Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Geofísica

Tarea 3

Ajuste Ortogonal de una Linea Recta usando Métodos de Muestreo Bayesianos: Algoritmos de Transitional Markov Chain Monte Carlo (TMCMC) + Remuestreo

Integrantes: Catalina Zapata
Constanza Santori
Karina Hernández
Teresa Peralta
Profesor: Francisco Ortega Culaciati
Fecha de entrega: 4 de agosto de 2025
Santiago de Chile

1. Programación del Algoritmo TMCMC

El módulo `tmcmc.py` implementa el algoritmo Transitional Markov Chain Monte Carlo (TMCMC) (Ching & Chen, 2007), que genera muestras de la distribución posterior utilizando una secuencia de distribuciones intermedias con un parámetro $\beta \in [0, 1]$. Esta transición progresiva evita que la cadena se concentre en regiones de alta verosimilitud, permitiendo una exploración más uniforme del espacio de modelos.

1.1. Módulo `calc_dbeta.py`

El algoritmo Transitional Markov Chain Monte Carlo (TMCMC) introduce una secuencia de distribuciones intermedias entre la distribución a priori $p(\mathbf{m})$ y la distribución posterior $p(\mathbf{m}|\mathbf{d})$ mediante un parámetro $\beta_k \in [0, 1]$, que controla la influencia de la verosimilitud en cada etapa.

La distribución intermedia está definida como:

$$f_k(\mathbf{m}) = p(\mathbf{m}) \cdot \mathcal{L}(\mathbf{m})^{\beta_k}$$

donde $L(\mathbf{m})$ es la función de verosimilitud, y β_k crece desde 0 hasta 1 en pasos $\Delta\beta$.

1.1.1. Effective Sample Size (ESS)

El valor de $\Delta\beta$ se determina para asegurar que una fracción γ del conjunto de muestras actual sea representativa de la siguiente distribución intermedia. Para esto, se utiliza el concepto de **Effective Sample Size (ESS)**, definido como:

$$ESS_{\Delta\beta} = \frac{1}{\sum_{i=1}^{N_S} w_i^2}$$

donde los pesos normalizados (w_i) se calculan como:

- **Caso 1:** Si se trabaja con los valores de verosimilitud directamente:

$$\texttt{use\_log\_likelihood = False} \quad \Longrightarrow \quad w_i = \frac{\mathcal{L}(\mathbf{m}_i)^{\Delta\beta}}{\sum_{j=1}^{N_S} \mathcal{L}(\mathbf{m}_j)^{\Delta\beta}}$$

- **Caso 2:** Si se trabaja con el logaritmo natural de los valores de verosimilitud:

$$\texttt{use\_log\_likelihood = True} \quad \Longrightarrow \quad w_i = \frac{e^{\Delta\beta \log \mathcal{L}(\mathbf{m}_i)}}{\sum_{j=1}^{N_S} e^{\Delta\beta \log \mathcal{L}(\mathbf{m}_j)}}$$

En ambos casos, es importante mencionar que, para evitar problemas de *overflow*, se implementa una corrección numérica estándar restando el máximo valor del logaritmo natural de la verosimilitud.

Selección adaptativa de $\Delta\beta$

Se define la función objetivo:

$$\phi(\Delta\beta) = ESS_{\Delta\beta} - \gamma N_S$$

con la fracción deseada $\gamma = 0.5$ como criterio estándar (en este caso, por enunciado corresponde al 50 % de estas sean efectivas de la fdp definida para el valor actualizado de β),

- Si existe un cambio de signo en el intervalo $[\ln(\Delta\beta_{\min}), \ln(\Delta\beta_{\max})]$, se aplica el método de Brent sobre $\phi(\ln \Delta\beta)$. Este método es eficiente y robusto para encontrar raíces cuando se garantiza que la función cruza cero en el intervalo dado.
- En caso contrario, se minimiza $\phi^2(\ln \Delta\beta)$ mediante `minimize_scalar`, ya que Brent no puede aplicarse si ϕ no cambia de signo. Esta alternativa permite encontrar un valor de $\Delta\beta$ que aproxima la condición deseada aún cuando no existe una raíz exacta dentro del intervalo.

Verificación de factibilidad con Brent:

Para verificar si se puede utilizar Brent, se evalúa el producto:

$$\phi(\ln \Delta\beta_{\min}) \cdot \phi(\ln \Delta\beta_{\max}) < 0$$

Esto indica un cambio de signo y garantiza que existe una raíz en el intervalo. Este resultado se guarda como un `status` booleano que determina si se aplica Brent o la minimización acotada alternativa.

En la implementación desarrollada, la búsqueda del incremento óptimo $\Delta\beta$ se realiza sobre su logaritmo natural ($\ln(\Delta\beta)$) para mejorar la estabilidad numérica de las operaciones exponenciales. En el caso en que los valores de verosimilitud se utilicen en escala logarítmica (`use_log_likelihood = True`), se aplica una normalización adicional del tipo `log-sum-exp` para evitar desbordamientos numéricos. Asimismo, se evita la evaluación de logaritmos de cero mediante la inclusión de un umbral inferior. Finalmente, si el valor calculado de $\beta + \Delta\beta$ supera 1, se ajusta automáticamente $\Delta\beta$ para asegurar que β no exceda el valor máximo permitido, cerrando así la última etapa del algoritmo.

1.2. Módulo `tmcmc.py`

El algoritmo TMCMC permite aproximar distribuciones complejas mediante una transición desde la distribución a priori. Esto evita problemas de mala exploración del espacio de modelos al incrementar gradualmente la influencia de la verosimilitud. Dentro de `tmcmc.py` se encuentra la función `tmcmc_pool` que genera muestras de la distribución posterior mediante una secuencia de distribuciones intermedias parametrizadas por un exponente $\beta \in [0, 1]$. En cada iteración, se actualiza el valor de β mediante la función `calc_dbeta`,

la cual garantiza que el conjunto de muestras mantenga una diversidad mínima (ESS). Luego, se generan nuevas muestras usando un esquema de Metropolis en paralelo. Para cada modelo del conjunto, se calcula su verosimilitud y su f_{prior} (o sus logaritmos) dependiendo del valor que toma `use_log_likelihood`.

- **Si no se usa el logaritmo verosimilitud (`use_log_likelihood = False`):** se evalúa $f(\mathbf{m}) = p(\mathbf{m}) \cdot \mathcal{L}(\mathbf{m})^{\beta}$. Este enfoque es más directo e intuitivo, pero puede generar problemas de *underflow* numérico cuando los valores de la verosimilitud $\mathcal{L}(\mathbf{m})$ son muy pequeños.
- **Si se utiliza el logaritmo de la verosimilitud (`use_log_likelihood = True`):** se evalúa $\log f(\mathbf{m}) = \log p(\mathbf{m}) + \beta \cdot \log \mathcal{L}(\mathbf{m})$. Esta formulación es numéricamente más estable, ya que evita el *underflow*.

Ambos enfoques representan la misma distribución posterior. Sin embargo, el uso de logaritmos es preferido en la práctica por razones de precisión y eficiencia computacional.

El proceso se repite hasta que $\beta = 1$, generando una secuencia de distribuciones cada vez más cercanas a la posterior. Así, luego se obtiene un conjunto de modelos y las razones de aceptación de cada una de las etapas.

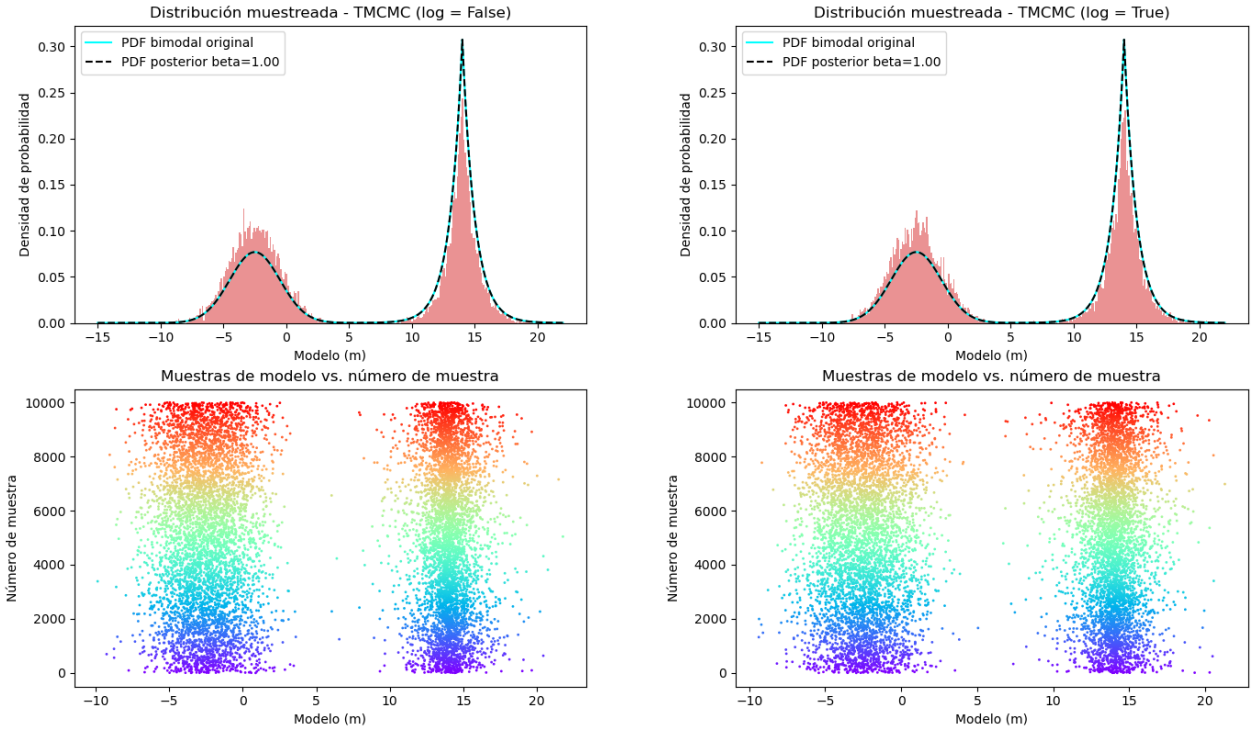
1.3. Verificación del algoritmo

Para verificar la correcta implementación del algoritmo TMCMC, se generaron muestras de una distribución bimodal utilizando tanto el algoritmo implementado como el algortimo de Metropolis en paralelo (versión paralelizada implementada en la Tarea 2). Las distribuciones de densidad de probabilidad obtenidas con ambos métodos fueron comparadas para validar el funcionamiento del algoritmo propuesto.

Los resultados obtenidos con el algoritmo TMCMC, mostrados en la Figura 1, demuestran una correcta aproximación a la distribución posterior bimodal, centrados en aproximadamente en -2.5 y 14, tanto al usar directamente la verosimilitud como su forma logarítmica. En ambos casos se alcanzó $\beta = 1.0$ en solo dos etapas, con valores similares de $\Delta\beta$ y altas tasas de aceptación, lo que confirma la adecuada implementación del módulo `calc_dbeta.py` y la correcta integración del esquema de Metrópolis en paralelo en `tmcmc.py`. Las distribuciones de las muestras reflejan con claridad la estructura bimodal de la densidad objetivo, concentrándose en las dos modas principales sin sesgos aparentes. Aunque se utilizaron distintas semillas aleatorias, los resultados obtenidos son equivalentes y reproducibles, reforzando la validez de la implementación.

Cabe destacar una ventaja importante del uso del logaritmo de la verosimilitud (`use_log_likelihood = True`): aunque no se refleje necesariamente en cambios visibles en la forma

de la distribución muestreada, esta formulación presenta mayor estabilidad numérica. Específicamente, permite manejar con mayor precisión diferencias pequeñas en la verosimilitud, especialmente en las etapas tempranas del algoritmo donde β es pequeño. Adicionalmente, reduce el riesgo de errores de precisión numérica al trabajar con valores extremadamente bajos de verosimilitud, aspecto que puede resultar crítico en problemas de alta dimensionalidad o cuando la verosimilitud decrece rápidamente. Por estas razones, aunque ambos enfoques son válidos y producen resultados coherentes, la versión logarítmica se considera más robusta y se recomienda para aplicaciones que demanden mayor precisión numérica.



(a) caso con $\mathcal{L}$ (b) caso con $\log \mathcal{L}$

Figura 1: Distribución de las muestras obtenidas con TMCMC

El hecho de que tanto la formulación con log-verosimilitud como la directa converjan a las mismas modas confirma la robustez del algoritmo y la consistencia de la implementación.

2. Resolución del problema inverso: Programación del Algoritmo TMCMC + Remuestreo

En esta etapa de la tarea se programará y verificará el algoritmo de TMCMC al cual se ha agregado un remuestreo de las muestras después de la actualización del valor del exponente β. Para ello, se utiliza como base el algoritmo TMCMC recién programado.

El algoritmo TMCMC con remuestreo integra, después de cada incremento de β, un paso adicional que reconstituye el conjunto de muestras para mantener su diversidad y representatividad. Esta adaptación evita que, a medida que aumenta la influencia de la verosimilitud, la mayoría de los pesos se concentren en un reducido número de modelos, fenómeno conocido como degeneración de muestras. Al activar la opción `use_resampling = True`, el algoritmo selecciona nuevas muestras de acuerdo a pesos normalizados calculados para el incremento actual de β. Este procedimiento produce un conjunto actualizado que refleja de manera más fiel la distribución intermedia, manteniendo un tamaño efectivo de muestra elevado en cada transición.

2.1. Algoritmo de Remuestreo

Cuando se avanza de $\beta + \Delta\beta$, queremos que las muestras representen bien la distribución posterior con el nuevo valor de β. Para ello, se calcula un peso para cada muestra anterior:

Una parte fundamental del algoritmo TMCMC consiste en mantener la representatividad de las muestras a medida que se incrementa el exponente β utilizado en la ponderación de la función de verosimilitud.

Para ello, luego de cada actualización del valor de β, se puede aplicar un **algoritmo de remuestreo** que seleccione, con reemplazo, un nuevo conjunto de muestras de acuerdo a la probabilidad de que cada muestra pertenezca a la distribución posterior actualizada.

Dado un conjunto de muestras $\{m_i\}_{i=1}^{N}$ generadas para un valor actual de β, al pasar a un nuevo valor $\beta' = \beta + \Delta\beta$, se definen **pesos no normalizados** w_i para cada muestra según:

- Si se trabaja con la verosimilitud directamente:

$$w_i \propto \mathcal{L}(m_i)^{\Delta\beta}$$

- Si se trabaja con el logaritmo natural de la verosimilitud:

$$w_i \propto \exp\left(\Delta\beta \cdot \log \mathcal{L}(m_i)\right)$$

Una vez calculados los pesos, se normalizan de forma que:

$$\sum_{i=1}^{N} w_i = 1$$

Luego, se generan N nuevas muestras seleccionando con reemplazo los modelos m_i del conjunto original, con probabilidad proporcional a w_i. Esta técnica corresponde a un remuestreo multinomial. Se evalúan los pesos de cada muestra de acuerdo al incremento de β y al tipo de verosimilitud.

Esta función se ejecuta automáticamente dentro del algoritmo TMCMC cuando se activa la opción `use_resampling = True`, asegurando así que las muestras reflejen adecuadamente la distribución posterior con el nuevo valor de β.

2.2. TMCMC + Remuestreo

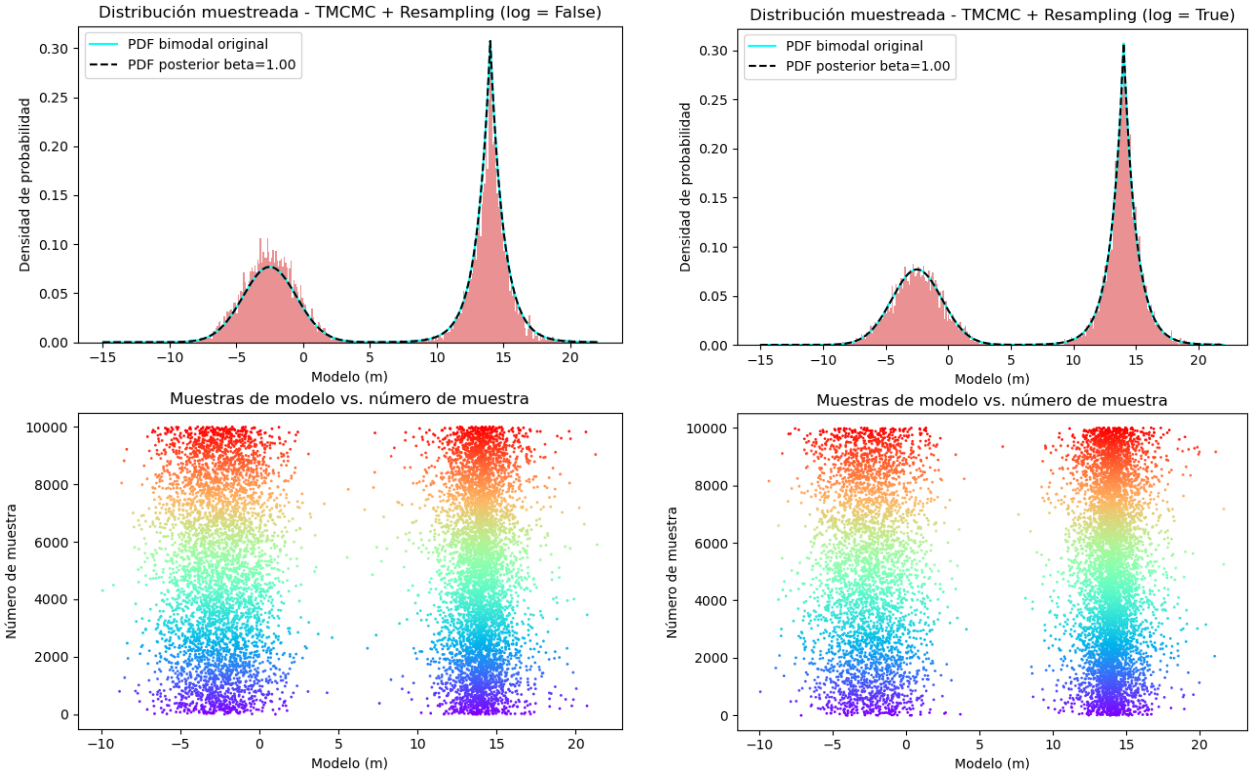
La integración del remuestreo dentro del algoritmo TMCMC permite que, tras cada incremento de β, el conjunto de modelos siga representando fielmente la distribución intermedia correspondiente. De este modo, se evita la degeneración del conjunto de muestras y se mantiene un tamaño efectivo alto durante todo el proceso de transición hacia la distribución posterior. El módulo `tmcmc.py` se modificó para llamar automáticamente a la función `resampling()` después de actualizar el valor de β, cuando se utiliza la opción `use_resampling = True`. Esta mejora hace que el algoritmo explore de forma más equilibrada el espacio de parámetros, generando resultados más robustos y estables en comparación con la versión sin remuestreo.

2.3. Verificación del Algoritmo

En la Figura 2 se muestran los resultados obtenidos al aplicar el algoritmo TMCMC con remuestreo, tanto en el caso con verosimilitud directa (izquierda) como con log-verosimilitud (derecha), como en el caso sin remuestreo. Donde, en ambos casos se observa un ajuste adecuado, la distribución posterior estimada mediante el histograma nuevamente coincide con las dos modas de la densidad objetivo, lo cual confirma que el algoritmo con remuestreo mantiene la representatividad de las muestras en cada transición de β.
Al comparar los resultados obtenidos en esta etapa con los de la sección 1.3, se observa que el remuestreo no modifica la ubicación de las modas principales ni la forma general de la distribución posterior, ya que en ambos casos la densidad se concentra alrededor de los mismos valores característicos de la función objetivo bimodal.

Sin embargo, observando los histogramas de la Figura 2, se aprecia una distribución más uniforme de los pesos a lo largo de las iteraciones, es decir, que se ajustan más a las distribuciones que en el caso sin resampling. Esto se debe a que, gracias al remuestreo, se evita que pocas muestras con pesos altos dominen la población, lo que en la Figura 1 comenzaba a notarse en una ligera concentración de puntos repetidos cuando β crecía.

De esta manera, los resultados obtenidos confirman que la integración del remuestreo en el módulo `tmcmc.py` es consistente con lo esperado teóricamente (Ching & Chen, 2007; Tarantola, 2005) y mejora la robustez de la cadena, sin alterar la correcta aproximación a la distribución posterior observada en la sección 1.3.



(a) con la verosimilitud (b) con el logaritmo de la verosimilitud

Figura 2: Distribución de las muestras obtenidas con TMCMC + remuestreo.

3. Aplicación al Problema de Ajuste Ortogonal a la Recta

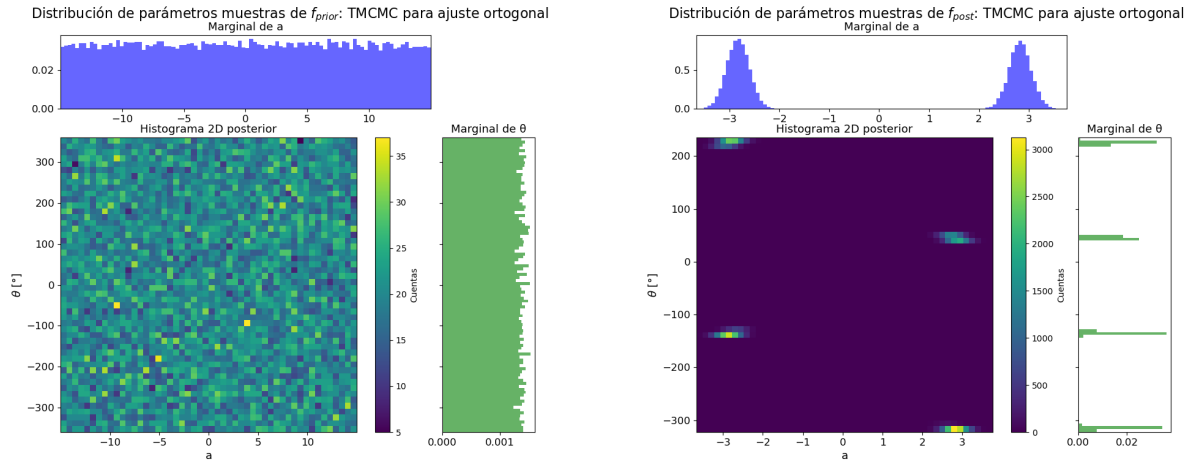
Posteriormente, se utiliza la implementación del algoritmo Transitional Markov Chain Monte Carlo (TMCMC) con resampling para estimar los parámetros de la recta en forma normal (a, θ) que mejor se ajusta ortogonalmente a los datos sintéticos de una elipse, al igual que en la tarea anterior. Se configuraron distribuciones *prior* uniformes para los parámetros restringidos a los intervalos a ∈ [-15, 15] y $\theta \in$ [-360º, 360º], utilizando un ensemble inicial de 50,000 modelos generados mediante muestreo de la distribución *prior* (f_{prior}) (Figura 3a). La función de verosimilitud se construyó asumiendo residuos normalizados con distribución normal estándar, y se empleó una distribución de propuesta normal multivariada con matriz de covarianza diagonal ($\sigma_a = \sigma_\theta = 0.5$). La variable booleana `use_log_likelihood = True` fue seleccionada para garantizar estabilidad numérica, evitando problemas de *underflow* al trabajar con productos de probabilidades muy pequeñas y mejorando la precisión en los cálculos durante las transiciones del algoritmo.

Con esto, como resultado de la aplicación del algoritmo, se obtiene la distribución de muestras a posteriori de la Figura 3b, que muestra una distribución marginal para a muy similar a dos distribuciones normales centradas en torno a ±2.8 y cuatro zonas de mayor probabilidad para θ en torno a los $45º \pm n \times 180º$ con una distribución marginal compuesta por un conjunto de peaks de densidad que, en términos de distribuciones, corresponderían aproximadamente a un conjunto de deltas de Dirac (centradas en 45º) con periodicidad constante de 180º (π en radianes). Es decir, las regiones de máxima probabilidad para θ se asemejan a un "Peine de Dirac" (normalizado) puesto que cada región de alta probabilidad es prácticamente un peak en torno a un valor (45º) con periodicidad constante de 180º (π en radianes).

Comparación con resultados de la Tarea 2

Se comparan los resultados obtenidos de la Aplicación al Problema de Ajuste Ortogonal a la Recta, considerando dos casos. El primero, aplicando el algoritmo TMCMC con remuestreo; el segundo, usando el algoritmo de Metropolis en paralelo sin remuestreo (P4.4, Tarea 2). Para cada caso se analizan la evolución de la cadena y las distribuciones conjunta y marginales finales.

Antes de analizar la evolución de las cadenas de muestreo, se comparan las distribuciones de probabilidad generadas a partir del muestreo de la distribución a priori (f_{prior}) y posterior (f_{post}). La Figura 3 muestra que, mientras las muestras de f_{prior} exploran de forma casi uniforme el espacio de parámetros (a, θ), las de f_{post} se concentran en cuatro zonas específicas con valores característicos de θ y a. Este comportamiento evidencia que el algoritmo TMCMC con remuestreo logró capturar adecuadamente las regiones de alta verosimilitud definidas por los datos sintéticos generados a partir de una elipse.



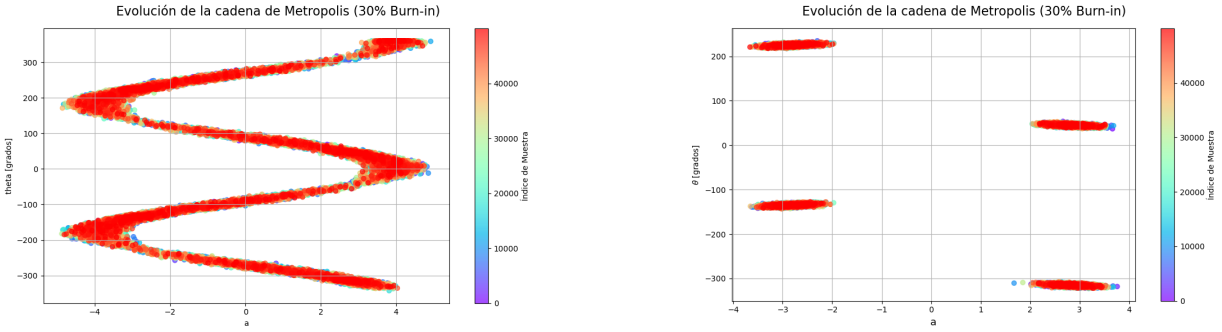
(a) (b)

Figura 3: Distribución de las muestras para el problema de Ajuste Ortogonal a la Recta con TMCMC y remuestreo. a) Muestras previas a TMCMC (de f_{prior}). b) Muestras post TMCMC (de f_{post}).

En la Figura 4, se presenta la comparación entre las muestras finales obtenidas mediante el algoritmo de Metropolis en paralelo sin remuestreo y el algoritmo TMCMC con remuestreo.

La Figura **??** muestra la evolución de la cadena obtenida con el algoritmo de Metropolis en paralelo sin remuestreo. Si bien también se identifican las cuatro regiones de alta probabilidad en la Fig. 5.a, la distribución de las muestras presenta mayor dispersión y una menor definición en las bandas. Esto indica que, en ausencia del mecanismo de remuestreo, el algoritmo puede converger hacia las zonas correctas, pero con menor eficiencia, acumulando muestras con mayor variabilidad e incluso reteniendo modelos de baja verosimilitud. Además, la cadena tiende a perder diversidad más rápidamente, especialmente al final del muestreo, lo que puede afectar la representatividad estadística del conjunto final de muestras. En otras palabras, al algoritmo le cuesta saltar de una distribución a otra. Por otro lado, en la Figura **??** se observa la evolución de la cadena generada mediante TMCMC con remuestreo con 30 % *burn_in*. Las muestras iniciales exploran uniformemente el espacio de parámetros a y θ, y progresivamente se concentran en cuatro bandas diagonales bien definidas, que corresponden a soluciones equivalentes por simetría de la elipse generadora. Estas bandas reflejan los valores característicos de θ y las correspondientes combinaciones de a, como es esperable por la periodicidad y simetría del modelo.

La Figura 5 muestra la comparación entre las distribuciones de muestras obtenidas para el problema de Ajuste Ortogonal a la Recta. Ambas figuras presentan los histogramas marginales de los parámetros a y θ, así como la distribución conjunta estimada a partir de las muestras generadas en cada caso. En ambas figuras, se identifican las mismas soluciones, con a distribuyéndose en torno a ± 3.

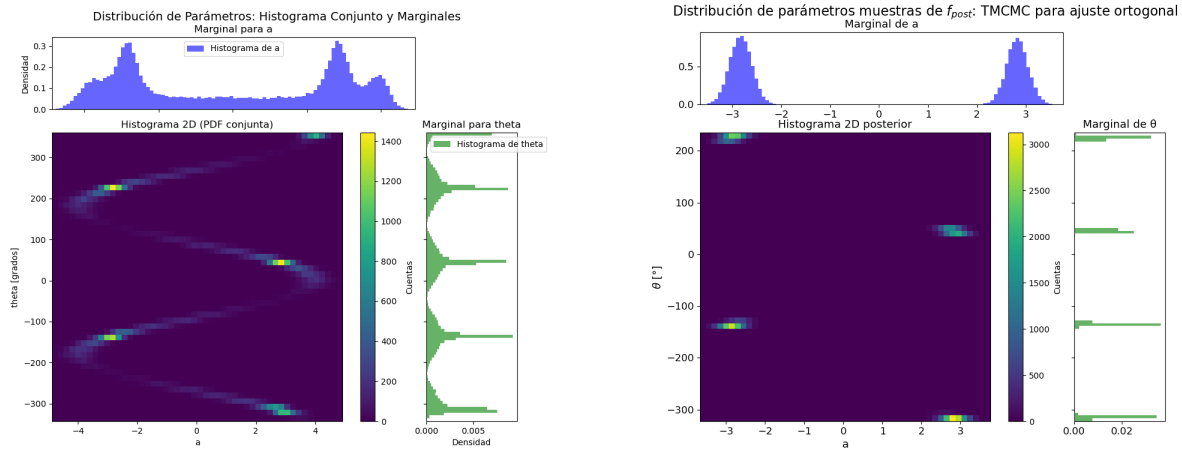


(a) (b)

Figura 4: Comparación entre TMCMC con remuestreo y Metropolis. a) Muestras finales de cada cadena Metropolis (Tarea 2). b) Muestras finales de cada cadena TMCMC con remuestreo.

En la Figura 5.a se observan cuatro bandas diagonales bien definidas en la distribución conjunta, distribuidas simétricamente en el espacio (a, θ). Estas corresponden a soluciones equivalentes por simetría geométrica del problema, donde los valores de a se agrupan en torno a magnitudes positivas y negativas similares, reflejando las distintas orientaciones posibles de la recta que ajusta ortogonalmente los datos de la elipse.

Al comparar con la Figura 5.b, se aprecia una mayor concentración de las muestras en torno a los máximos de verosimilitud, con modas más definidas. Esto se debe a que el remuestreo estabiliza los pesos de las muestras en cada transición de β, manteniendo un tamaño efectivo alto y una representación más fiel de la distribución a posteriori.



(a) (b)

Figura 5: Distribución de las muestras obtenidas para el problema de Ajuste Ortogonal a la Recta. a) Solo Metrópolis en Paralelo ($\beta = 1$). b) TMCMC con remuestreo.

Cabe destacar que este cambio observado en la distribución de las muestras obtenidas al utilizar el algoritmo de TMCMC con remuestreo y las obtenidas sin TMCMC ni remuestreo (Figura 5) es consistente con en comportamiento ejemplificado por Minson et al. (2013) pa-

ra el proceso de remuestreo, donde al incorporar el remuestreo por importancia se obtiene una distribución mucho más concentrada en regiones de mayor probabilidad, pues aquellas muestras con menor probabilidad fueron reemplazadas por otras de mayor probabilidad.

En nuestro caso, este efecto se refleja en la forma y definición de los histogramas marginales. Aunque ambos métodos identifican correctamente las soluciones alrededor de valores de a próximos a ± 3, la distribución obtenida con TMCMC con remuestreo presenta modos mucho más estrechos y definidos. Mientras que, la distribución de Metropolis es más dispersa, con colas más anchas y presencia de ruido.

De manera similar, la distribución marginal de θ en TMCMC exhibe picos delgados y concentrados, como un *"Peine de Dirac"*, mientras que en Metrópolis los mismos aparecen más suavizados y extendidos, lo que reduce la resolución modal.

En resumen, mediante esta comparación, se evidencia que el TMCMC con remuestreo mejora la calidad de las estimaciones respecto al Metropolis en paralelo utilizado en la Tarea 2. Las soluciones encontradas son consistentes en ambos métodos, pero el TMCMC mantiene diversidad en el *ensemble*, evita la degeneración de partículas y permite una convergencia más clara hacia las regiones de alta verosimilitud. La elección de trabajar con log-verosimilitud (`use_log_likelihood = True`) resultó adecuada para garantizar estabilidad numérica, tal como recomiendan Tarantola (2005) y Ching & Chen (2007).

Referencias

[1] Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics (SIAM).

[2] Ching, J., & Chen, Y. C. (2007). Transitional Markov Chain Monte Carlo method for Bayesian model updating, model class selection, and model averaging. *Journal of Engineering Mechanics*, 133(7), 816–832.

[3] S. E. Minson, M. Simons, J. L. Beck (2013). Bayesian inversion for finite fault earthquake source models I—theory and algorithm, Geophysical Journal International, Volume 194, Issue 3, Pages 1701–1726, https://doi.org/10.1093/gji/ggt180