EE553 TERM PROJECT

Novel two phase algorithm to design three loop autopilot using parameter plane technique and particle swarm optimization.

Ömer Faruk Arslan

2231256

27.01.2023

Introduction

In this project we are dealing with classical three-loop autopilot design problem in aircrafts/missiles. Three-loop autopilot approach is based on “cascade control” algorithm and it is designed by according to cascade controller rules. First consideration in this approach is “stability”, design should be ensure desired stability constraints. Second consideration is “performance”, autopilot should perform better than minimum requirements.

ω_d	ε_d	K_1	K_3	T_α	A_{11}	A_{12}
22.4	0.052	-1116.5	0.6477	0.676	0.001054	-0.00081

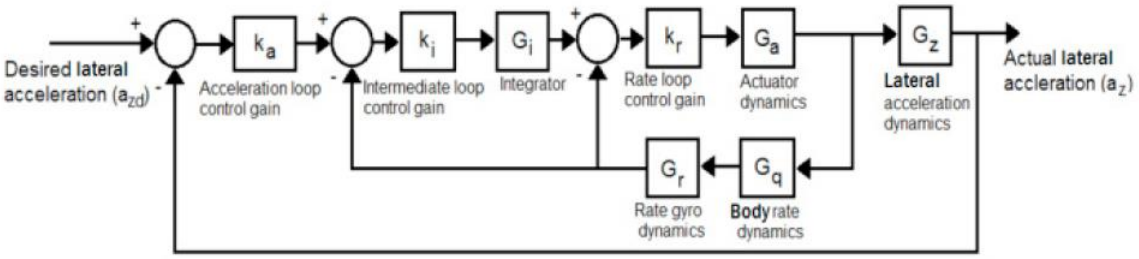

Fig.1 : Three Loop Autopilot Structure and parameters

As a control people we know that stability & performance are inversely correlated, this means that when one of them getting better while other one getting worse and worse. To solve this engineering problem we can use “optimization tools”. Obviously this trade-off between two measures directly refers an optimization problem. In the following parts of the paper, we will develop an approach that while satisfying the required stability measures, it increases the performance to the maximum level. To be specific about organization of paper, phase-1 will include stability analysis, phase-2 will introduce an optimization tool which is called “Particle Swarm Optimization (PSO)” and application to the current problem. Phase-3 will add performance optimization with PSO algorithm and at the results section, figures and final discussion will be included.

Phase #1 – Stability

For three loop autopilot design robustness is assigned by classical gain and phase margin definitions. One important thing here is we need to assigned stability requirements all three loops independently. Since if one of them go wrong cascaded controller structure will break apart.

To assign required robustness level we first obtain three open-loop transfer functions which are obtained by breaking each loop once in the error signal branch. Basic idea is when error signal takes noise inside, what will we see after closing the loop? After obtaining open-loop transfer functions for each loop we will run a “for loop” to obtain frequency dependent robustness plot.

While running these loops we assign a value for one of the loop gains which is “rate loop gain kr”. In this loop system will face with actuator dynamics and as a rule of thumb we choose “kr” to make inner loop bandwidth is approximately 1/3rd of the actuator bandwidth.

Tracing the space for desired solutions in 2D problems is called “parameter plane technique” and by using this approach we found a set of solutions for “integral gain ki” and “acceleration gain ka” .

After analyzing the system several times, we ensure that “5 dB gain margin” and “30 degree phase margin” is suitable for these dynamics. After assigning the required phase and gain margin we solve a set of equation to find out corresponding “ki & ka” values at that given frequency variable “w”.

$$G(j\omega) = \frac{N(j\omega)}{D(j\omega)} = -\frac{1}{A}e^{j\theta}$$

$$D(j\omega) + Ae^{-j\theta}N(j\omega) = 0$$

$$M(j\omega) = D(j\omega) + Ae^{-j\theta}N(j\omega)$$

In this equation M(jw) is called “gain-phase margin tester function” where A is gain margin and theta value is phase margin in terms of radian. Setting M(jw) for each loop we obtained the following plot which shows pair of gains that satisfy the equation.

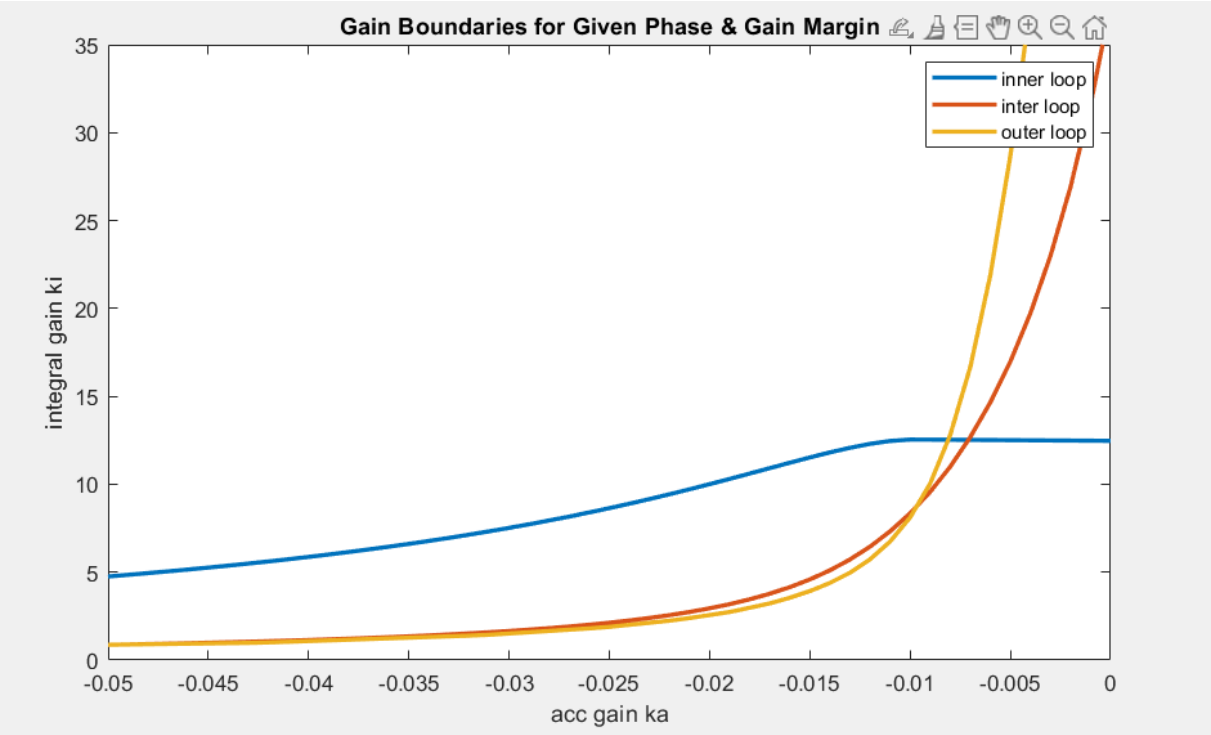

Fig.2 : Three Loop Autopilot Gain Pairs for Robustness

We all know that if we choose smaller gains in the controller we can obtain higher robustness levels. Therefore we will choose minimum values of these 3-plots as an upper boundary for solution space. Then, we know that pairs will produce “at least” 30 degree and 5 dB phase and gain margins.

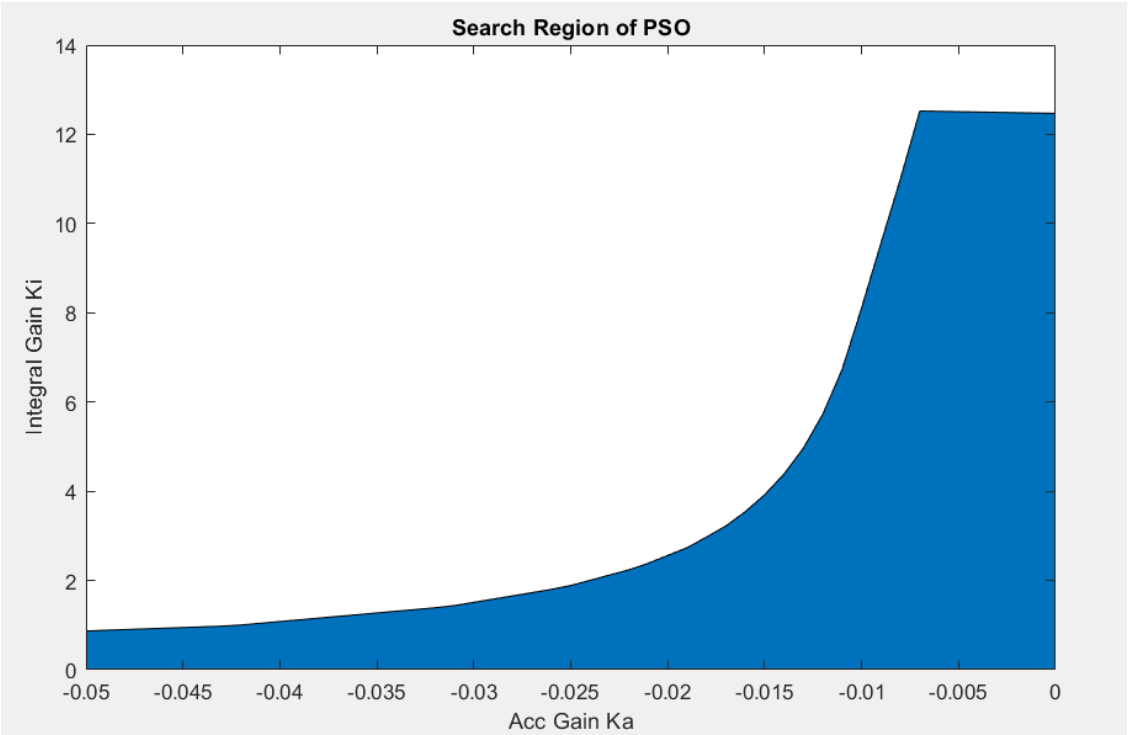

Fig.3 : Loop gains search space, guaranteed robustness

Phase #2 – PSO Introduction & Search Region

After finding the set of pairs, next step is the defining the optimization problem and approach to solve it. For this problem our optimization problem is based on “time domain performance requirements”. Specifically, undershoot, overshoot, settling time and rise time performance of the closed loop system.

To solve this problem we use an optimization approach called “particle swarm optimization(PSO)”. In this approach, problem search space is filled with given number of particles which are designed to move toward “global best location” according to given cost function definition.

In this optimization approach optimization parameters are inertia of particle, coefficient of the local best and coefficient of the global best for position update. Overall, we assign a “maximum number of iteration” and “swarm size” for the whole system.

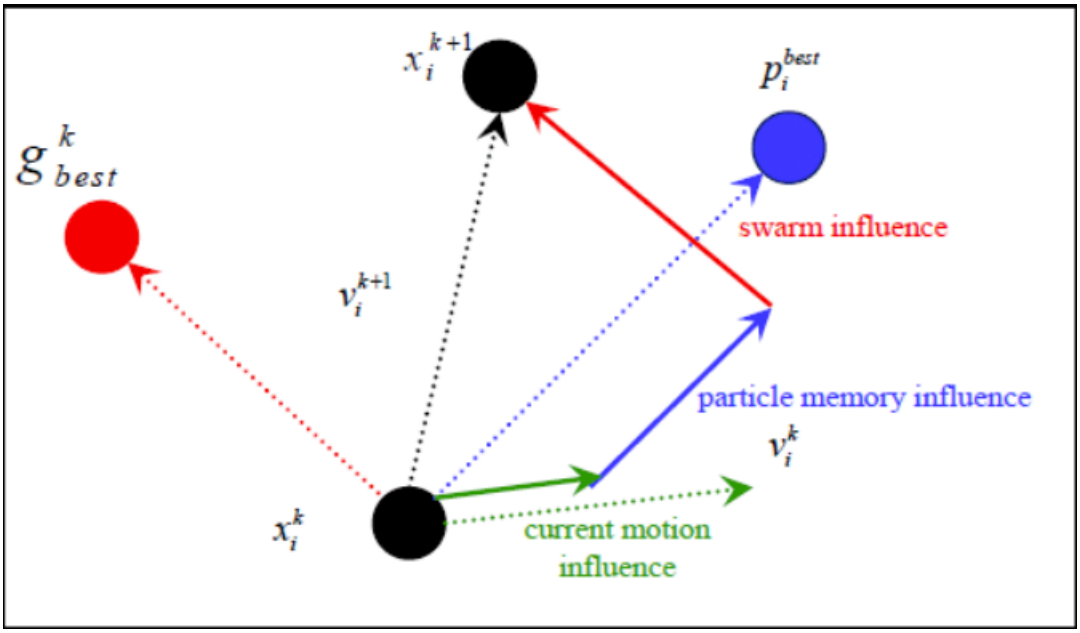

Fig.4 : Particle Swarm Optimization Algorithm

As we can see in this figure motion of the particle at each iteration is determined by current motion influence, particle memory influence(local best) and swarm influence(global best) . After running algorithm several times, particles which are “randomly placed” in the search space move towards to “best” positions.

One should be aware of that, PSO can stuck in the local minima. To overcome this problem 2 solution approach is advised. First one is increasing the swarm size and second one is resetting the PSO when convergence happens as iteration goes. Resetting particle positions randomly will decrease probability of local minima convergence problem, and in this paper we choose second approach.

Before entering the optimization phase, we should prepare the problem. In our problem we need to choose tow controller gains which are dependent on each other. Therefore, position updates cannot be done independently. To assign a relation between them, we fit the given controller gain data and increase the order until we catch the original fashion in the parameters.

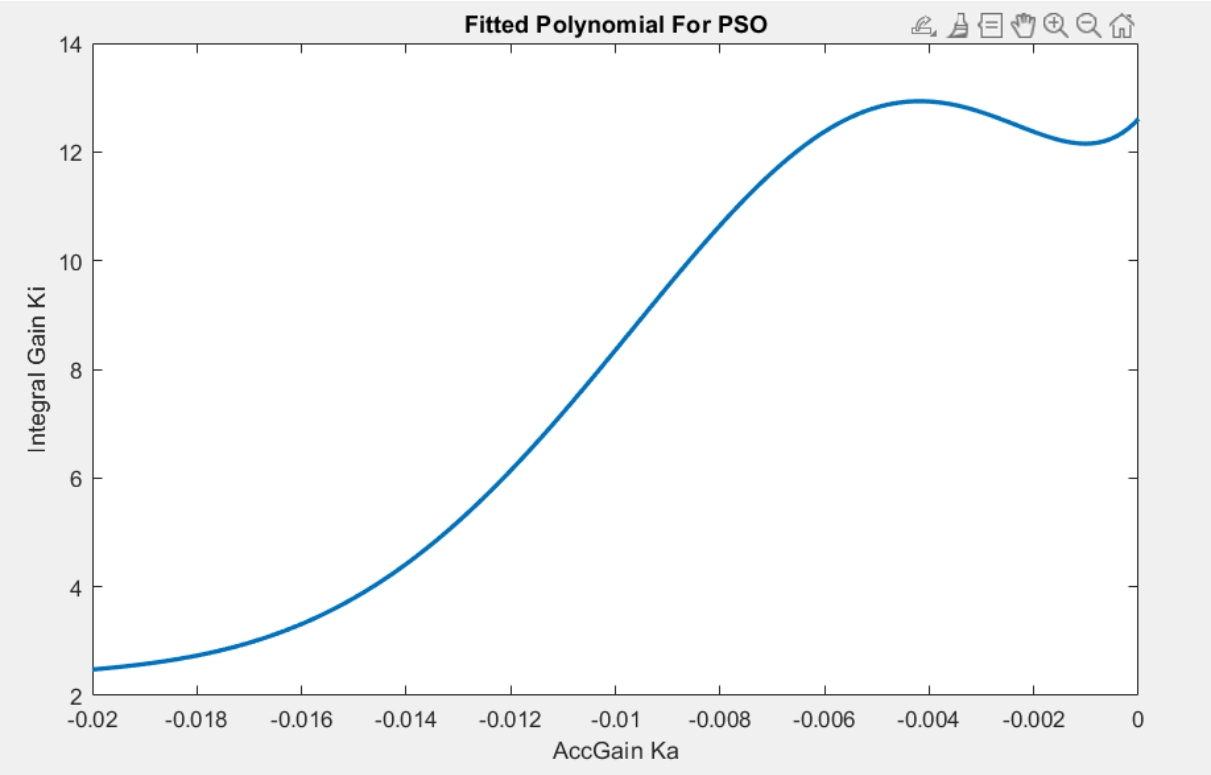

Fig.5 : Gain pair dependency plot, for PSO algorithm

Phase #3 – Performance Optimization with PSO

In the optimization phase, first we will talk about structure of PSO and how it moves at time goes by. Then we will discuss results of the PSO algorithm and their meanings.

First of all, to make code more suitable and readable we design PSO code as function which takes optimization parameters from designer. Specifically, we give maximum iteration number, swarm size, inertia weight, damping coefficient and local & global best coefficients.

MaxIter = 100	**SwarmSize = 10**	**w = 1**	**wdamp = 0.99**	**c1 = 2**	**c2 = 2**

Then, PSO function takes these parameters and sets the equations of position update and velocity update. According to these equations, particles are expected to move global best position in the search region. Initializing the particle distribution in the search space is done by randomly. After initializing, at each iteration step by using velocity formula position of the particles are updated.

The possible problem is local minima convergence. Particles are designed to move towards global minima but none of them has the information of the “global minima position”. Each particle knows its own best and only collective information of the swarm assign the global minimum. If the somehow particles close each other in terms of cost function value, they cannot leave that local minima since none of them break that barrier. To solve this problem, we reset PSO algorithm several times to decrease probability of local convergence. Since each reset distribute the swarm randomly , local convergence will be less possible option.

Specifically, we designed code such that it resets PSO 4 times therefore, we need to assign “maximum number of iteration” as a multiplicative of 5. One can see the effect of this reset approach in the following PSO cost results.

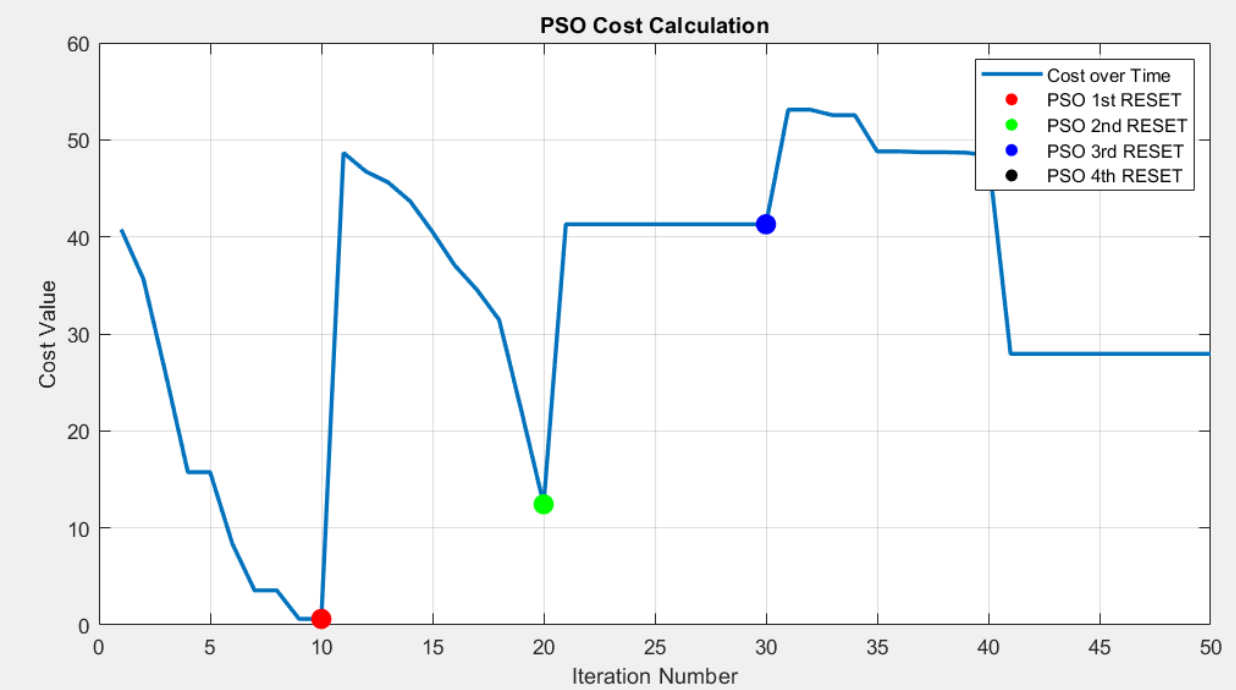

Fig.6 : PSO cost function minimization and RESETS

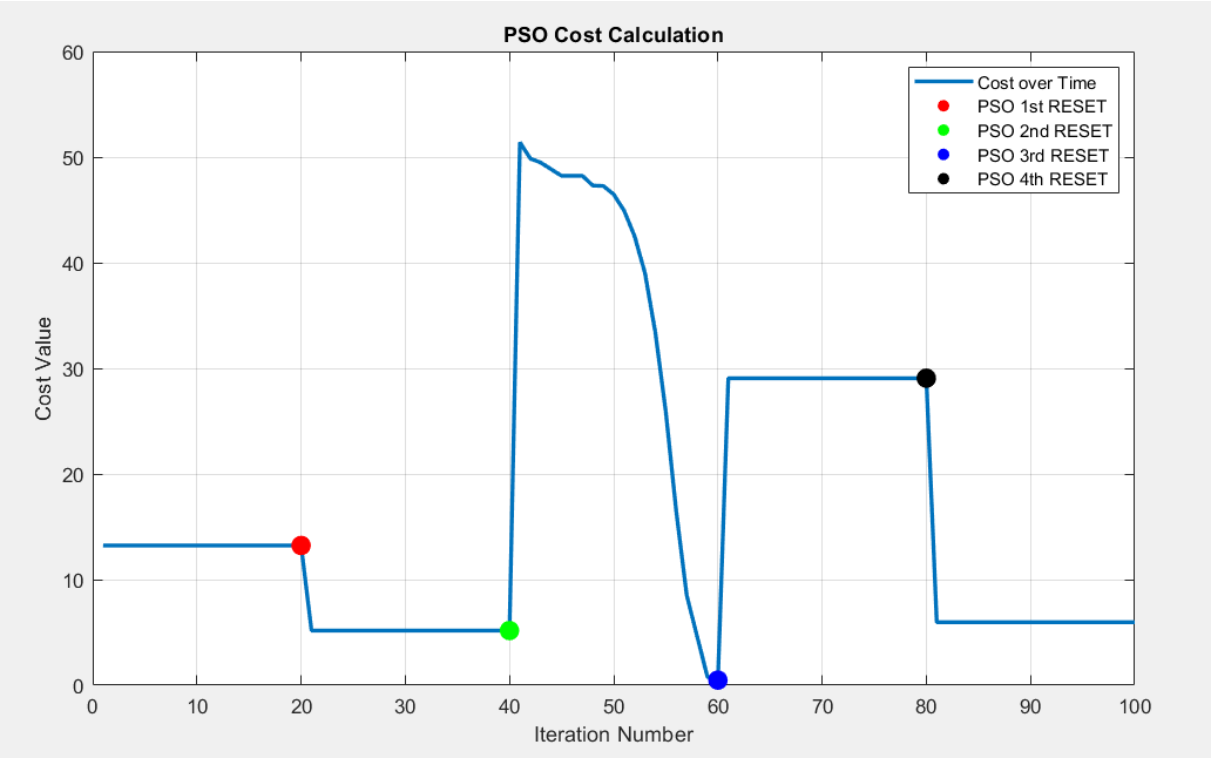

Fig.7 : PSO cost function minimization and RESETS

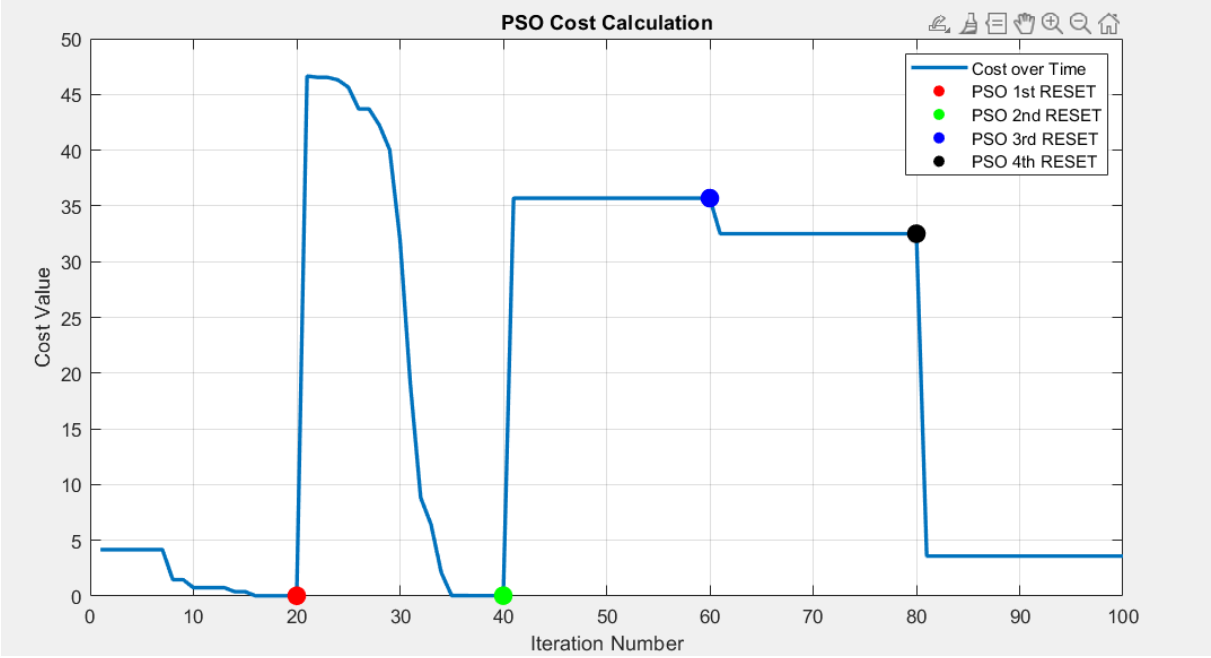

Fig.8 : PSO cost function minimization and RESETS

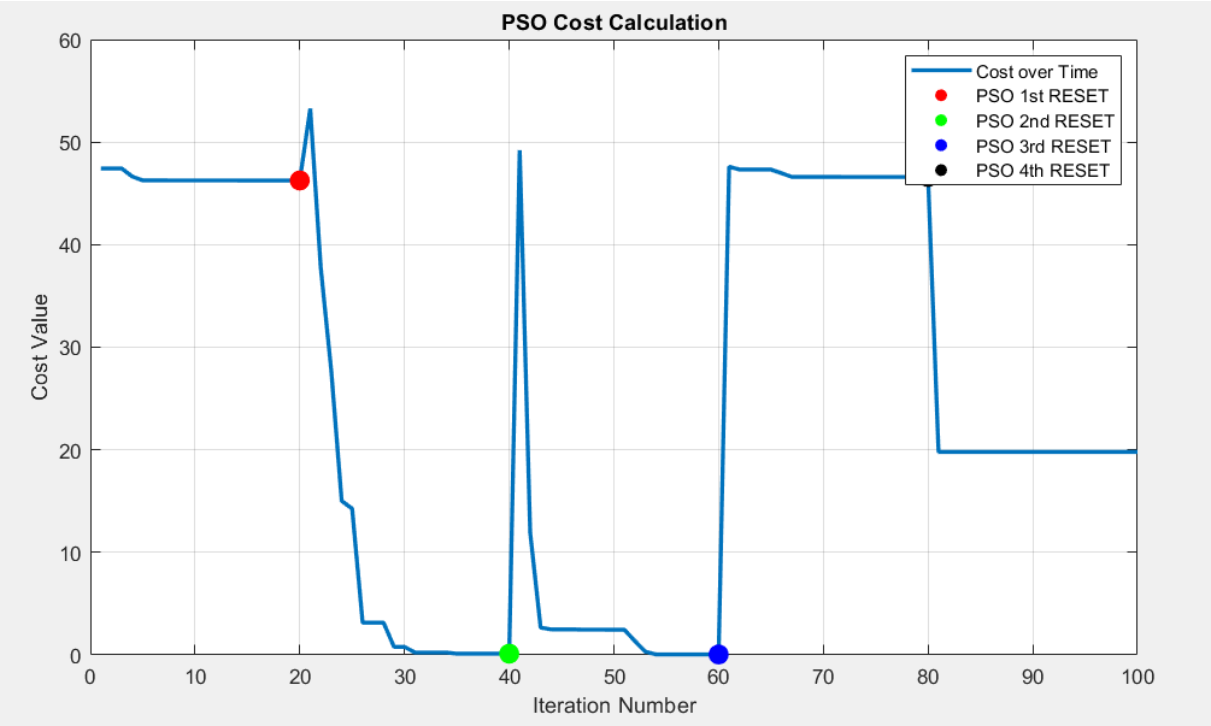

Fig.9 : PSO cost function minimization and RESETS

Notice that, since each PSO starts randomly distributed particles each cost trajectory will be different. We can see the local convergences at each plot some of the sections. Luckily, resetting the PSO solve the problem almost each trial during the design and test phase of the paper.

During optimization phase we used previously mentioned cost function definition. Which includes time domain performance measures. We take into account undershoot performance, overshoot performance, rise time and settling time with their corresponding weight assignments.

$$\mathbf{Cost = w1*(us-usd) + w2*(os-osd) + w3*(ts-tsd) +w4*(rs-rsd)}$$

Where

us = undershoot & usd = desired undershoot

os = overshoot & osd = desired overshoot

ts = settling time & tsd = desired settling time

rs = rise time & rsd = desired rise time.

We used following values for cost weights and desired performance measures;

W1 = 1 & usd = 10%

W2 = 1 & osd = 2%

W3 = 10 & tsd = 0.3 sec

W4 = 10 & rsd = 0.15 sec

For this parameters and weight configuration, PSO find out that optimal gain pair should be around **(ki = 8.65 and ka = -0.0059)** which are close enough to optimal solutions given in the paper.

Paper Best	**Ki = 8.6182**	**Ka = -0.006**
PSO Results	**Ki = 8.6276**	**Ka = -0.0059511**

Results

After running PSO part we obtained the results of the optimization problem. Which minimizes the predefined cost function of time domain performance.

```
#############
Best Solution = 8.6276  -0.0059511
Best Cost = 0.057497
Time Domain Performance
Error in Desired Overshoot    = %30.9494
Error in Desired Undershoot   = %2.9364
Error in Desired RiseTime     = %2.6299
Error in Desired SettlingTime = %10.7128
All 3 Loop has at least 5 dB GM and 30 deg PM.
#############
```

Finally, we can look at the gain and phase margins of the inner, intermediate and outer loops. Which can be found in the following figures;

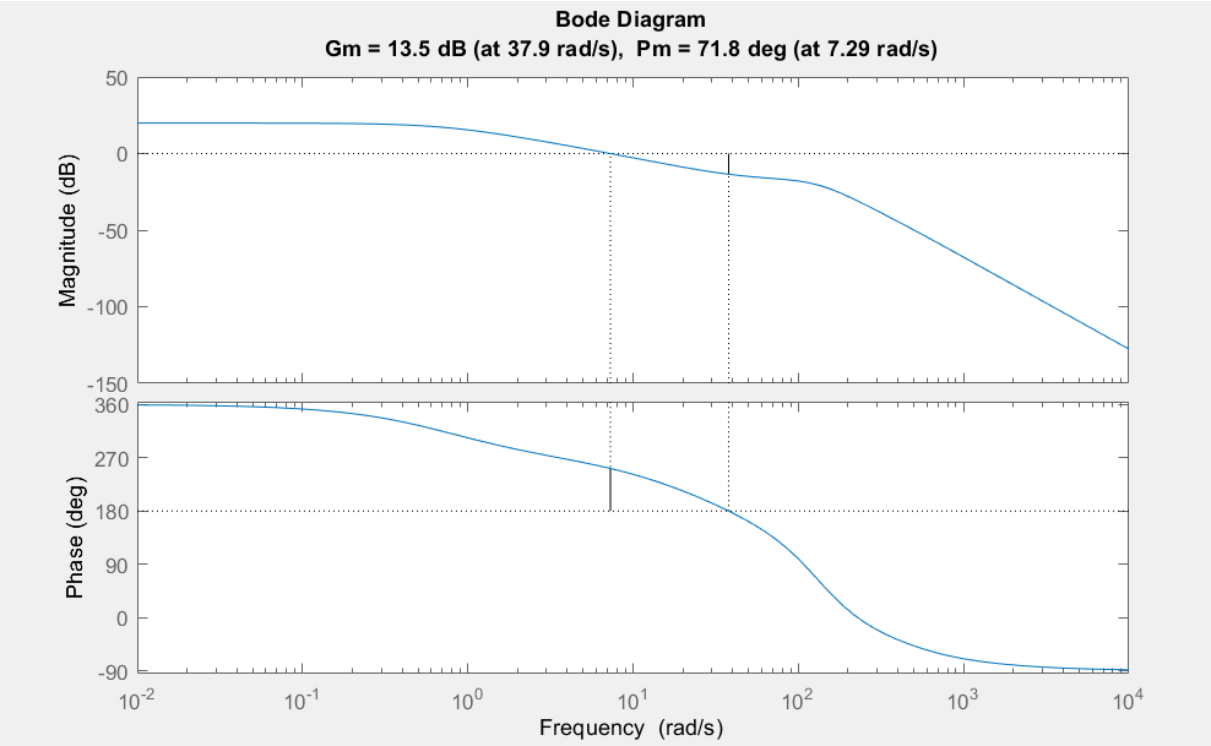

Fig.10 : Outer Loop Phase & Gain Margin after PSO

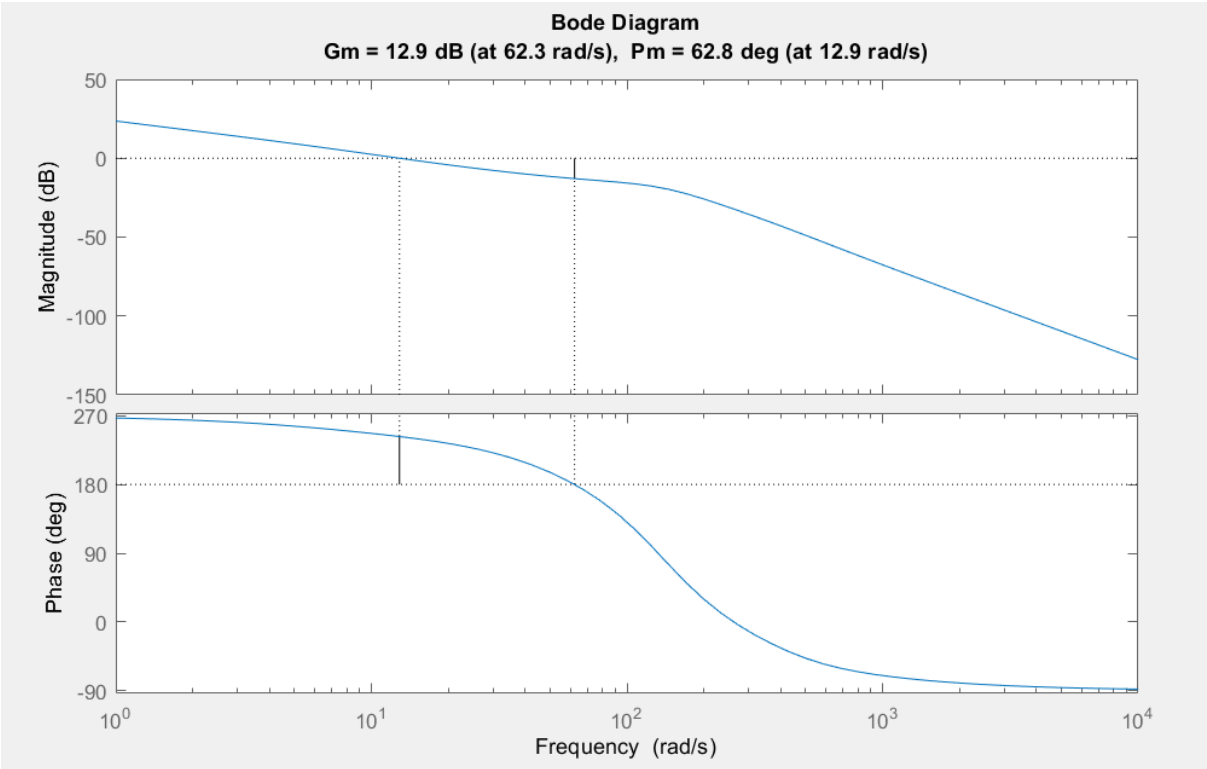

Fig.11 : Intermediate Loop Phase & Gain Margin after PSO

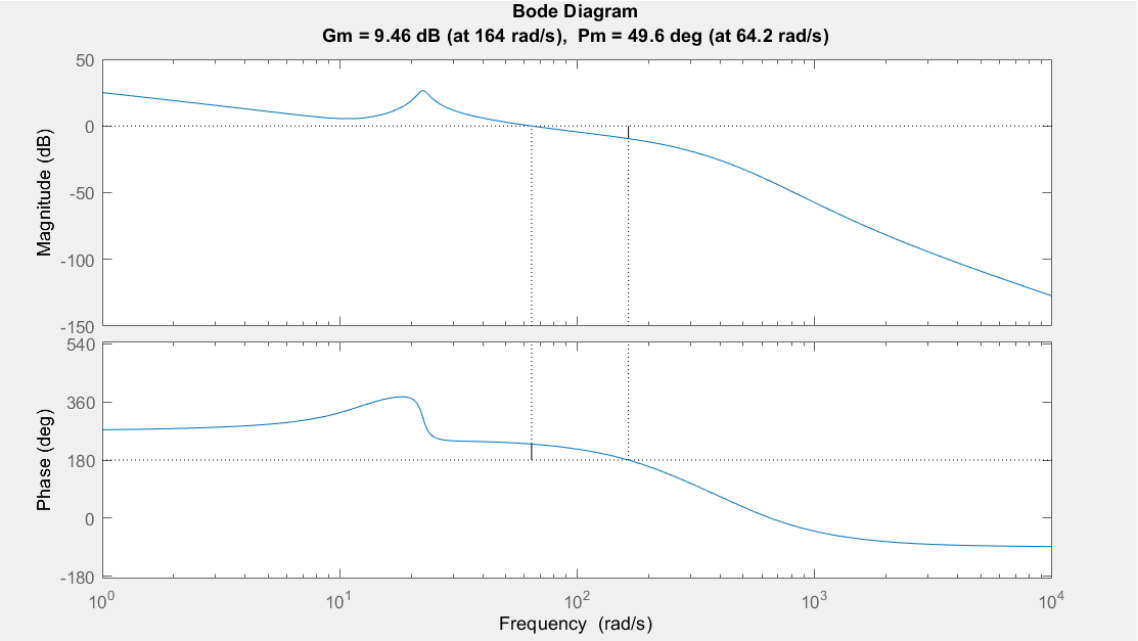

Fig.12 : Inner Loop Phase & Gain Margin after PSO

Appendix

To make project more compact and visually better, we designed a MATLAB GUI using App Designer Tool(available after R2016). One can see the PSO cost calculation, time domain performance results, robustness results etc. After pushing to "RUN PSO" button we can follow iterations from MATLAB workspace.

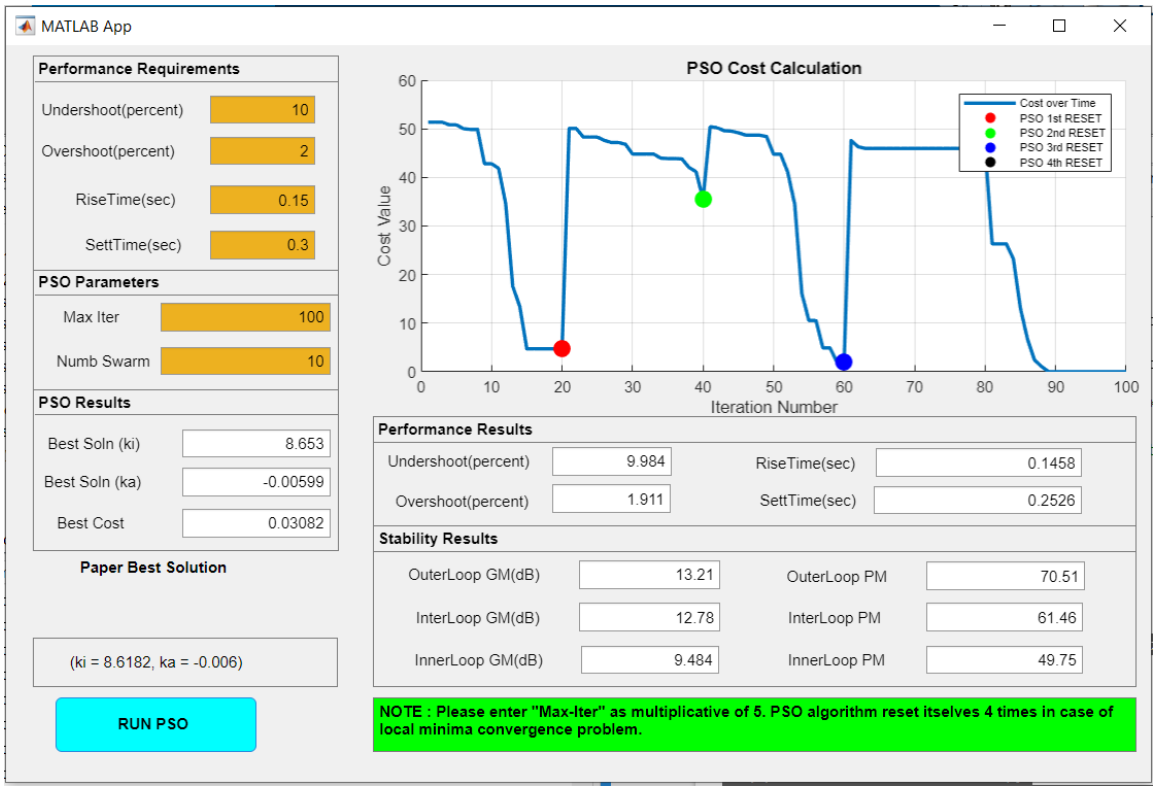

Fig.13 : Graphical User Interface Design for project

References

1. Kedarisetty, S.: Novel two phase algorithm to design three loop autopilot using parameter plane technique and particle swarm optimization. IFAC-PapersOnLine 49(1), 830–835 (2016)

2. Defu, Lin & Junfang, Fan & Zaikang, Qi & Yu, Mou. (2009). Analysis and improvement of missile three-loop autopilots. Systems Engineering and Electronics, Journal of. 20. 844-851.