Explicación del código:

1. `generate_matrix_with_condition_number(n, cond)`:
   - Genera dos matrices ortogonales aleatorias U y V.
   - Crea un vector de n valores singulares espaciados logarítmicamente entre 1 y `cond`. Esto asegura que la razón entre el valor singular más grande y el más pequeño sea aproximadamente `cond`.
   - Construye la matriz A utilizando la descomposición en valores singulares: $A = USV^T$, donde S es una matriz diagonal con los valores singulares.
2. `exact_line_search_step(A, b, x_k, grad_fk)`:
   - Implementa la fórmula para el tamaño del paso óptimo α_k en el método de descenso por gradiente para problemas de mínimos cuadrados. La función objetivo es $f(x) = \|Ax - b\|^2 = (Ax - b)^T(Ax - b)$.
   - El gradiente de $f(x)$ es $\nabla f(x) = 2A^T(Ax - b)$.
   - La dirección de descenso es $d_k = -\nabla f(x_k) = -2A^T(Ax_k - b)$. Sin embargo, la constante 2 no afecta el tamaño del paso óptimo, por lo que usamos $d_k = A^T(b - Ax_k)$ o equivalentemente $-\nabla f(x_k)/2$. En el código, se usa el gradiente $\nabla f(x_k) = A^T(Ax_k - b)$.
   - El tamaño del paso óptimo α_k se encuentra minimizando $f(x_k - \alpha d_k)$ con respecto a α. Esto lleva a la fórmula:

$$\alpha_k = \frac{d_k^T d_k}{d_k^T A^T A d_k} = \frac{(A^T r_k)^T (A^T r_k)}{(A^T r_k)^T A^T A (A^T r_k)} = \frac{r_k^T A A^T A r_k}{r_k^T A A^T A A^T A r_k}$$

donde $r_k = b - Ax_k$ es el residuo. **Corrección:** La fórmula correcta para el tamaño del paso con el gradiente $\nabla f(x_k) = A^T(Ax_k - b)$ es:

$$\alpha_k = \frac{(A^T(Ax_k - b))^T(A^T(Ax_k - b))}{(A^T(Ax_k - b))^T A^T A(A^T(Ax_k - b))} = \frac{\|A^T r_k\|^2}{\|A(A^T r_k)\|^2}$$

donde $r_k = b - Ax_k$. En el código se utiliza $grad\_fk = A^T(Ax_k - b)$, por lo que la fórmula implementada es correcta para ese gradiente.

3. `gradient_descent_exact_linesearch(A, b, x_0, max_iter=1000, tol=1e-6)`:
   - Implementa el algoritmo de descenso por gradiente.
   - Inicializa el punto de partida $x_k = x_0$.
   - Almacena la historia del error cuadrático (norma del residuo al cuadrado) en cada iteración.
   - En cada iteración:
     - Calcula el residuo $r_k = Ax_k - b$.
     - Calcula el gradiente $\nabla f(x_k) = A^T r_k$.
     - Determina el tamaño del paso óptimo α_k usando `exact_line_search_step`.
     - Actualiza la solución $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$.
     - Verifica la condición de convergencia (si el cambio en x es menor que la tolerancia)

- Verifica la condición de convergencia (si el cambio en x es menor que la tolerancia).

4. `pseudo_inverse_solution(A, b)`:
   - Calcula la pseudo-inversa de A usando `la.pinv(A)`.
   - Obtiene la solución $x_{pinv} = A^{+}b$.
   - Calcula el error cuadrático para esta solución.
5. `study_convergence(n=100, condition_numbers=[10, 100, 1000, 10000], num_runs=5)`:
   - Esta función experimenta con diferentes números de condición para la matriz A.
   - Para cada número de condición:
     - Genera múltiples matrices A y vectores b aleatorios.
     - Ejecuta el descenso por gradiente y calcula la solución con la pseudo-inversa.
     - Almacena la historia de la convergencia (norma del residuo al cuadrado).
   - Grafica la norma del residuo al cuadrado promedio en función del número de iteraciones para diferentes números de condición.
   - Imprime los residuales finales promedio obtenidos por el descenso por gradiente y el residual obtenido por la pseudo-inversa para cada número de condición.
6. **Bloque** `if __name__ == "__main__":`:
   - Define el tamaño del problema (n) y un número de condición de ejemplo.
   - Genera una matriz A y un vector b.
   - Resuelve el problema de mínimos cuadrados utilizando ambos métodos y muestra los resultados.
   - Llama a la función `study_convergence` para analizar el efecto del número de condición en la convergencia.

Cómo interpretar los resultados:

- **Comparación con la pseudo-inversa:** La solución obtenida por el descenso por gradiente debería converger a la misma solución que la obtenida por la pseudo-inversa (en ausencia de errores numéricos y con suficientes iteraciones). El residual final del descenso por gradiente debería acercarse al residual de la pseudo-inversa.
- **Convergencia y número de condición:**
  - Se espera que la convergencia del descenso por gradiente sea más rápida para matrices con un número de condición bajo (cercano a 1).
  - A medida que el número de condición de A aumenta, el problema se vuelve más "mal condicionado", y el descenso por gradiente tenderá a converger más lentamente, mostrando una trayectoria en "zig-zag" hacia la solución. Esto se debe a que la función objetivo tendrá contornos elípticos muy alargados.
  - La gráfica generada por `study_convergence` debería ilustrar esta relación, mostrando curvas de convergencia más lentas para números de condición más grandes.
  - Los residuales finales obtenidos por el descenso por gradiente también podrían ser ligeramente mayores para matrices mal condicionadas con un número finito de iteraciones.

Ejecuta este código y observa los resultados y la gráfica generada para comprender cómo el número de condición de la matriz A afecta la eficiencia del algoritmo de descenso por gradiente con *exact line search* al resolver problemas de mínimos cuadrados.