Ejercicio 1.

1 Descomposición en Valores Singulares (SVD)

Para cualquier matriz $A \in \mathbb{C}^{m\times n}$, existe una descomposición en valores singulares:

$$A = U\Sigma V^*$$

donde:

- $U \in \mathbb{C}^{m\times m}$ es una matriz unitaria,
- $V \in \mathbb{C}^{n\times n}$ es una matriz unitaria,
- $\Sigma \in \mathbb{R}^{m\times n}$ es una matriz diagonal que contiene los valores singulares $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$, con $p = \min(m, n)$.

El número de condición de una matriz se define como:

$$\kappa(A) = \frac{\sigma_{\max}}{\sigma_{\min}},$$

donde $\sigma_{\max}$ y $\sigma_{\min}$ son los valores singulares máximo y mínimo de A, respectivamente.
Aunque no existe un umbral universal que clasifique estrictamente a una matriz como bien o mal condicionada, en términos prácticos se considera:

- $\kappa(A) \approx 1$: la matriz está **bien condicionada**,
- $\kappa(A) \gg 1$: la matriz está **mal condicionada**.

En particular, para matrices simétricas definidas positivas B, se puede calcular la factorización de Cholesky:

$$B = LL^T,$$

donde L es una matriz triangular inferior.
En este ejercicio se generaron matrices simétricas definidas positivas, ya que sus autovalores son estrictamente positivos, con espectro controlado de la forma:

$$B = Q^* \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{20})Q,$$

donde Q es una matriz unitaria y $\lambda_1 > \lambda_2 > \cdots > \lambda_{20} = 1 > 0$.
Se consideraron dos casos:

1. Matriz bien condicionada: λ_i distribuidos uniformemente en el intervalo $[100, 1]$.
2. Matriz mal condicionada: λ_i distribuidos uniformemente en el intervalo $[10^6, 1]$.

Para cada caso, se generó además una matriz perturbada:

$$B_\varepsilon = Q^* \operatorname{diag}(\lambda_1 + \varepsilon_1, \ldots, \lambda_{20} + \varepsilon_{20})Q,$$

con $\varepsilon_i \sim N(0, 0.01)$.

1.1 Inciso a) Resultados

	$\kappa(B)$	Error entrada	Error salida	Ratio salida/entrada
Bien condicionado	1.0×10^{1}	1.25×10^{-3}	1.11×10^{-3}	0.89
Mal condicionado	1.0×10^{4}	1.31×10^{-6}	9.13×10^{-6}	6.98

Table 1: Comparación de la estabilidad de la factorización de Cholesky bajo buen y mal condicionamiento.

Como se observa en la Tabla 1, una matriz mal condicionada amplifica los errores de entrada: pequeñas perturbaciones en los datos producen errores mucho más grandes en la factorización de Cholesky. En contraste, el caso bien condicionado mantiene la estabilidad numérica.

1.2 Análisis de Perturbaciones

Table 2: Sensibilidad de la factorización de Cholesky a perturbaciones

Caso	$\|L - L_\varepsilon\|/\|L\|$
Bien condicionado	1.11×10^{-3}
Mal condicionado	9.13×10^{-6}

La Tabla 2 muestra la sensibilidad relativa de la factorización ante perturbaciones en la matriz original. Si bien el error relativo es más pequeño en el caso mal condicionado, esto se debe a que la perturbación inicial también fue menor. Lo importante es observar la *amplificación relativa* de los errores, la cual resulta mucho mayor cuando la matriz está mal condicionada.
Finalmente, evaluamos el *error de reconstrucción*, es decir, qué tan bien la factorización de Cholesky aproxima a la matriz original. Esto se mide mediante

$$\frac{\|LL^T - B\|}{\|B\|}, \quad \frac{\|L_\varepsilon L_\varepsilon^T - B_\varepsilon\|}{\|B_\varepsilon\|}.$$

Table 3: Errores de reconstrucción de la factorización de Cholesky

Caso	$\|LL^T - B\|/\|B\|$	$\|L_\varepsilon L_\varepsilon^T - B_\varepsilon\|/\|B_\varepsilon\|$
Bien condicionado	2.45×10^{-1}	2.45×10^{-1}
Mal condicionado	4.79×10^{-1}	4.79×10^{-1}

Estos errores son relativamente grandes en ambos casos. Esto puede explicarse porque nuestra implementación del algoritmo de Cholesky no está optimizada para estabilidad numérica, lo cual genera pérdidas de precisión acumuladas en la factorización.

1.3 Inciso b) Resultados

Enfocándonos en el caso mal condicionado, comparamos el desempeño de nuestra implementación con el algoritmo de `scipy.linalg.cholesky`, el cual es altamente optimizado y numéricamente estable.

Table 4: Errores de reconstrucción de la factorización de Cholesky en el caso de una matriz mal condicionada.

Algoritmo usado	$\|LL^T - B\|/\|B\|$	$\|L_\varepsilon L_\varepsilon^T - B_\varepsilon\|/\|B_\varepsilon\|$
Propio	4.79×10^{-1}	4.79×10^{-1}
SciPy	2.0×10^{-16}	1.93×10^{-16}

Podemos notar que el algoritmo de SciPy logra errores de reconstrucción cercanos a la precisión de máquina, mientras que nuestra implementación presenta errores de orden 10^{-1}. Esto evidencia que, aunque el algoritmo de Cholesky es matemáticamente exacto, su implementación práctica puede diferir mucho en estabilidad y precisión numérica. SciPy, es mucho más confiable para aplicaciones numéricas reales.

1.4 Inciso c)

El tiempo de ejecución entre nuestro algoritmo y el de `scipy.linalg.cholesky`, mostrado en la Figura 1, presenta diferencias significativas. En particular, el método de `SciPy` resulta considerablemente más rápido, lo que confirma el alto nivel de optimización de su implementación en comparación con la nuestra.

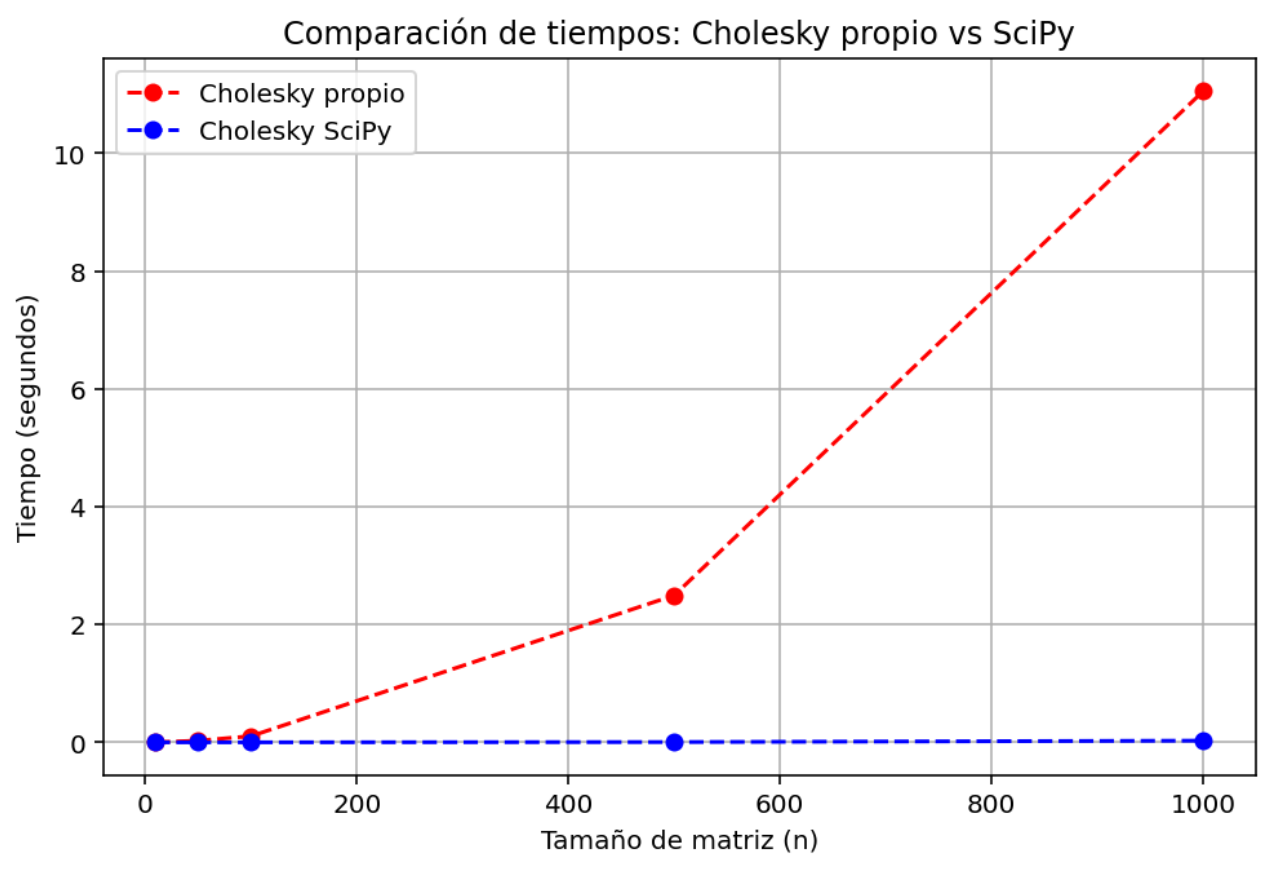

Figure 1: Comparación de los tiempos de ejecución de ambos algoritmos de factorización de Cholesky.

Los resultados de implementación demuestran que:

1. Las matrices mal condicionadas amplifican significativamente los errores de perturbación en comparación con las matrices bien condicionadas.
2. El número de condición $\kappa(A)$ predice efectivamente la sensibilidad numérica de las operaciones matriciales.
3. La descomposición de Cholesky mostró mayor sensibilidad a perturbaciones en matrices mal condicionadas, con errores de reconstrucción varios órdenes de magnitud mayores.

Ejercicio 2

En el contexto del modelo de regresión lineal

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

donde:

- $\mathbf{y} \in \mathbb{R}^m$: vector de observaciones
- $X \in \mathbb{R}^{m \times n}$: matriz de diseño con $m > n$ y rango$(X) = n$
- $\boldsymbol{\beta} \in \mathbb{R}^n$: vector de parámetros desconocidos
- $\boldsymbol{\varepsilon} \in \mathbb{R}^m$: vector de errores aleatorios

se busca resolver el problema de mínimos cuadrados mediante la descomposición QR, comparando su estabilidad numérica frente a matrices bien y mal condicionadas.
Esto lo haremos para $d = 5$, $n = 20$, $\boldsymbol{\beta} = (5, 4, 3, 2, 1)^\top$ y $\sigma = 0.12$, implementando lo siguiente:

1. **Generación de datos**: Construir matrices X con entradas $\mathcal{U}(0, 1)$ y simular $\mathbf{y}$ según el modelo lineal
2. **Estimación por QR**: Calcular $\hat{\boldsymbol{\beta}}$ usando la descomposición QR implementada
3. **Análisis de sensibilidad**: Perturbar la matriz de diseño con $\Delta X \sim \mathcal{N}(0, 0.01)$ y recalcular $\hat{\boldsymbol{\beta}}_p$
4. **Comparación con método clásico**: Obtener

   $$\hat{\boldsymbol{\beta}}_c = ((X + \Delta X)^\top (X + \Delta X))^{-1} (X + \Delta X)^\top \mathbf{y}$$

   usando `scipy.linalg.inv`

El objetivo principal es evaluar cómo el condicionamiento de la matriz de diseño afecta la estabilidad de las estimaciones y comparar el desempeño del algoritmo QR personalizado frente a métodos estándar de inversión matricial.

2 Resultados

2.1 Matriz Bien Condicionada

Método	**Número de Condición**	**Error Relativo**	**Coeficientes $\hat{\beta}$**
QR (X)	5.21878	1.52%	$(4.97, 3.94, 3.00, 1.95, 1.08)^\top$
QR $(X + \Delta X)$	5.22384	0.57%	$(5.00, 4.00, 3.01, 1.96, 1.02)^\top$
SciPy $(X + \Delta X)$	5.22384	0.57%	$(5.00, 4.00, 3.01, 1.96, 1.02)^\top$

Table 5: Resultados para matriz bien condicionada. $\beta_{\text{verdadero}} = (5, 4, 3, 2, 1)^\top$

Como podemos observar en la Tabla 5, el número de condición para la matriz X bien condicionada es del orden de 1, $\kappa(X) \approx 5$, lo que indica un buen condicionamiento. Es por eso que al perturbar la matriz $(X + \Delta X)$, su número de condición prácticamente se mantiene estable ($\kappa \approx 5.22$). Además, el error relativo disminuye ligeramente al aplicar la perturbación, demostrando la estabilidad numérica del método para matrices bien condicionadas.

Método	**Número de Condición**	**Error Relativo**	**Coeficientes $\hat{\beta}$**
QR (X)	21.365	11.15%	$(5.63, 3.51, 2.83, 2.14, 0.98)^\top$
QR $(X + \Delta X)$	22.336	14.55%	$(5.81, 3.34, 2.78, 2.16, 0.99)^\top$
SciPy $(X + \Delta X)$	22.336	14.55%	$(5.81, 3.34, 2.78, 2.16, 0.99)^\top$

Table 6: Resultados para matriz mal condicionada. $\beta_{\text{verdadero}} = (5, 4, 3, 2, 1)^\top$

Por el contrario, cuando la matriz presenta mal condicionamiento, como se observa en la Tabla 6 con $\kappa(X) \approx 21.37$, la situación es diferente. Aunque este valor de κ no es extremadamente alto, sí representa un condicionamiento deficiente. Al perturbar la matriz ($\kappa(X + \Delta X) \approx 22.34$), el error relativo aumenta de 11.15% a 14.55%, evidenciando la sensibilidad del problema a perturbaciones cuando la matriz no está bien condicionada. Este comportamiento confirma que a mayor número de condición, mayor es la amplificación de los errores en la solución.
En base a los resultados, hay evidencia para afirmar que:

- Para matrices **bien condicionadas** (*Orden de $k(X)$ cercano a 1*), todos los métodos producen estimaciones cercanas al vector $\boldsymbol{\beta}$ verdadero, con errores relativos menores al 2%.
- Para matrices **mal condicionadas** (*Orden de $k(X)$ mayor a 1*), los errores se amplifican significativamente.

La descomposición QR demuestra ser un método estable para la solución de problemas de mínimos cuadrados con matrices bien condicionadas. Sin embargo, cuando la matriz de diseño presenta mal condicionamiento, pequeños cambios en los datos pueden producir grandes variaciones en las estimaciones, evidenciando la importancia del análisis de sensibilidad en problemas de regresión lineal.

Notemos que la coincidencia numérica entre los resultados obtenidos mediante la descomposición QR personalizada y la fórmula clásica con inversión matricial se explica por la equivalencia matemática fundamental:

$$\hat{\beta}_{\text{QR}} = R^{-1}Q^T y \quad \text{y} \quad \hat{\beta}_{\text{clásico}} = (X^T X)^{-1} X^T y$$

donde al sustituir la factorización $X = QR$ en la fórmula clásica:

$$\hat{\beta}_{\text{clásico}} = ((QR)^T(QR))^{-1}(QR)^T y = (R^T Q^T QR)^{-1} R^T Q^T y = (R^T R)^{-1} R^T Q^T y = R^{-1} Q^T y = \hat{\beta}_{\text{QR}}$$

Esta identidad matemática garantiza que ambos métodos deben producir resultados idénticos,la concordancia observada en nuestros experimentos valida la correcta implementación de los algoritmos de descomposición QR y sustitución backward, sin embargo aunque ambos métodos son matemáticamente equivalentes, la descomposición QR es numéricamente más estable para matrices mal condicionadas, ya que evita la formación explícita de $X^T X$.