Grey Wolf Optimizer Implementation: A Bio-Inspired Optimization Technique

Irfan Mahmood **Emmanuel Pamboy**

Abdoulie J Bah **Bilal Masih**

Faculty of Applied Mathematics

Silesian University of Technology

RealMath

Introduction to GWO

The **Grey Wolf Optimizer (GWO)** is a bio-inspired optimization algorithm developed by Mirjalili et al. in 2014. It is modeled after the social hierarchy and hunting behavior of grey wolves. The algorithm features a leadership hierarchy of alpha, beta, delta, and omega wolves, which mimics their cooperative hunting strategy. By dynamically adjusting wolf positions, the algorithm balances exploration and exploitation to find optimal solutions. GWO is known for its simplicity, flexibility, and effectiveness, making it suitable for various applications including engineering, machine learning, and data mining. Its ease of implementation and competitive performance has made it a popular choice in optimization problems.

Key Features of GWO

- **Leadership Hierarchy:** GWO mimics the leadership hierarchy of grey wolves, which consists of alpha, beta, delta, and omega wolves.
- **Hunting Mechanism:** The algorithm simulates the cooperative hunting strategy of grey wolves, including encircling prey, pursuing, attacking, and searching for prey.
- **Exploration and Exploitation:** GWO balances exploration (searching for new solutions) and exploitation (refining existing solutions) by dynamically adjusting the position of wolves in the solution space.

Applications of GWO

The **Grey Wolf Optimizer (GWO)** is widely used across various fields due to its simplicity and efficiency. In engineering, it optimizes design parameters, improving performance and reducing costs. In machine learning, GWO tunes hyperparameters, enhancing model accuracy and training speed. Data mining applications benefit from GWO in feature selection and clustering, extracting meaningful patterns from large datasets. Additionally, GWO is applied in scheduling, image processing, and control systems, solving complex optimization problems. Its robust performance in diverse domains demonstrates GWO's adaptability and effectiveness, making it a valuable tool for researchers and practitioners seeking optimal solutions.

Implementation details

The implementation of the Grey Wolf Optimizer (GWO) involves applying it to three benchmark functions: **Sphere, Rastrigin, and Rosenbrock**. For each function, the optimization is performed across **three dimensions (2, 5, and 10), four population sizes (10, 20, 50, and 100), and three iteration counts (10, 30, and 50).** The process begins by initializing a population of grey wolves randomly within the defined search space. The wolves' positions are updated iteratively, simulating the hunting behavior of grey wolves, including encircling prey, hunting, and attacking.

For each combination of dimension, population size, and iteration count, the GWO algorithm is run multiple times to ensure robustness. The best solutions from each run are recorded and analyzed. This setup allows for comprehensive testing of GWO's performance on different types of optimization landscapes and under varying computational conditions. The results are then aggregated and visualized, highlighting GWO's efficacy and providing insights into its behavior across different problem settings.

```
CreateWolf[fitness_,dim_,minx_,maxx_,seed_]:=Module
[{position,rnd,wolf},

  SeedRandom[seed];

  position=Table[0.0,{dim}];

  Do[position[[i]]=(maxx-minx)
RandomReal[]+minx,{i,dim}];

  fitnessValue=fitness[position];

  wolf=<|"Position"->position,"Fitness"-
>fitnessValue|>;

  wolf]
```

Mathematica code

Classifying Wolf

```
gwo[fitness_,maxIter_,n_,dim_,minx_,maxx_]:=Module[{rnd,
population,alphaWolf,betaWolf,gammaWolf,Iter,a,A1,A2,A3,
C1,C2,C3,X1,X2,X3,Xnew,fnew},rnd=RandomReal[1,{n,dim}];

   population=Table[<|"position"-
>RandomReal[{minx,maxx},dim],"fitness"-
>fitness[RandomReal[{minx,maxx},dim]]|>,{n}];

   population=SortBy[population,#fitness&];

   {alphaWolf,betaWolf,gammaWolf}=population[[;;3]];

   Iter=0;


While[Iter<maxIter,If[Mod[Iter,10]==0&&Iter>1,Print["Ite
r = ",Iter," best fitness = ",alphaWolf["fitness"]]];

    a=2 (1-Iter/maxIter);

    Do[A1=a (2 RandomReal[]-1);

     A2=a (2 RandomReal[]-1);

     A3=a (2 RandomReal[]-1);

     C1=2 RandomReal[];

     C2=2 RandomReal[];

     C3=2 RandomReal[];
```

Mathematica code

Grey Wolf Optimizer

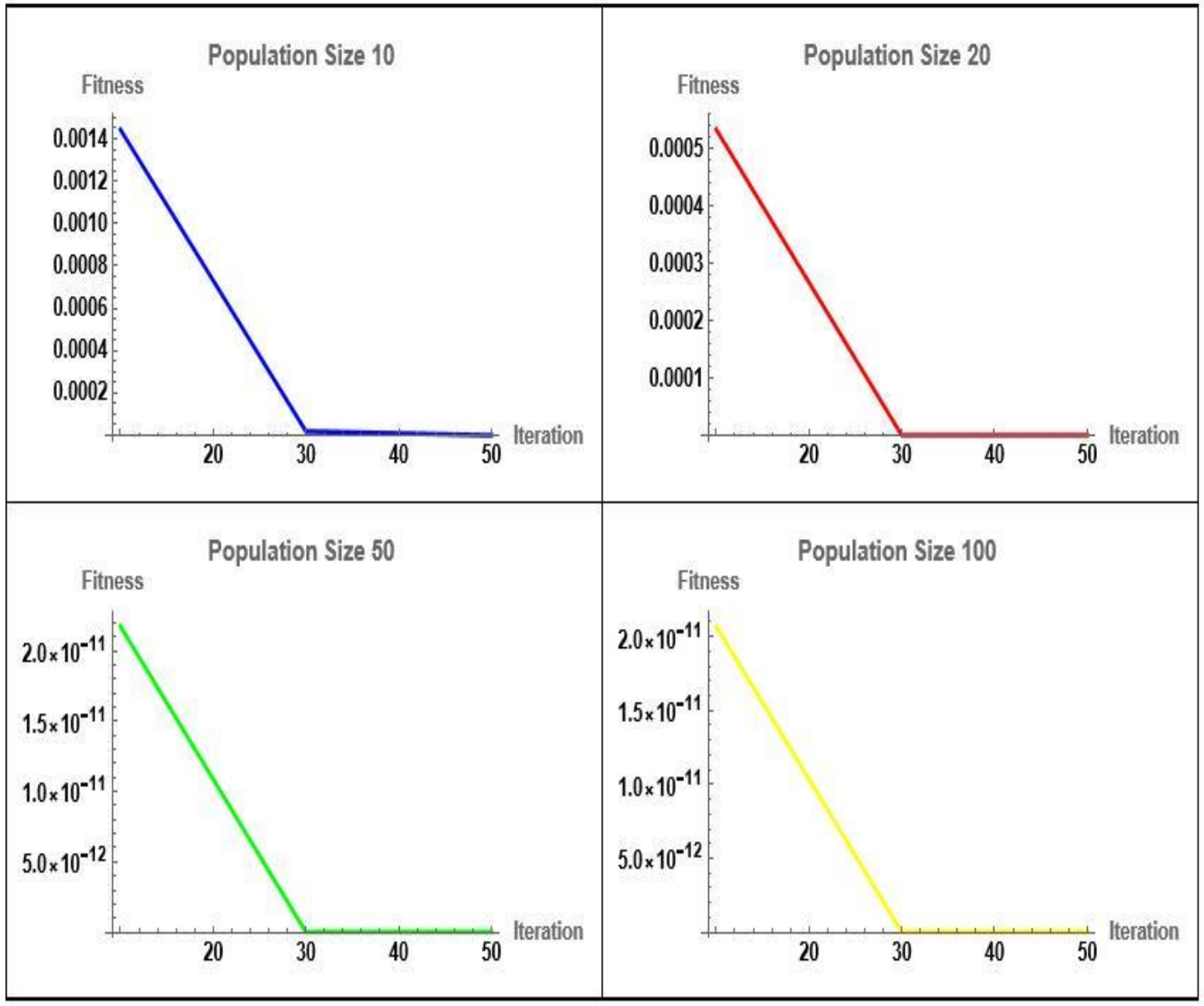
```
 X1=alphaWolf["position"]-A1 Abs[C1
alphaWolf["position"]-population[[i,"position"]]];
     X2=betaWolf["position"]-A2 Abs[C2
betaWolf["position"]-population[[i,"position"]]];
     X3=gammaWolf["position"]-A3 Abs[C3
gammaWolf["position"]-population[[i,"position"]]];
     Xnew=Mean[{X1,X2,X3}];
     fnew=fitness[Xnew];

If[fnew<population[[i,"fitness"]],population[[i,"po
sition"]]=Xnew;
      population[[i,"fitness"]]=fnew;],{i,n}];
    population=SortBy[population,#fitness&];

{alphaWolf,betaWolf,gammaWolf}=population[[;;3]];
    Iter++;];
   alphaWolf["position"]];
```

Mathematica code

Grey Wolf Optimizer

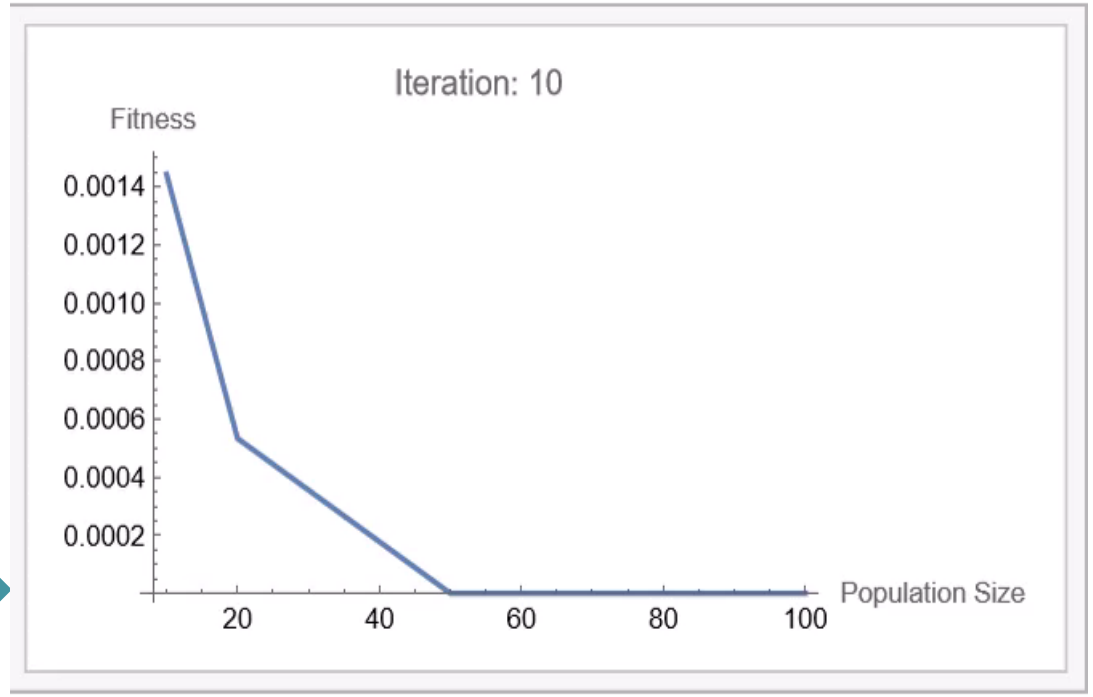
Implementation: **Sphere function for dimension 2**

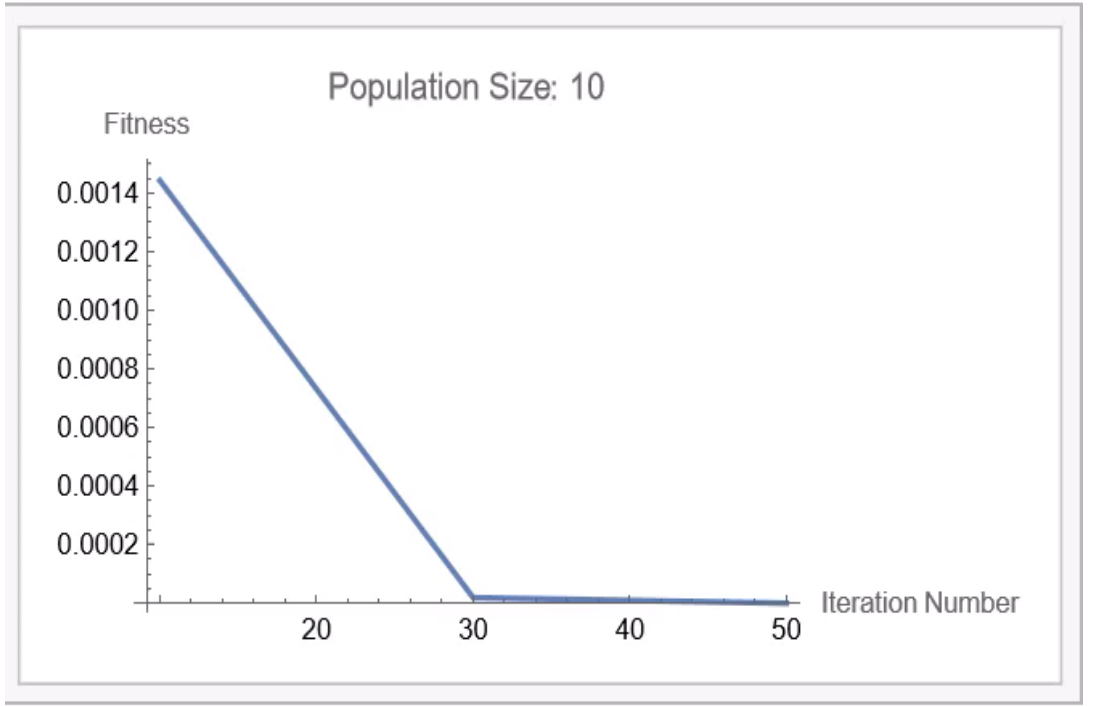
Dimension	Population Size	Iteration Number	Fitness
2	10	10	0.00144304
2	10	30	0.0000191292
2	10	50	0.00000000000933022
2	20	10	0.000533731
2	20	30	0.0000000343572
2	20	50	0.000000000000874519
2	50	10	0.0000000000217473
2	50	30	0.00000000000000185949
2	50	50	0.000000000000000000084182
2	100	10	0.0000000000207014
2	100	30	0.000000000000000000030072
2	100	50	0.000000000000000000000000000000000000000138064



Implementation:
Sphere function for dimension 2

Animation: **Sphere function for dimension 2**

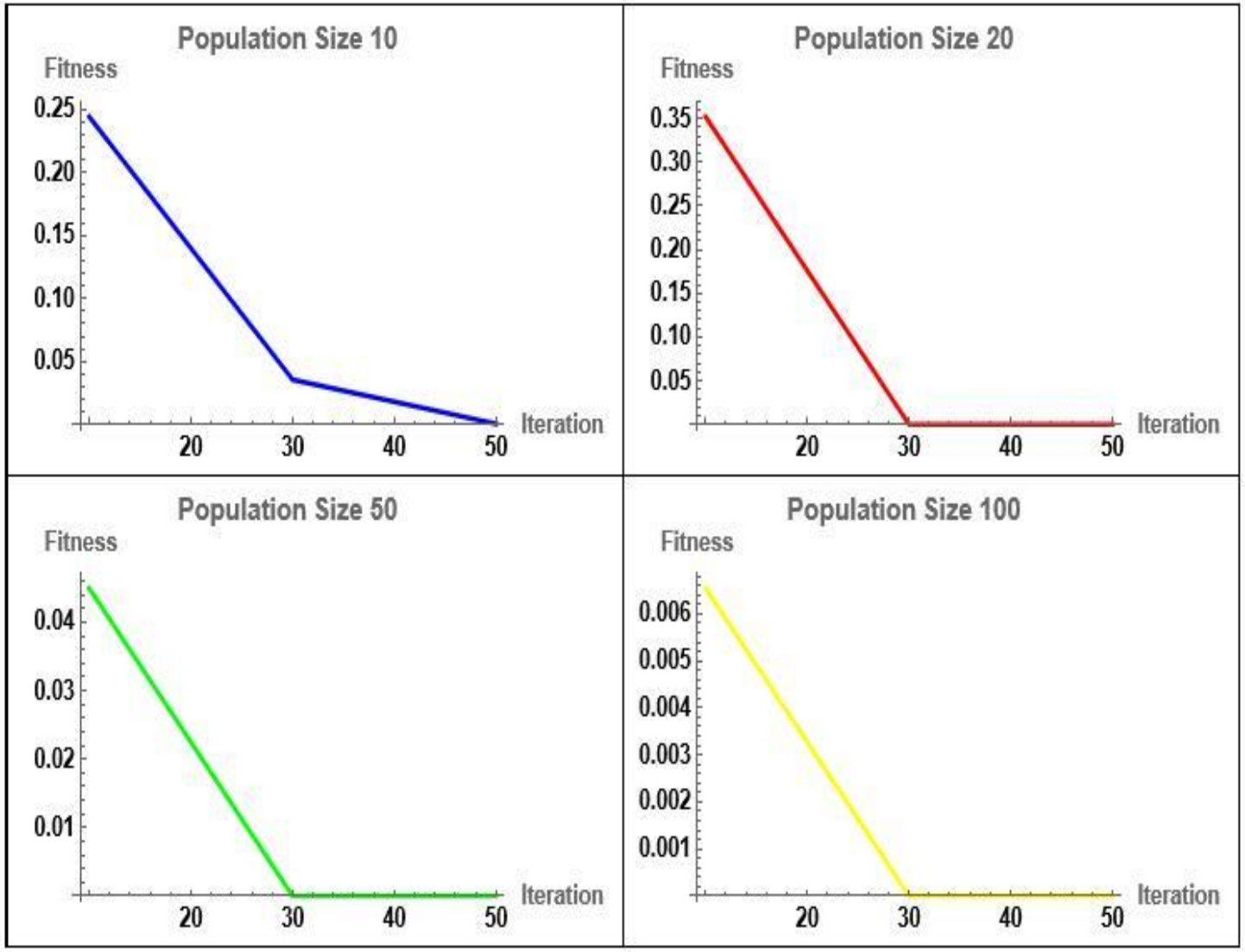




Implementation of Sphere function for dimension 2

Implementation: **Sphere function for dimension 5**

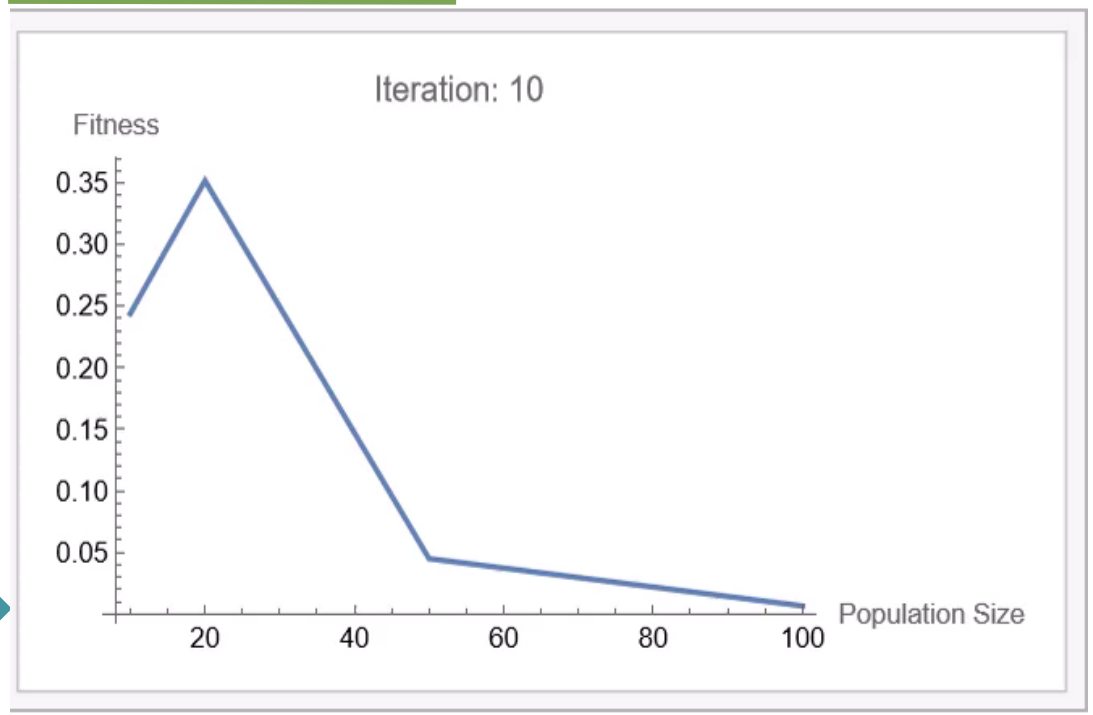
Dimension	Population Size	Iteration Number	Fitness
5	10	10	0.24396
5	10	30	0.0354883
5	10	50	0.000526752
5	20	10	0.351842
5	20	30	0.000132099
5	20	50	0.000000252848
5	50	10	0.0449576
5	50	30	0.00000187366
5	50	50	0.0000000000507158
5	100	10	0.00654089
5	100	30	0.000000151883
5	100	50	0.0000000000000239625

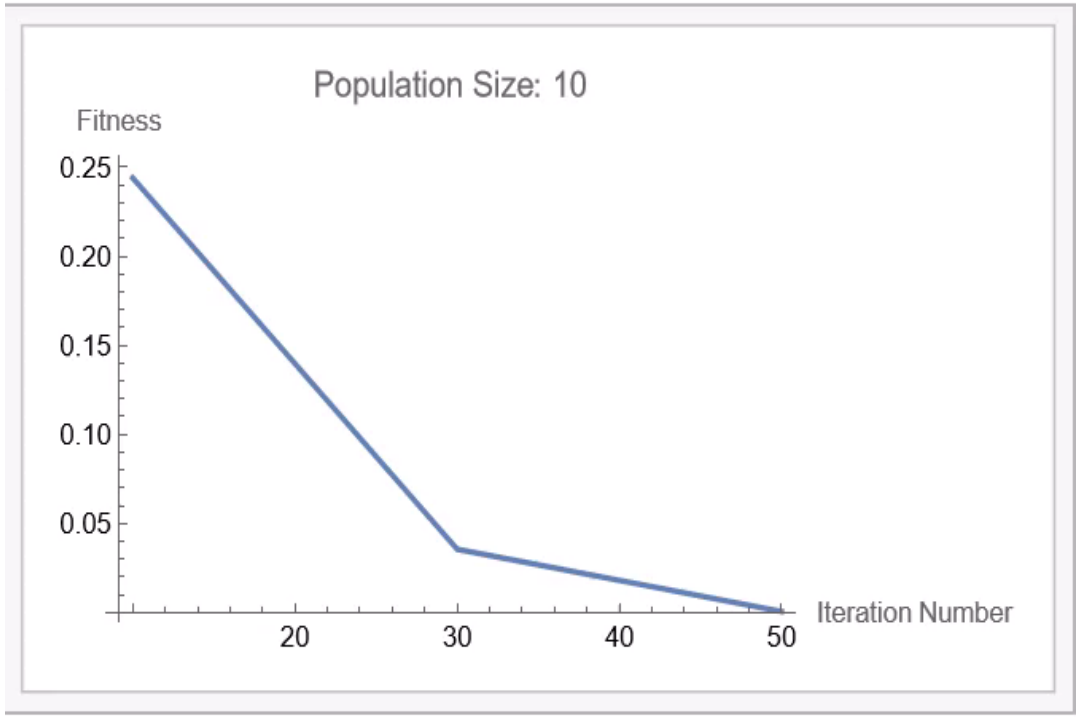


Implementation:

Sphere function for dimension 5

Animation: **Sphere function for dimension 5**

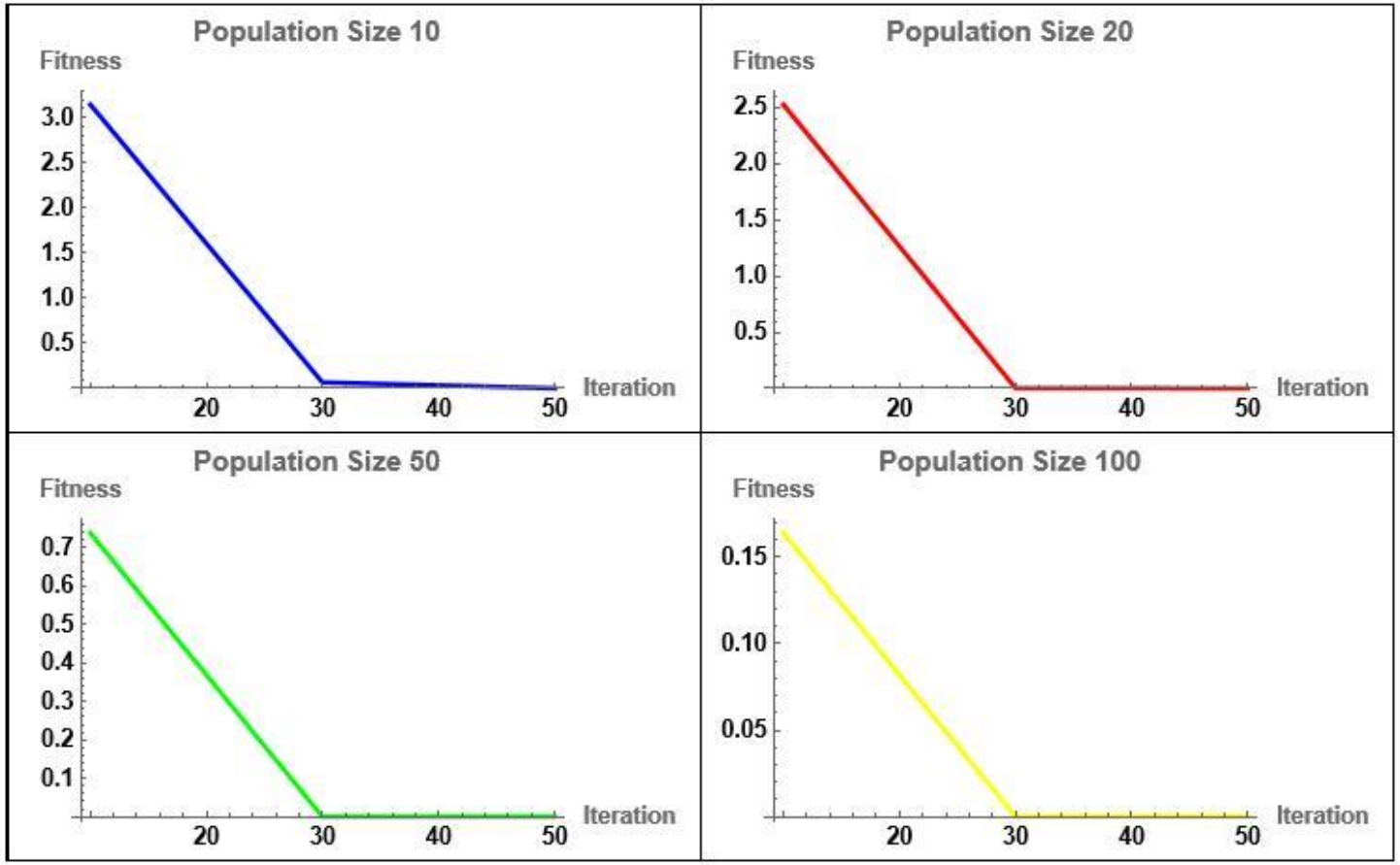




Implementation of Sphere function for dimension 5

Implementation: **Sphere function for dimension 10**

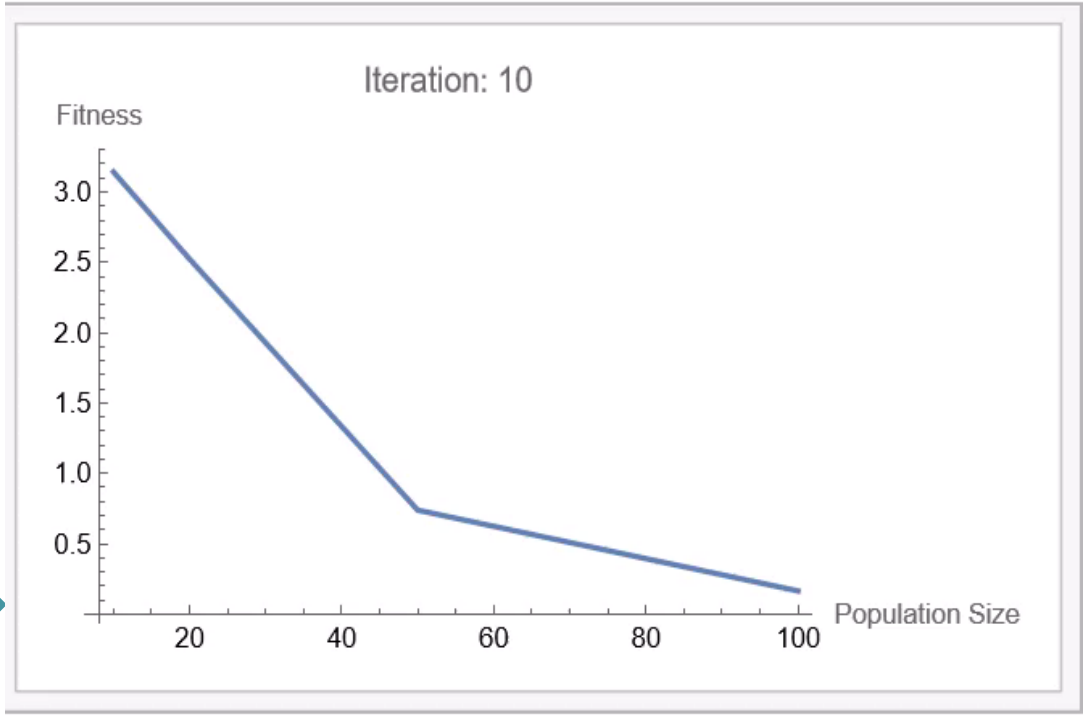
Dimension	Population Size	Iteration Number	Fitness
10	10	10	3.14278
10	10	30	0.0660663
10	10	50	0.00138231
10	20	10	2.52328
10	20	30	0.00295417
10	20	50	0.000104125
10	50	10	0.737973
10	50	30	0.0000240545
10	50	50	0.00000000982341
10	100	10	0.163715
10	100	30	0.00000107959
10	100	50	0.000000000219652

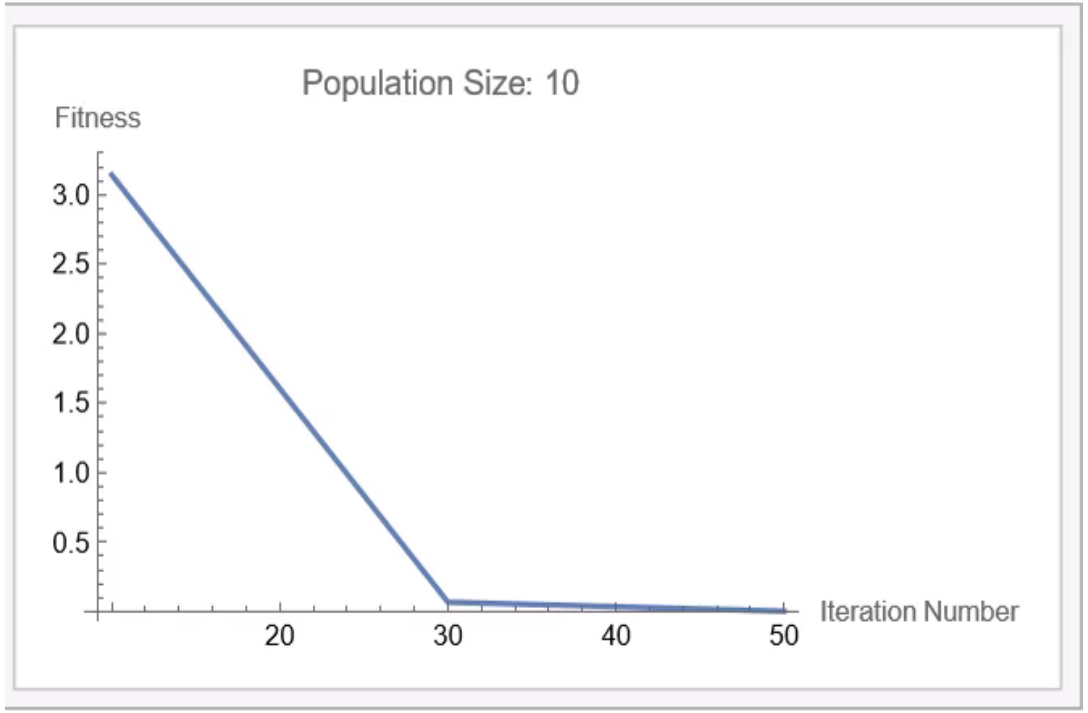


Implementation:

Sphere function for dimension 10

Animation: **Sphere function for dimension 10**

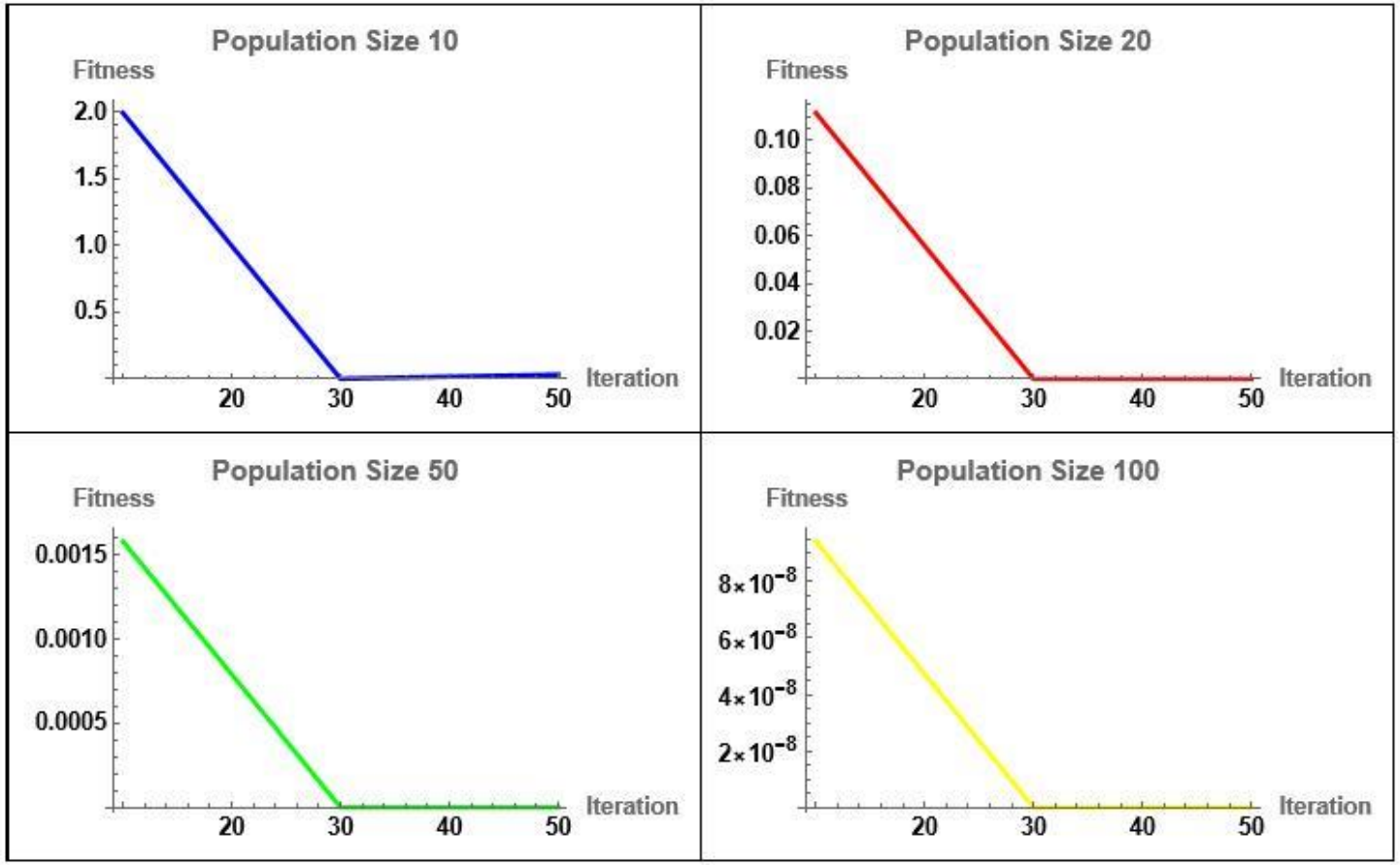




Implementation of Sphere function for dimension 10

Implementation: **Rastrigin function for dimension 2**

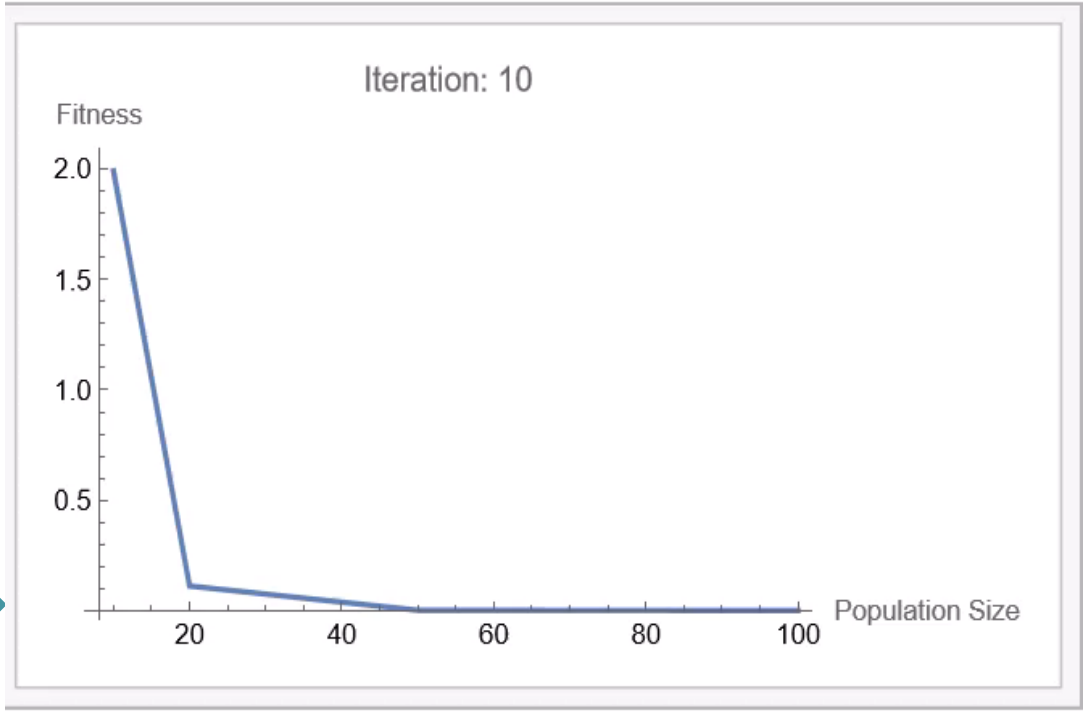
Dimension	Population Size	Iteration Number	Fitness
2	10	10	1.99199
2	10	30	0.00166134
2	10	50	0.0330409
2	20	10	0.111558
2	20	30	0.00000044973
2	20	50	0.0000199621
2	50	10	0.00158206
2	50	30	0.000000000000127898
2	50	50	0.000000000000959233
2	100	10	0.000000942996
2	100	30	0.000000000000710543
2	100	50	0.0

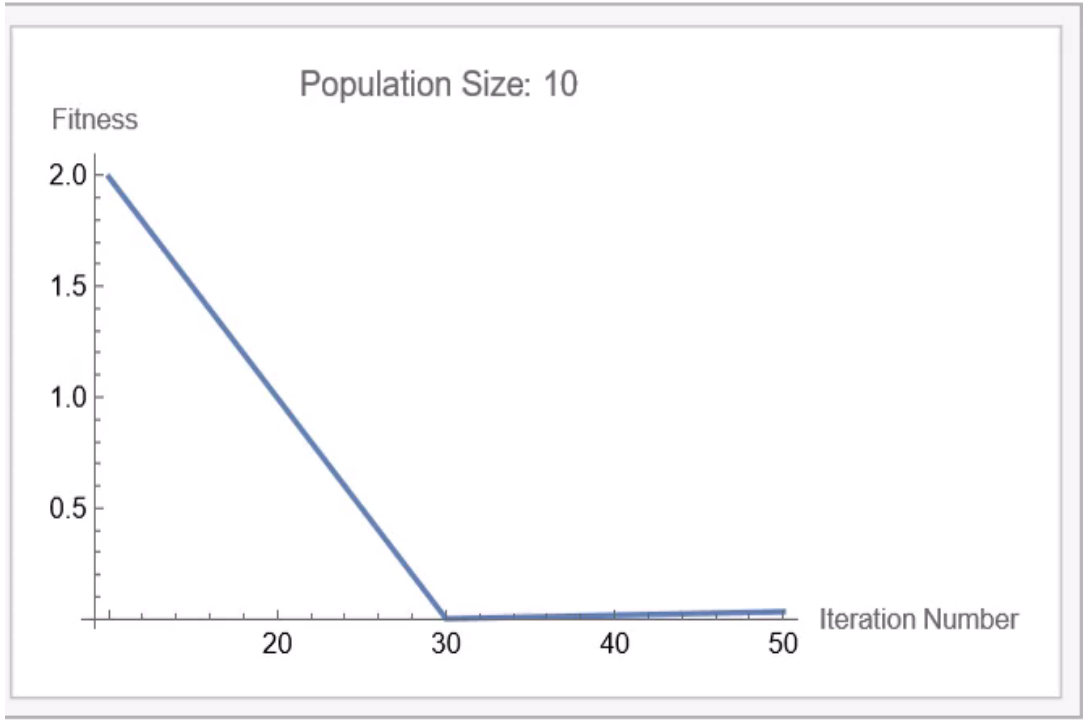


Implementation:

Rastrigin function for dimension 2

Animation: **Rastrigin function for dimension 2**

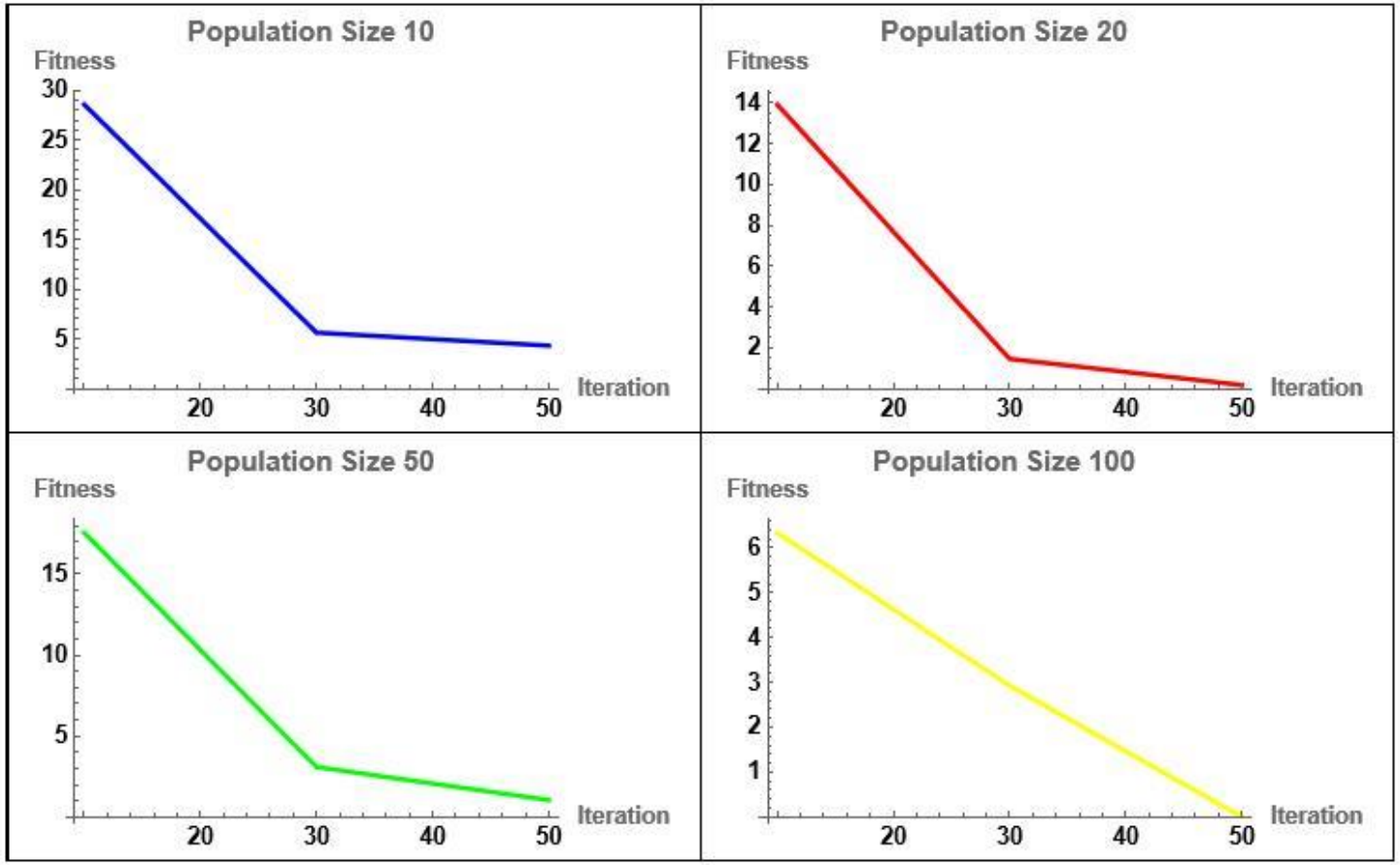




Implementation of Rastrigin function for dimension 2

Implementation: **Rastrigin function for dimension 5**

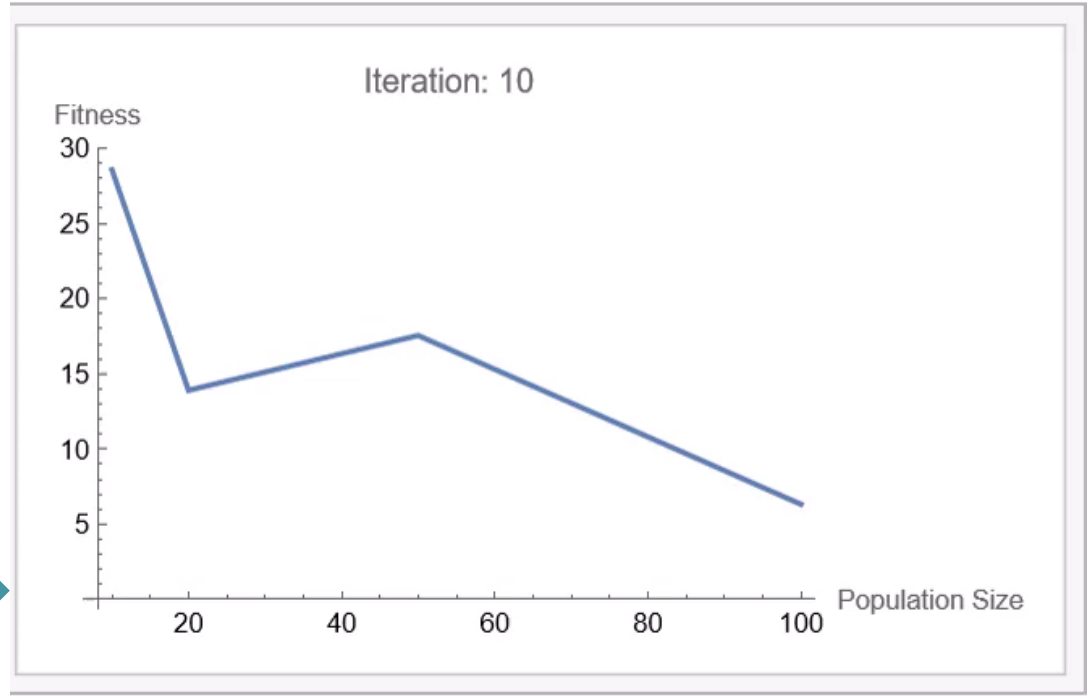
Dimension	Population Size	Iteration Number	Fitness
5	10	10	28.5656
5	10	30	5.61939
5	10	50	4.30767
5	20	10	13.9053
5	20	30	1.44955
5	20	50	0.167691
5	50	10	17.5531
5	50	30	3.08183
5	50	50	1.0382
5	100	10	6.33065
5	100	30	2.92388
5	100	50	0.000000281149

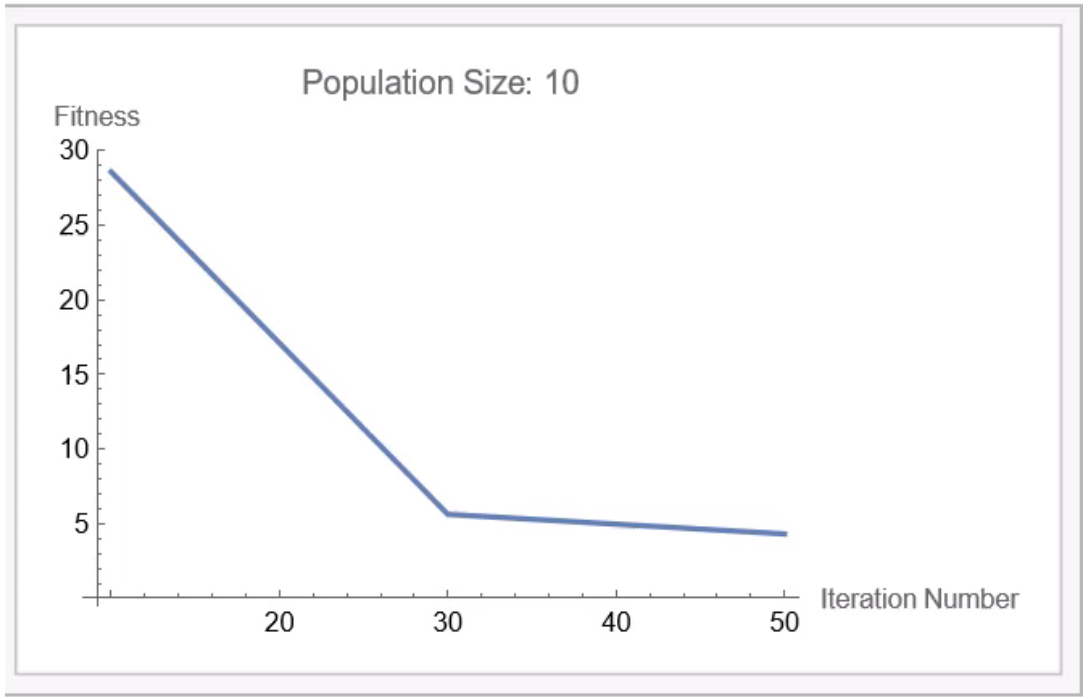


Implementation:

Rastrigin function for dimension 5

Animation: **Rastrigin function for dimension 5**

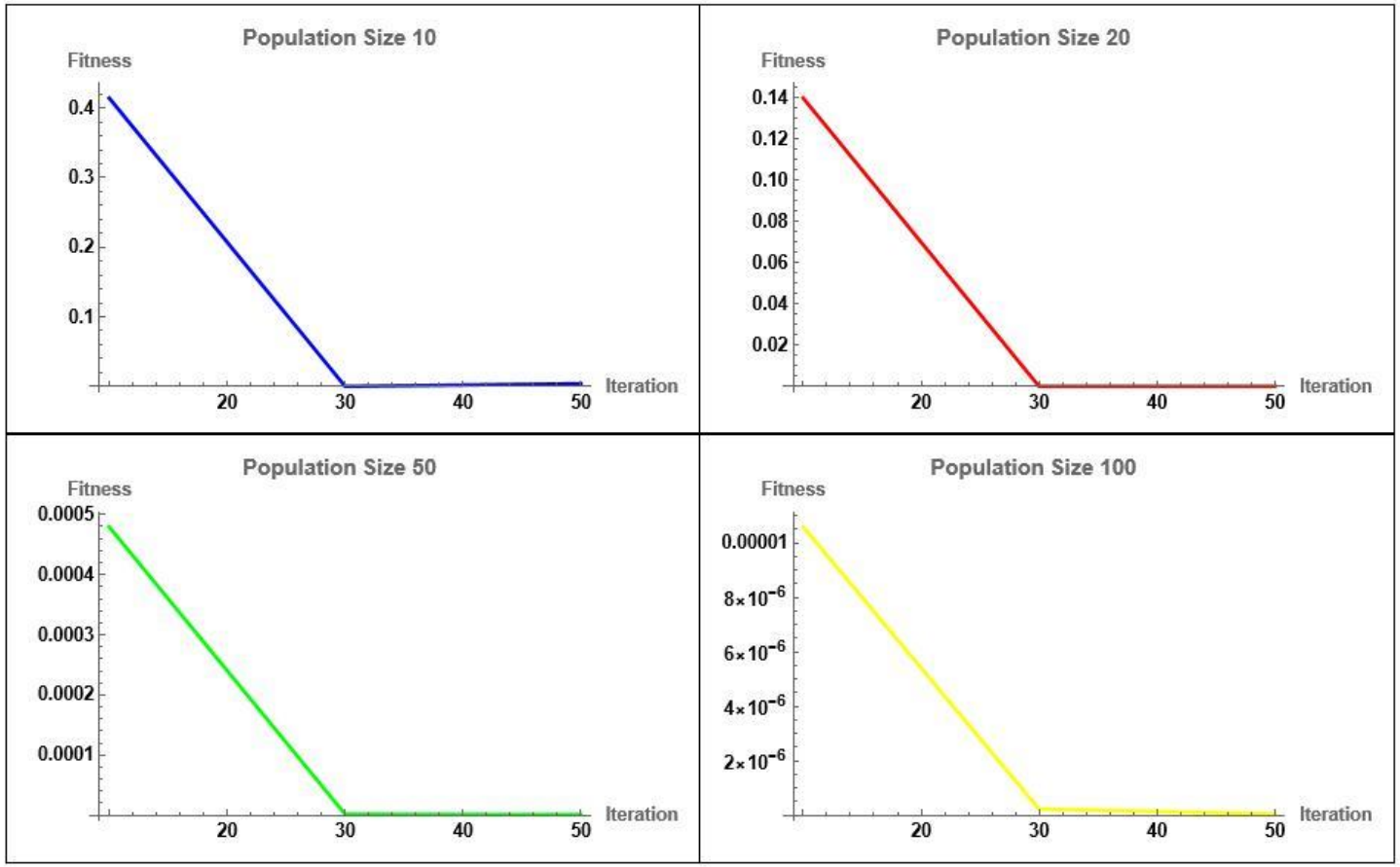




Implementation of Rastrigin function for dimension 5

Implementation: **Rosenbrock function for dimension 2**

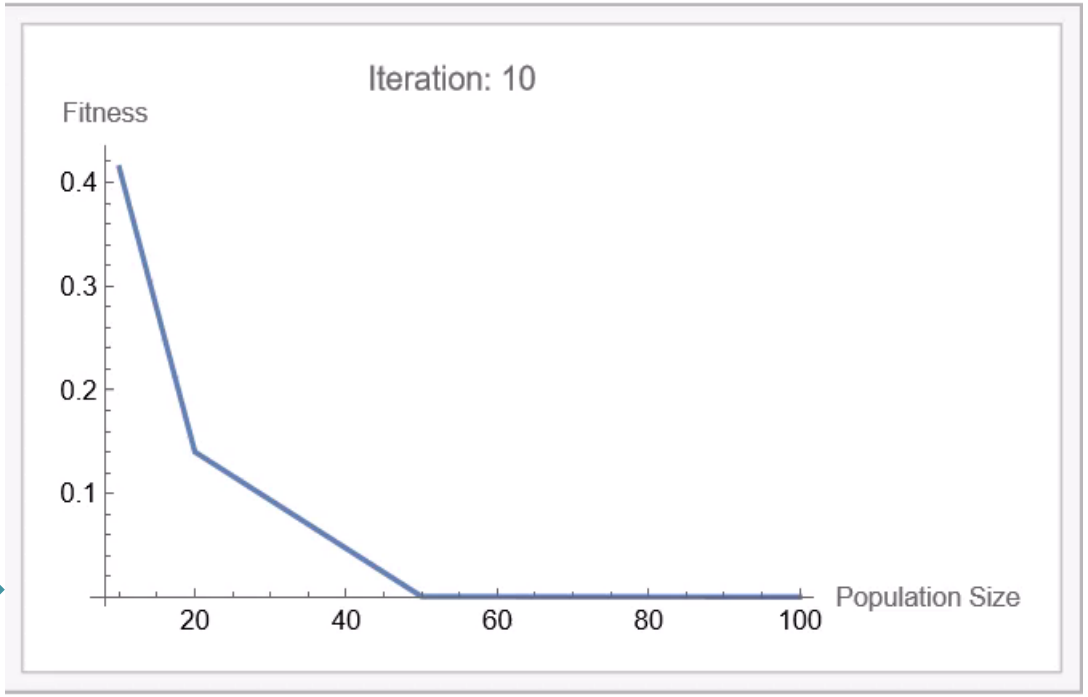
Dimension	Population Size	Iteration Number	Fitness
2	10	10	0.414082
2	10	30	0.0000897467
2	10	50	0.00379143
2	20	10	0.139944
2	20	30	0.0000102165
2	20	50	0.00000286412
2	50	10	0.000479116
2	50	30	0.00000165891
2	50	50	0.000000864358
2	100	10	0.0000105843
2	100	30	0.000000227351
2	100	50	0.0000000429961

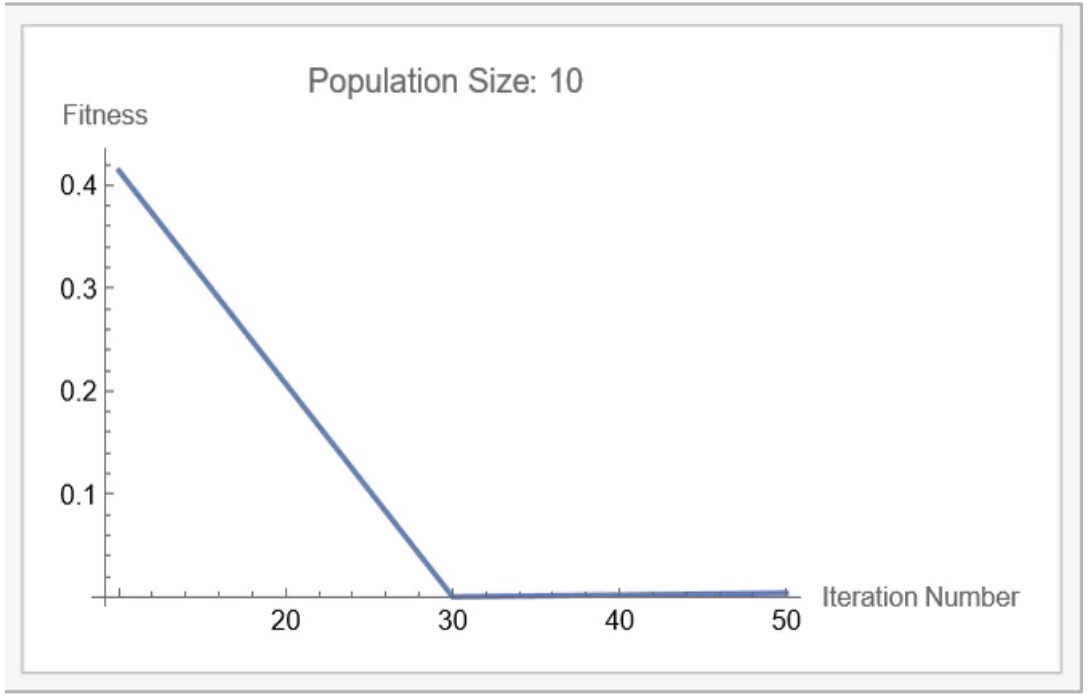


Implementation:

Rosenbrock function for dimension 2

Animation: **Rosenbrock function for dimension 2**

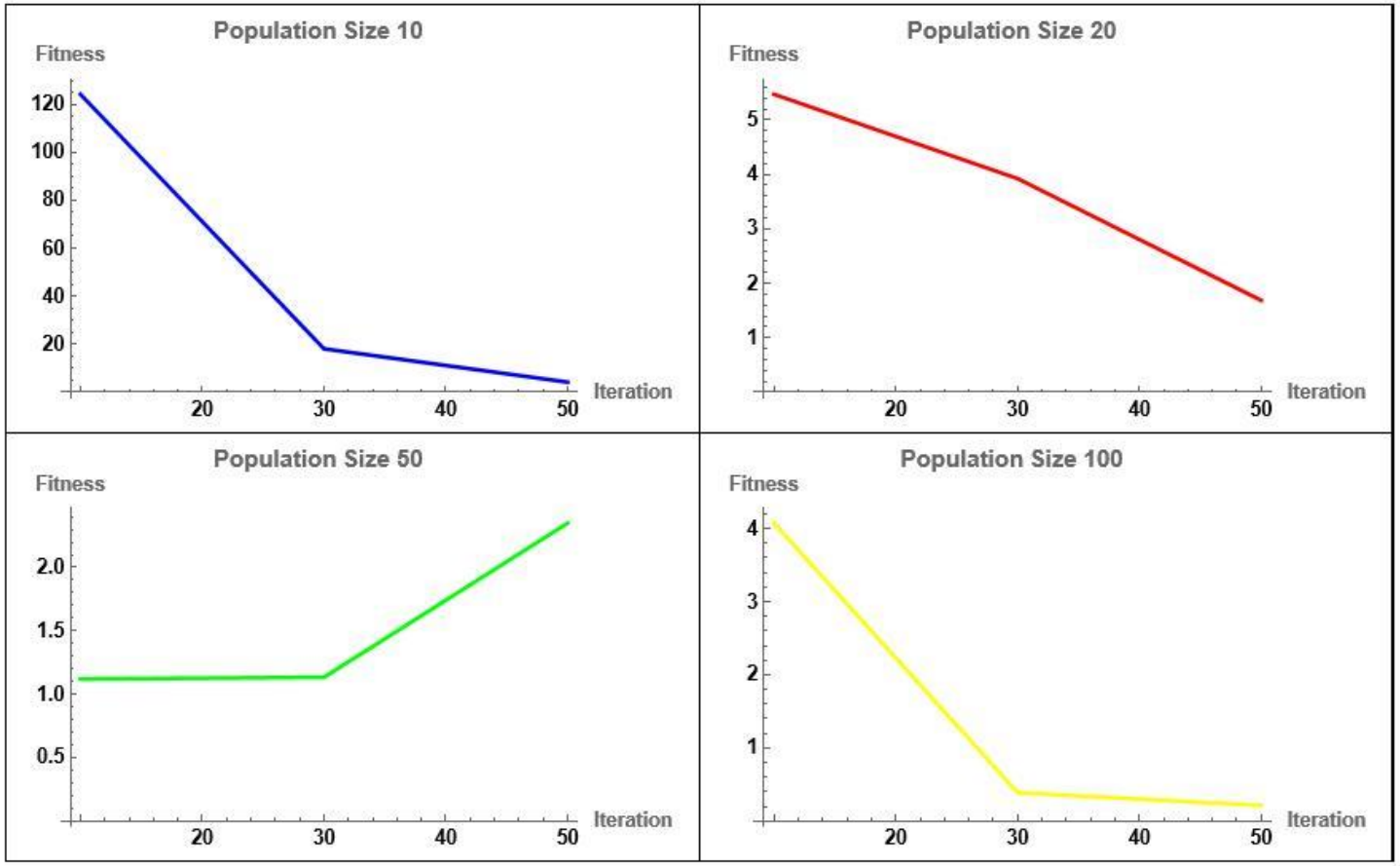




Implementation of Rosenbrock function for dimension 2

Implementation: **Rosenbrock function for dimension 5**

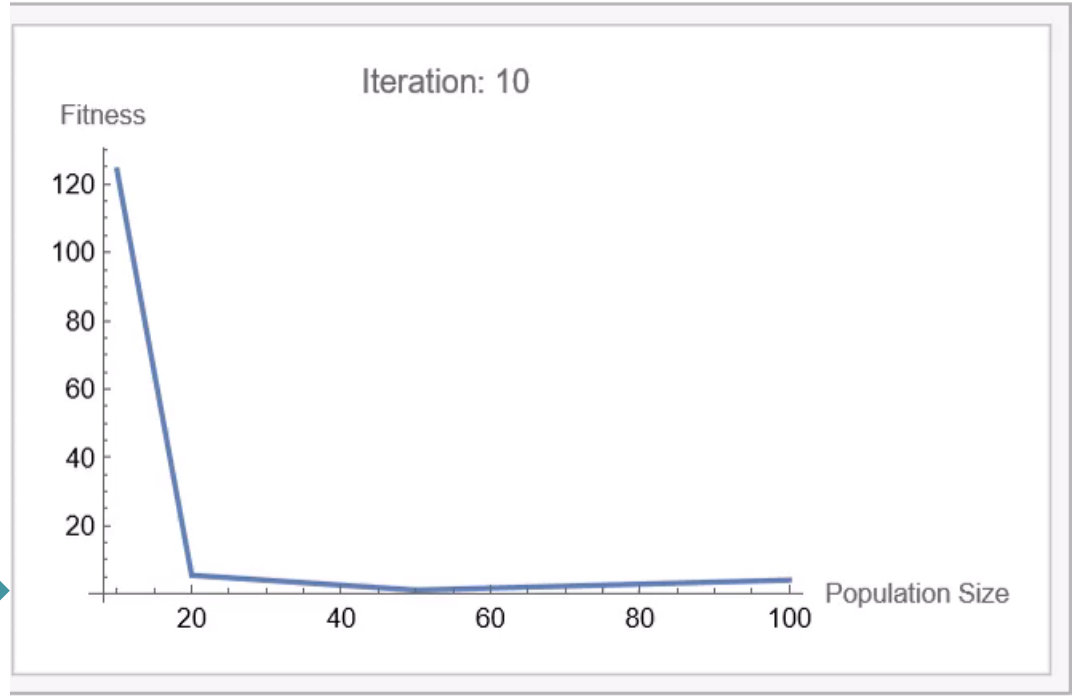
Dimension	Population Size	Iteration Number	Fitness
5	10	10	124.132
5	10	30	18.0843
5	10	50	4.18853
5	20	10	5.46988
5	20	30	3.92218
5	20	50	1.68702
5	50	10	1.12014
5	50	30	1.1331
5	50	50	2.34874
5	100	10	4.07383
5	100	30	0.388195
5	100	50	0.213145

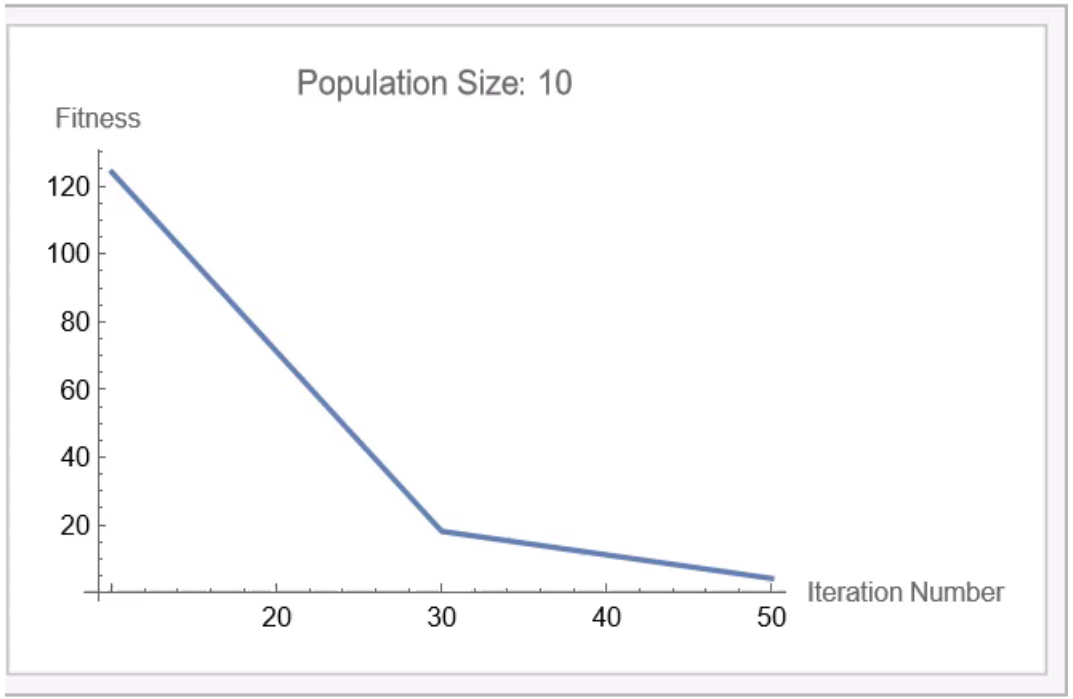


Implementation:

Rosenbrock function for dimension 5

Animation: Rosenbrock function for dimension 5





Implementation of Rosenbrock function for dimension 5

Summary

In this study, we successfully implemented the Grey Wolf Optimizer (GWO) to solve optimization problems using three benchmark functions: Sphere, Rastrigin, and Rosenbrock. The implementation process involved testing the GWO across different dimensions (2, 5, 10), population sizes (10, 20, 50, 100), and iterations (10, 30, 50).

Our results demonstrated the GWO's efficacy in finding optimal solutions, as evidenced by the progressively improving fitness values. We presented and analyzed these results using data visualization techniques, creating animations and detailed plots to highlight the performance trends over iterations. This approach provided clear insights into the GWO's convergence behaviour and robustness across various scenarios.

The findings confirm that the GWO is a powerful metaheuristic algorithm for tackling complex optimization problems, making it a valuable tool for researchers and practitioners in fields requiring efficient optimization solutions. Future work could explore hybrid models combining GWO with other optimization techniques and applying GWO to real-world optimization challenges.

Overall, this study provides a comprehensive understanding of the GWO's capabilities and its practical implementation, showcasing its potential and laying the groundwork for further exploration and application in optimization problems.

Thank you for your patience and understanding