Cómputo Científico para Probabilidad, Estadística y Ciencia de Datos

Ezau Faridh Torres Torres

TAREA 7: MCMC: Metropolis-Hastings II

Fecha de entrega: 23/Oct/2024.

NOTA: Los ejercicios se encuentran repartidos en los archivos:

- ejercicio1_tarea7.py
- ejercicio2_tarea7.py
- ejercicio3_tarea7.py

Con el algoritmo Metropolis-Hastings (MH), simular lo siguiente:

---

1. Sean $x_i \sim Ga(\alpha, \beta)$; $i = 1, 2, \ldots, n$. Simular datos x_i con $\alpha = 3$ y $\beta = 100$ considerando los casos $n = 5$ y 40.

   Con $\alpha \sim \mathrm{U}(1, 4), \beta \sim \exp(1)$ distribuciones a priori, se tiene la posterior

   $$f(\alpha, \beta \mid \bar{x}) \propto \frac{\beta^{n\alpha}}{\Gamma(\alpha)^n} r_1^{\alpha-1} e^{-\beta(r_2+1)} \cdot \mathbb{1}_{(1\leq\alpha\leq 4)} \cdot \mathbb{1}_{(\beta>1)} \tag{1}$$

   con

   $$r_2 = \sum_{i=1}^{n} x_i \text{ y } r_1 = \prod_{i=1}^{n} x_i \tag{2}$$

   En ambos casos, grafica los contornos para visualizar dónde está concentrada la posterior. Utilizar la propuesta

   $$q\left(\begin{pmatrix}\alpha_p \\ \beta_p\end{pmatrix} \middle| \begin{pmatrix}\alpha \\ \beta\end{pmatrix}\right) = \begin{pmatrix}\alpha \\ \beta\end{pmatrix} + \begin{pmatrix}\varepsilon_1 \\ \varepsilon_2\end{pmatrix} \tag{3}$$

   donde

   $$\begin{pmatrix}\varepsilon_1 \\ \varepsilon_2\end{pmatrix} \sim \mathcal{N}_2\left(\begin{pmatrix}0 \\ 0\end{pmatrix}, \begin{pmatrix}\sigma_1^2 & 0 \\ 0 & \sigma_2^2\end{pmatrix}\right). \tag{4}$$

Respuesta:

En el archivo ejercicio1_tarea7.py se implementa la función *METROPOLIS_HASTINGS_SIM()* la cual aplica el algoritmo Metropolis-Hastings para propuestas simétricas simulando una cadena de Markov en $\mathbb{R}^n$ y toma los siguientes argumentos:

- La función objetivo f (en este caso es la posterior (1)).
- La distribución propuesta q_{gen} (en este caso es la propuesta (3)).
- El valor inicial x_0 (en este caso es (α, β) con $\alpha \sim \mathrm{U}(1, 4), \beta \sim \exp(1)$).
- El número de iteraciones del algoritmo (casi siempre se usa $N = 10,000$).

Regresa la cadena de Markov simulada en $\mathbb{R}^n$ y usa el criterio de aceptación: si y_t es la propuesta dada por $q_{gen}(\cdot|x_t)$ en x_t, entonces se acepta y_t con probabilidad $\rho(x_t, y_t)$ con

$$\rho(x, y) = \min\left\{1, \frac{f(y)}{f(x)}\right\} \tag{5}$$

(en este caso, $\frac{q(x|y)}{q(y|x)} = 1$ ya que la propuesta es simétrica) y se rechaza con probabilidad $1 - \rho(x_t, y_t)$. A continuación, se define la función *SIMULAR_GAMMA()*, la cual hace las simulaciones $x_i \sim Ga(\alpha, \beta)$; $i = 1, 2, \ldots, n$

Después, se define la función posterior (1) llamada *posterior()*, y la función *propuesta_gen()* dada por (3) la cual, como vimos en clase, es simétrica. Dentro de la función *posterior()* se aplicó el logaritmo a los términos para evitar desbordamientos numéricos.

Finalmente:

- Se simulacion $x_i \sim Ga(\alpha = 3, \beta = 100)$; $i = 1, 2, \ldots, n$ para los casos $n_1 = 5$ y $n_2 = 40$.
- Se dio el punto inicial $x_0 = (\alpha_0, \beta_0)$ con $\alpha_0 \sim \mathrm{U}(1, 4), \beta_0 \sim \exp(1)$. Tal punto resultó ser: $(3.886, 1.007)$.
- Se aplicó el algoritmo Metropolis-Hastings para generar las cadenas correspondientes para n_1 y n_2 usando $N = 10,000$ iteraciones.
- Se usaron los parámetros de varianza de la propuesta $\sigma_1 = 0.1$ y $\sigma_2 = 10$ (se comenta más de esta elección al final).
- Se generaron los histogramas resultantes de α y β para ambs casos y se graficaron los contornos para visualizar la concentración de la posterior.

Se obtuvieron los siguientes resultados: para $n_1 = 5$ se obtuvo una tasa de aceptación del 18.45% y el promedio de la cadena para cada parámetro resultó ser: $(1.163, 6.006)$ Mientras que para $n_2 = 40$ se obtuvo una tasa de aceptación del 21.42% y el promedio de la cadena para cada parámetro resultó ser: $(1.131, 19.920)$. Los histogramas resultantes se encuentran en la figura siguiente:

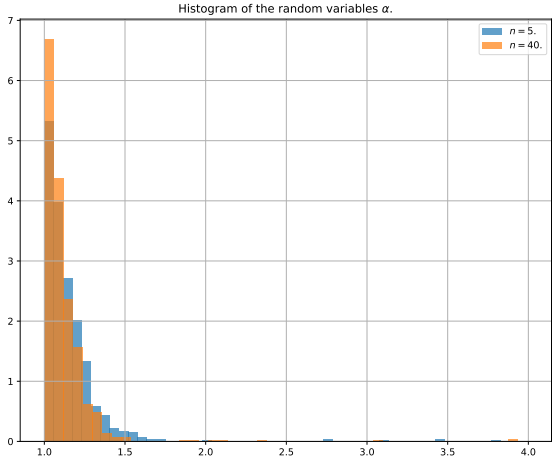


Figure 1: α.

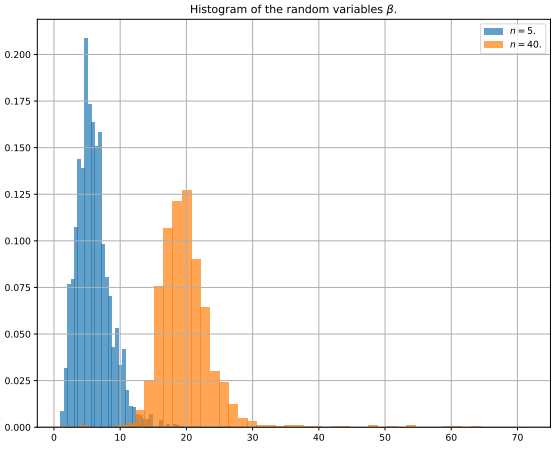
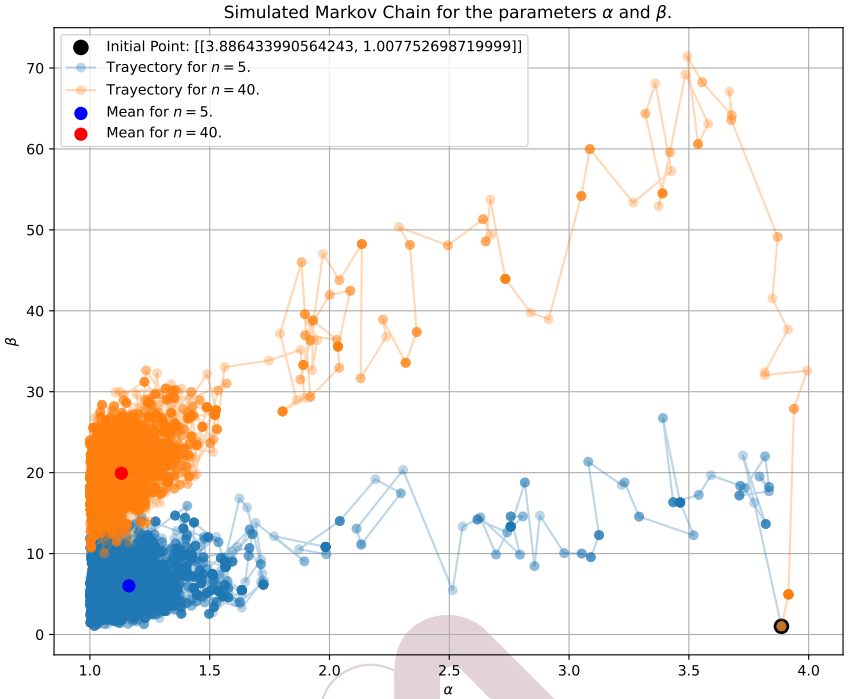
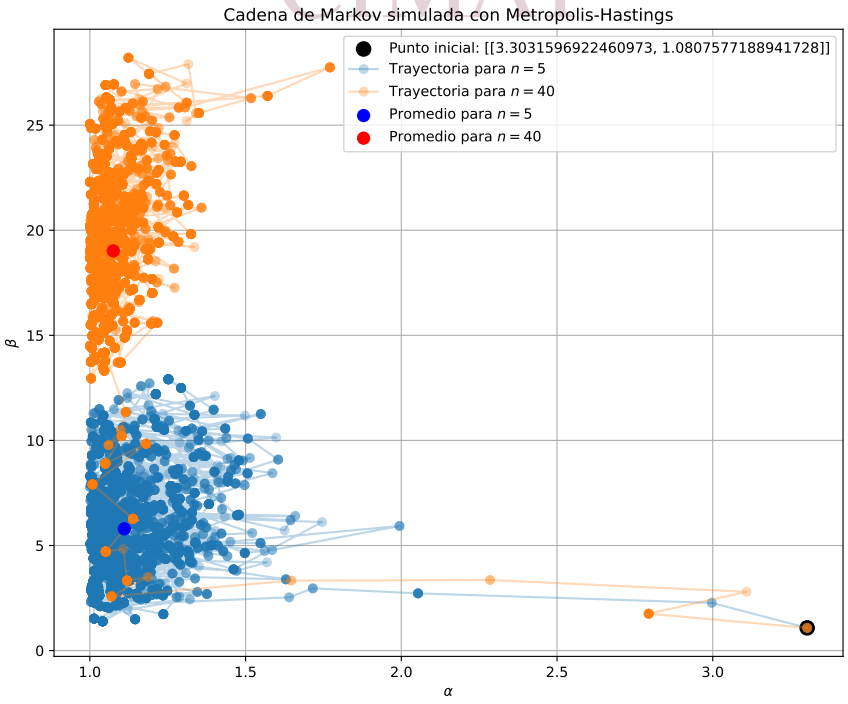


Figure 2: β.

El gráfico de contornos es:



Nótese que los valores promedio de las cadenas generadas se encuentran los puntos rojo y azul del gráfico anterior. En este caso particular, se puede ver que existe más varianza en el eje vertical, esto debido a que se tomó σ_2 100 veces mayor que σ_1. Otro ejemplo con varianzas distintas es:



En este último caso, se usó $\sigma_1 = \sigma_2 = 1$. Sin importar el punto inicial, las varianzas elegidas o llevar el número de iteraciones a valores muy grandes, el algoritmo siempre terminaba alrededor de los mismos puntos, lo cual no corresponde con lo esperado ya que sabemos que los valores usados para generar las muestras fueron $\alpha = 3$ y $\beta = 100$. Incluso se varió n para tener muestras más grandes y el resultado era similar (aunque sí hacía crecer los promedios de la cadena) o se volvía imposible de obtener debido a que el cálculo de r_1 causaba muchas indeterminaciones.

2. Simular de la distribución $Gamma(\alpha, 1)$ con la propuesta $Gamma([\alpha], 1)$, donde $[\alpha]$ denota la parte entera de α. Además, realizar el siguiente experimento: poner como punto inicial $x_0 = 950$ y graficar la evolución de la cadena, es decir, $f(X_t)$ vs t.

Respuesta:

En el archivo ejercicio2_tarea7.py se implementa la función *METROPOLIS_HASTINGS()* la cual aplica el algoritmo Metropolis-Hastings para cualquier dimensión n y toma los siguientes argumentos:

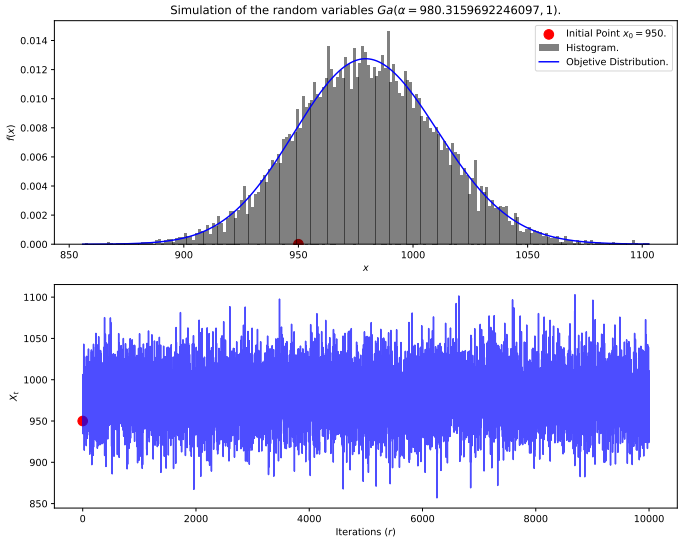
- La función objetivo f (en este caso es una distribución $Gamma(\alpha, 1)$).
- La distribución propuesta q_{gen} (en este caso es una distribución $Gamma([\alpha], 1)$).
- La función q_{pdf} la cual es función de densidad de probabilidad de la propuesta q_{gen}.
- El valor inicial x_0 (en este caso es $x_0 = 950$).
- El número de iteraciones del algoritmo (casi siempre se usa $N = 10,000$).

Esta función regresa la cadena de Markov simulada. Usa el criterio de aceptación: si y_t es la propuesta dada por $q(\cdot|x_t)$ en x_t, entonces se acepta y_t con probabilidad $\rho(x_t, y_t)$ con

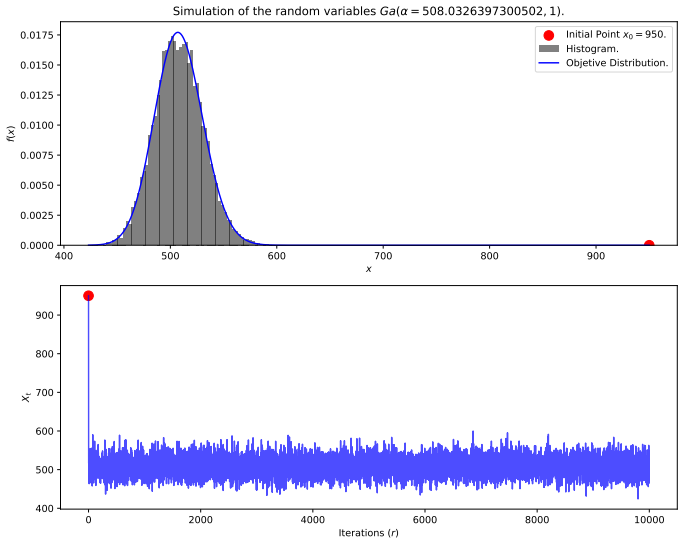
$$\rho(x, y) = \min\left\{1, \frac{f(y)}{f(x)}\frac{q(x|y)}{q(y|x)}\right\} \tag{6}$$

y se rechada con probabilidad $1 - \rho(x_t, y_t)$ (para este caso se utiliza q_{pdf}). Se define la función objetivo *f()* (que es la pdf de una distribución $Gamma(\alpha, 1)$), y las funciones propuesta *q_gen()* y *q_pdf()* dada por la distribución $Gamma([\alpha], 1)$. Finalmente, se simulan variables aleatorias $Gamma(\alpha, 1)$ con 3 alphas distintos y se comparan las distribuciones obtenidas con un histograma, también se genera el gráfico que describe la evolución de la cadena. Dado que $x_0 = 950$ es relativamente grande, α no tiene que ser tan alejado de 950 para evitar indeterminaciones y el riesgo de no converger. Para esto, se tomaron los 3 valores de α tales que:

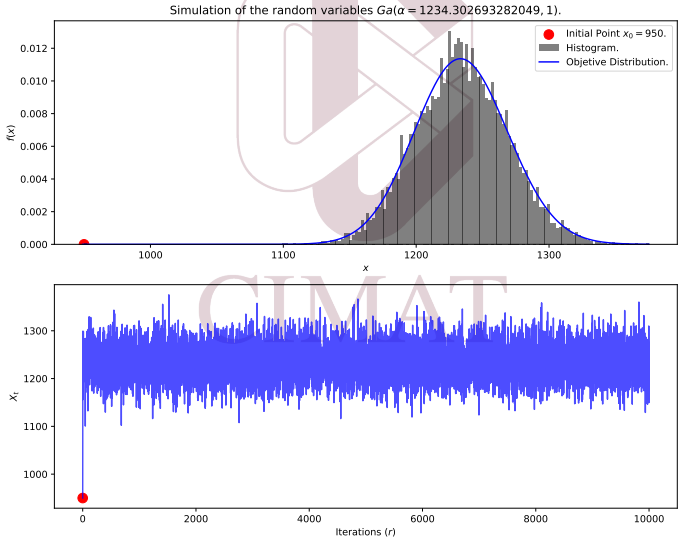
- $\alpha \in [x_0 - 200, x_0 + 200]$



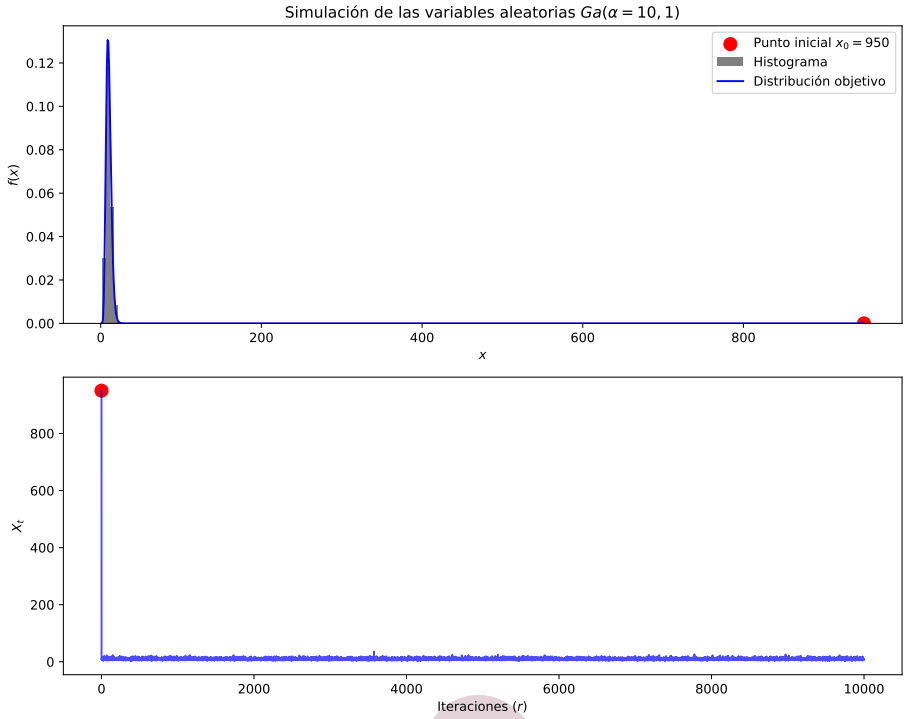
- $\alpha \in [x_0 - 500, x_0]$



- $\alpha \in [x_0, x_0 + 500]$



Se simularon varios ejemplos con valores de α más pequeños pero, al estar tan lejos del soporte, de la distribución, tiende a generar indeterminaciones al comienzo de la cadena de Markov. Un ejemplo es el siguiente para $\alpha = 10$, en donde se nota que la cadena demora un poco más en llegar a la región donde se concentra el soporte de $Gamma(10, 1)$:


Simulación de las variables aleatorias $Ga(\alpha = 10, 1)$
Punto inicial $x_0 = 950$
Histograma
Distribución objetivo
$f(x)$
x
X_t
Iteraciones (r)

.

3. Implementar Random Walk Metropolis Hasting (RWMH) donde la distribución objetivo es $\mathcal{N}_2(\mu, \Sigma)$, con

$$\mu = \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}. \tag{7}$$

   Utilizar como propuesta $\varepsilon_t \sim \mathcal{N}_2(0, \sigma I)$. ¿Cómo elegir σ para que la cadena sea eficiente? ¿Qué consecuencias tiene la elección de σ?

   Como experimento, elige como punto inicial $x_0 = \binom{1000}{1}$ y comenta los resultados.

Para todos los incisos del ejercicio anterior:

- Establece cual es tu distribución inicial.
- Grafica la evolución de la cadena.
- Indica cuál es el Burn-in.
- Comenta qué tan eficiente es la cadena.
- Implementa el algoritmo MH considerando una propuesta diferente.

Respuesta:

En el archivo ejercicio3_tarea7.py se implementan las funciones

- *contornos()*: grafica los contornos de la densidad (se muestran los contornos de la distribución real debajo y se sobrepone la cadena generada.
- *marginal_histograms()*: genera los histogramas de las cadenas marginales.
- *plot_evolution_burn_in()*: grafica la evolución de las cadenas de Markov marginales y sobrepone el burn-in calculado.
- *moving_average()*: calcula la media móvil.
- *estimate_burn_in()*: Determinar cuándo se estabiliza la media móvil. Esta técnica consiste en observar la media móvil de las muestras y encontrar cuándo esta se estabiliza. Es una aproximación para el burn-in.

Luego, se define la función *RANDOM_WALK_METROPOLIS_HASTINGS()* la cual implementa algoritmo Random Walk Metropolis-Hastings en $\mathbb{R}^n$. Toma como argumentos:

- La función objetivo f (en este caso es (7)).
- El valor inicial x_0.
- La matriz de covarianza (en este caso es σI).
- El número de iteraciones del algoritmo (casi siempre se usa $N = 10,000$).

Regresa la cadena de Markov simulada en $\mathbb{R}^n$ y usa el criterio de aceptación: si y_t es la propuesta dada por $y_t = x_t + \varepsilon_t$ en x_t, entonces se acepta y_t con probabilidad $\rho(x_t, y_t)$ con

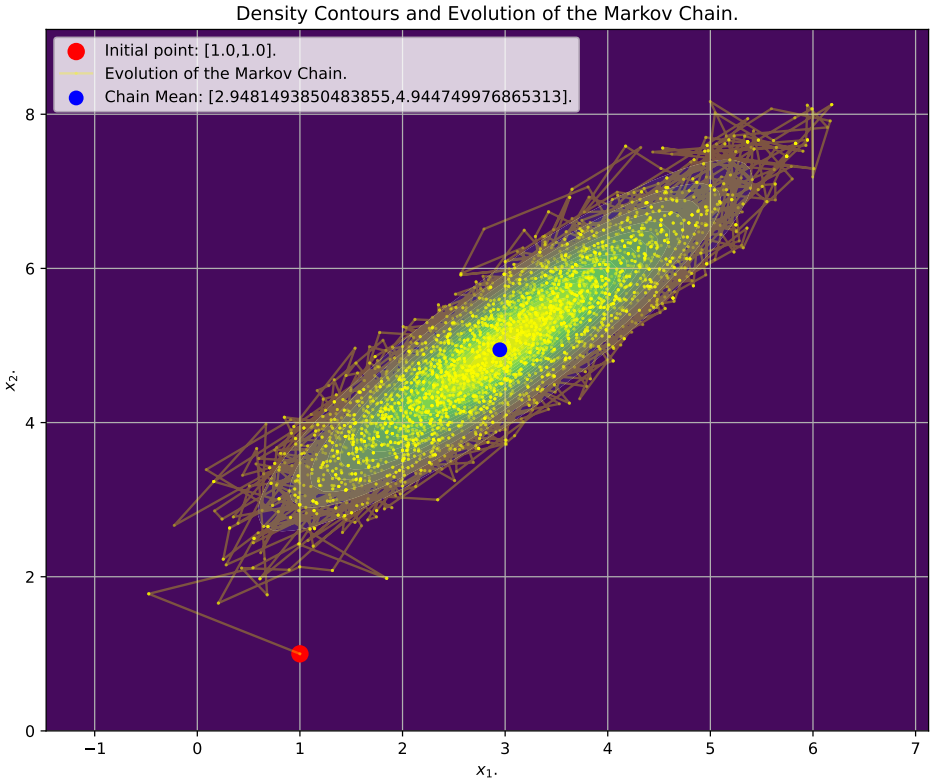
$$\rho(x, y) = \min\left\{1, \frac{f(y)}{f(x)}\right\} \tag{8}$$

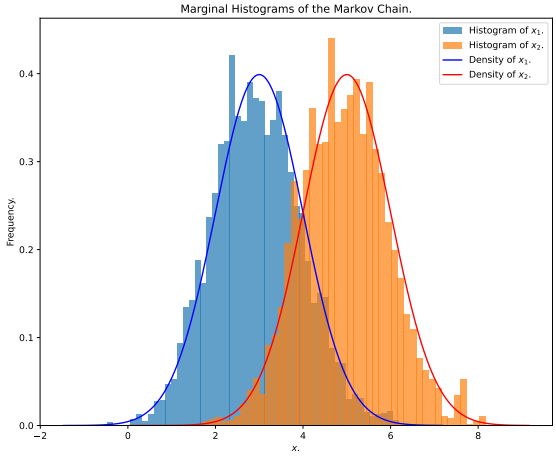
y se rechaza con probabilidad $1 - \rho(x_t, y_t)$. Luego, se define la función *f_pdf()* la cual implementa (7).

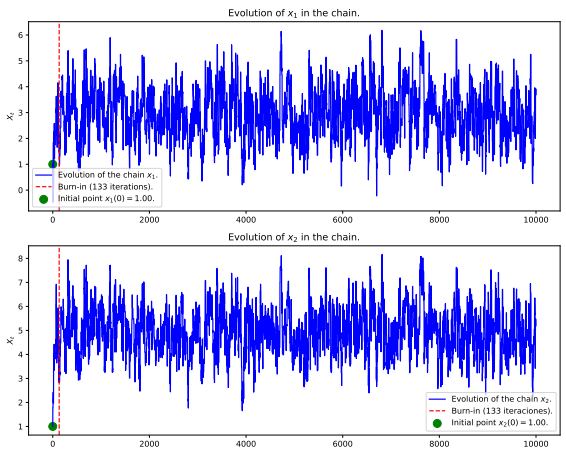
Finalmente, se simulan 4 cadenas de Markov en $\mathbb{R}^2$ con diferentes puntos iniciales y matrices de covarianza para la propuesta. Se grafican los contornos de la densidad y la evolución de la cadena de Markov en $\mathbb{R}^2$, los histogramas marginales y la evolución de las cadenas de Markov marginales con el burn-in estimado.

Ejemplo 1: Se usó el punto inicial $x_0 = (1, 1)$, y $\sigma = 1$, i.e., la matriz de covarianzas fue I y se obtuvieron los siguientes resultados:

Tasa de aceptación: 31.10%, burn-in estimado: 133 iteraciones, promedio de la cadena: $[2.948, 4.944]$ y covarianza de la cadena: $\begin{pmatrix} 1.016 & 0.911 \\ 0.911 & 1.007 \end{pmatrix}$. El algoritmo fue eficiente debido a la similitud en la magitud de σ respecto a la lejanía de x_0 de $(3, 5)$.

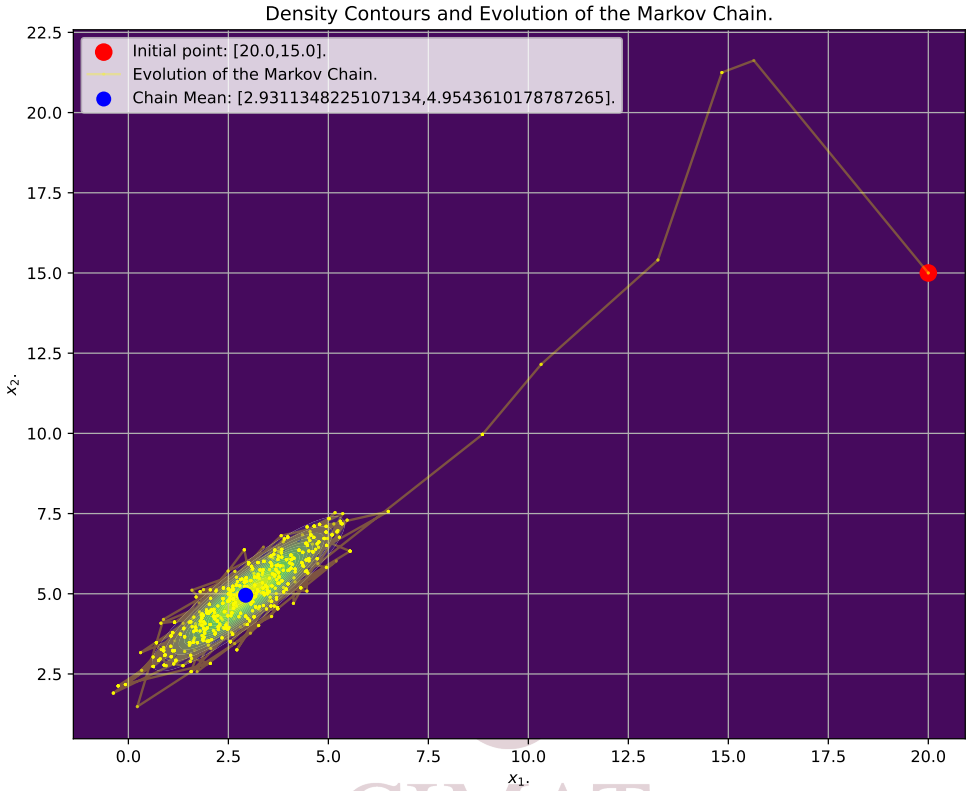


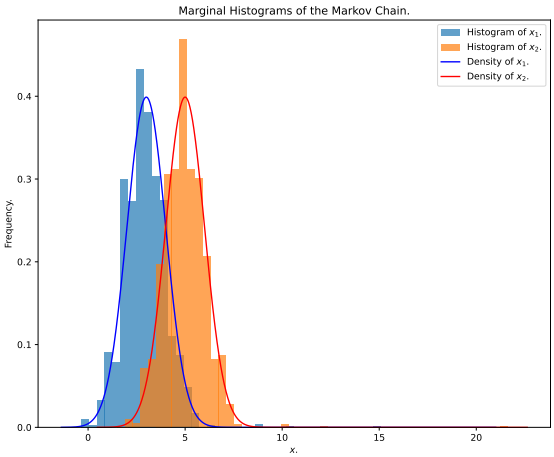


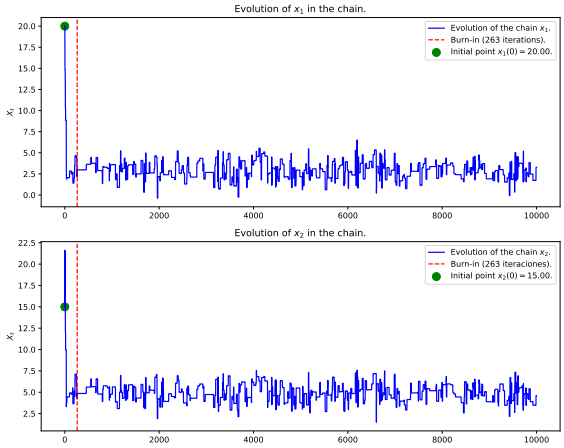


Ejemplo 2: Se usó el punto inicial $x_0 = (20, 15)$, y se usó $\sigma = 20$, i.e., la matriz de covarianzas fue $20I$ y se obtuvieron los siguientes resultados:

Tasa de aceptación: 4.06%, burn-in estimado: 263 iteraciones, promedio de la cadena: $[2.931, 4.954]$ y covarianza de la cadena: $\begin{pmatrix} 1.282 & 1.163 \\ 1.163 & 1.266 \end{pmatrix}$. El algoritmo fue eficiente debido a la similitud en la magitud de σ respecto a la lejanía de x_0 de $(3, 5)$.

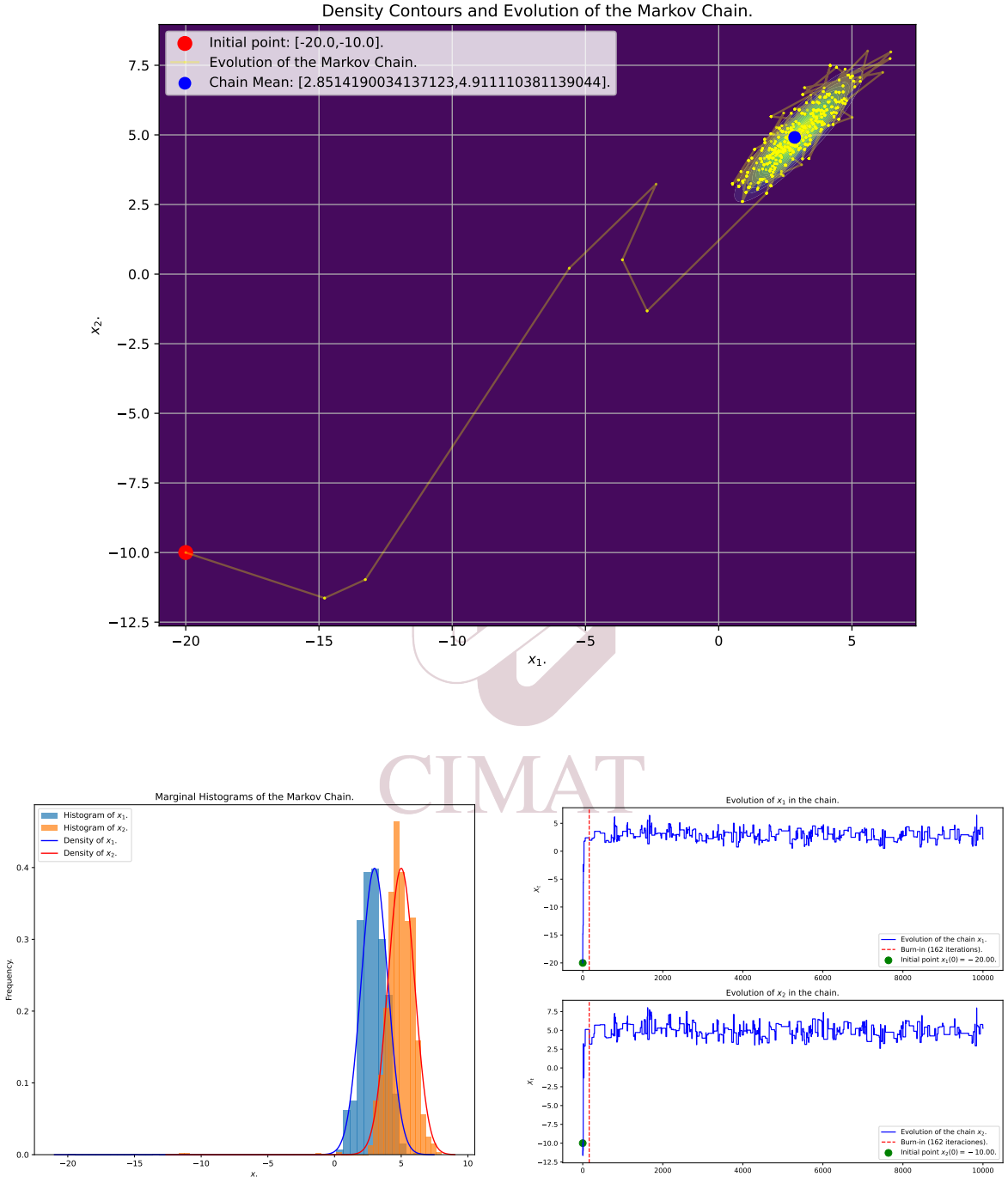






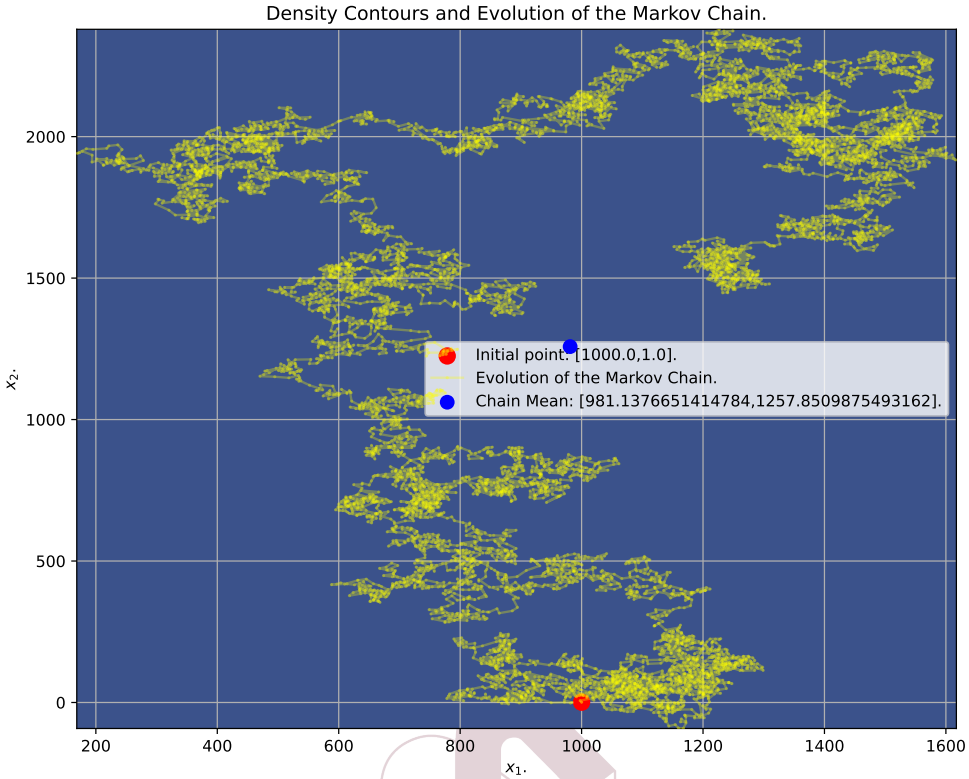
Ejemplo 3: Se usó el punto inicial $x_0 = (-20, -10)$, y se usó $\sigma = 25$, i.e., la matriz de covarianzas fue $25I$ y se obtuvieron los siguientes resultados:

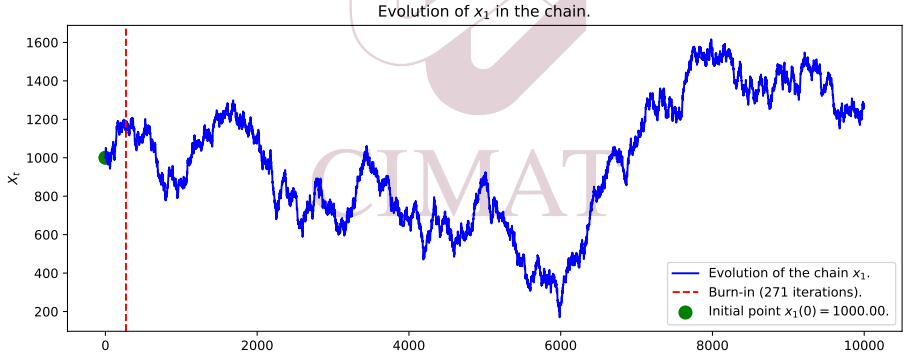
Tasa de aceptación: 4.05%, burn-in estimado: 162 iteraciones, promedio de la cadena: $[2.851, 4.911]$ y covarianza de la cadena: $\begin{pmatrix} 1.281 & 1.113 \\ 1.113 & 1.147 \end{pmatrix}$. El algoritmo fue eficiente debido a la similitud en la magitud de σ respecto a la lejanía de x_0 de $(3, 5)$.

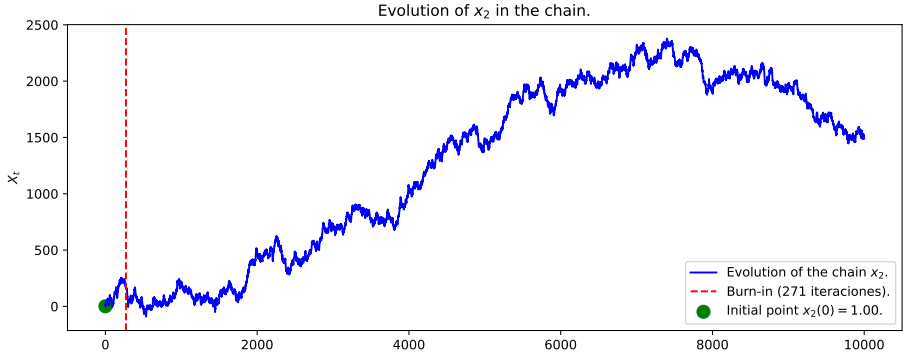


Ejemplo 4: Se usó el punto inicial $x_0 = (1000, 1)$, y se usó $\sigma = 100$, i.e., la matriz de covarianzas fue $100I$ y se obtuvieron los siguientes resultados:

Tasa de aceptación: 99.99%, burn-in estimado: 271 iteraciones, promedio de la cadena: $[981.137, 1257.851]$ y covarianza de la cadena: $\begin{pmatrix} 110724.962 & 37422.424 \\ 37422.424 & 613530.619 \end{pmatrix}$. El algoritmo no fue eficiente para ninguna configuración de parámetros debido a la lejanía de x_0 de $(3, 5)$.







Comentarios finales:

Mientras se revisaban diversos ejemplos variando parámetros como σ y el punto inicial x_0, iba resultando que, según que tan lejos eligiéramos x_0 del punto $(3, 5)$, iba a necesitar mayor

varianza o un número mayor de iteraciones para que la cadena se encontrara cerca de $(3, 5)$. El parámetro σ controla el tamaño de los pasos en el proceso de Metropolis-Hastings. Se notó que las diferentes consecuencias de elegir diferentes valores de σ son las siguientes:

- σ muy pequeño: Las propuestas estarán muy cerca del valor anterior, lo que puede resultar en una tasa de aceptación muy alta, pero la cadena se moverá muy lentamente por el espacio de estados, lo que lleva a una mayor correlación entre las muestras (un ejemplo de esto es el ejemplo 1). En este caso, se necesitarán más iteraciones para cubrir toda la distribución objetivo, y el burn-in será más largo.
- σ muy grande: Las propuestas serán más lejanas, lo que puede llevar a una baja tasa de aceptación, ya que muchas propuestas serán rechazadas (ejemplos de esto son los ejemplos 2 y 3). La cadena puede quedarse atrapada en ciertas regiones, y será difícil explorar completamente el espacio de estados. Aquí también el burn-in puede será más largo, ya que la cadena podría tardar en estabilizarse.

Respecto a el punto inicial $x_0 = (1000, 1)$ (se tiene un ejemplo en el ejemplo 4), como se encontraba demasiado lejos de la zona donde se concentra el soporte de la distribución objetivo, generaba demasiados errores e indeterminaciones al momento de aplicar Metropolis-Hastings (específicamente al calcular la tasa de aceptación $\frac{f(y)}{f(x)}$). Los "mejores" resultados se tenían al usar varianzas muy grandes, sin embargo, en ninguno de los casos se logró la convergencia (no se logró la eficiencia). Esto se debe a que la probabilidad de este x_0 bajo la distribución objetivo es 0, debido a su lejanía del punto $(3, 5)$.