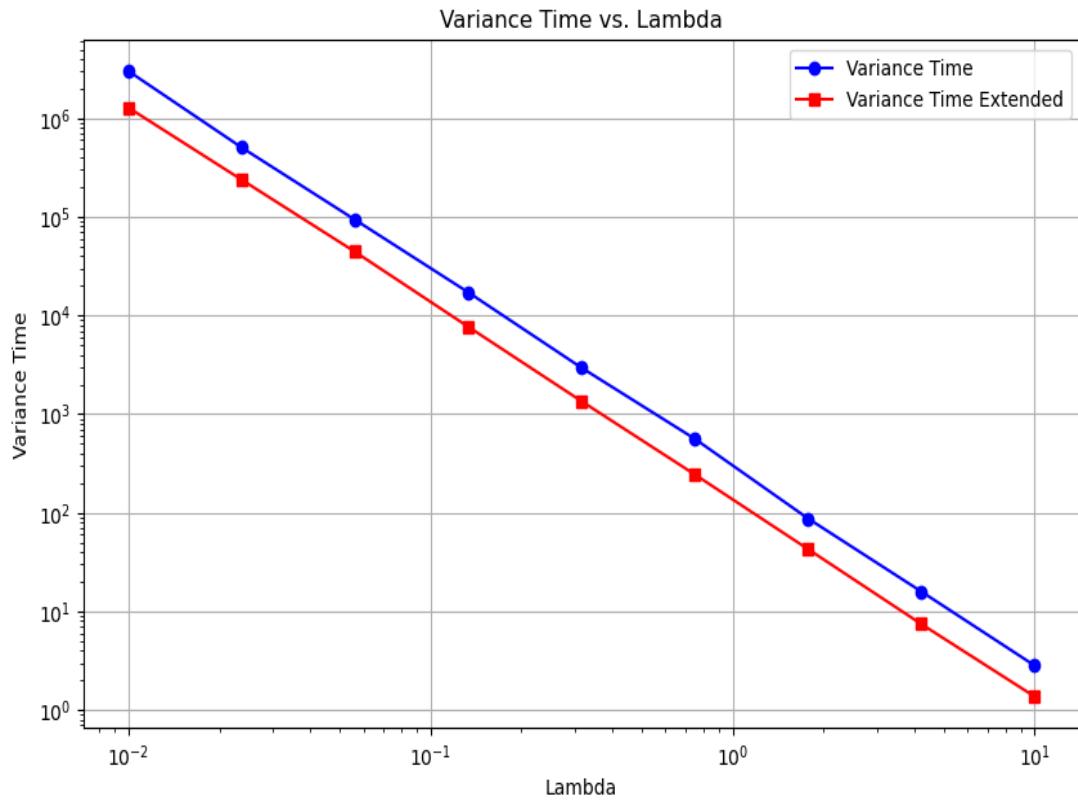


Figura 28: Comparación de varianza en función de λ .

5.6. Conclusión.

Utilizando una taza de fallos constante modelada por un proceso Poisson se puede simular y modelar el tiempo de fallo de un equipo compuesto por diferentes piezas que funcionan en paralelo. Donde bajo ciertos supuestos (que pueden extender de este trabajo) se puede buscar estimar el valor de la taza de fallo para predecir el tiempo de falla total del equipo, aún cuando el daño que reciba este sea aleatorio y sea independiente de la taza de falla.

Además, se comenta que se pueden utilizar múltiples equipos conectados en paralelo para obtener una varianza del tiempo de falla del sistema más controlada y su vez una esperanza de vida del sistema mayor.