Optimización de Mallas usando NSGA-II

Juliana Restrepo Tobar
Andrea Carvajal Maldonado

November 25, 2021

1 Introducción

El objetivo general de la generación de mallas es descomponer una figura geométrica en elementos. Los elementos son restringidos en tipo y forma, y el número de elementos que se utilizan no debe de ser muy grande[Ede01]. Las mallas pueden ser aplicadas para resolver ecuaciones en derivadas parciales en el método de elementos finitos, así como para la renderización de imágenes.
Para los fines de este trabajo, los elementos que componen las mallas serán triángulos. En particular, estos se hallarán por medio de la triangulación de Delaunay, que se describirá más adelante.
Se hará uso de la herramienta de la optimización multiobjetivo NSGA-II para generar mallas con mínima desviación estándar entre las áreas de sus triángulos y mínimo error absoluto en la aproximación al área real. Las diferentes soluciones o salidas del modelo pueden generar "trade offs". Esto es que la malla óptima para un objetivo puede no serlo para el resto. Por esto, no existe una sola solución óptima, sino más bien un conjunto de soluciones Pareto eficientes (soluciones no dominadas por otras).

2 Planteamiento del problema

Sea la función

$$f = \frac{xy(x^2 - y^2)}{x^2 + y^2 + 1} \quad (x, y) \in [0, 3] \times [0, 3]$$

El presente trabajo tiene como fin encontrar mallas que se adapten a la función f con la siguientes condiciones:
- Mayor similitud en las áreas de los triángulos que la componen, es decir, se minimiza la desviación estándar, dado por:

$$\sum_{i=1}^{m} \frac{(a(i) - \bar{a})^2}{m},$$

donde $a(i)$ es el área del triángulo i, $\bar{a}$ es la media de las áreas, y m es el número de triángulos.
- Mejor aproximación a la función real, es decir que se minimiza el siguiente error absoluto:

$$|A - \hat{A}|$$

donde A es el área real de la función, calculada mediante la integral de superficie: $\int_0^3 \int_0^3 (f_x + f_y + 1) dxdy$; y $\hat{A}$ es el área de la malla, calculada mediante la suma de las áreas de todos los triángulos que la componen, que se obtienen por: $\frac{||\mathbf{u} \times \mathbf{v}||}{2}$.

3 Teoría y herramientas computacionales

Como se mencionó anteriormente, se usará la triangulación de Delaunay para la resolución de este problema. Esta retorna una malla de triángulos convexa y conexa, que además cumple con la condición de Delaunay, es decir, que la circunferencia circunscrita de cada triángulo no contenga ningún vértice de otro triángulo [Ede01]. Para hallar esta triangulación se usó el módulo de Scipy en Python, que

contiene un método que realiza el procedimiento.
Asimismo, se hizo uso de NSGA-II descrito por primera vez por *Deb et al* en el 2002 [DPAM02].
Este algoritmo consta de diversas etapas. Primero se inicializa la población inicial de tamaño n con valores arbitrarios o escogidos por el decisor. Luego empieza la etapa del crossover, que consiste en tomar pares de cromosomas y generar hijos a partir de ellos. Una vez se hayan generado n hijos, este proceso se detiene. Ahora, se juntan los n hijos con los n padres para crear una población de tamaño 2n. En algunas ocasiones, se hace un proceso de mutación. Este consiste en cambiar aleatoriamente un gen del cromosoma y se realiza con una probabilidad pequeña de $\frac{1}{t}$, donde t es el tamaño del cromosoma. Después, se evalúa el desempeño de cada individuo según las funciones objetivo para calcular su ranking de no dominancia. El ranking de no dominancia indica cuántas soluciones dominan a la solución en cuestión, es decir que si su ranking es cero, la solución es no dominada. Por lo tanto, no existe otra solución que no empeore en ningún aspecto y que sea mejor en al menos uno de ellos.
Adicionalmente, se calcula la distancia de apilamiento d de las soluciones para mantener la diversidad de estas. Para esto se inicia calculando una distancia para cada función objetivo, asignándole una distancia infinita a las soluciones con mejor y peor desempeño, y calculando el resto de distancias de la siguiente manera:

$$d_i = d_i + \frac{m_{i+1} - m_{i-1}}{max(m) - min(m)}$$

donde d_i es la distancia del individuo i para la función objetivo m, max(m) y min(m) son las soluciones máximas y mínimas para la función m, y m_{i-1}, m_{i+1} son la solución anterior y posterior a la solución i con respecto al objetivo m. Al final, se suman las distancias de un mismo individuo para cada objetivo y este resultado es la distancia de apilamiento.
Para terminar, se ejecuta un torneo de selección binario usando la distancia de apilamiento y el ranking de no dominancia. En el torneo se toman dos individuos de manera aleatoria y se comparan sus rankings, si el individuo 1 tiene menor ranking que el individuo 2, se elige el individuo 1 para ser padre en la próxima generación. Análogamente, en el caso contrario. Sin embargo, si ambos rankings son iguales, se elige el individuo con mayor distancia de apilamiento.
Estos pasos se pueden efectuar cuantas veces considere el decisor, tomando en cuenta que cada iteración es una nueva generación.

El desarrollo práctico del proyecto se realizó en el lenguaje de programación Python, haciendo uso de las librerías Scipy, Numpy y Pygmo para hacer los procedimientos matemáticos requeridos y las librerías Matplotlib y Plotly para graficar.

4 Resolución del problema

Para empezar se construyó una red rectangular con n nodos equidistantes de la forma $(x, y) \in [0, 3] \times [0, 3]$ que servirá como referencia. Dicha red se trianguló aplicando la triangulación de Delaunay.

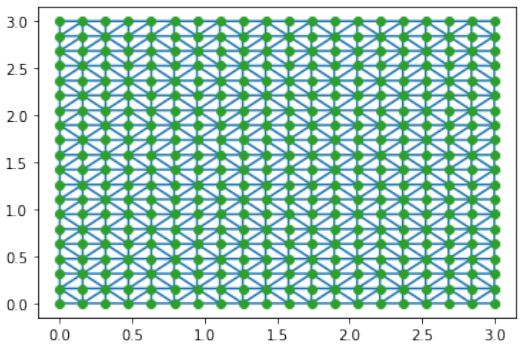

Figure 1: Ejemplo de red de referencia de 20x20 nodos

Se calculó el área de de la función real por medio de una integral de superficie planteada de la

siguiente manera:

$$\int_0^3 \int_0^3 (f_x + f_y + 1) dx dy;$$

y se calculó el área de los triángulos de la malla a partir de los coordenadas de sus vértices, así:

$$\frac{|u \times v|}{2}$$

donde $u = (x2 - x1, y2 - y1, z2 - z1)$ y $v = (x3 - x1, y3 - y1, z3 - z1)$.
Se definió una población inicial de 100 redes a partir de la red de referencia. En este caso los individuos representan conjuntos de nodos que conforman la red.
Se decidió realizar una codificación binaria, es decir, representar los cromosomas como cadenas de 1s y 0s [CÁCD13]. En el contexto particular del proyecto, los 1s simbolizan que el nodo (x, y) está en la red y 0 simboliza lo contrario. Es importante notar que se determinó que los nodos de las esquinas siempre forman parte del cromosoma para evitar errores.
Se aplica el algoritmo NSGA-II con todas sus fases. Para el caso del crossover, se utiliza el crossover uniforme, y para la mutación se cambia un gen que tenga el valor de 1 (y que no coincida con una esquina) por 0, y viceversa. Para saber si una red es una buena solución, se evalúa cada nodo en la función f y se construye la malla para calcular su respectivo error absoluto y desviación estándar.
Finalmente, cuando el algoritmo devuelve las soluciones, estas se filtran para eliminar las soluciones repetidas.
El código del trabajo se encuentra en:
https://github.com/jrestrepot/OptimizacionII/blob/main/MOLP/Mesh%20Optimization/MESH%20SIMPLIFICATION%20WITH%20NSGAII.py

En la próxima sección se pueden observar las soluciones obtenidas corriendo el algoritmo por 10 generaciones.

4.1 Resultados

4.2 Red 5x5

Para comenzar se tiene la siguiente red uniforme de referencia con 5x5 nodos con su respectiva malla:

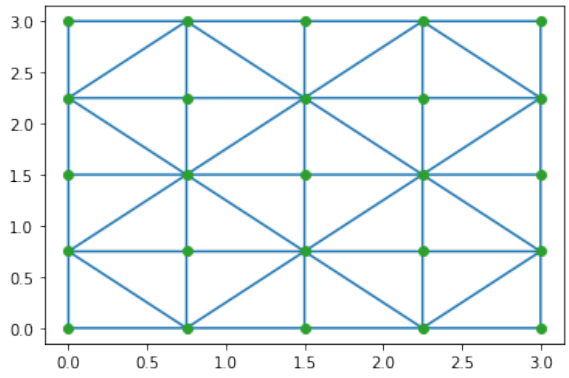

Figure 2: Red de referencia 5x5

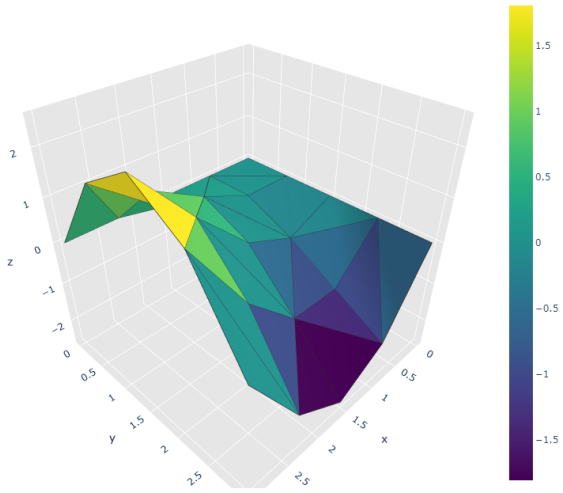

Figure 3: Malla a partir de la red de referencia

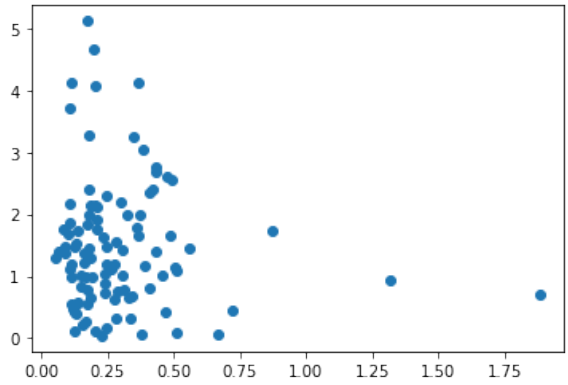

Figure 4: Diagrama de dispersión población inicial

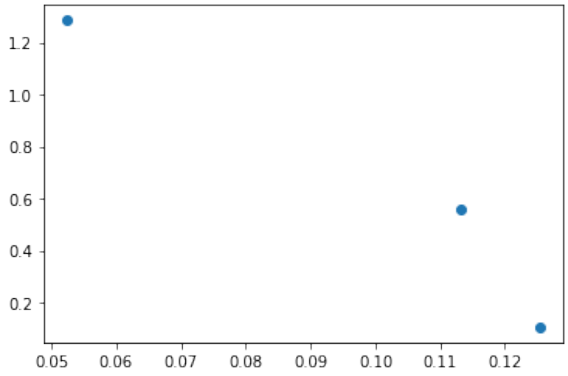

Figure 5: Diagrama de dispersión población final

El eje x de ambas imágenes representa la desviación estándar del área de los triángulos, y el eje y representa el error absoluto. El diagrama de dispersión de la figura 4 muestra las soluciones de los 100 individuos de la población inicial, mientras que el diagrama de la derecha representa las soluciones después de 10 generaciones.
Se observa que las desviaciones y errores siguen siendo altos, aunque se mejoraron muchísimo con respecto a aquellos de la población inicial.
El lector podría asumir que el algoritmo NSGA-II está devolviendo menos individuos que los iniciales, pero en realidad devuelve los 100 individuos, sólo que los cromosomas de varios son idénticos, por lo que en realidad solo se obtienen las 3 soluciones que se aprecian en la figura 5.
Las representación gráfica de las soluciones es la siguiente:

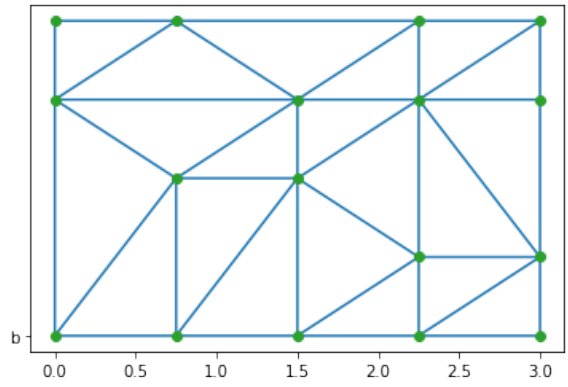

Figure 6: Red compuesta de 17 nodos

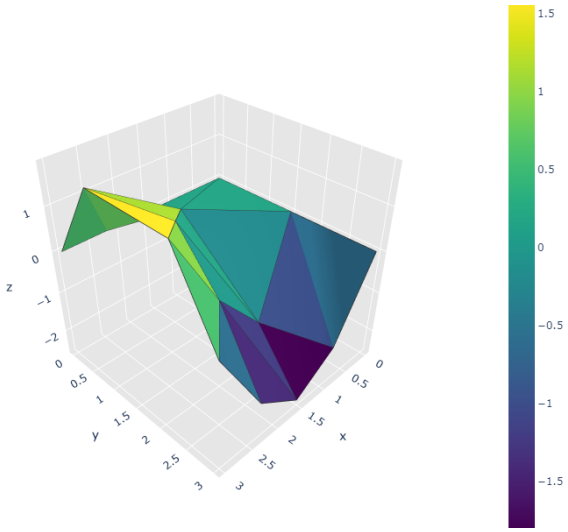

Figure 7: Malla de f

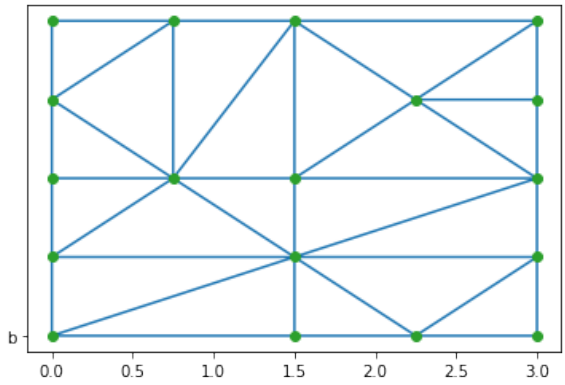

Figure 8: Red compuesta de 18 nodos

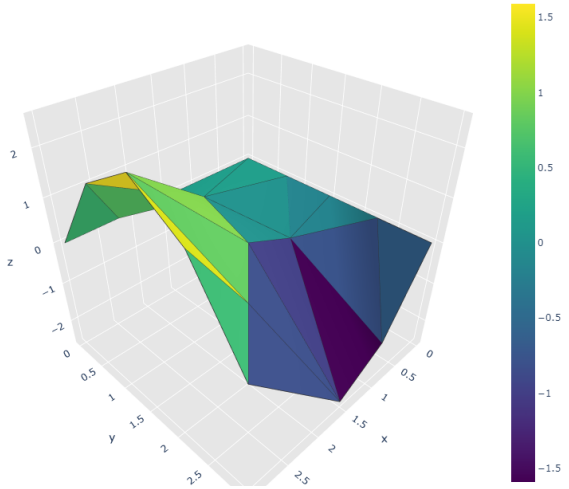

Figure 9: Malla de f

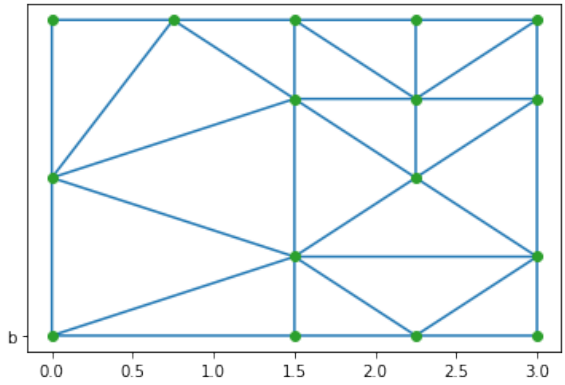

Figure 10: Red compuesta de 16 nodos

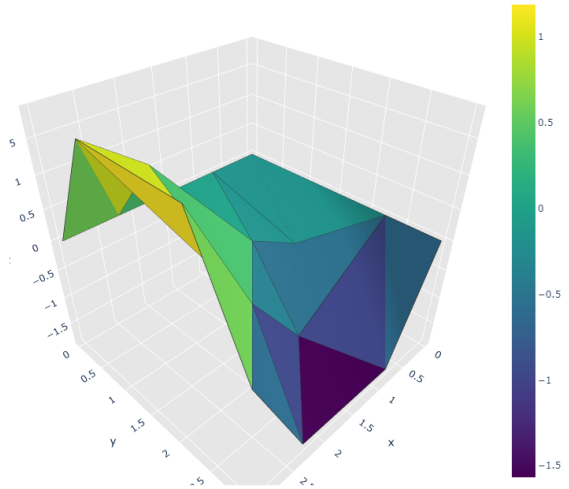

Figure 11: Malla de f

En todas las gráficas anteriores se ven bordes bruscos y esquinas muy rectas, lo que puede indicar que no es una buena aproximación a la función f.

4.3 Red 20x20

Ahora se tiene la siguiente red uniforme de referencia con 20x20 nodos con su respectiva malla:

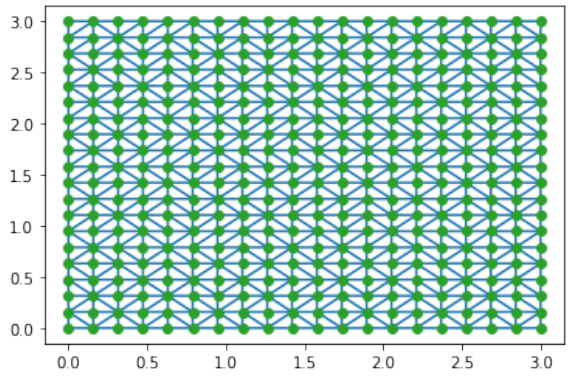

Figure 12: Red de referencia 20x20

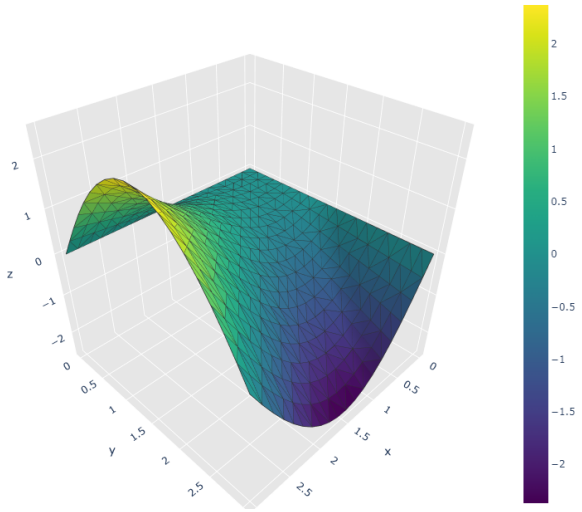

Figure 13: Malla a partir de la red de referencia

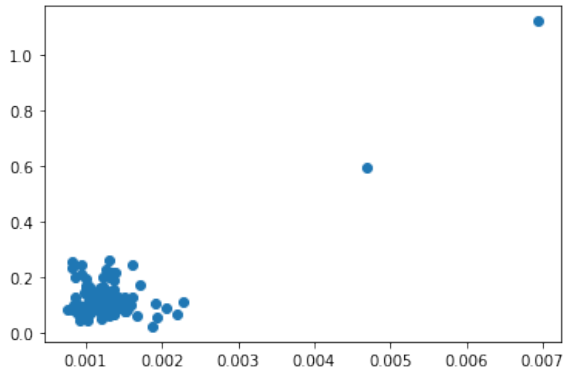

Figure 14: Diagrama de dispersión población inicial

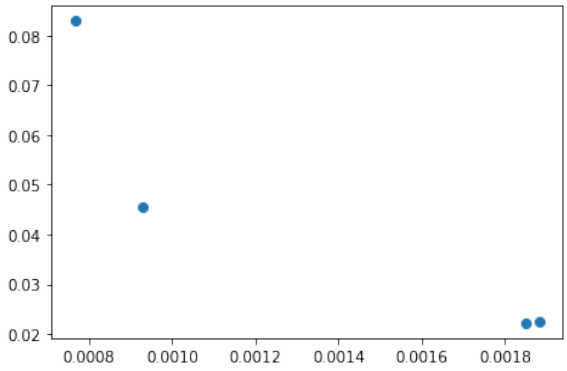

Figure 15: Diagrama de dispersión población final

El eje x de ambas imágenes representa la desviación estándar del área de los triángulos, y el eje y representa el error absoluto. El diagrama de dispersión de la figura 14 muestra las soluciones de los 100 individuos de la población inicial, mientras que la figura 15 representa las soluciones después de 10 generaciones.
Se evidencian grandes mejoras a los errores y desviaciones de la población inicial, y ya estos valores empiezan a ser aceptables.
Las representación gráfica de las soluciones es la siguiente:

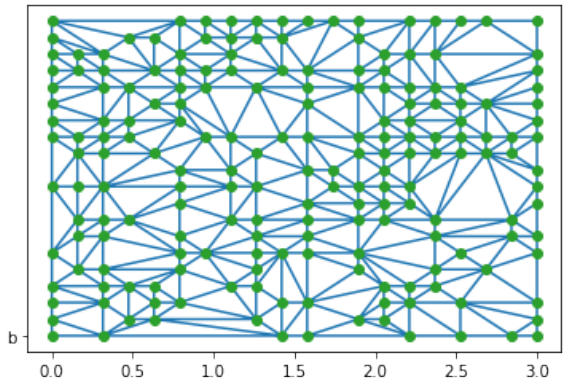

Figure 16: Red compuesta de 191 nodos

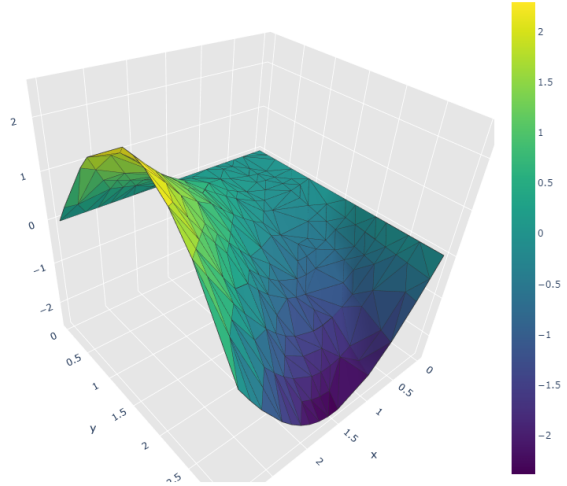

Figure 17: Malla de f

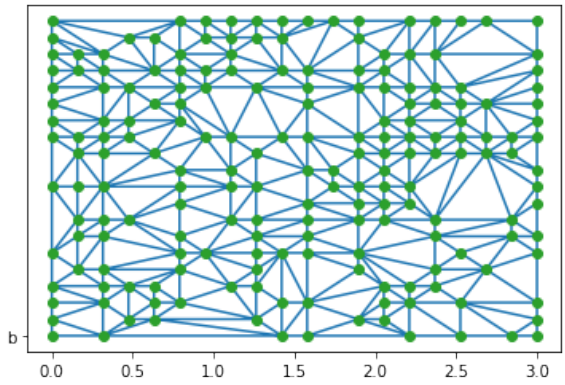

Figure 18: Red compuesta de 212 nodos

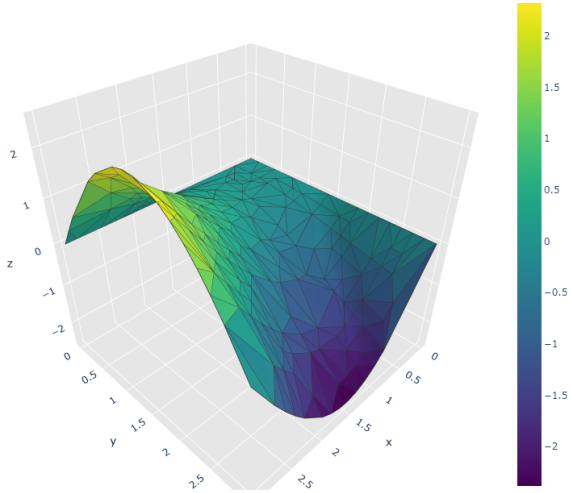

Figure 19: Malla de f

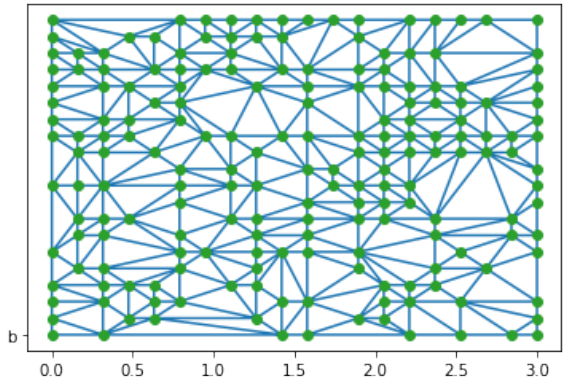

Figure 20: Red compuesta de 190 nodos

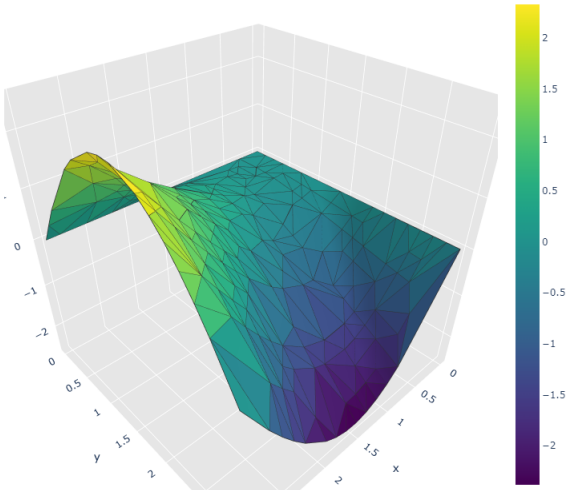

Figure 21: Malla de f

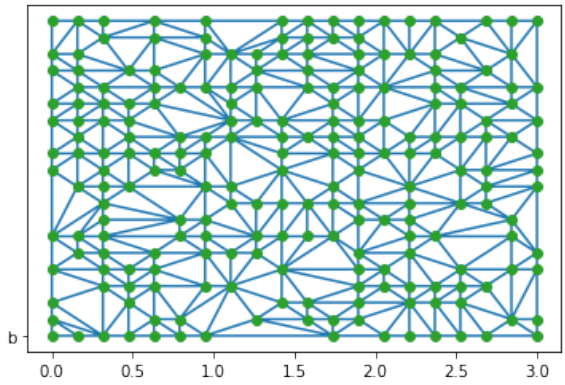

Figure 22: Red compuesta de 217 nodos

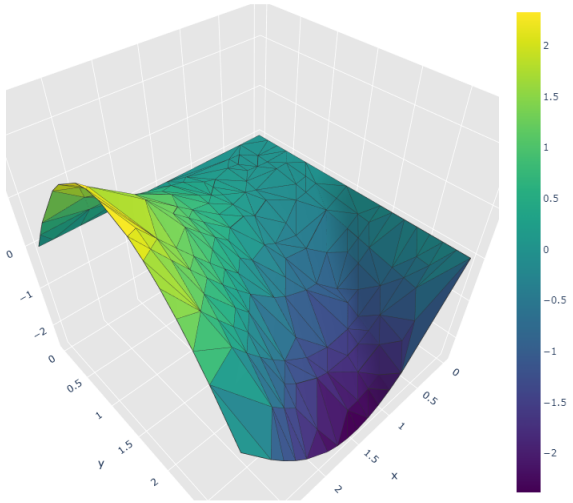

Figure 23: Malla de f

Las gráficas se ven mucho más suaves que las de las mallas construidas a partir de la red 5x5 y los valores de los errores demuestran que estas son mejores aproximaciones.

5 Conclusiones

El algoritmo logra disminuir el número de nodos con respecto a la red de referencia, alcanzando una muy buena aproximación de la función con incluso la mitad de los nodos originales. Este resultado se alcanza sin sacrificar considerablemente el error y la desviación estándar, aunque cabe resaltar que la aproximación de la red original sí es más precisa que la de las simplificaciones. Es claro que entre más nodos tenga la red de referencia, mejor será la aproximación de sus simplificaciones.

References

[CÁCD13] B Rosario Campomanes-Álvarez, Oscar Cordon, and Sergio Damas. Evolutionary multi-objective optimization for mesh simplification of 3d open models. *Integrated Computer-Aided Engineering*, 20(4):375–390, 2013.

[DPAM02] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[Ede01] Herbert Edelsbrunne. *Geometry and Topology for Mesh Generation*. Cambridge, 2001.