ASSIGNMENT #2

ENGR 5010G – Advanced Optimization

Course Instructor: Prof. Md Asif Khan

Bao Khang Nguyen 100975684

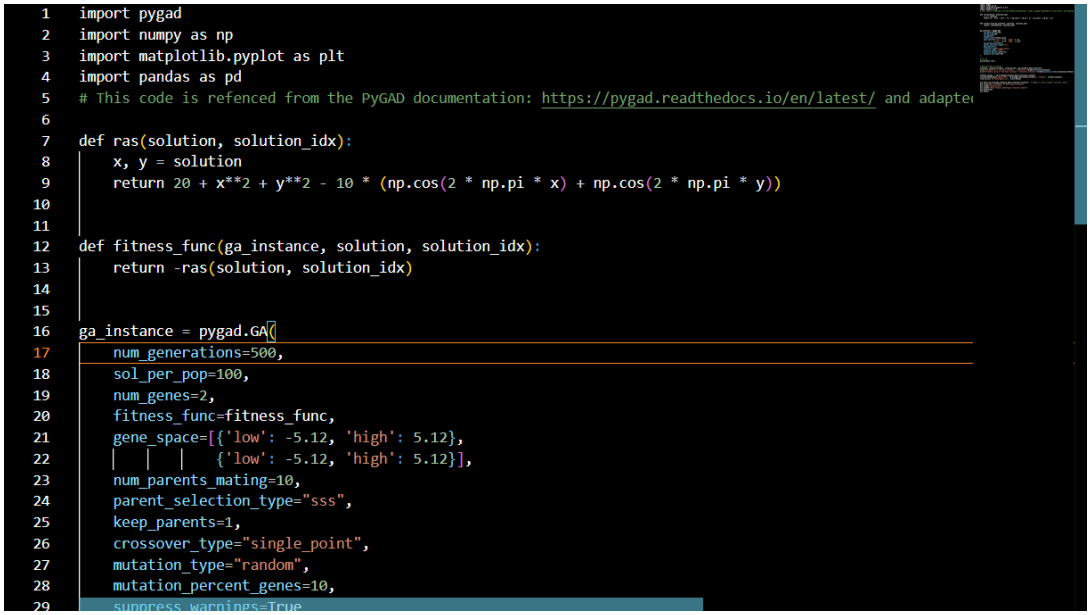
July 30, 2025

All the algorithms will be conducted in python for convenience

1. Genetic Algorithm

Genetic Algorithm (GA) is conducted on PyGad library in python for convenience. PyGad offers a dedicated library especially for GA, so utilizing it will make the code more reliable and trustworthy.

The code is as follows:

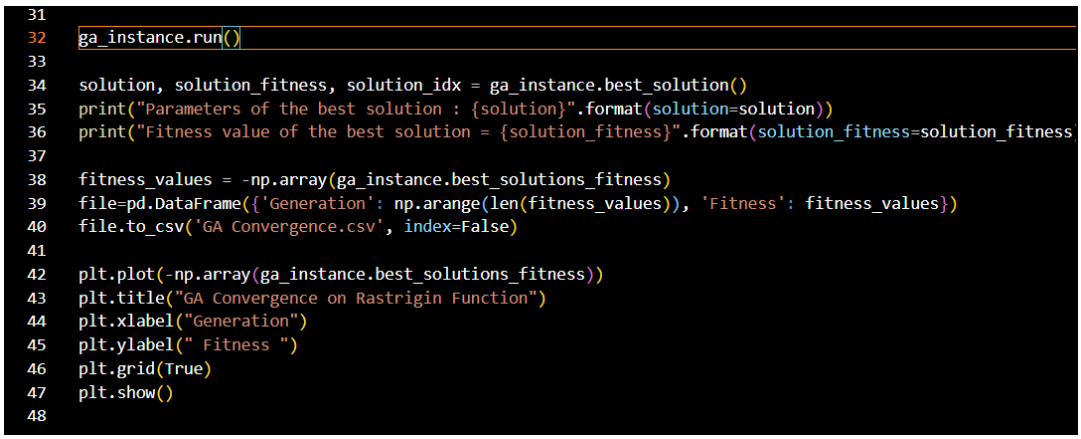


```python
import pygad
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# This code is refenced from the PyGAD documentation: https://pygad.readthedocs.io/en/latest/ and adapte

def ras(solution, solution_idx):
    x, y = solution
    return 20 + x**2 + y**2 - 10 * (np.cos(2 * np.pi * x) + np.cos(2 * np.pi * y))


def fitness_func(ga_instance, solution, solution_idx):
    return -ras(solution, solution_idx)


ga_instance = pygad.GA(
    num_generations=500,
    sol_per_pop=100,
    num_genes=2,
    fitness_func=fitness_func,
    gene_space=[{'low': -5.12, 'high': 5.12},
                {'low': -5.12, 'high': 5.12}],
    num_parents_mating=10,
    parent_selection_type="sss",
    keep_parents=1,
    crossover_type="single_point",
    mutation_type="random",
    mutation_percent_genes=10,
    suppress_warnings=True
```



```python

ga_instance.run()

solution, solution_fitness, solution_idx = ga_instance.best_solution()
print("Parameters of the best solution : {solution}".format(solution=solution))
print("Fitness value of the best solution = {solution_fitness}".format(solution_fitness=solution_fitness

fitness_values = -np.array(ga_instance.best_solutions_fitness)
file=pd.DataFrame({'Generation': np.arange(len(fitness_values)), 'Fitness': fitness_values})
file.to_csv('GA Convergence.csv', index=False)

plt.plot(-np.array(ga_instance.best_solutions_fitness))
plt.title("GA Convergence on Rastrigin Function")
plt.xlabel("Generation")
plt.ylabel(" Fitness ")
plt.grid(True)
plt.show()

```

Code Description:

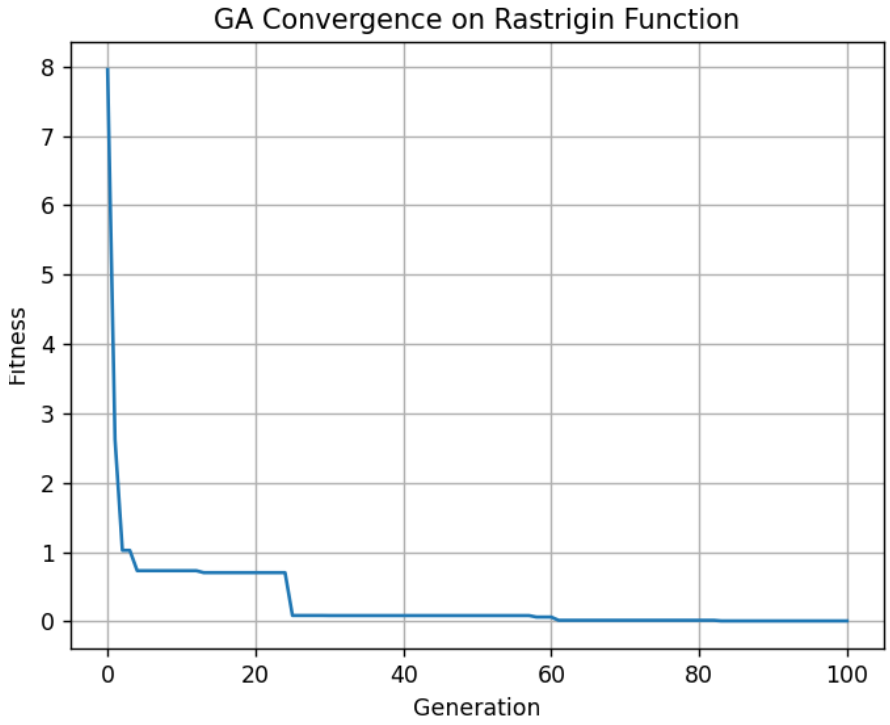
First, we define the rastrigin function for later use as fitness functions. Since GA algorithm offer by PyGad only calculates the global maxima the function is considered as the opposite of the rastrigin function (- rastrigin function). The fitness function is defined for later used in the genetic algorithm function.

The parameter of our GA is as follows: 500 generations, 100 solutions per generation, 10 parents are used for cross over and mutation, keep 1 parent in the 10 parent to the next generation with mutation is carried out randomly with 10% chance.

Best fitness table

Generation	Fitness
0	7.9584995
1 to 12	2.624324395 to 0.732739455
13 to 57	0.704894936 to 0.08571304
58 to 82	0.064541076 to 0.01762365
82-100	0.008851

Converge graph:



Analysis:

GA has a medium convergence speed with nearly 60 generations. The solution quality is good with the final value match the global minimum. However, GA is high sensitive due to dependence on crossover rate and mutation rate

2. Differential Evolution (DE)

The same principle is done with DE, and we will use scipy libraries with dedicated DE functions to perform the benchmarking for the rastrigin function

The code is as follows:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import differential_evolution
import pandas as pd
# Rastrigin function optimization using Differential Evolution is refenced based on the manual from SciP
#Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.h
def ras(X):
    x, y = X
    return 20 + x**2 + y**2 - 10*(np.cos(2*np.pi*x) + np.cos(2*np.pi*y))

convergence=[]

def record(intermediate_result):
    convergence.append(intermediate_result.fun)
    print(intermediate_result.x)



result=differential_evolution(ras, bounds=[(-5.12, 5.12), (-5.12, 5.12)], maxiter=500, popsize=100,
                              mutation=0.5, recombination=0.7,seed=1, callback=record)
```

```python
file= pd.DataFrame({'Generation': np.arange(len(convergence)), 'Fitness': convergence})
file.to_csv('DE Convergence.csv', index=False)

plt.plot(convergence, label='Fitness', color='blue')
plt.title('Convergence of Differential Evolution on Rastrigin')
plt.xlabel('Generation')
plt.ylabel('Function Value')
plt.ylim([0, 10])
plt.xlim([0, len(convergence)])
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

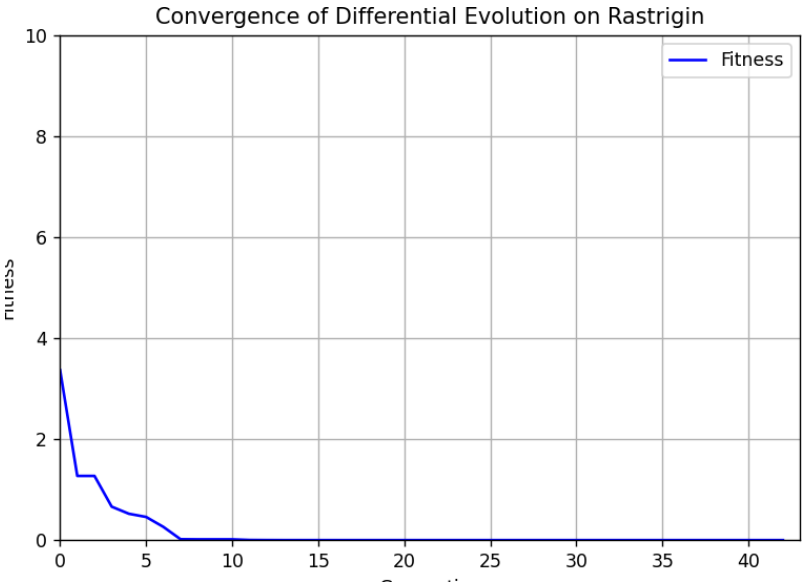
Code description:

The code is based on the SciPy DE algorithm manual and adjusted to meet the functional requirements. Our parameter running DE is kept the same and as follows: 500 generations, 100 solutions per generation, mutation probability is 0.5, recombination rate is 0.7. The seed value here is considered as a way to fix our initial population so that we get consistent results. The plotting part is the same as GA. The first 2 line are for creating a csv file and plotting the convergence of DE

Fitness Table

Generation	Fitness
0	3.364
1 to 13	1.27 to 0.00012
14 to 33	7.52E-05 to 3.55E-15
33 to 42	0

Convergence graph



Analysis:

DE has the fastest convergence speed out of the 3 algorithms. Solution quality is good with the algorithm reaching the global minimum at (0,0). DE is still sensitive to parameter setting like mutation factor and crossover rate, as this can impact convergence speed.

3. Particle Swarm Optimization (PSO)

The same principle is done with PSO, and we will use pyswarms libraries with dedicated PSO functions to perform the benchmarking for the rastrigin function

The code is as follows:

```python
import numpy as np
import matplotlib.pyplot as plt
import pyswarms as ps
import pandas as pd
# Particle Swarm Optimization (PSO) for Rastrigin Function is referenced from the pws library documentat
#Link: https://pyswarms.readthedocs.io/en/latest/tutorials/optimization.html
def ras(X):
    x = X[:, 0]
    y = X[:, 1]
    return 20 + x**2 + y**2 - 10*(np.cos(2*np.pi*x) + np.cos(2*np.pi*y))

options = {'c1': 0.5, 'c2': 0.3, 'w': 0.9}
optimizer = ps.single.GlobalBestPSO(n_particles=100, dimensions=2, options=options,bounds=([-5.12, -5.12
cost,pos= optimizer.optimize(ras, iters=500)

file = pd.DataFrame({'Generation': np.arange(len(optimizer.cost_history)),'Fitness': optimizer.cost_hist
file.to_csv('PSO Convergence.csv', index=False)

plt.plot(optimizer.cost_history, label='Fitness', color='blue')
plt.title('PSO Convergence  on Rastrigin Function')
plt.xlabel('Generation')
plt.ylabel('Fitness Value')
plt.grid(True)
plt.legend()
plt.show()
```

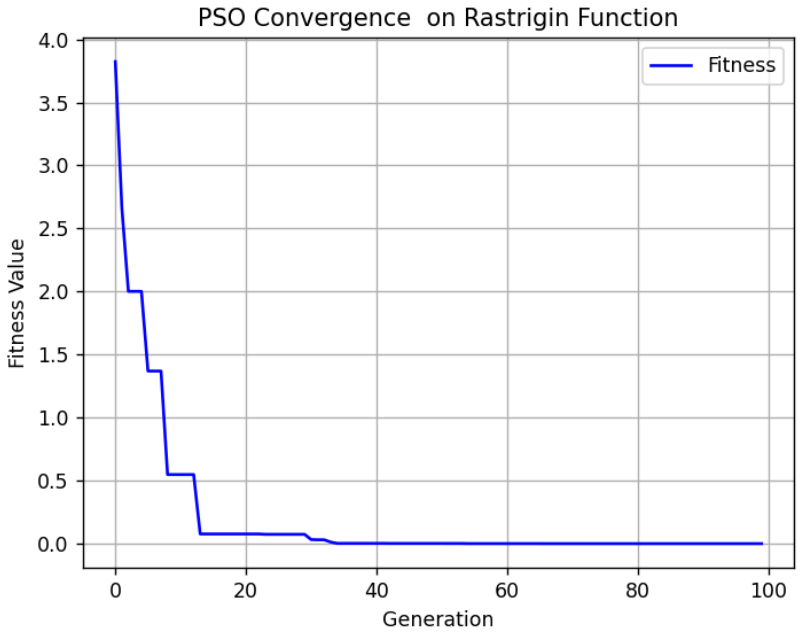
Code description:

The first function definition is for creating rastrigin function. The options functions are the parameter for the PSO. C1 is the cognitive coefficient, C2 is the social coefficient, ,W is the inertia coefficient. This value is passed onto the PSO function to perform the optimization. The plotting part is the same as the previous 2 algorithm with 1 part dedicated to converting to csv file and the last part is for plotting the convergence.

Fitness Table

Generation	Fitness
0	3.823917471
1 to 29	2.664068028 to 0.073776843
30 to 70	0.031982918 to 5.66E-05
71 to 100	4.65E-06

Convergence Graph



Analysis:

PSO has a quick convergence speed with approximately 30 generations. The solution quality is good with the algorithm reaching the global minimum. However, it is still highly sensitive to parameter settings like c1,c2 and w to reach global minimum. Changing in these parameters can possibly impact convergence speed of the system.