**1. ¿Qué representa lambda\_val = h² / k²? ¿Por qué se define así?**

**Respuesta esperada**:  
Es el cociente entre los pasos de malla en las direcciones x y y, elevado al cuadrado. Es una constante que aparece al discretizar la ecuación de Poisson con mallas rectangulares no uniformes. Permite balancear el peso de los términos ∂²u/∂x² y ∂²u/∂y² en la ecuación.

**2. ¿Qué pasaría si usas una malla no uniforme, es decir, si h y k cambian con i y j?**

**Respuesta esperada**:  
El método dejaría de ser directamente aplicable en su forma actual. Requerirías una versión adaptativa del algoritmo que calcule h(i) y k(j) en cada punto. Además, las derivadas segundas se deben aproximar con fórmulas más complejas para mallas no uniformes.

**3. ¿Por qué usamos Gauss-Seidel y no Jacobi?**

**Respuesta esperada**:  
Gauss-Seidel es un método iterativo que usa inmediatamente los valores actualizados en cada iteración, lo que acelera la convergencia respecto a Jacobi. Es especialmente útil en ecuaciones elípticas como la de Poisson.

**4. ¿Qué criterio asegura la convergencia del método?**

**Respuesta esperada**:  
Convergencia está garantizada para matrices diagonales dominantes o simétricas positivas definidas. En este caso, la matriz generada por discretización de Poisson con condiciones de Dirichlet es SPD(simétrica positiva dominante). Además, el método detiene las iteraciones cuando NORM ≤ tol.

**5. ¿Cuál es la complejidad computacional del método?**

**Respuesta esperada**:  
Por iteración es O(n·m), ya que se actualizan todos los nodos internos. El número total de iteraciones depende de tol y del comportamiento espectral del sistema, aunque se puede acotar por N\_max.

**6. ¿Qué representa w[i, j] en la malla? ¿Qué contiene full\_w?**

**Respuesta esperada**:  
w[i, j] contiene la aproximación de la solución u(x,y) en los nodos internos, excluyendo fronteras. full\_w es la malla completa con condiciones de contorno incorporadas, de tamaño (n+1)x(m+1).

**7. ¿Qué impacto tiene cambiar tol o N\_max?**

* Reducir tol → Mejora la precisión pero aumenta las iteraciones y el tiempo.
* Reducir N\_max → Puede cortar el proceso antes de converger (falla del método).

**8. ¿Por qué rellenas full\_w después de la convergencia?**

* **Respuesta esperada**:  
  Porque w contiene solo los valores internos de la solución. Para graficar o comparar con la solución analítica, se necesita reconstruir la malla completa incluyendo las condiciones de borde.

**9. ¿Qué tipo de error estamos cometiendo al usar diferencias finitas?**

* **Respuesta esperada**:  
  Error de truncamiento debido a la aproximación de derivadas por diferencias. En este caso es de orden O(h² + k²), típico de métodos con esquema de 5 puntos.

**10. ¿Qué pasa si uso np.ones(...) en vez de np.zeros(...) en la inicialización?**

* **Respuesta esperada**:  
  Cambiaría el punto de partida de la iteración, pero Gauss-Seidel debería converger si el sistema es bien condicionado. Sin embargo, puede requerir más iteraciones o generar valores intermedios inestables.

**11. ¿Qué pasa si intercambio x e y en la implementación?**

* **Respuesta esperada**:  
  El resultado debería ser el mismo en teoría si el dominio es simétrico. Pero en la práctica, puede alterar el orden en que se actualizan los valores y afectar la tasa de convergencia por cuestiones de cache y acceso a memoria.

**12. ¿Qué ventaja tendría usar NumPy vectorizado en vez de for-loops?**

* **Respuesta esperada**:  
  Podríamos aprovechar operaciones de álgebra lineal optimizadas en C para acelerar la iteración. Sin embargo, dado que Gauss-Seidel depende del orden secuencial de actualización, no se puede vectorizar fácilmente sin reestructurar el método.

**13. ¿Por qué usamos cos(x + y) + cos(x - y) y no directamente 2cos(x)cos(y)?**

* **Respuesta esperada**:  
  Porque el problema original está formulado con esa suma. Aunque se puede simplificar analíticamente, para fines numéricos seguimos la formulación dada, pues los errores de redondeo o aliasing pueden comportarse distinto en cada forma.