**Optimization, Modeling, and Simulation of Physics-based Systems**

**Modeling & Simulation Tools for Industrial Applications**

**Introduction**

Effective design and operation of mechanical and physical systems can lead to significant savings in resources such as materials, labor, and time. There are a variety of methods available for optimizing these systems, both theoretically and numerically. Among these methods, Newton's method is a fundamental tool used in numerical analysis, optimization, and control. Its wide range of applications in scientific research, industrial settings, and financial practices has made it an essential technique for optimizing engineering problems. In fact, it serves as a foundation for some of the most effective procedures in linear and nonlinear programming.

However, due to the challenges associated with guessing initial values and dealing with nonconvex problems, genetic algorithm (GA) has become a popular alternative technique for solving complex engineering problems. GA is a hybrid computational approach that involves selection, crossover, and mutation operators to manage a searching system strategy. Its main idea is derived from natural selection and genetic concepts, and it is motivated by the evolution process. GA is part of a larger class of evolutionary algorithms used in informatics and computational mathematics.

Evolutionary algorithms are particularly useful for problems that don't have a well-defined efficient solution. They help to achieve optimal solutions and are used in modeling and simulation where randomness function is utilized. GA, specifically, is a candidate solution for optimizing problems that aim to develop a better solution. Each candidate solution has a set of characteristics or an array of random numbers that can be evolved and changed. Evolution begins with a group of randomized individuals and is an iterative process where the population is reproduced by an appropriate method of generation for each reproduction. At each generation step, the fitness of each individual in the population is measured, usually using the value of the objective function being solved. Based on their fitness, sufficiently fit individuals are probabilistically selected from the existing population, and their genes are modified to create a new generation for all. The newer generation of candidate solutions is then utilized over the next generation of the process. The algorithm typically ends when a maximum number of generations have been generated or when the optimization is satisfied.

In this study, we compare Newton's method and GA method in both theoretical and numerical contexts. We consider a simple minimization problem to highlight the advantages and challenges associated with each method. Additionally, we evaluate the advantages of GA in solving the swarm of unmanned aerial vehicles (UAVs).

**Methodology**

* Theoretical Analysis of optimization problem

Optimization problem for minimization is as follows.

(1)

The minimum point of function, has to satisfy the following equation.

(2)

Once one solution to be satisfied to eq. (2) is found, if , the point is minimum point and oppositely, maximum value.

* Newton method

The Taylor’s expansion of is given by following equation.

(3)

Attains its extremum when solves the linear equation (4).

(4)

From eq. (4), provided that is a twice-differentiable function and the initial guess is selected close enough to optimal point, the sequence defined by equation (5).

(5)

* Genetic Algorithm

It is assumed that the number of design parameters is as follows.

, (6)

1. Compute initial population.

, (7)

where is the number of strings.

1. Select individuals for reproduction.

Individuals for reproduction are selected according to the value of objective function(fitness). At this time, fitness for every individual is determined as following equation.

, (8)

In this stage, first, the population is rearranged according to the fitness value. After it, given the number of parents, , first genetic strings are selected as parent string.

1. Create offspring’s by crossing individuals.

Crossover operation is as follows.

, (9)

, (10)

By eq. (9) and (10), children genetic strings are obtained, where and are random numbers different for each component between 0 and 1.

1. Mutation of some individuals

genetic strings, () are replaced randomly.

Finally, new generation is following.

(11)

1. Repeat 1~4.

**Theory-Based Exercises**

* **Hessian and Gradient of function**

,

,

* **Minimum values of function**

If is minimum value, and . And then if is maximum value, and .

For , , thus, . . Finally, is the minimum value of function .

For , , Thus, and .

And then, and .Thus, is also minimum value of function .

* **Challenging using a gradient-based method for minimizing**

Because a gradient-based method is just one of local optimization methods and has some extreme points.

* **Velocity formulation for UAV Swarm**

From Newton second law,

(12)

where and are the ground speed of UAV and wind speed, respectively and it is assumed that . And then is propulsion of UAV and the second term of right side of eq. (12) is drag.

Then,

(13)

, , (14)

In here, is airspeed of UAV.

Thus,

(15)

Because and are constant,

(16)

From eq. (16),

(17)

(18)

Based on initial condition of , .

(19)

When , the maximum value of is .

* **Forward Euler equation for time discretization**

The Forward Euler equation of eq. (15) for time discretization is

(20)

is the airspeed of UAV at current time step and is the airspeed of UAV at previous time step.

And then, and are propulsion of UAV and drag at previous time step, respectively.

* **If , or are negative?**

From eq. (20), if is negative, and if is negative, .

* **Analytical solution for gradient of surface**

Rearranging equation in terms of surface,

* **Incidence angle and**

Incidence angle:

Velocity after reflection:

1. ,

1. ,

1. ,

* **Tracking ray**

From the fact of flat surface,

Writing velocity as vector, .

Thus, velocity after reflection,

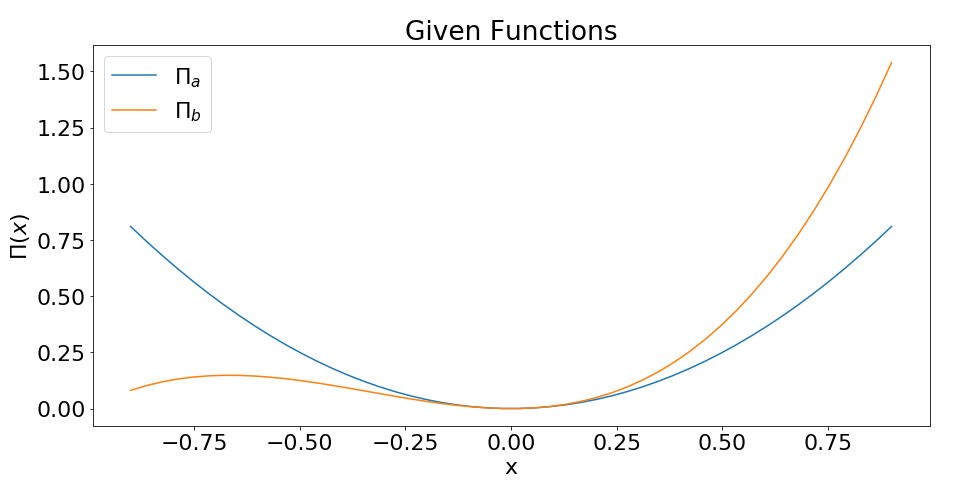
The time for ray to reach to ground:

Thus,

Finally, and total time is .

**Coding Exercises**

* **Define both objective functions as symbolic functions**
* import numpy as np
* p = {
* 'x': np.arange(-0.9, 1, 0.1), # Spatial domain of problem
* 'TOL': 1e-8, # Tolerance for Newton's method
* 'maxit': 20, # Max number of iterations
* 'x0': np.array([-0.8, -0.4, 0, 0.4, 0.8]), # Initial conditions for Newton's method
* 'parents': 12, # Number of parents for evolutionary algorithm
* 'S': 50, # Number of design strings
* 'G': 100, # Number of generations for evolutionary algorithm
* 'dv': 1, # Number of design variables
* 'SB': [-1, 1] # Search bounds for design variables
* }
* p['piA']= [x\*\*2 for x in p['x']], # First given function
* p['piB']= [x\*\*3+x\*\*2 for x in p['x']], # Second given function
* **Plot both objective functions**



* **Write a function using Newton’s method**

############################### Problem 2.2 ###################################

def myNewton(func, x0, TOL, maxit):

its = 0

f\_df = func.diff(x)

df = lambdify(x,f\_df)

f\_ddf = f\_df.diff(x)

ddf = lambdify(x,f\_ddf)

hist = np.zeros([1, maxit])

while its<maxit:

sol = x0 - df(x0)/ddf(x0)

if np.abs(sol)<TOL:

break

hist[0][its] = sol

its = its + 1

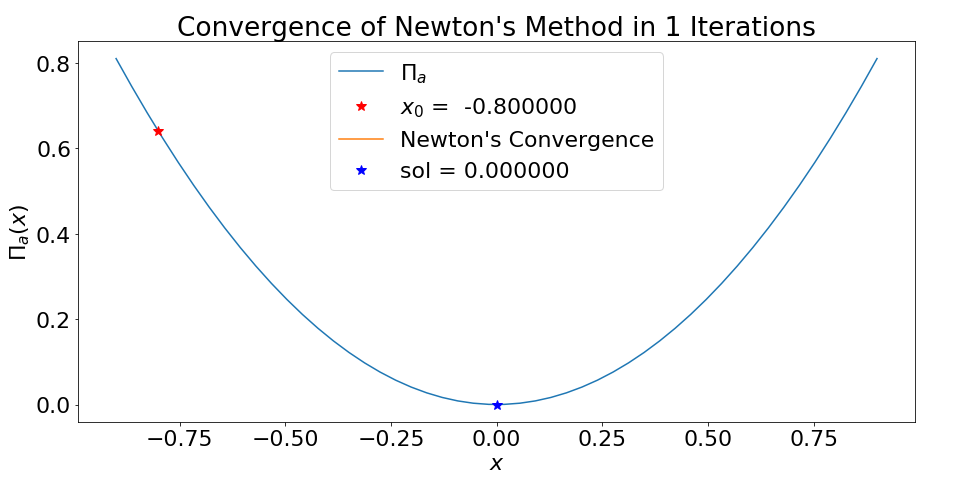
x0 = sol

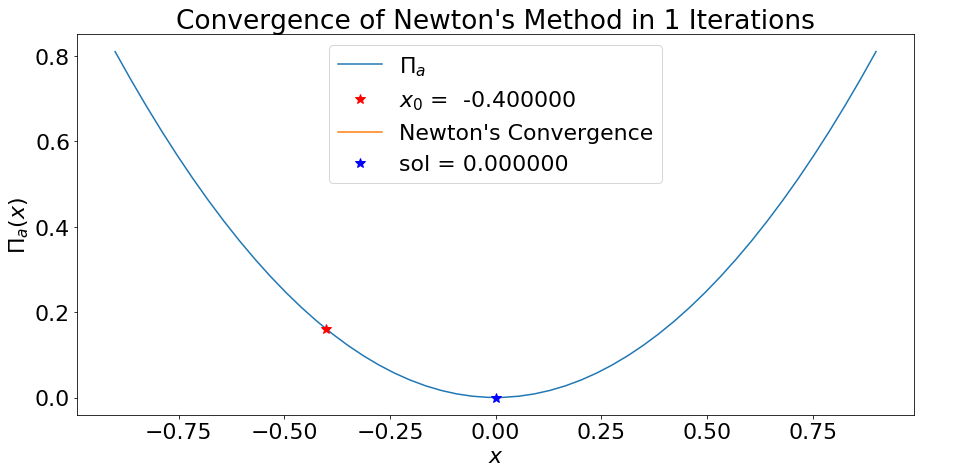
hist = hist[0][0:its]

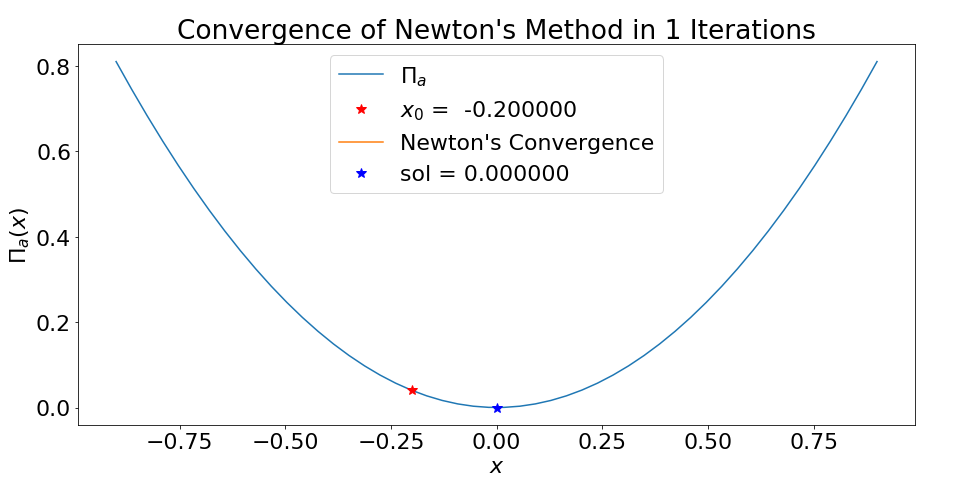
its = its+1

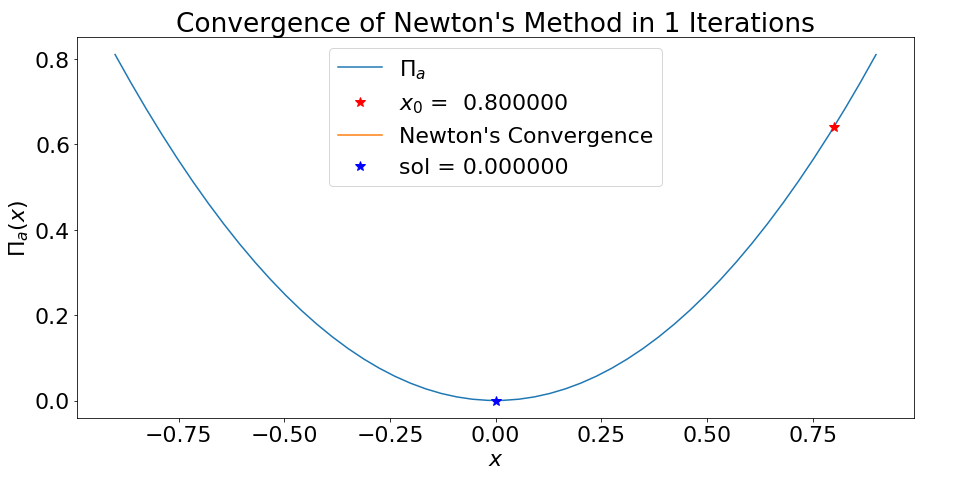
return sol, its, np.array(hist,dtype=object)

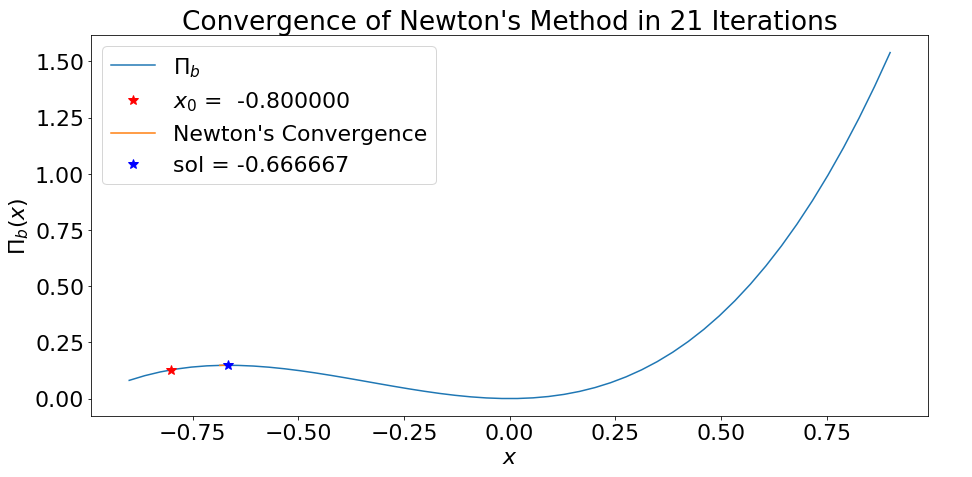
* **Minimizing and using Newton’s Method**

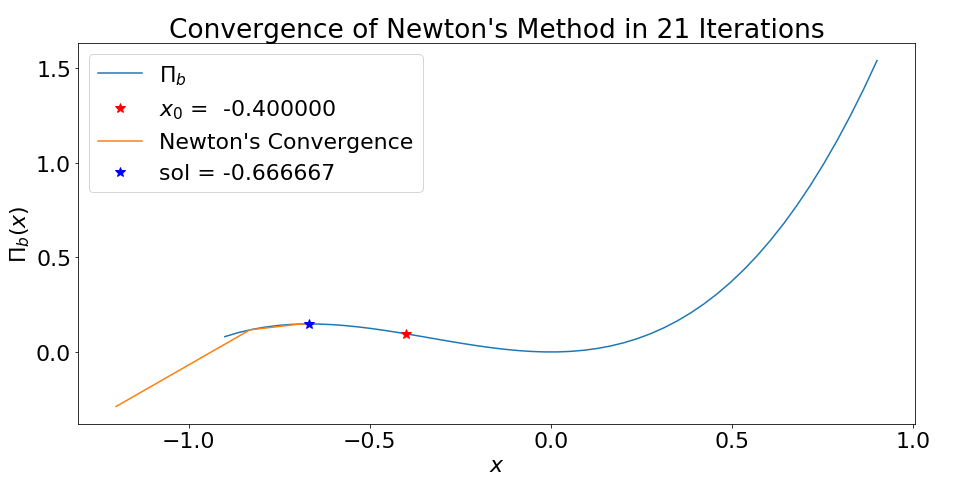


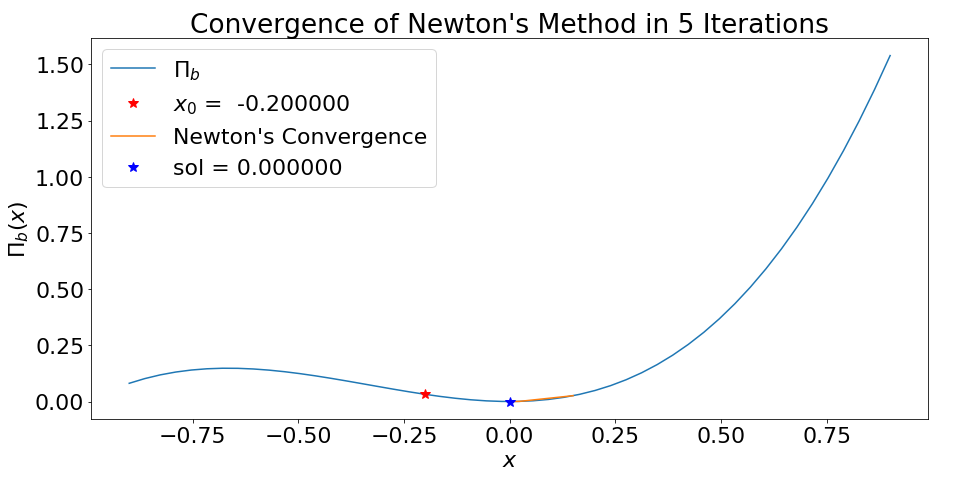


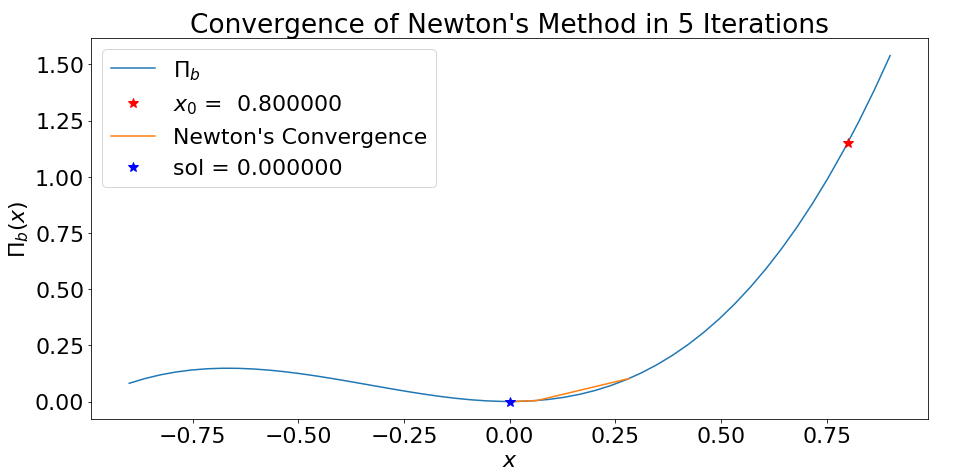












**-Write Genetic Algorithm function**

########################### Problem 2.4 #######################################

def myGA(parents, G, S, dv, func, SB):

cost = np.zeros(S) # All costs in an individual generation

func\_sym = lambdify(x,func)

# Generate initial rnadom population for all generations

Lambda = np.zeros([G, S])

# Generate initial random population

Lam = (SB[1] - SB[0])\*np.random.rand(dv,S) + SB[0]

Orig = Lam

Child = parents #Number of Childrens

Par = parents #Number of Parents

for i in range(G): # Loop through generations

# Calculate fitness of unknown design string costs

for j in range(S): # Evaluate fitness of strings

cost[j] = func\_sym(Lam[:,j])

# Sort cost and design strings based on performance

ind = np.argsort(cost)

cost = np.sort(cost)

Lam = Lam[:,ind]

# Generate offspring radnom parameters

phi = np.random.rand(dv,Child)

ind1 = range(0,Child,2)

ind2 = range(1,Child,2)

# Concatonate original parents children, and new random strings all together into new design string array

Lam = np.hstack((Lam[:,0:Par], phi[:,ind1]\*Lam[:,ind1] + (1-phi[:,ind1])\*Lam[:,ind2],

phi[:,ind2]\*Lam[:,ind2] + (1-phi[:,ind2])\*Lam[:,ind1],

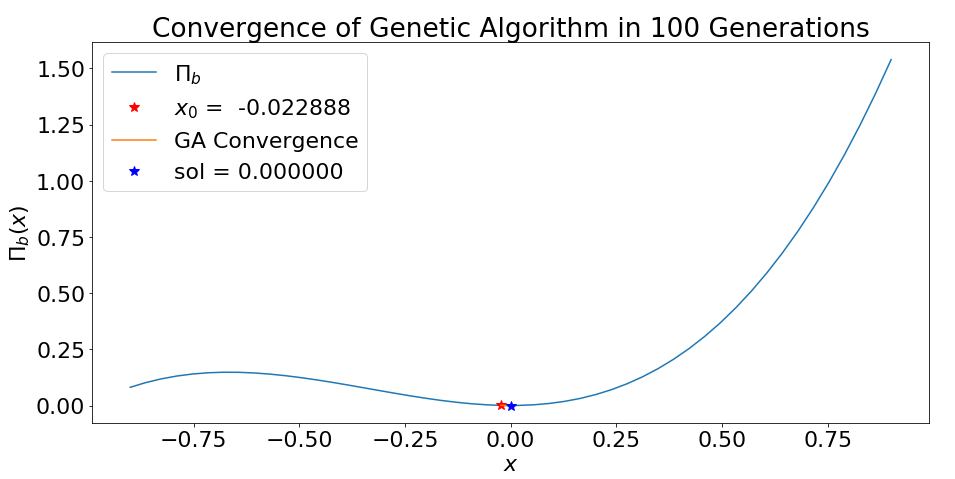
(SB[1] - SB[0])\*np.random.rand(dv,S-Par-Child)+SB[0]));

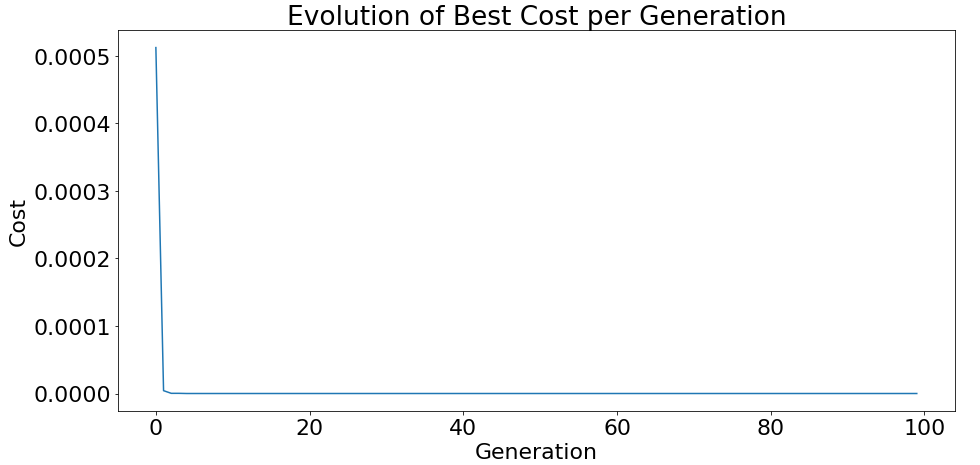
Lambda[i][:] = Lam

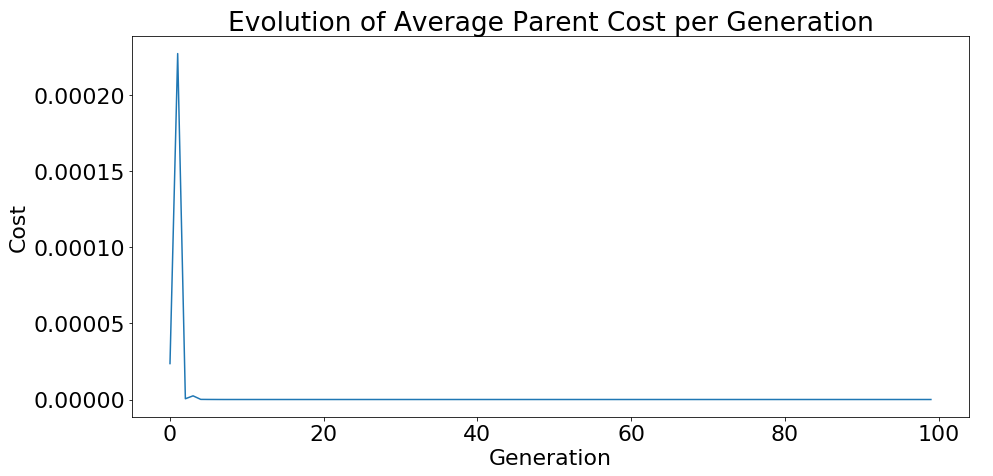
PI = cost

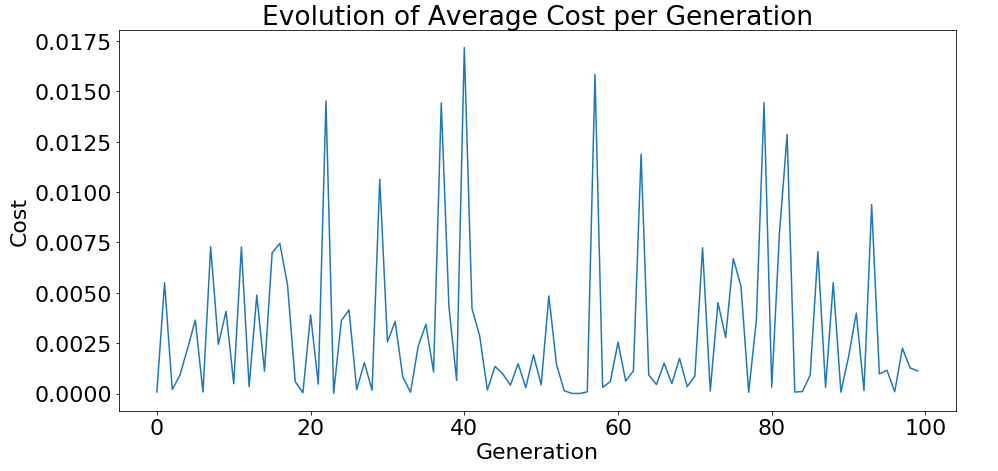
return PI, Orig, Lambda

* **Minimizing using GA**



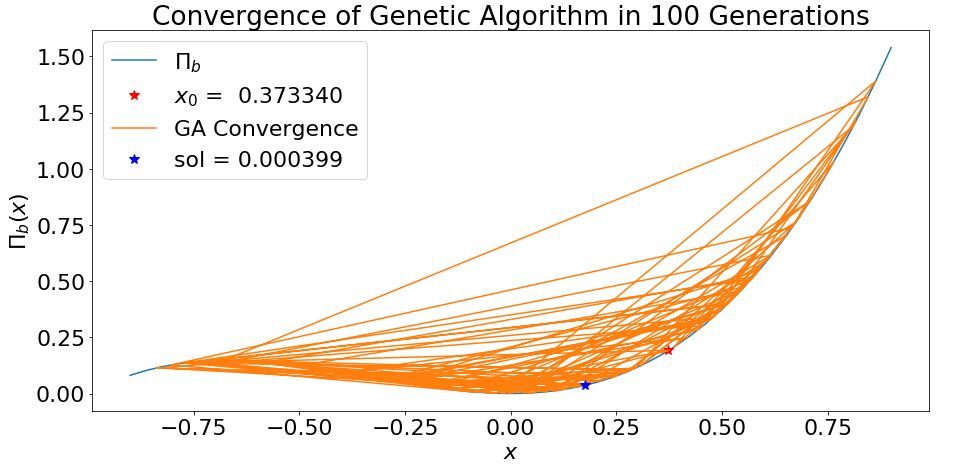






**Analyzing Your Results**

* **What does a genetic algorithm with zero parents represent? How would it perform compared to GA used in previous section?**



Zero parent means that there is no crossover operation in GA. Thus, GA with zero parent can’t predict the minimum value correctly. The solution of GA with 12 parent is 0.00000, but one with zero parent is 0.000399.

* **Advantages and disadvantages of Newton’s method**

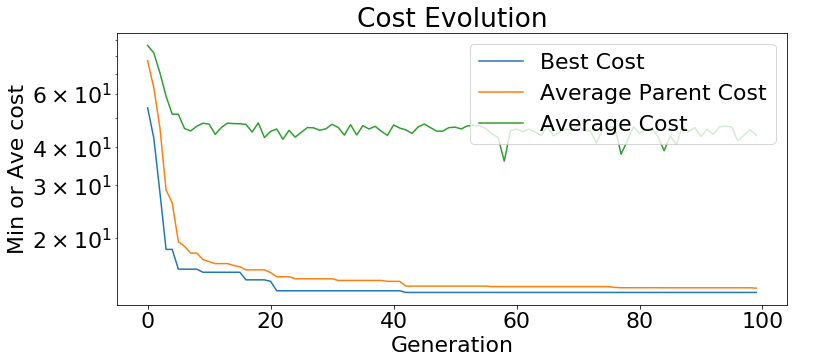
For global optimization, a complicated function has some extreme points. In the case with some extreme points, Newton method can’t find the minimum value correctly. Newton’s method is valid only for local optimization.

GA can find the minimum value correctly for global optimization with some extreme points.

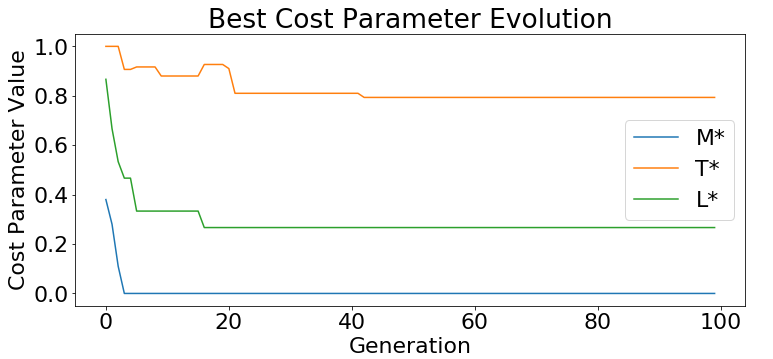
Run time of Newton’s method is much less than one of GA. In fact, for every iteration, Newton’s method performs two differential operations and one adding operation(total 3 operations).

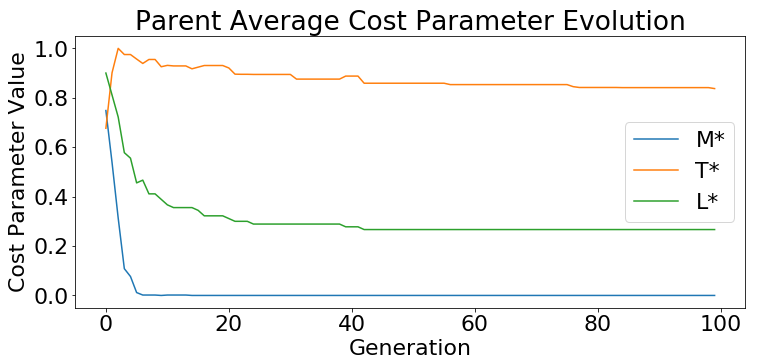
But, in the case of GA, a lot of operations should be performed. Only in the stage of selection of individuals, the number of operations for fitness calculation is same to the number of genetic strings. And then huge operations are needed for other stages.

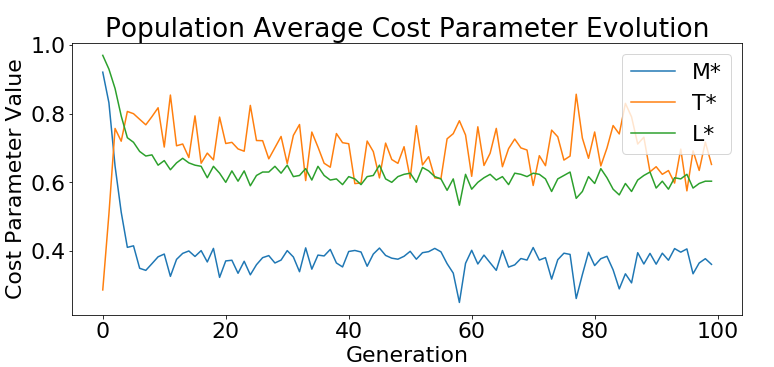
* **Convergence plot for UAV swarms**



* **Individual performance components**





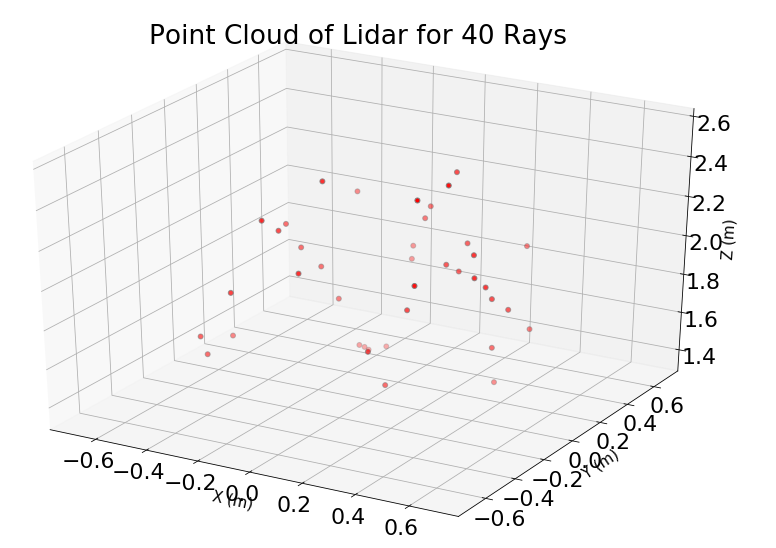


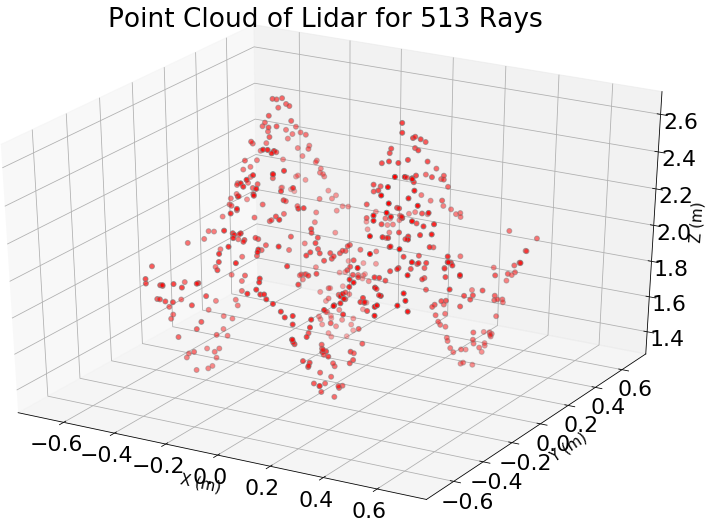
According to selection of initial population and the maximum value of generation, optimal solution will be changed.

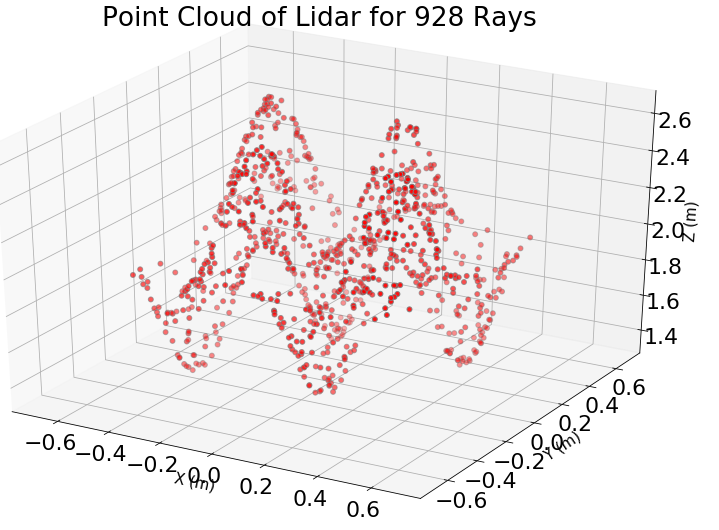
**-Report best Problem 3.5**

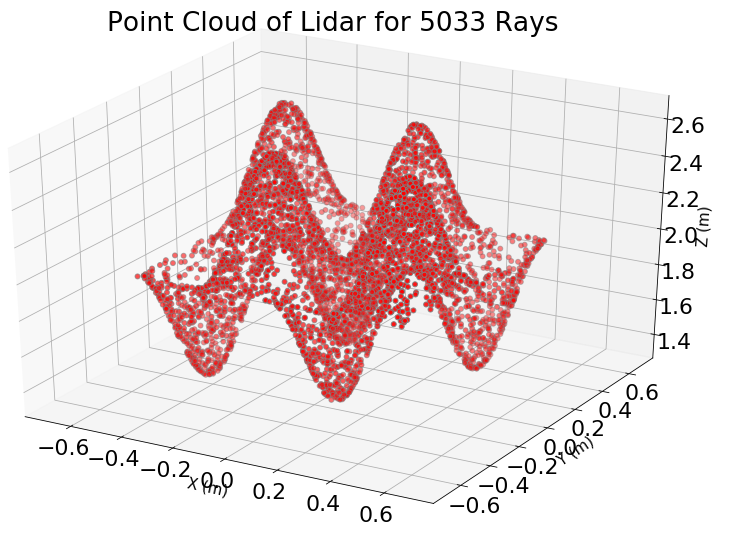
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Design | 1 | 2 | 3 | 4 |
|  | 0.70481715 | 0.75061932 | 0.74065342 | 0.74186925 |
|  | 0.71579409 | 0.55656309 | 0.69279235 | 0.62860335 |
|  | 0.34778258 | 0.36285478 | 0.28038367 | 0.3371252 |
|  | 0.73943194 | 0.7495778 | 0.71321392 | 0.72105264 |
|  | 1.15142005 | 0.89720236 | 1.03051392 | 0.9999106 |
|  | 0.7619586 | 0.57646548 | 0.61136691 | 0.58763727 |
|  | 0.84488251 | 0.85096052 | 1.02598467 | 0.91108161 |
|  | 0.27339561 | 0.94266324 | 0.6460415 | 0.77856814 |
|  | 1.08908027 | 1.2517174 | 1.1964974 | 1.20862391 |
|  | 0.21343601 | 0.20452123 | 0.21001227 | 0.20452697 |
|  | 1.35481761 | 1.66156477 | 1.37228777 | 1.65567125 |
|  | 1.71926715 | 1.71744247 | 1.7427102 | 1.72186177 |
|  | 1.05845983 | 1.08856907 | 0.95777498 | 0.99848814 |
|  | 0.77399672 | 0.91546729 | 0.86133801 | 0.91269131 |
|  | 0.52130213 | 0.55263838 | 0.54682213 | 0.54799389 |
| Best Cost | 9.966666666666667 | 10.233333333333334 | 10.233333333333334 | 10.333333333333334 |

* **Provide the point-cloud reconstruction plots for each nRays value**









* **Describe the relation between number of rays and LiDAR accuracy**

The liDAR accuracy is increased as the number of rays is increased.

* **List three assumption made in raytracing/LiDAR algorithm**

1. The features of the surface to be irradiated has an order of magnitude larger than the wavelength of the incident radiation at least. In this case, a geometrical ray tracing theory is applicable.
2. Ray-tracing is valid for rapid approximate solutions to wave equations for high-frequency/small-wavelength applications.
3. A ray becomes corrupt if the ray reflects more than once.

**Conclusion**

In here, the optimizations using Newton’s method and GA were compared.

First, two methods were compared for simple objective functions to evaluate performances of both methods.

In this case, Newton method didn’t find minimum value for initial value -0.8 and -0.4, but GA method found minimum value correctly.

As shown from above result, Newton’s method can’t be applied for global optimization problem with various extreme points. GA is useful for global optimization.

Also, the advantage of GA was proved for swarm problem of UAV.