Taller 3

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Integrantes:

Juan Pablo Sotelo Rativa – 20211020113

Andrés Felipe Salazar Malagón - 20202020043

Julián David Pérez Chaparro - 20192020017

Docente:

Duvan Andres Tellez Castro

Universidad Distrital Francisco José de Caldas

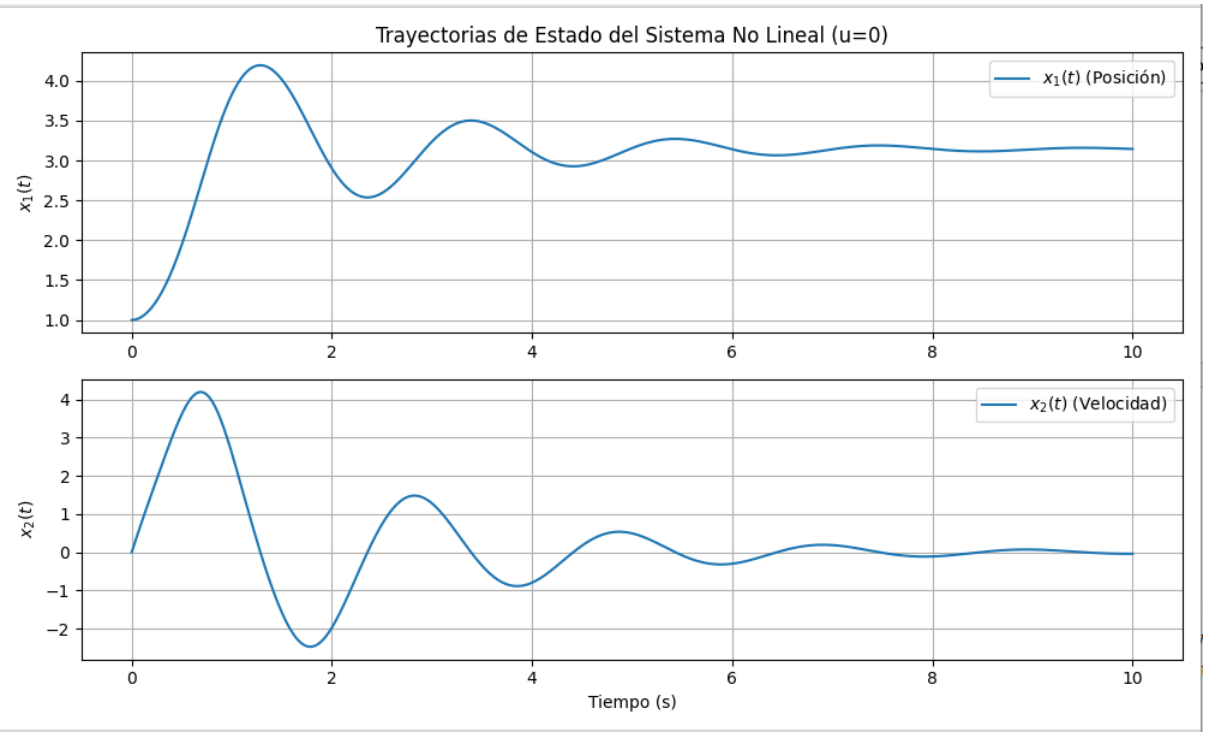
Facultad de Ingeniería

Cibernética III

2025

Punto 1:

Inicialmente usando python construimos el sistema para obtener las salidas x1 y x2 del sistema no lineal en u = 0, para comprobar que dichas trayectorias coincidan con las presentadas en el documento, obtuvimos lo siguiente:



Comparando con las trayectorias presentadas en el documento vemos que tienen exactamente el mismo comportamiento x1 y x2.

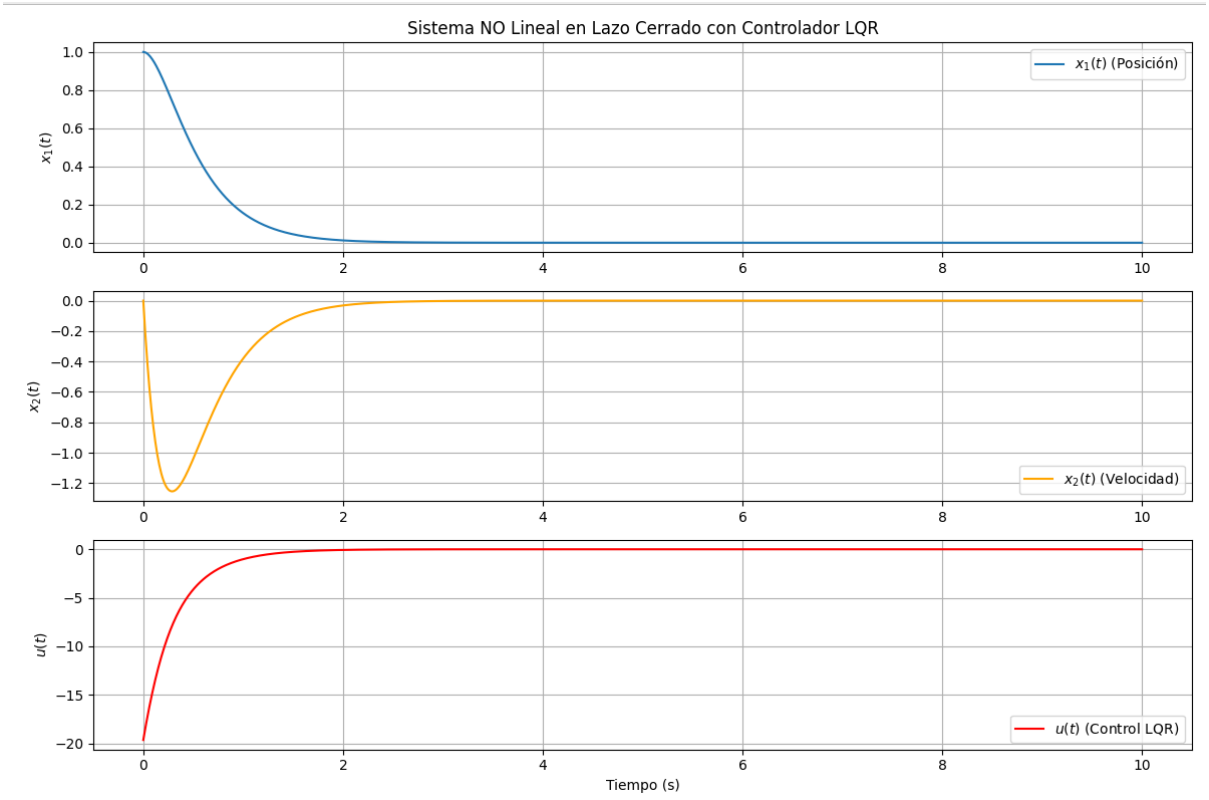
Para diseñar la ley de control óptima, se formuló un problema de regulación basado en un controlador LQR con las condiciones planteadas en el taller. Con las matrices dadas se resolvió la ecuación Algebraica de Riccati asociada al sistema linealizado obteniendo lo siguiente:

```
Matriz P (Solución de Riccati):
[[72.3370642  19.65088828]
 [19.65088828  5.34836802]]
```

Para después obtener la ganancia óptima K:

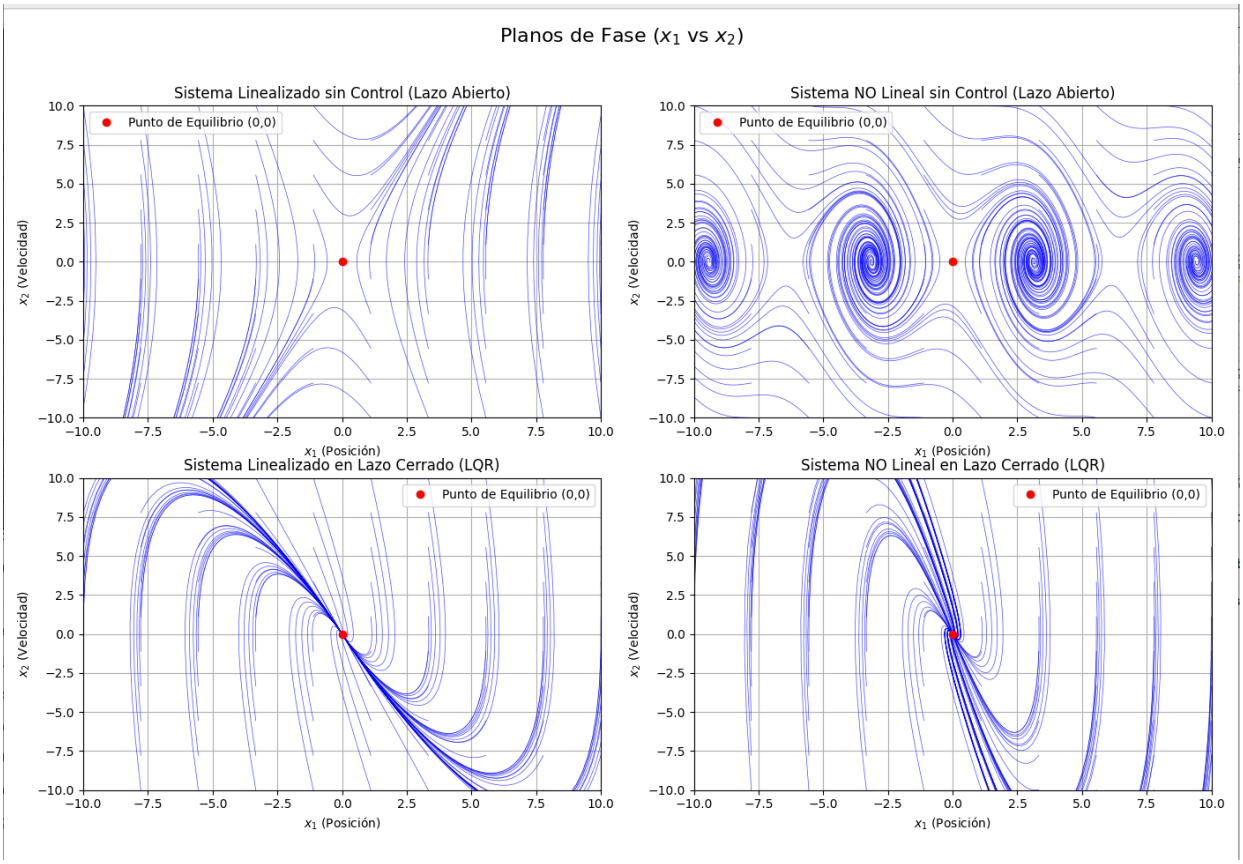
```
Matriz de Ganancia K:
[[19.65088828  5.34836802]]
```

Una vez determinada la ley de control óptima se simuló en lazo cerrado el sistema aplicándole el control, con las condiciones iniciales de x1(0) = 1 y x2(0) = 0 en 10 segundos obteniendo los siguientes resultados:



Como verificamos la dinámica del sistema bajo control converge hacia el punto de equilibrio.

Por último, se realizaron simulaciones tanto de lazo abierto y cerrado del sistema lineal y no lineal para analizar el comportamiento dinámico. Se generaron planos de fase (x1 vs x2) que permitieron visualizar la evolución de los estados a partir de diferentes condiciones iniciales obteniendo lo siguiente:



1. El sistema lineal en lazo abierto permite observar que el punto de equilibrio es inestable, las trayectorias divergen para las diferentes condiciones iniciales.
2. El sistema no lineal en lazo abierto muestra que las trayectorias tienen una naturaleza oscilatoria y no lineal, con comportamientos periódicos y divergentes por regiones.
3. El sistema linealizado en lazo cerrado muestra que todas las trayectorias convergen hacia el origen que es nuestro punto de equilibrio confirmando la estabilización.
4. El sistema no lineal en lazo cerrado igualmente se aproximó hacia el punto de equilibrio mostrando que el controlador diseñado fue capaz de estabilizar el sistema real en una región alrededor del origen.

Punto 2:

El objetivo es resolver un problema convexo cuadrático con una única restricción lineal. El enfoque principal utilizado fue el método primal–dual en tiempo continuo, y los métodos KKT y fmincon se aplicaron únicamente para validar la solución obtenida por el dinámico.

Método primal–dual

Se construye la Lagrangiana:

$$L(x, \lambda) = \frac{1}{2} x^T Q x + c^T x + \lambda (a^T x - 1)$$

y se plantean las dinámicas continuas:

$$\dot{x} = -\nabla_x L = -(Qx + c + \lambda a)$$

$$\dot{\lambda} = a^T x - 1.$$

El sistema completo evoluciona hasta que:

- $\dot{x}$=0,
- $\dot{\lambda}$=0,

lo cual corresponde exactamente a las ecuaciones KKT.
Se integra numéricamente durante un intervalo de tiempo hasta que los valores se estabilizan.

Resultados

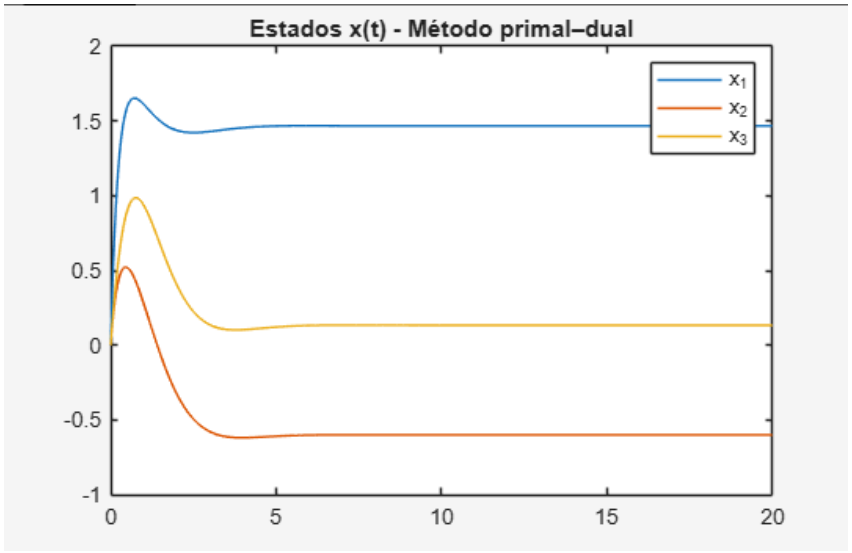
```
=== INTEGRACIÓN PRIMAL-DUAL ===
x* (primal-dual) =
    1.4667
   -0.6000
    0.1333

λ* (primal-dual) =
    2.7333
```

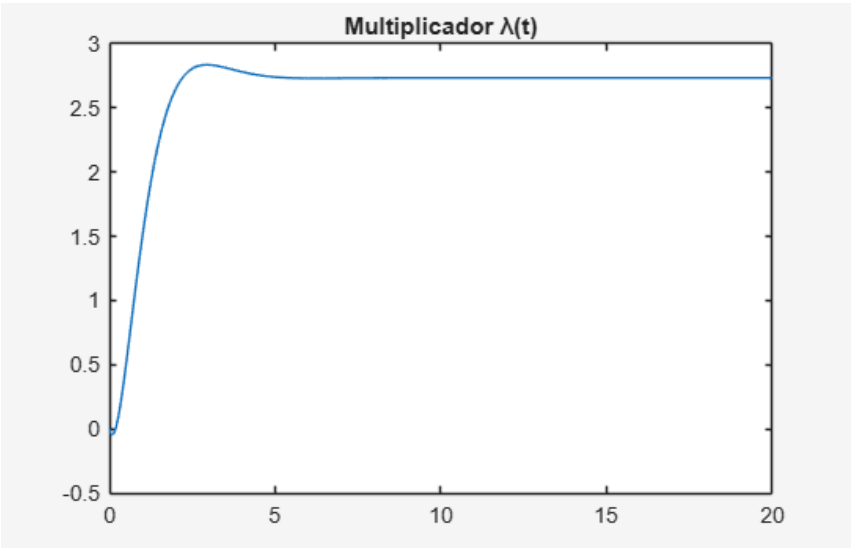
- Los valores finales de x∗ son el punto óptimo que minimiza la función objetivo bajo la restricción lineal.
- El valor de λ∗ es el multiplicador de Lagrange, indicando la "fuerza" con la que la restricción afecta la solución.

Las gráficas de *x(t)* y *λ(t)* muestran cómo:

- hay un pequeño sobresalto inicial (respuesta transitoria),
- el sistema se estabiliza suavemente,
- y converge rápidamente al equilibrio.

Esto es típico de los sistemas acoplados primal–dual y refleja que la dinámica está bien planteada.





Al ser convexo, solo existe un mínimo global, y el sistema primal–dual no tiene ambigüedad:siempre converge al mismo punto independientemente de la condición inicial.

Esto explica por qué las gráficas muestran una convergencia suave hacia el equilibrio.

Método KKT

El método KKT resuelve directamente las ecuaciones de optimalidad del problema.
Estas ecuaciones combinan dos condiciones:

Optimalidad interna:Esta condición dice que el gradiente del objetivo está balanceado por el efecto del multiplicador λ, que incorpora la restricción.

Satisfacción de la restricción:Resolver este sistema determina exactamente el único punto que simultáneamente minimiza la función cuadrática,y cumple la restricción.

El resultado fue

```
=== SOLUCION POR KKT ===
x* (KKT) =
    1.4667
   -0.6000
    0.1333

λ* (KKT) =
    2.7333

Valor óptimo f(x*) =
   -6.5333
```

Esto confirma que el equilibrio obtenido por el método primal–dual coincide con la condición de optimalidad teórica del problema.

Solución con fmincon

El problema se implementa en MATLAB mediante:

- una función objetivo cuadrática,
- una restricción lineal,
- condiciones iniciales.

`fmincon` utiliza un método de optimización numérica (interior-point o SQP).
Durante las iteraciones:

- disminuye la función objetivo,
- reduce la violación de la restricción,
- ajusta gradualmente el gradiente hacia cero.

El resultado final coincide perfectamente:

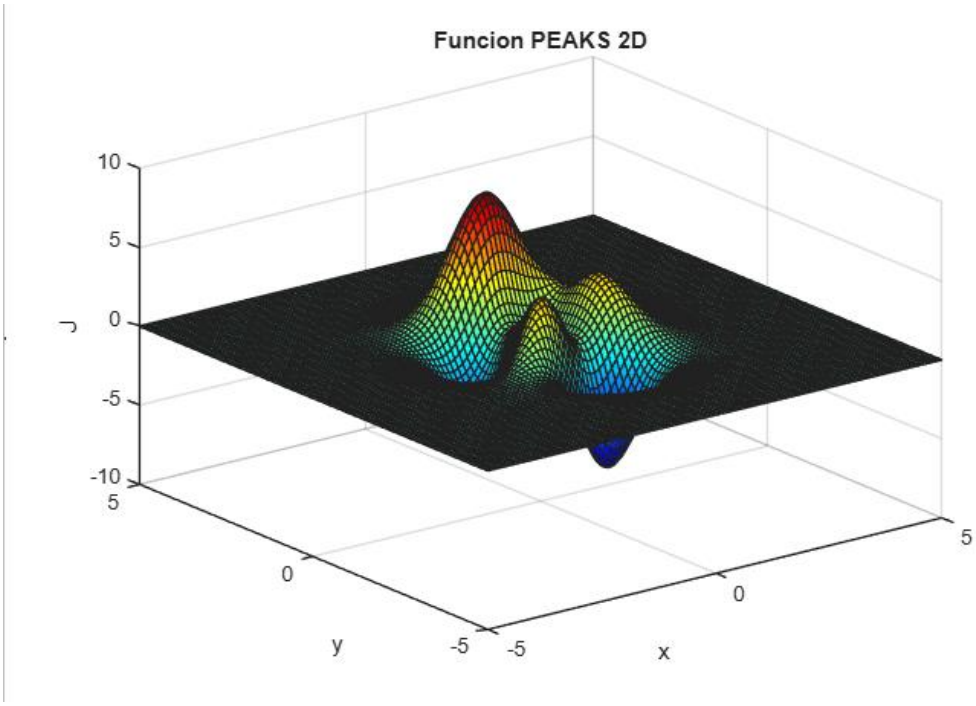
```
=== VALIDACIÓN CON fmincon ===
                                          First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0       4    0.000000e+00    1.000e+00    3.333e+00
    1       9   -4.277778e+00    5.000e-01    3.167e+00    2.062e+00
    2      13   -6.376319e+00    0.000e+00    8.587e-01    1.136e+00
    3      17   -6.522304e+00    0.000e+00    2.412e-01    5.147e-01
    4      21   -6.533246e+00    0.000e+00    2.436e-02    1.187e-01
    5      25   -6.533333e+00    0.000e+00    1.835e-03    8.483e-03
    6      29   -6.533333e+00    0.000e+00    2.684e-05    7.317e-04
    7      33   -6.533333e+00    0.000e+00    4.996e-07    1.678e-05

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
x* (fmincon) =
    1.4667
   -0.6000
    0.1333

Valor óptimo f(x*) con fmincon =
   -6.5333
```

Conclusiones

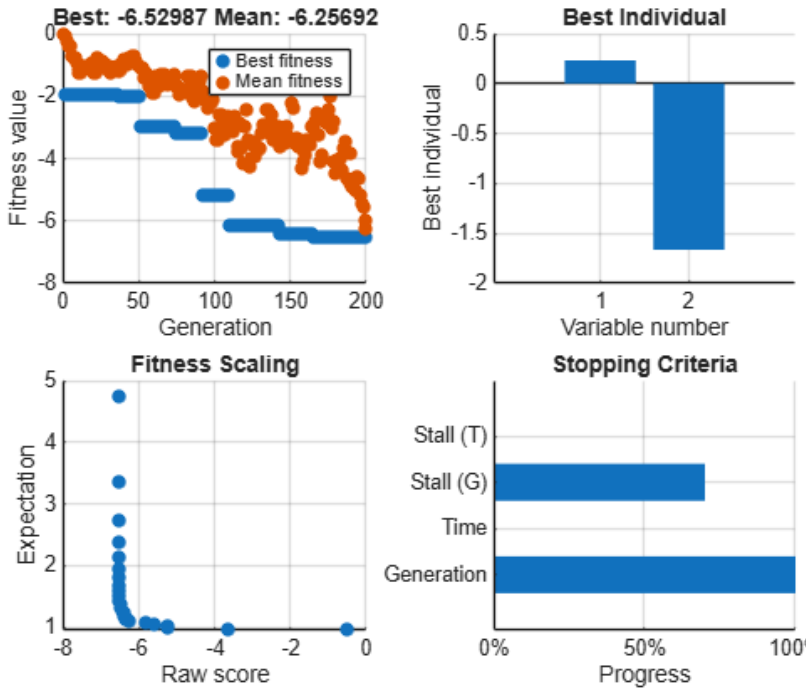
- El problema es estrictamente convexo, por lo que solo existe un único mínimo global.No hay múltiples soluciones ni mínimos locales.
- El método primal–dual, al alcanzar su punto de equilibrio, satisface automáticamente las condiciones KKT, que son la caracterización teórica del óptimo.
- fmincon, aunque usa un enfoque distinto (numérico-iterativo), también llega al mismo punto porque la superficie de la función objetivo guía cualquier algoritmo correctamente formulado hacia ese óptimo global.
- El valor óptimo de la función es −6.5333, que corresponde al mínimo alcanzable bajo la restricción

Punto 3: Para abordar este ejercicio, se realizó la optimización de diversas funciones de prueba bidimensionales ampliamente utilizadas en la evaluación de algoritmos metaheurísticos, entre ellas PEAKS, Himmelblau, Rastrigin, Circles, Rosenbrock y Shaffer. Con el fin de comparar el desempeño entre diferentes métodos de optimización global, se implementaron los algoritmos particleswarm y ga disponibles en MATLAB. Para cada función se analizó qué tan cerca se encontraba la solución obtenida del óptimo global teórico y el número de iteraciones necesarias para converger.

PEAKS 2D

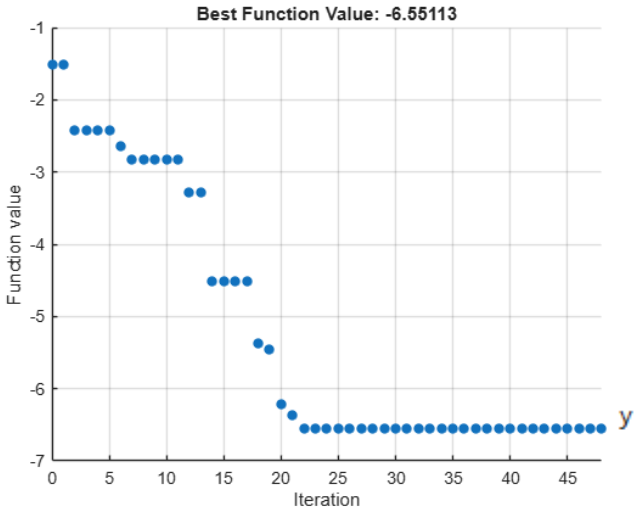


Resultados algoritmo genético:



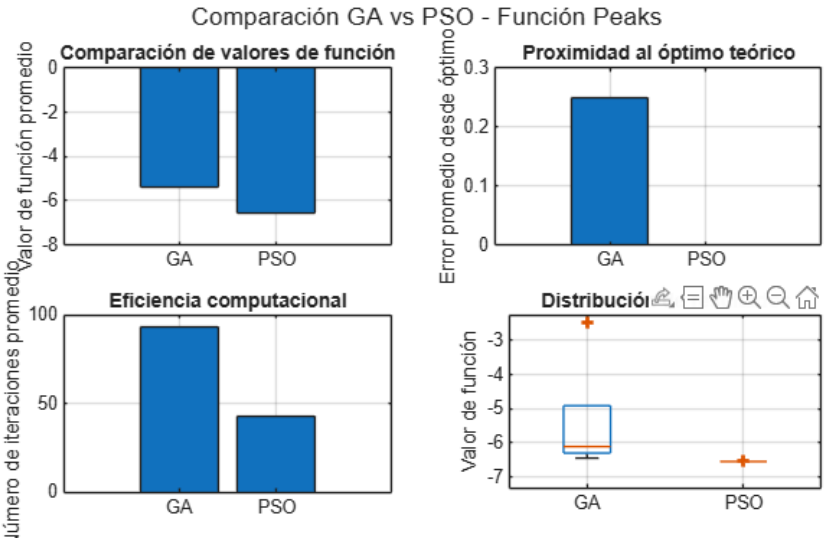

```
y =
    0.2315   -1.6637
```

Resultados particleswarm:




```
y =
    0.2283   -1.6255
```

Resultados Comparación: Se utilizó un óptimo teórico de la función Peaks y el mínimo global está aproximadamente en (0.228, -1.626) con valor ≈ -6.55



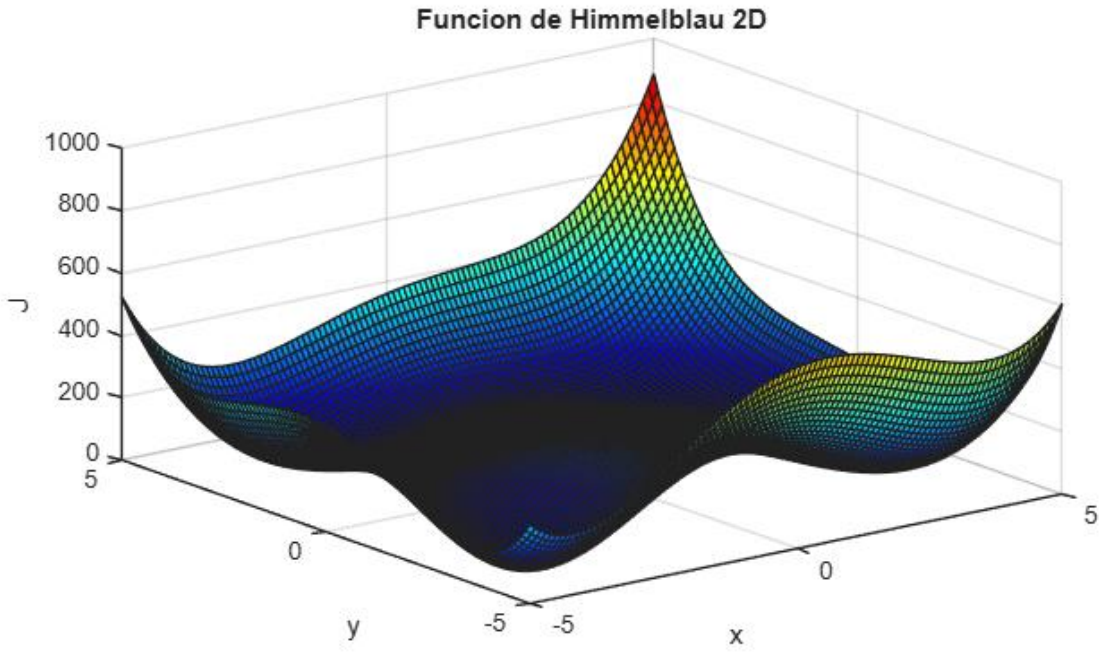

```
=== RESULTADOS ESTADÍSTICOS ===
ALGORITMOS GENÉTICOS (GA):
  Media: x = [0.2508, -1.5091], f = -5.406279
  Desv. Estándar: x = [0.1775, 0.2451], f = 1.652537
  Error promedio desde óptimo: 0.249465
  Iteraciones promedio: 93.6

PARTICLE SWARM OPTIMIZATION (PSO):
  Media: x = [0.2283, -1.6255], f = -6.551133
  Desv. Estándar: x = [0.0000, 0.0000], f = 0.000000
  Error promedio desde óptimo: 0.000543
  Iteraciones promedio: 42.8

=== CONCLUSIÓN ===
Mejor algoritmo (menor valor de función): PSO
Diferencia en valor de función: 1.144855
```

Conclusión:

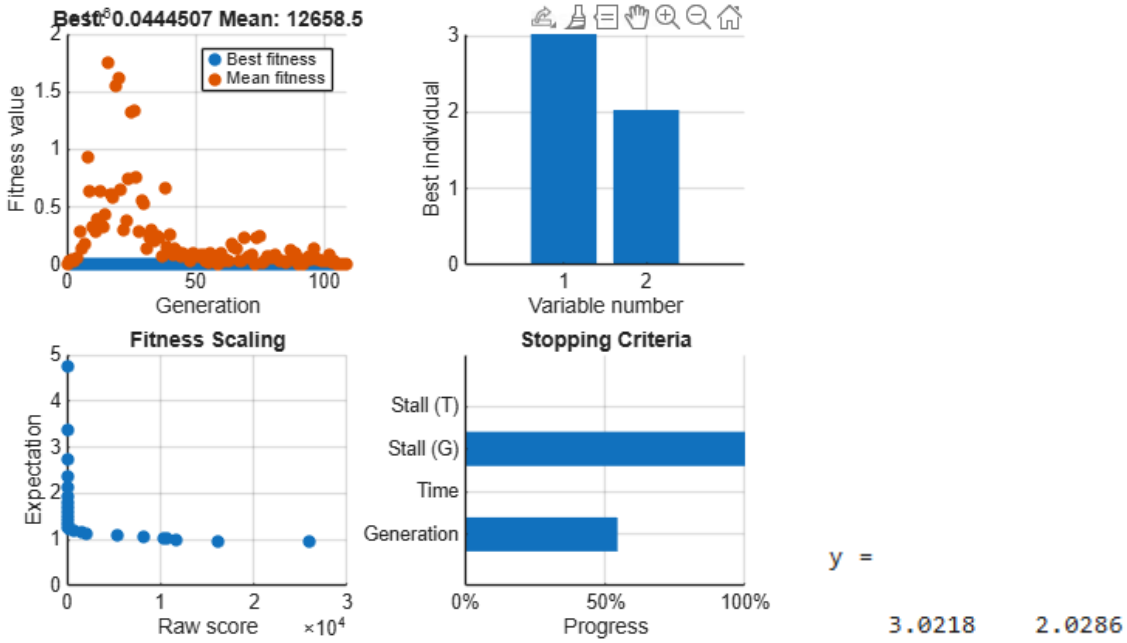
En la función Peaks, el desempeño del algoritmo PSO fue consistentemente superior al del algoritmo genético (GA). PSO no solo alcanzó un valor de función más cercano al óptimo teórico, sino que además mostró una desviación prácticamente nula entre ejecuciones, lo que evidencia una alta estabilidad y reproducibilidad. El error promedio respecto al punto óptimo fue notablemente menor en PSO, demostrando una mejor capacidad de convergencia hacia la solución global.

En contraste, el GA presentó mayor variabilidad en las posiciones encontradas y un error más alto respecto al óptimo, además de requerir más del doble de iteraciones en promedio para converger, lo que refleja una menor eficiencia computacional. Aunque ambos métodos lograron aproximaciones válidas, los resultados muestran que PSO ofrece una mejor combinación de precisión, rapidez y consistencia para esta función, superando al GA por una diferencia aproximada de 1.14 unidades en el valor final de la función.

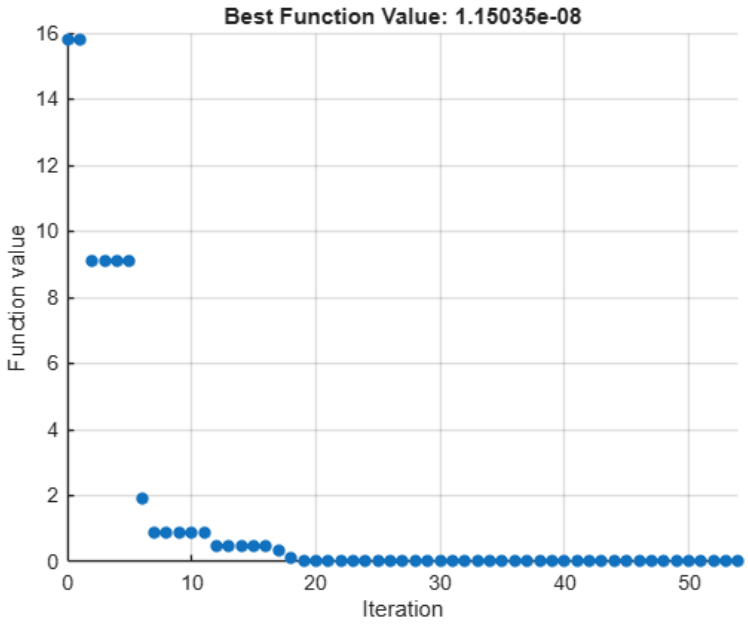
Himmelblaus



Resultados algoritmo genético:

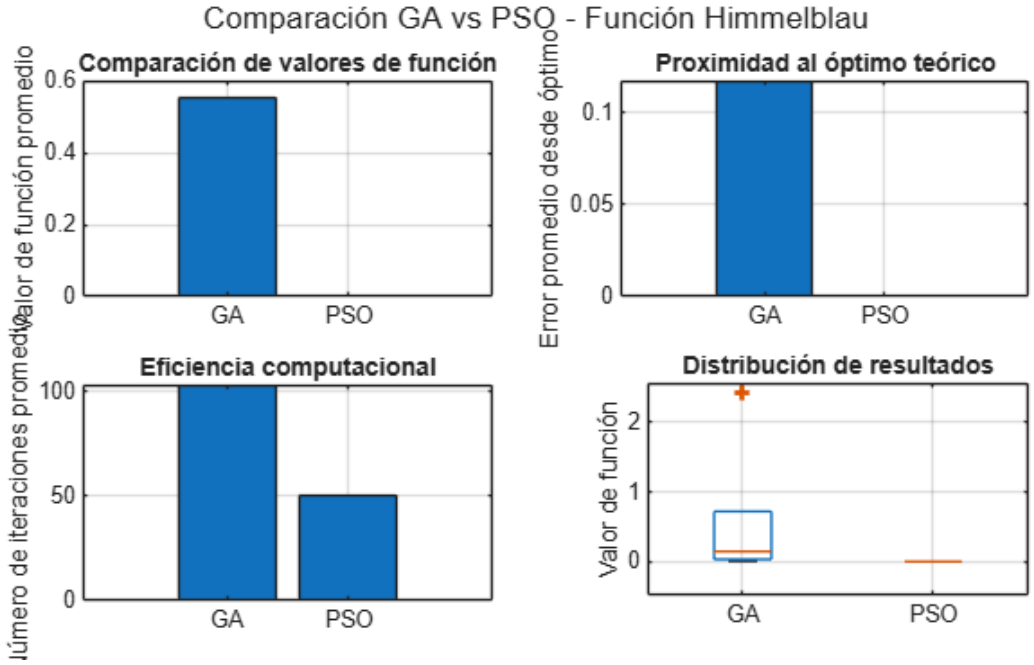


Resultados particleswarm:




```
y =

    3.0000    2.0000
```

Resultados Comparación: Se utilizó un óptimo teórico de la función Himmelblau, tiene 4 mínimos globales: (3,2), (-2.805118, 3.131312), (-3.779310, -3.283186), (3.584428, -1.848126)



```
=== RESULTADOS ESTADÍSTICOS ===
ALGORITMOS GENÉTICOS (GA):
  Media: x = [1.8002, 2.2822], f = 0.553116
  Desv. Estándar: x = [2.5459, 0.4714], f = 1.045378
  Error promedio desde óptimo más cercano: 0.116896
  Iteraciones promedio: 103.2

PARTICLE SWARM OPTIMIZATION (PSO):
  Media: x = [3.1169, 1.2304], f = 0.000000
  Desv. Estándar: x = [0.2614, 1.7209], f = 0.000000
  Error promedio desde óptimo más cercano: 0.000002
  Iteraciones promedio: 50.2

=== CONCLUSIÓN ===
Mejor algoritmo (menor valor de función): PSO
Diferencia en valor de función: 0.553116
```

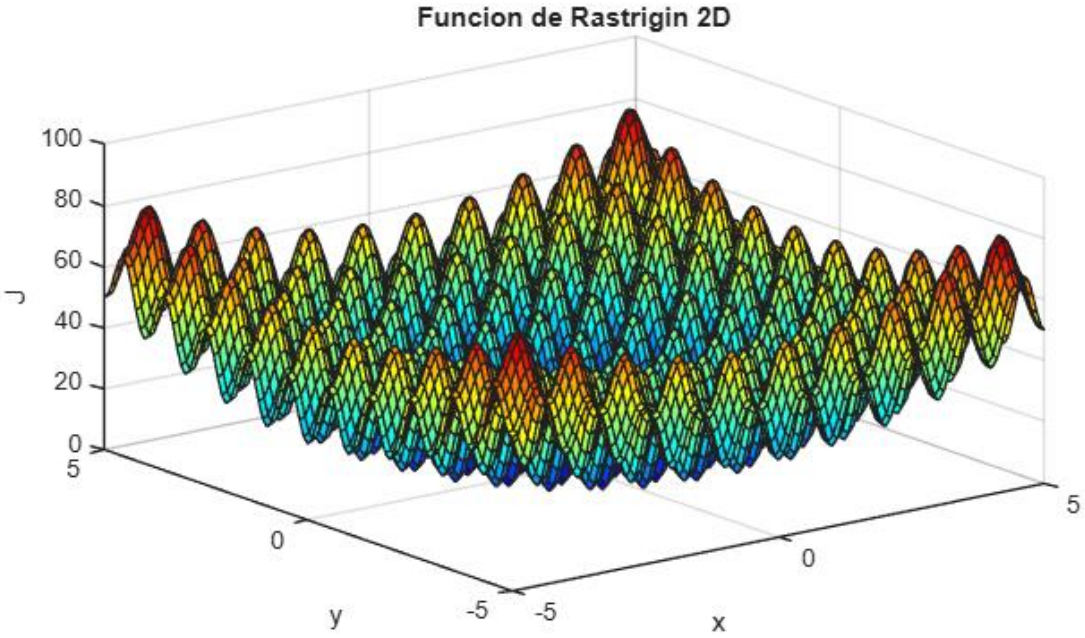
Conclusión:

En la función Himmelblau, el desempeño del algoritmo Particle Swarm Optimization (PSO) fue superior al del Algoritmo Genético (GA). El PSO alcanzó consistentemente valores de función más cercanos al óptimo teórico f = 0 logrando una media prácticamente nula y un error promedio desde el punto óptimo de solo 0.000002, lo que demuestra una excelente capacidad de convergencia hacia la solución global. Además, la desviación estándar de los resultados en PSO fue mínima, indicando una alta estabilidad y reproducibilidad entre ejecuciones.

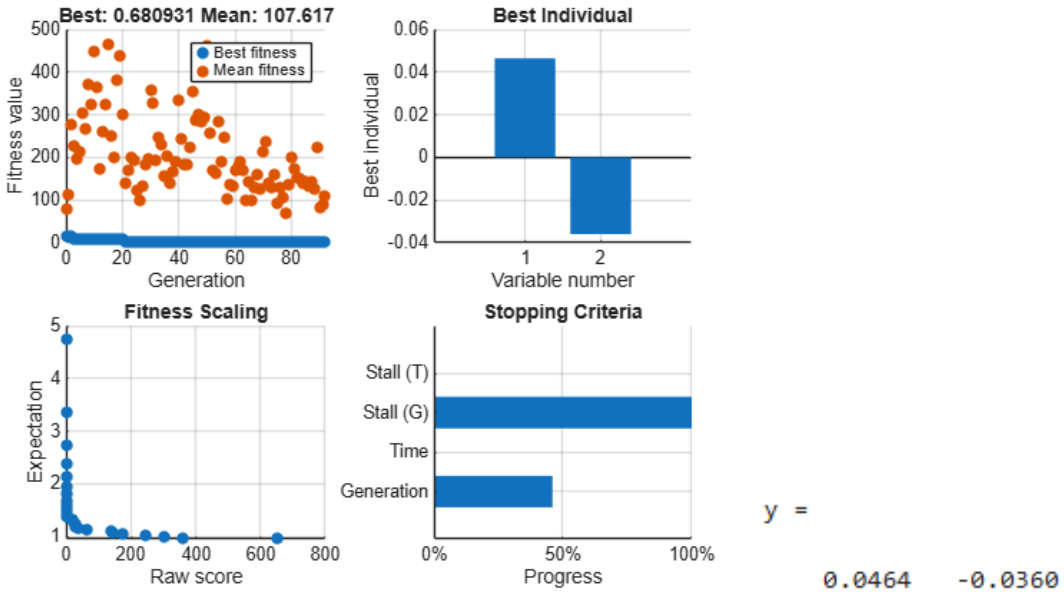
Por el contrario, el GA presentó una media de función notablemente más alta (f = 0.5531) y una mayor variabilidad en las posiciones finales, reflejada en desviaciones estándar considerables en las coordenadas de las soluciones. Asimismo, requirió más del doble de iteraciones en promedio (103.2 vs 50.2) para converger, evidenciando una menor eficiencia computacional.

En conjunto, estos resultados indican que PSO supera al GA en términos de precisión, estabilidad y eficiencia al optimizar la función Himmelblau, alcanzando soluciones más exactas y consistentes con menor esfuerzo computacional. La diferencia en el valor promedio de la función fue de 0.5531 unidades, consolidando al PSO como el método más efectivo para este caso.

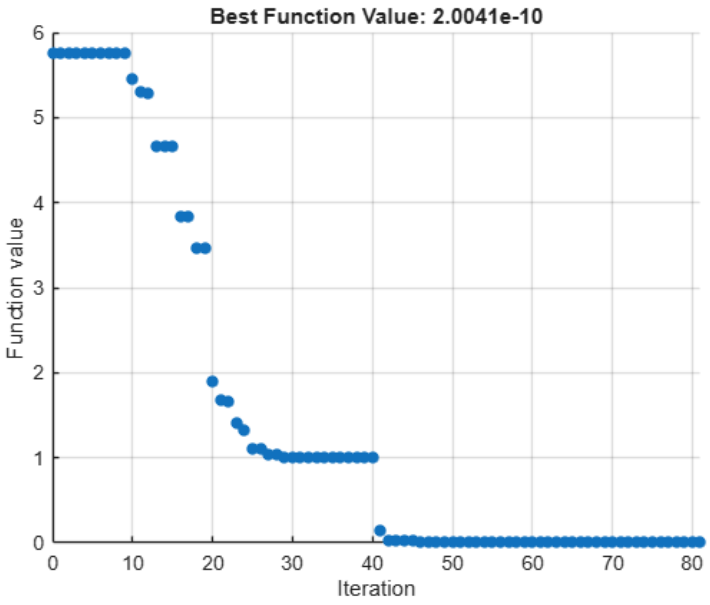
Rastigin



Resultados algoritmo genético:

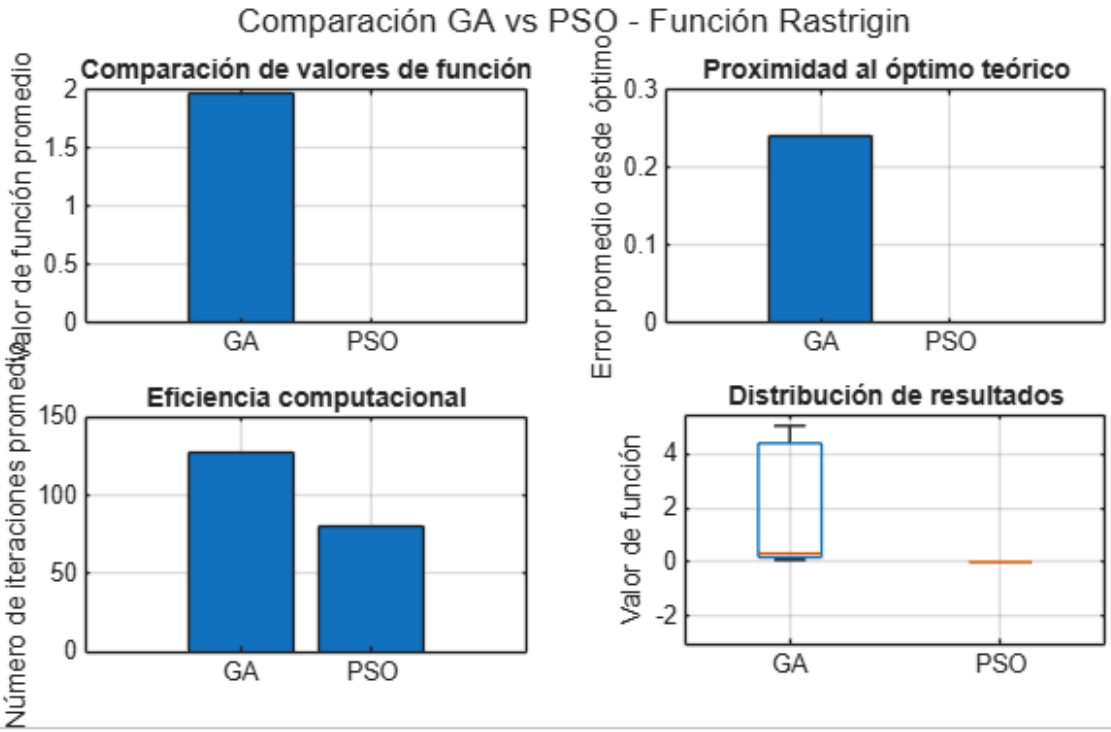


Resultados particleswarm:




```
y =

   1.0e-06 *

   -0.7908    0.6203
```

Resultados Comparación: Se utilizó un óptimo teórico de la función Rastrigin, el mínimo global está en (0,0) con valor f = 0



```
=== RESULTADOS ESTADÍSTICOS ===
ALGORITMOS GENÉTICOS (GA):
  Media: x = [-0.1729, -0.0197], f = 1.964638
  Desv. Estándar: x = [0.4387, 0.0801], f = 2.435510
  Error promedio desde óptimo: 0.240069
  Iteraciones promedio: 127.0

PARTICLE SWARM OPTIMIZATION (PSO):
  Media: x = [-0.0000, -0.0000], f = 0.000000
  Desv. Estándar: x = [0.0000, 0.0000], f = 0.000000
  Error promedio desde óptimo: 0.000000
  Iteraciones promedio: 79.8

=== CONCLUSIÓN ===
Mejor algoritmo (menor valor de función): PSO
Diferencia en valor de función: 1.964638
```

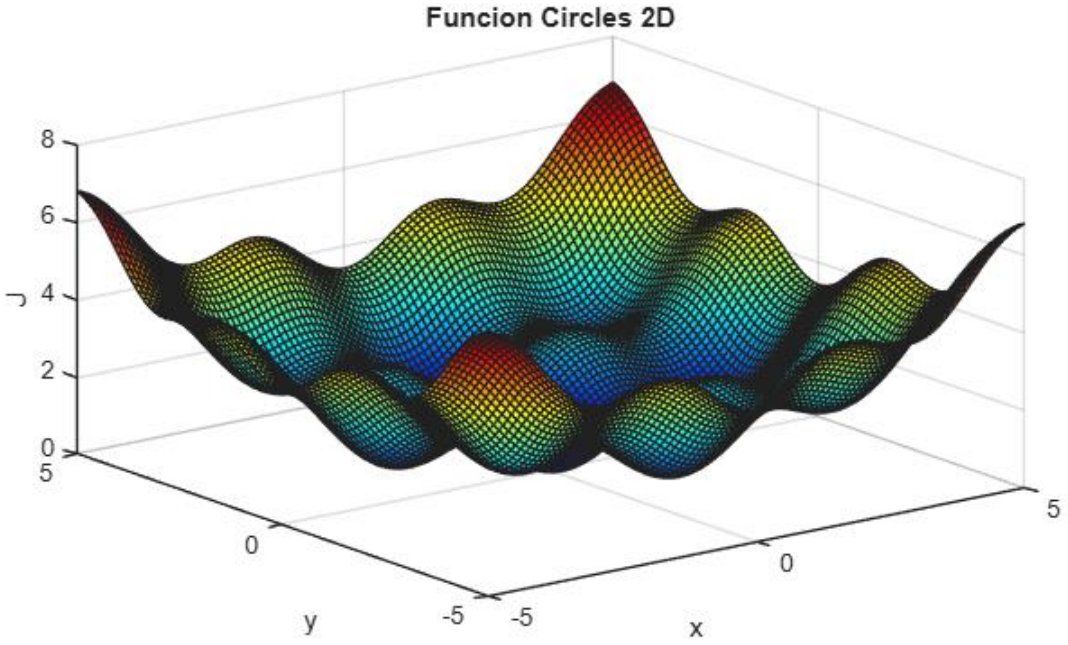
Conclusión:

En la función Rastrigin, el desempeño del algoritmo Particle Swarm Optimization (PSO) fue superior al del Algoritmo Genético (GA). El PSO alcanzó de forma consistente el óptimo teórico en (0, 0), con un valor de función promedio de f = 0 y error promedio nulo, demostrando una convergencia perfecta hacia la solución global. Además, la desviación estándar en las coordenadas y el valor de función fue prácticamente cero, lo que evidencia una excelente estabilidad y reproducibilidad entre ejecuciones.

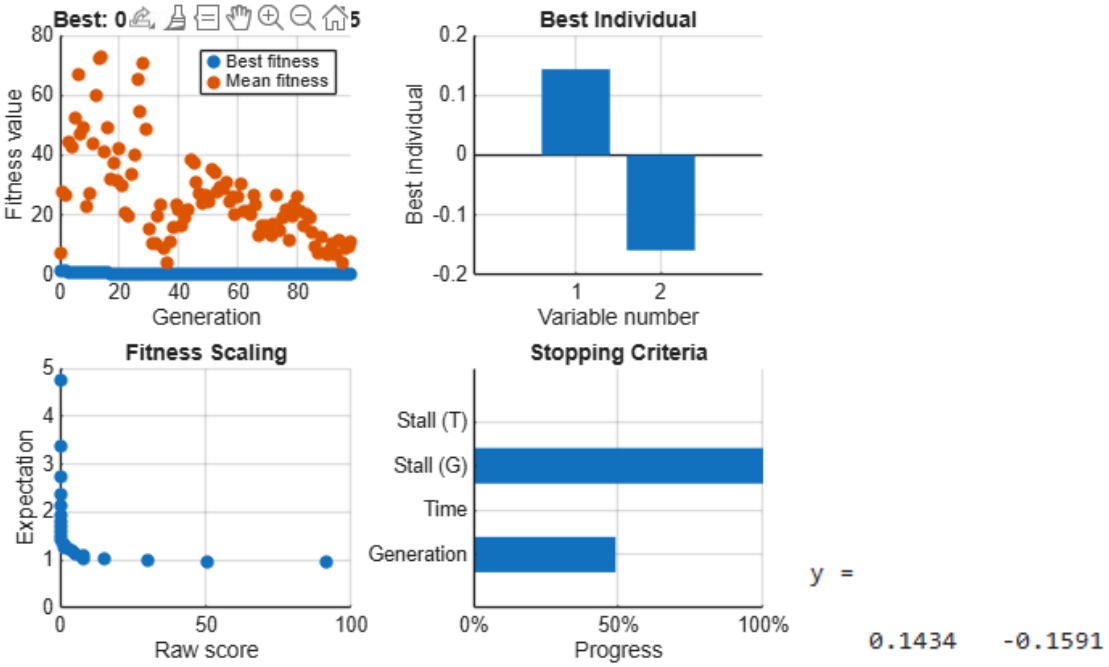
Por otro lado, el GA presentó un valor de función promedio significativamente mayor f = 1.9646 una variabilidad notable tanto en las soluciones encontradas como en los valores de función desviación estándar 2.4355. Asimismo, el algoritmo genético requirió más iteraciones en promedio (127 vs 79.8) para converger, indicando una menor eficiencia computacional frente al PSO.

En conjunto, los resultados muestran que PSO ofrece una ventaja significativa en precisión, velocidad y consistencia al optimizar la función Rastrigin. Su capacidad para alcanzar el mínimo global de forma estable lo convierte en la opción más efectiva para este tipo de función multimodal, superando al GA por una diferencia de 1.96 unidades en el valor promedio de la función.

Circles

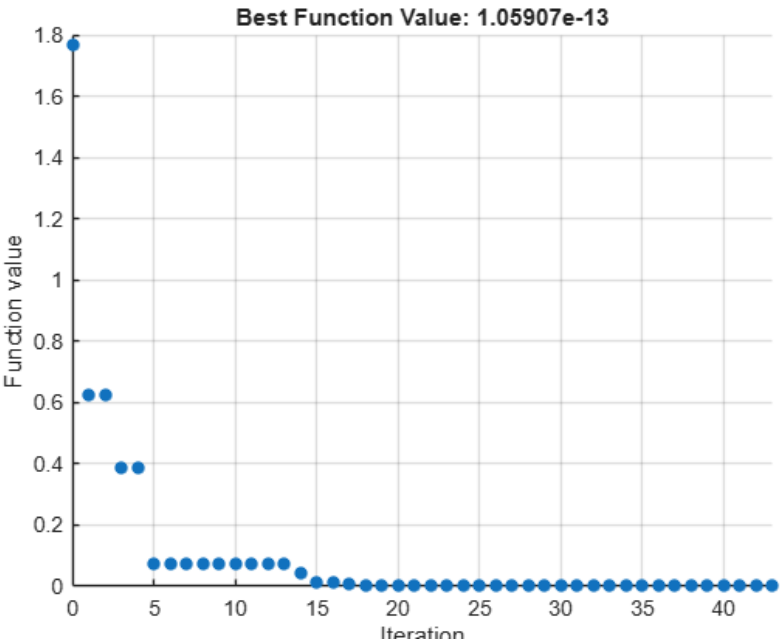


Resultados algoritmo genético:



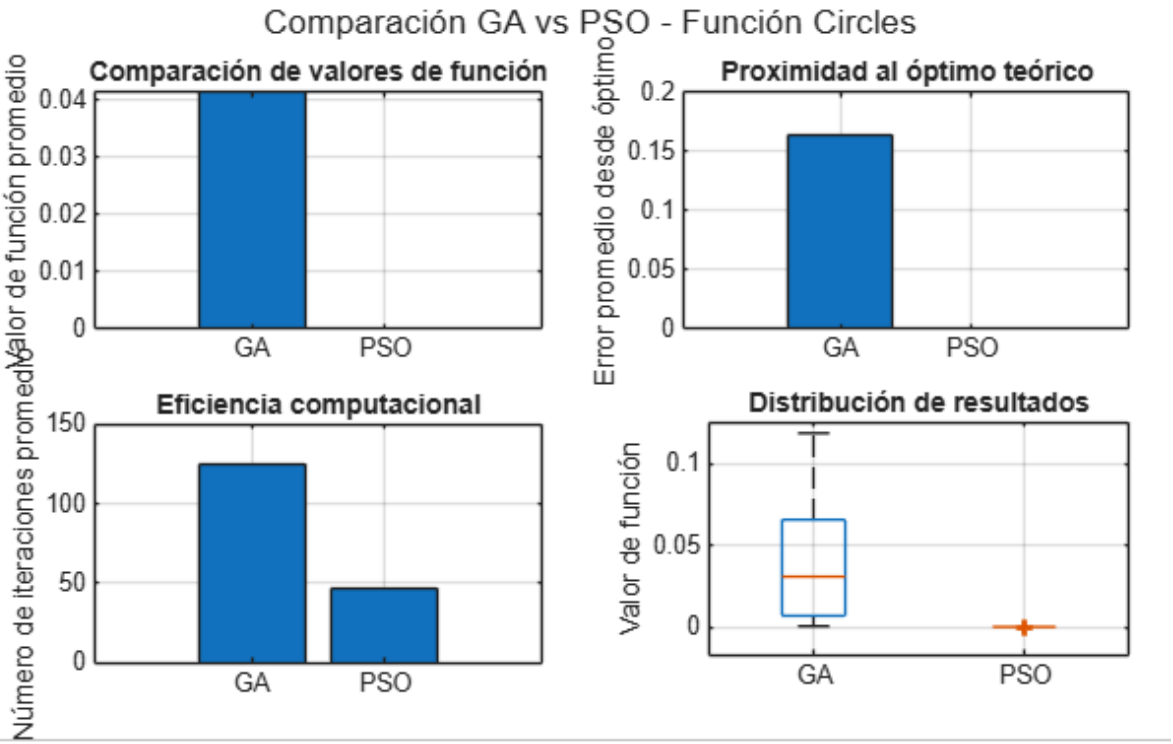

```
y =

    0.1434   -0.1591
```

Resultados particleswarm:




```
y =

   1.0e-06 *

   -0.0329   -0.3085
```

Resultados Comparación: Se utilizó un óptimo teórico de la función Circles f(x,y) = sin(x)^2 + sin(y)^2 + 0.1*(x^2 + y^2), el mínimo global está en (0,0) con valor f(0,0) = 0.



```
=== RESULTADOS ESTADÍSTICOS ===
ALGORITMOS GENÉTICOS (GA):
  Media: x = [0.1220, 0.0265], f = 0.041409
  Desv. Estándar: x = [0.1195, 0.1182], f = 0.047135
  Error promedio desde óptimo: 0.163824
  Iteraciones promedio: 124.6

PARTICLE SWARM OPTIMIZATION (PSO):
  Media: x = [0.0000, 0.0000], f = 0.000000
  Desv. Estándar: x = [0.0000, 0.0000], f = 0.000000
  Error promedio desde óptimo: 0.000008
  Iteraciones promedio: 46.4

=== CONCLUSIÓN ===
Mejor algoritmo (menor valor de función): PSO
Diferencia en valor de función: 0.041409
```

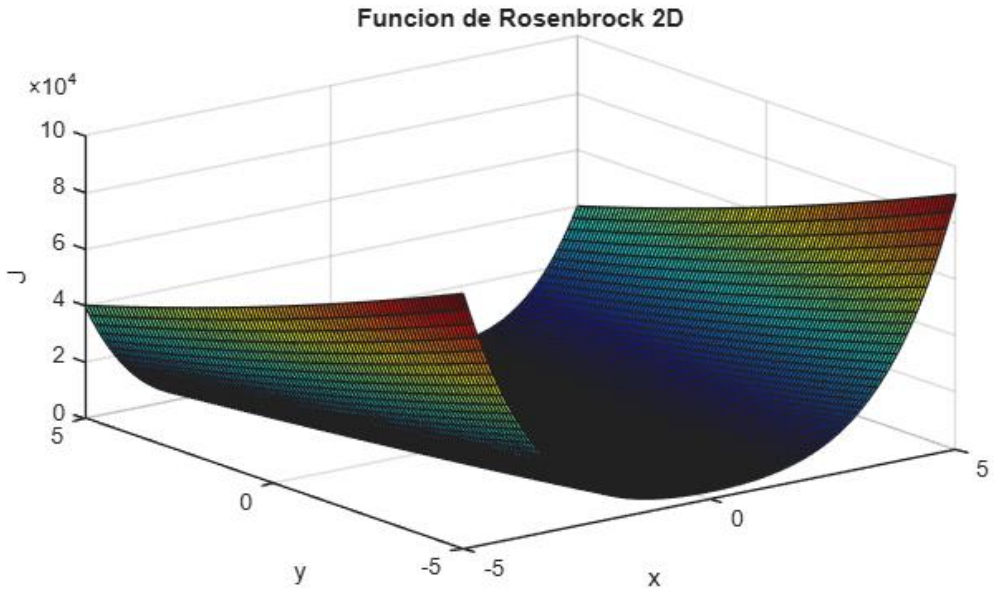
Conclusión:

En la función Circles, el rendimiento del algoritmo Particle Swarm Optimization (PSO) fue notablemente superior al del Algoritmo Genético (GA). El PSO alcanzó de manera consistente el óptimo teórico en el punto (0,0), obteniendo un valor promedio de función f = 0 sin error respecto al mínimo global. Además, la desviación estándar tanto en las coordenadas como en los valores de función fue esencialmente nula, lo que evidencia una estabilidad sobresaliente y una excelente reproducibilidad entre ejecuciones.

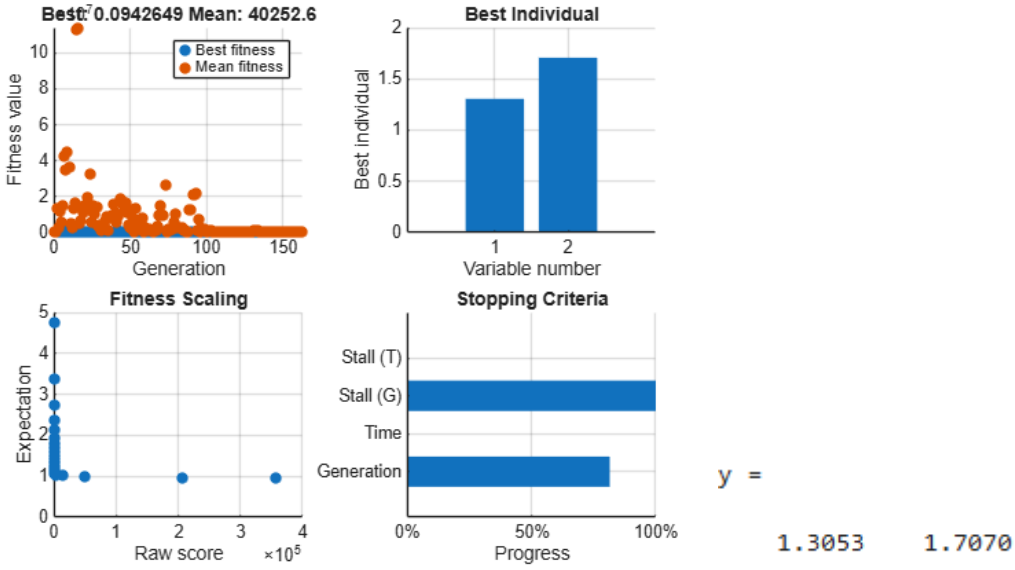
En contraste, el GA presentó un comportamiento claramente inferior: obtuvo un valor promedio de función de aproximadamente f = 0.041, con una variabilidad apreciable en sus soluciones. El error promedio respecto al óptimo también fue mayor, indicando dificultad para converger exactamente al punto mínimo. Asimismo, el algoritmo genético requirió cerca de 125 iteraciones en promedio, frente a solo 46 iteraciones del PSO, lo que demuestra una menor eficiencia computacional.

En conjunto, los resultados muestran que el PSO supera al GA en precisión, rapidez y consistencia al optimizar la función Circles. Su capacidad para alcanzar el mínimo global de manera estable y con muy baja dispersión convierte al PSO en la opción claramente más efectiva para esta función. La diferencia en el valor promedio de la función entre ambos métodos aproximadamente 0.041 confirma esta ventaja significativa.

Rosenbrock

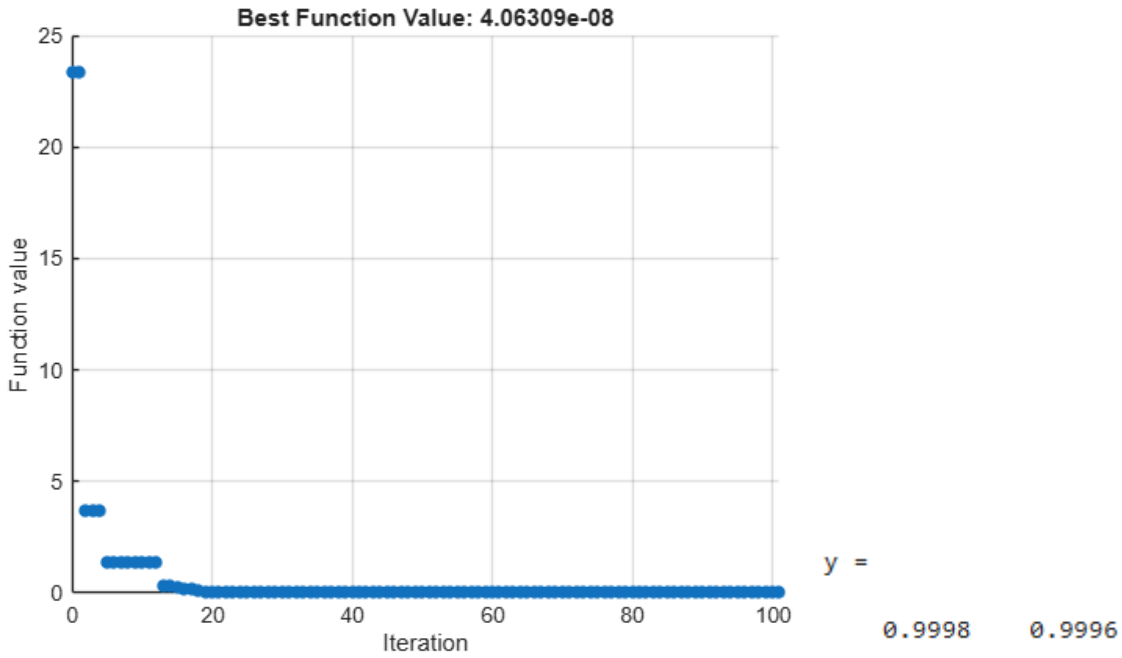


Resultados algoritmo genético:

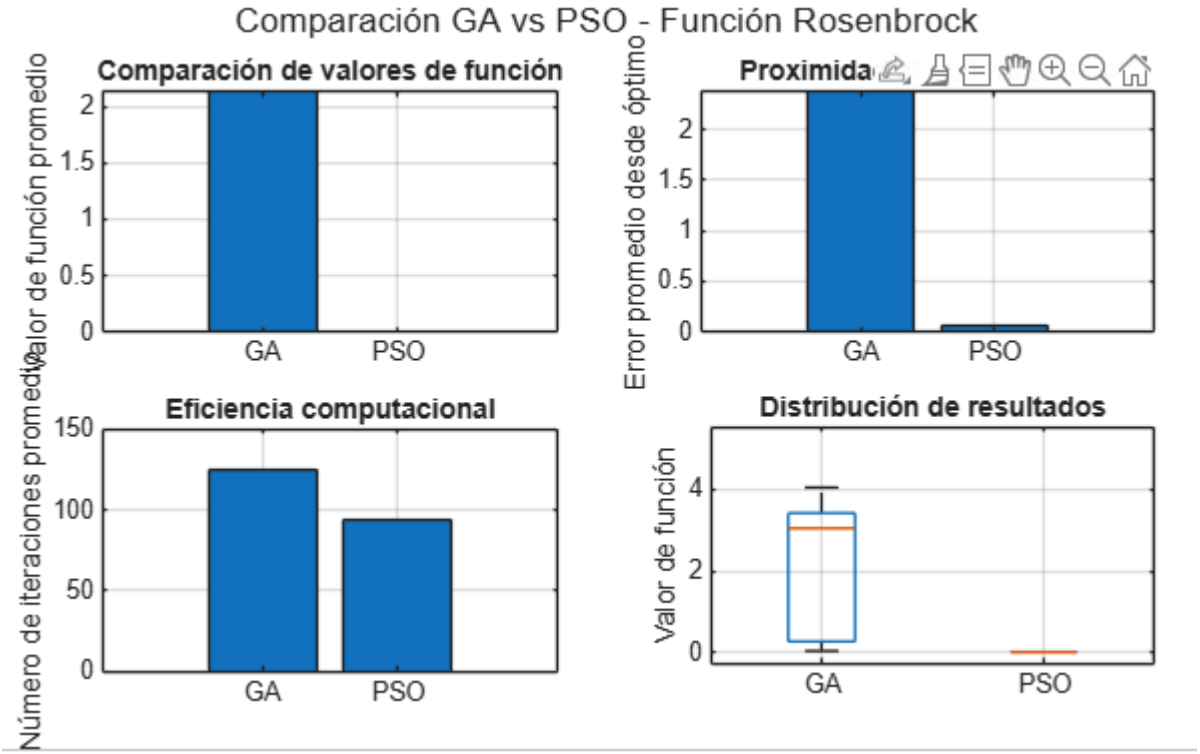



```
y =

    1.3053    1.7070
```

Resultados particleswarm:



Resultados Comparación: Se utilizó un óptimo teórico de la función Rosenbrock, el mínimo global está en (1,1) con valor f = 0



```
=== RESULTADOS ESTADÍSTICOS ===
ALGORITMOS GENÉTICOS (GA):
  Media: x = [1.0269, 2.5444], f = 2.134117
  Desv. Estándar: x = [1.3568, 3.0578], f = 1.823324
  Error promedio desde óptimo: 2.369298
  Iteraciones promedio: 125.2

PARTICLE SWARM OPTIMIZATION (PSO):
  Media: x = [0.9948, 0.9922], f = 0.002197
  Desv. Estándar: x = [0.0424, 0.0855], f = 0.003010
  Error promedio desde óptimo: 0.055857
  Iteraciones promedio: 93.6

=== CONCLUSIÓN ===
Mejor algoritmo (menor valor de función): PSO
Diferencia en valor de función: 2.131920
```

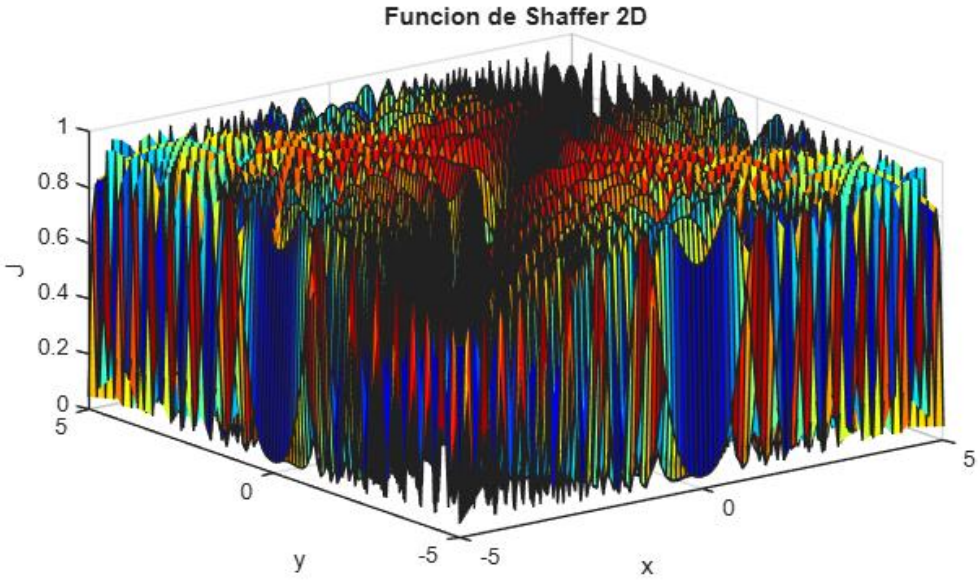
Conclusión:

En la función Rosenbrock, el algoritmo Particle Swarm Optimization (PSO) mostró un desempeño significativamente superior al del Algoritmo Genético (GA). El PSO logró aproximarse de manera consistente al óptimo teórico (1, 1), obteniendo un valor de función promedio $f \approx 0.0022$, muy cercano al mínimo global. La desviación estándar de los valores de función fue extremadamente baja (≈ 0.003), lo que indica una convergencia estable y reproducible entre ejecuciones. Además, el error promedio respecto al óptimo fue de apenas 0.0558, confirmando su capacidad para acercarse con precisión a la solución global.

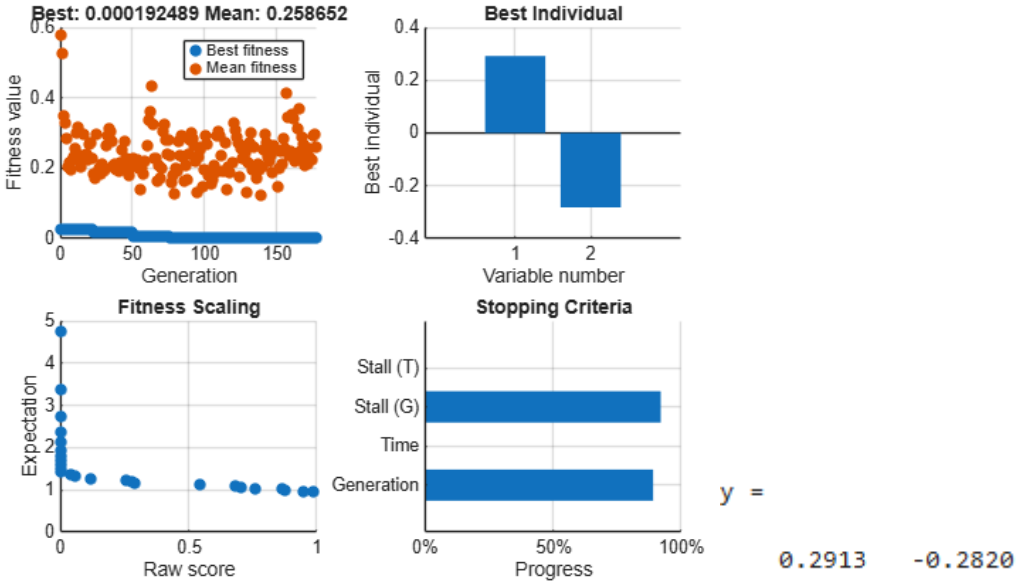
En contraste, el GA obtuvo un valor promedio de función mucho mayor, $f \approx 2.134$, mostrando dificultades claras para encontrar el valle estrecho característico de Rosenbrock. La variabilidad en los resultados tambiér fue considerable, con una desviación estándar de 1.823, lo que evidencia una baja consistencia y una tendencia a soluciones subóptimas. Asimismo, el GA necesitó más iteraciones en promedio (125.2) que el PSO (93.6 iteraciones), demostrando una menor eficiencia computacional.

En conjunto, los resultados evidencian que el PSO ofrece una ventaja contundente en precisión, estabilidad y velocidad al optimizar la función Rosenbrock. La diferencia entre ambos algoritmos aproximadamente 2.13 unidades en el valor promedio de la función confirma que, para este problema en particular, el PSO es considerablemente más efectivo que el GA.

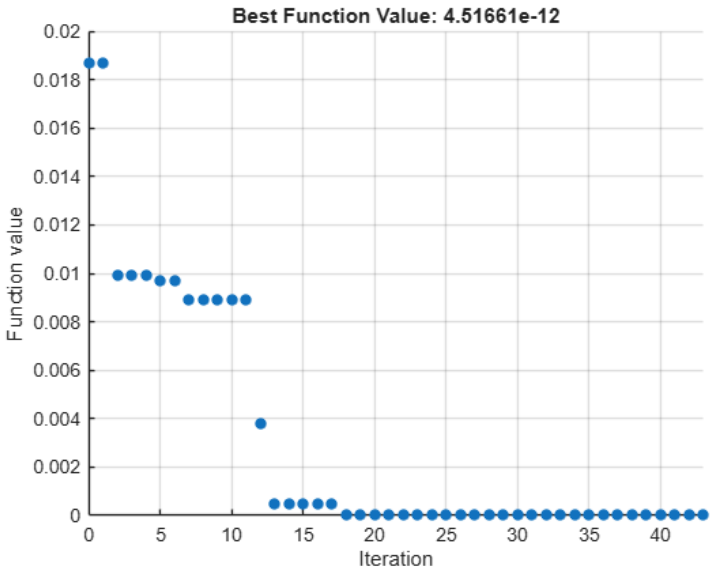
Shaffer



Resultados algoritmo genético:

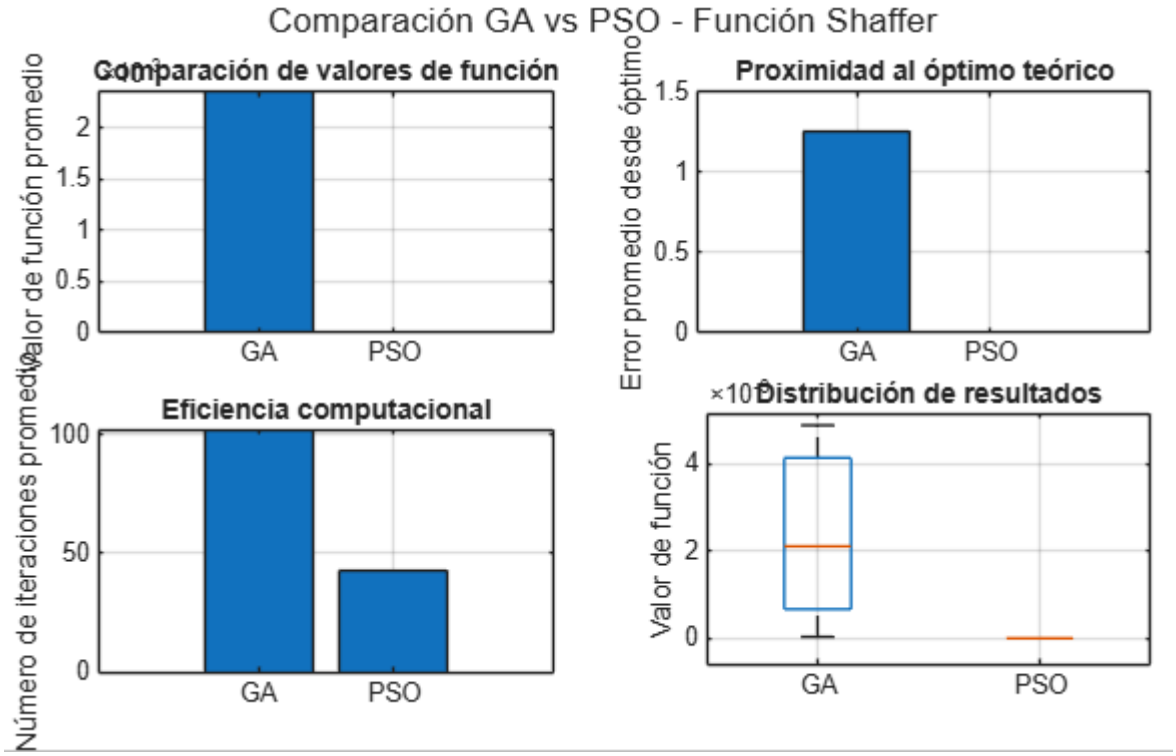


Resultados particleswarm:




```
y =

    1.0e-04 *

   -0.6436   -0.1933
```

Resultados Comparación: Se utilizó un óptimo teórico de la función Shaffer, el mínimo global está en (0,0) con valor f = 0



```
=== RESULTADOS ESTADÍSTICOS ===
ALGORITMOS GENÉTICOS (GA):
  Media: x = [-0.0439, 0.2403], f = 0.002357
  Desv. Estándar: x = [1.3177, 0.9454], f = 0.002023
  Error promedio desde óptimo: 1.257990
  Iteraciones promedio: 102.0

PARTICLE SWARM OPTIMIZATION (PSO):
  Media: x = [0.0001, -0.0002], f = 0.000000
  Desv. Estándar: x = [0.0003, 0.0005], f = 0.000000
  Error promedio desde óptimo: 0.000443
  Iteraciones promedio: 43.0

=== CONCLUSIÓN ===
Mejor algoritmo (menor valor de función): PSO
Diferencia en valor de función: 0.002357
```

Conclusión:

En la función Shaffer, el algoritmo Particle Swarm Optimization (PSO) demostró un rendimiento claramente superior al del Algoritmo Genético (GA). El PSO alcanzó de forma consistente el valor de función óptimo f = 0, correspondiente al mínimo global en (0, 0). Tanto la media como la desviación estándar de los resultados fueron prácticamente nulas, lo que evidencia una convergencia extremadamente precisa y una estabilidad sobresaliente entre ejecuciones. Además, su error promedio respecto al óptimo fue de apenas 0.000443, mostrando una alta capacidad para localizar de manera exacta la solución global.

Por el contrario, el GA obtuvo un valor promedio de función de aproximadamente f = 0.002357 considerablemente mayor que el del PSO. También presentó un error medio de 1.25799, indicando una clara dificultad para posicionarse adecuadamente en el mínimo global. La dispersión en las soluciones fue mucho más alta que en el PSO, lo que refleja una convergencia menos fiable y notable variabilidad entre ejecuciones. En términos computacionales, el GA requirió 102 iteraciones en promedio, mientras que el PSO logró converger en solo 43 iteraciones, demostrando una eficiencia significativamente mayor.

En conjunto, los resultados muestran que el PSO ofrece un desempeño contundentemente superior al GA en la optimización de la función Shaffer. Su precisión absoluta, rápida convergencia y mínima variabilidad lo convierten en la opción más eficiente para esta función. La diferencia de aproximadamente 0.002357 unidades en el valor promedio de la función confirma esta ventaja de manera cuantitativa.