The University Of Sheffield.

Multisensor and Decision Systems ACS6124

Part II: Decision Systems for Engineering Design

Assignment (ACS6124-002)

By

Ankur Tiwari

Registration Number 210158775

Contents

Executive Summary.

Problem Statement:
Tuning the gains for a Proportional-Integral (PI) controller such that a feedback control system satisfies a set of requirements.

Proposed Solution:

This problem involved optimization of multiple objectives (nine) that affects the stability, Transient performance, and Steady state performance of a plant, some of these objectives are conflicting in nature for example rise time and maximum overshoot hence the problem here is to find a multiple optimal trade-off solution. In this case deployment of an evolutionary algorithm made sense as their population-based approach can find multiple optimal solution in its final population. Genetic algorithm was chosen for this purpose as they generate high-quality solutions to optimization and search problems. Genetic algorithm relies on biologically inspired operators like mutation, crossover, and selection.

Genetic algorithm requires an initial population for implementation. To generate the initial population different types of sampling plans were used and appropriate plan was selected by evaluating them using evaluation matrices. In this case Sobol Sequence was selected because of its better space filling properties.

A mathematical formulation of problem was done to identify the decision parameters from the Transfer function. Mathematical formulation also helped in formulation of objective space.

Different types of data exploratory methods were used to identify correlation between data and establish relation between decision parameters and objectives.

The results of optimization process are analyzed and presented in a systematic manner and based on those results recommendations are made on tuning the PI controller for the plant. Also, key trade-offs were discussed, and alternative approaches were also suggested.

Section 1: Multi-objective optimization for Engineering Design

Introduction to Decision Systems for Engineering Design.

Decision making is an imperative activity in engineering design. Designers must make important decisions regarding:

- Pursuing an idea from a broad pool of ideas
- Selection of material for usage
- Creation of a model from different alternatives.

Success of a design is dependent on purchases decisions made by consumer and product decisions made by competitors hence success of a design depends upon the ability of designers to make good decision.

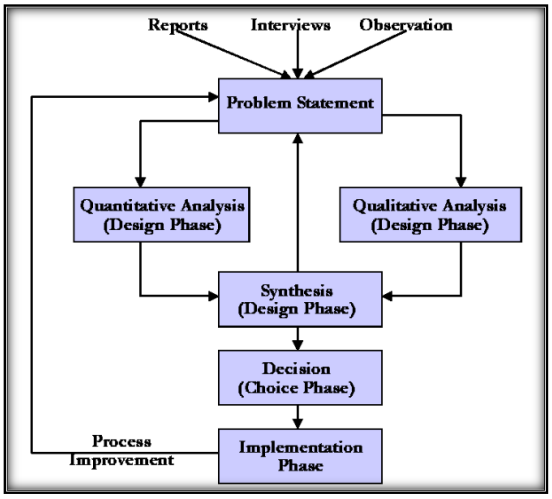


Figure 1 Figure Showing Steps involved in Multi-objective Decision Making

For the organizations involved in product development, Generating & refining design alternatives and then selecting a single design or even a set of design that fulfills specific objective is an imperative task in engineering design. An Engineering design decision problem can be stated as "*Provided several criteria for performance that requires simultaneous optimization, It's the process of determination of a method for comparison between design alternatives dependent on individual criteria for each alternative*."

Approaches used in decision making

In Engineering design decision-making methods are very effective, however designers prefer tested procedures and experience-based approaches. This can be attributed to the reasons such as consolidated design habits of companies and people, time consumption and lack of knowledge and tools. This section discuses most used decision-making methods and their application in automotive industry. In automotive industry decision-making is particularly critical as there so many high-quality parts that requires mass-production and should be designed in short time and the design decision is to adapt a specific design.

The Decision Problems can be sub classified into categories shown in Figure2

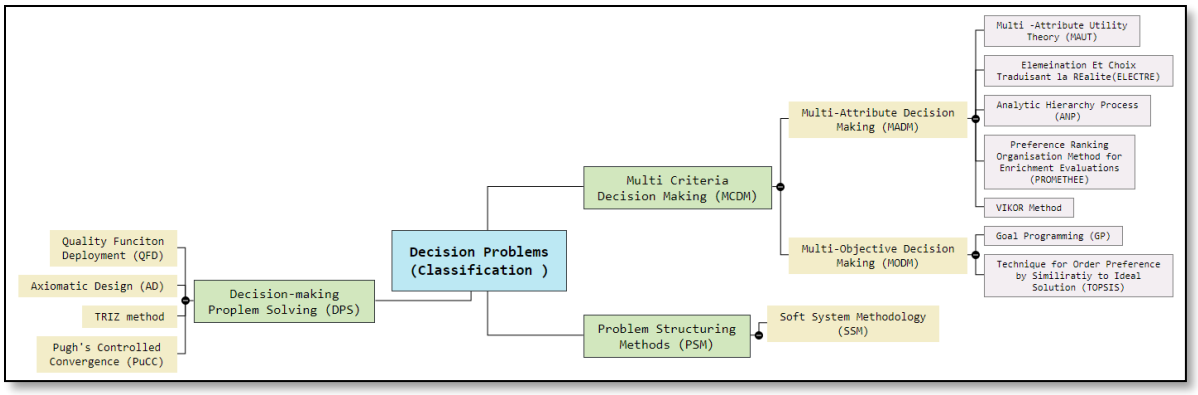


Figure 2 Classification of Decision-Making Techniques

Comparison between Different Approaches

Decision Making Technique	Technique Name	Description
MCDM Methods	MAUT	Steps involved are identifying decision-makers viewpoint , then alternatives , attribute evaluation , identifying factors for attribute evaluation, utility scale establishment for scoring factor
	ELECTRE	It chooses a set of promising actions, describes partial preferences, and generate priorities based on preference modeling.
	ANP	Identifies the interdependence and interactions between attributes in such a manner that lower-level criteria's lay foundation for higher level.
	PROMETHEE	Rank of finite set of alternative actions is determined in this method.
	VIKOR	It is used for multi-criteria optimization of complex problems. In order to achieve this, it makes a compromise ranking list and corresponding solutions. Using the initial weights, the weight stability intervals of preference of compromise solution is obtained.
	GP	Solution that satisfies each constraint portrays compromise result which is the minimum Euclidian distance of the attainment and aspiration levels of the objectives
PSM	SSM	Mainly used to solve Management, organizational and policy related problems.
DPS	AD	Mainly used to solve problems related to user needs related specific design requirements.
	QFD	Mainly used for identifying links between customer needs and company requirements.
	TRIZ	It is a creative technique used for new product development.
	PuCC	In this technique homonymous matrices are used to select a product during the conceptual design phase.

Examples of applied decision-making methods in automotive design.

Design Problem	Method	Description
Indicator Panel design	APH+Fuzzy-AHP	The parameters and functional requirements are determined by the fuzzy language scale and based on the AHP to obtain a weight indicating the importance of the functional requirement. The methodology was tested to develop a new panel indicator for passenger cars.
To evaluate the planning design of Four-door midsize sedan and sports car.	ANP+BBN (Bayesian Belief Network)	Customer satisfaction attributes were analyzed and prioritized using correlation between attributes. BBN was used to prioritize process from subjective expert judgements.
Design of a motor car Vehicle fatigue in design of truck cab	Mol+Fuzzy-QFD TOPSIS	Uncertainty related to engineering characteristics were quantified using Fuzzy set theory.Mol strategy was used to synthesise multiple ECs instead of commonly used weighted sum strategy.
New Diesel Engine Development case	QFD+AHP	Fuzzy set theory is used to find the uncertainty associated with design and engineering characteristics.
Car sunshade	QFD	QFD is used to concurrently consider both customer attitudes and engineering preferences.

Comparison between population-based optimizers

Genetic Algorithm:

Genetic algorithm solves a problem based on process of natural selection. A randomly generated population is taken during initialization and then the best result is obtained by iteratively selecting and mutating these candidates till the results converge to an optimal solution. Genetic algorithm uses tournament selection for identifying the fittest candidates from current population of candidates. The candidates that are selected are passed to next generation this is done with the help of a k-way tournament. Tournaments takes place till the final selection. The parameter that decides the selection of a candidate is called selection pressure, which is a probabilistic measure of a candidate's selection. Rate of convergence of the algorithm is decided by selection pressure.

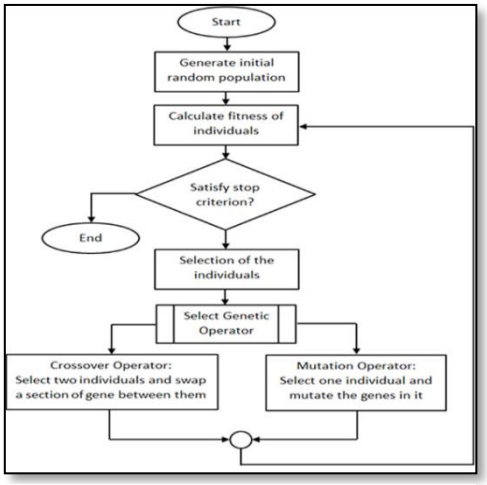


Figure 3 Flowchart showing the functioning of Genetic Algorithm *[2]*

Particle Swarm Optimization:

Particle swarm Optimization optimizes a problem by iteratively attempting to move forward a candidate arrangement with respect to given degree of quality. It solves the problem by moving the particles within a search space agreeing to a mathematical equation based on particle position and velocity. The best position identified by neighboring particles helps in guiding the movement of a particle to its optimal position. In this way swarm of particles attains an optimal solution. PSO does not employ any kind of mutation or crossover unlike genetic algorithm.

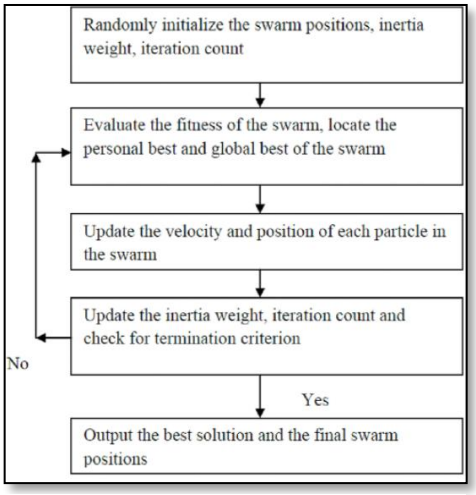


Figure 4 Figure showing the flowchart of Particle Swarm Optimization

Ant Colony Optimization:

This is a probabilistic technique used to identify best path through graph. Like real world pheromone-based communication is used. optimal solution is achieved by moving artificial ants through parameter space that represents a set of all possible solutions. Like real life ants' artificial ants record their position so that later ants can identify better solution.

Ant colony optimization-based solvers scan all possible solutions before deciding on selecting a particular solution element, this makes the optimization process computationally very expensive. To overcome this the element choice is restricted for each artificial ant.

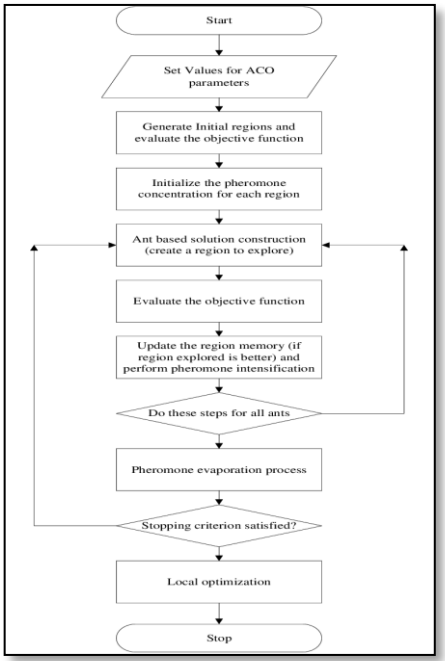


Figure 5 Figure showing the flowchart of ant colony optimization

Section 2: Problem Formulation

Problem Statement:

The goal of this problem is to use data modelling tools to explore the problem's design space and convey the relationship between design variables and performance requirements. The problem has a pre-determined architecture. A Proportional-Integral (PI) controller is used in a unity-gain feedback loop to control the digital system. Because the plant is analogue, digital-to-analogue and analogue-to-digital converters are necessary between the controller and the plant, which is sampled every one second. The choice of proportional and integral gains for the PI controller is the control system design problem.

System Model:

The plant has been modeled as first - order linear system in continuous time.
Where,
G(s) is Plant transfer function
G_0(s) is Digital to analogue converter modelled as zero-order hold
G_D(s) is digital PI controller

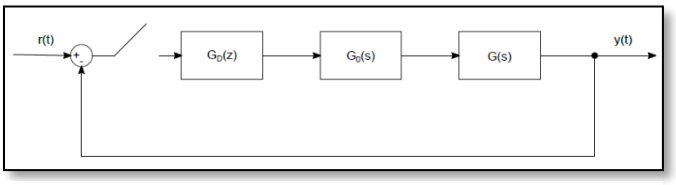


Figure 6 Model of plant

$K_P > 0$ and $K_I > 0$ are the proportional and integral gains respectively.
The open loop transfer function for the plant is given by:

$$G_D(z)G_0(z)G(z) = \frac{(1-e^{-1})((K_P+K_I)z-K_P)}{z^2(z-e^{-1})(z-1)}$$

Decision Variables:

Here, **K_P** and **K_I** are the decision variables, therefore Decision vector can be written as **X= [K_P, K_I]**

Objective / Performance Criteria:

Nine performance criteria have been identified for this system:

1.Stability Criteria	2.Transient Performance Criteria	3.Steady State performance criteria
I. Magnitude of largest pole in closed loop TF (Z_1)	IV. Rise time (Z_4)	IX. Steady state error (Z_9)
II. Gain margin (Z_2)	V. Peak Time (Z_5)	
III. Phase margin (Z_3)	VI. Maximum overshoot (Z_6)	
	VII. Maximum Undershoot (Z_7)	
	VIII. Settling time (Z_8)	

The performance criteria are evaluated using a difference-equation model

The performance space is given by **$Z = [Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7, Z_8, Z_9]$**

Goals:

Objective	Range	Priority
Largest closed-loop pole (Minimize)	<1 Hard constraint	Hard Constraint
Gain margin (Maximize)	6 dB High	High
Phase margin Range	30^o to 60^o	High
Rise time (Minimize)	2 seconds	Moderate
Peak time (Minimize)	10 seconds	Low
Maximum Overshoot (Minimize)	10 %	Moderate
Maximum Undershoot (Minimize)	8%	Low
Settling time (Minimize)	20 seconds	Low
Steady-state error (Minimize)	1 %	Moderate

These are the preference given by chief Engineer

The objective here is to find the value of decision variables **X** which gives the value of **Z** that satisfies the following Goals.

Section 3: Sampling Plan

Sampling plan involves assessing samples taken from the problem's design space (together with samples of the parameter space, in cases where parameters are defined). The approaches used for design of sampling plan are:

Random Sampling:

This technique provides each sample equal probability of being chosen. An unbiased representation of total population is given by a randomly selected sample. In random sampling there is under representation of areas of the design space which is one of its demerits and leads to better techniques such as space filling designs. This is due to the fact that random sampling does not consider the previously generated points.

Space Filling Design

Space filling designs lays out the points with the objective of attaining assortment of data. This is done in order to obtain smoother and more accurate predictions that is spread out in sample space. Space filling designs can be subdivided into Full factorial Design and Latin hypercube Sampling (LHS). These are described in more details below.

- **Full Factorial Designs**: In full factorial design the entire space is divided into segments called hypercubes and then a sample is defined in each hypercube, either within the hypercube or at its corner.
- **Latin Hypercube Sampling**: This approach takes the idea of perturbations a step further and also makes sure that the coverage of all the design variables is more uniform. Design regions equal to the number of variables are identified and are in taken in such a manner that they are equal. These divided regions are called hypercubes. Hypercubes are then randomly filled for each variable. LHS is one of the most commonly used sampling plan used to identify relationship between input and output variables.

Other approaches of Sampling:

- **Orthogonal arrays**: For using orthogonal arrays the important condition is that number of levels should be small. To apprehend the interaction between up to p variables at a moment, an orthogonal array of strength p is required.
- **Sobol Sampling**: It is a quasi-random number generator and has better space filling properties as compared to pseudo-random number generator. It is better than Latin hypercube sampling in conditions where the sample space is divided into S spaces, but LHS plan cannot produce good results if less than S samples spaces are checked because of lesser availability of resources than previously anticipated.

Figure 7 helps in visualizing the spread obtained by using different types of sampling plans to derive more concrete understanding of their working.

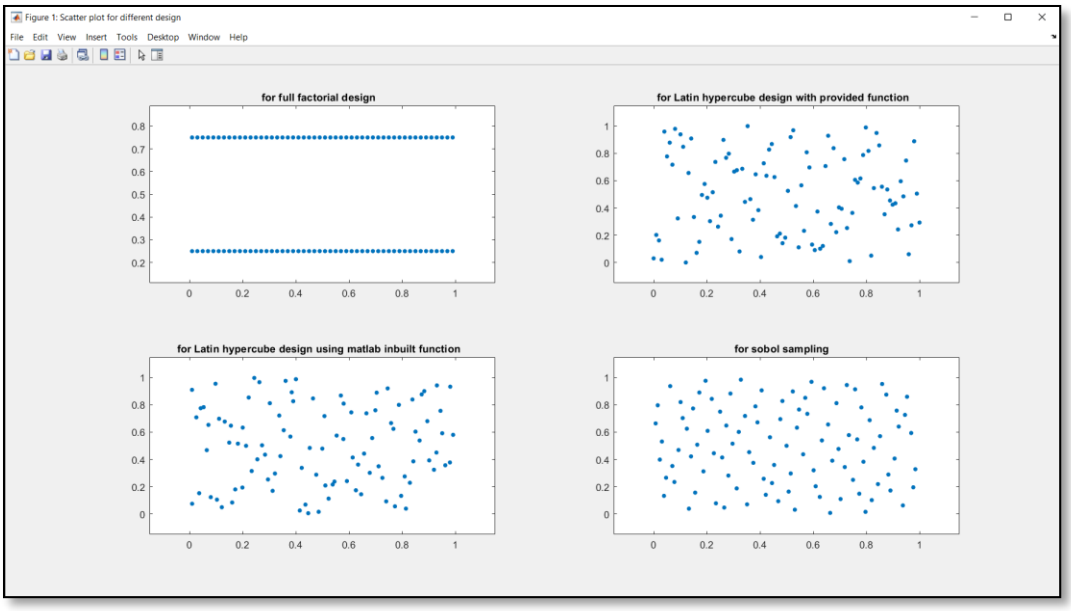


Figure 7 Scatter Plot for different types of sampling plans.

Space-filling metrics:

ϕ_q is a metric used to evaluate and compare different Sampling plans? This is done by calculating distance between all pair of points in a given space. All the distances are then ranked in ascending order. Then pairs of samples associated with each distance are then counted. The equation for computing ϕ_q is given by

$$\phi_q(x) = \left(\sum_{j=1}^{u} J_j \,/\, d_j^q\right)^{1/q}$$

Where, q is a tunable parameter that dictates how much the smaller distances in the question will dominate the final value for the metric.

Selection of Sampling plan to take forward:

The output of ϕ_q value obtained for different sampling plans is shown in Figure 8 . The least ϕ_q value is obtained by Sobol Sampling which indicates its superiority over other sampling plans hence Sobol sampling will be used further.

```
The phi_Q value for full Factorial Design is :79.092
The phi_Q value for Latin Hypercube sampling is :84.421680
The phi_Q value for Sobol Sampling plan is :38.099528
```

Figure 8 Figure showing Matlab output for φ_q value of different sampling plans

Section 4: Knowledge Discovery

Some interesting Interpretations from the results obtained by evaluating different sampling plans are discussed below:

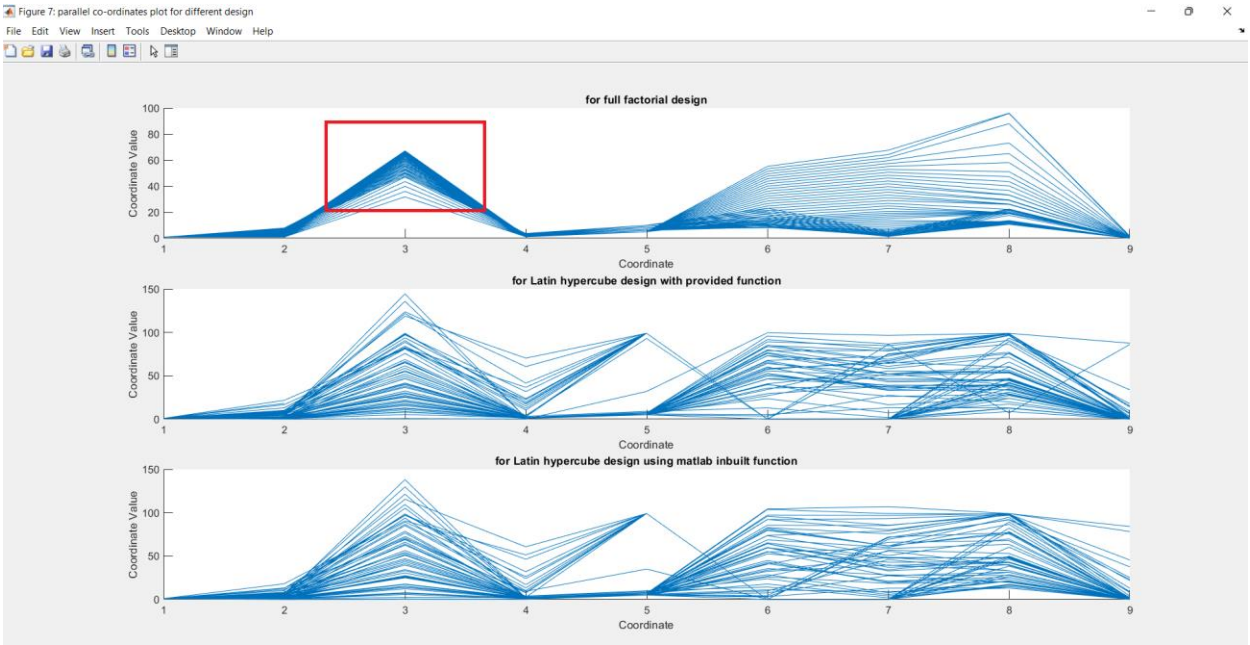


Figure 9 Parallel Coordinate plot of performance space for different sampling plans

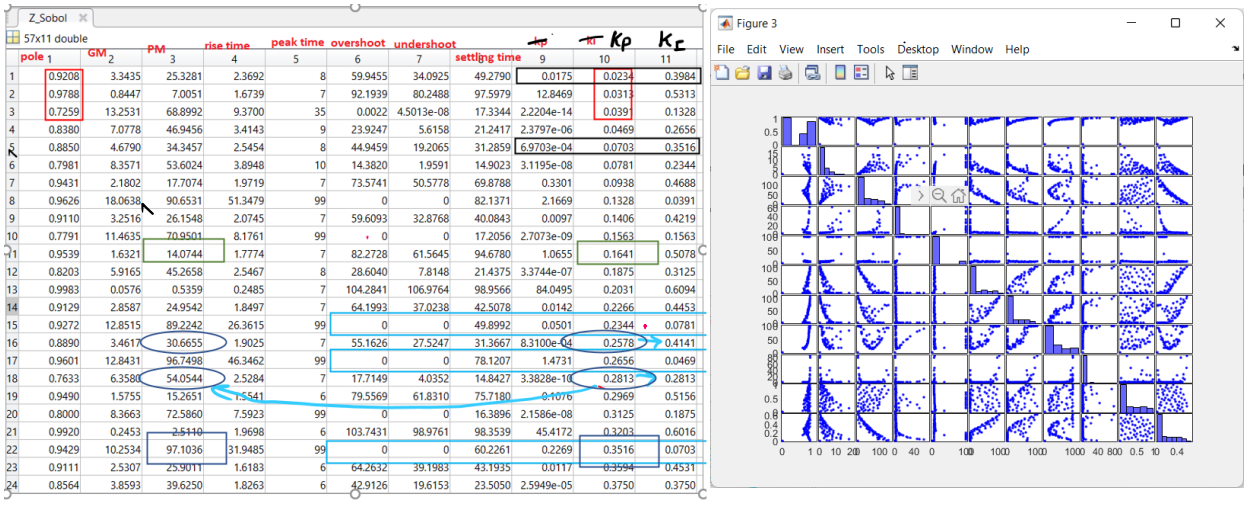
The following observations can be deduced from Figure 9

- The phase margin values obtained for full-factorial design seems concentrated in a range of 30 to 65 degree as indicated by red box in figure 9 however for Latin hyper cube and Sobol sampling these values seems to be scattered in all possible ranges. This is due to the fact that full-factorial design is linearly distributed.
- The parallel Co-ordinate plot for full factorial design is perfectly streamlined when compared to other 2 plots this is also due to the fact that the values in full factorial sampling plan are uniformly distributed whereas randomly distributed for other two sampling plans which gives better overall understanding of the performance domain.

The sampling plan chosen for further consideration is Sobol Sampling Plan. Following steps were taken to derive relationship between design variables and performance criteria Analytically

1. Add the value of kp and ki respectively to 10^{th} and 11^{th} column of the performance criteria.
2. All the rows with ‘NaN’ values were filtered out to avoid any confusion.

3. All the zero values were deleted from the obtained results
4. The values of performance criteria were sorted in increasing order of the kp values to deriver further interpretations.
5. The table of values obtained after following processing is shown below in Figure 10.



Z_Sobol — 57x11 double

	pole 1	GM 2	PM 3	rise time 4	peak time 5	overshoot 6	undershoot 7	settling time 8	Kp 9	Ki 10	Kc 11
1	0.9208	3.3435	25.3281	2.3692	8	59.9455	34.0925	49.2790	0.0175	0.0234	0.3984
2	0.9788	0.8447	7.0051	1.6739	7	92.1939	80.2488	97.5979	12.8469	0.0313	0.5313
3	0.7259	13.2531	68.8992	9.3700	35	0.0022	4.5013e-08	17.3344	2.2204e-14	0.0391	0.1328
4	0.8380	7.0778	46.9456	3.4143	9	23.9247	5.6158	21.2417	2.3797e-06	0.0469	0.2656
5	0.8850	4.6790	34.3457	2.5454	8	44.9459	19.2065	31.2859	6.9703e-04	0.0703	0.3516
6	0.7981	8.3571	53.6024	3.8948	10	14.3820	1.9591	14.9023	3.1195e-08	0.0781	0.2344
7	0.9431	2.1802	17.7074	1.9719	7	73.5741	50.5778	69.8788	0.3301	0.0938	0.4688
8	0.9626	18.0638	90.6531	51.3479	99	0	0	82.1371	2.1669	0.1328	0.0391
9	0.9110	3.2516	26.1548	2.0745	7	59.6093	32.8768	40.0843	0.0097	0.1406	0.4219
10	0.7791	11.4635	70.9501	8.1761	99	0	0	17.2056	2.7073e-09	0.1563	0.1563
11	0.9539	1.6321	14.0744	1.7774	7	82.2728	61.5645	94.6780	1.0655	0.1641	0.5078
12	0.8203	5.9165	45.2658	2.5467	8	28.6040	7.8148	21.4375	3.3744e-07	0.1875	0.3125
13	0.9983	0.0576	0.5359	0.2485	7	104.2841	106.9764	98.9566	84.0495	0.2031	0.6094
14	0.9129	2.8587	24.9542	1.8497	7	64.1993	37.0238	42.5078	0.0142	0.2266	0.4453
15	0.9272	12.8515	89.2242	26.3615	99	0	0	49.8992	0.0501	0.2344	0.0781
16	0.8890	3.4617	30.6655	1.9025	7	55.1626	27.5247	31.3667	8.3100e-04	0.2578	0.4141
17	0.9601	12.8431	96.7498	46.3462	99	0	0	78.1207	1.4731	0.2656	0.0469
18	0.7633	6.3580	54.0544	2.5284	7	17.7149	4.0352	14.8427	3.3828e-10	0.2813	0.2813
19	0.9490	1.5755	15.2651	1.5541	6	79.5569	61.8310	75.7180	0.1076	0.2969	0.5156
20	0.8000	8.3663	72.5860	7.5923	99	0	0	16.3896	2.1586e-08	0.3125	0.1875
21	0.9920	0.2453	2.5110	1.9698	6	103.7431	98.9761	98.3539	45.4172	0.3203	0.6016
22	0.9429	10.2534	97.1036	31.9485	99	0	0	60.2261	0.2269	0.3516	0.0703
23	0.9111	2.5307	25.9011	1.6183	6	64.2632	39.1983	43.1935	0.0117	0.3594	0.4531
24	0.8564	3.8593	39.6250	1.8263	6	42.9126	19.6153	23.5050	2.5949e-05	0.3750	0.3750

Figure 10 Figure Showing the interpretations derived from the performance space of the Sobol Sampling plan

One carefully analyzing the data following interpretations can be made:

- The phase margin values seem to lie in desirable range if Kp values lies near 0.25 and Ki value lies around 0.26 how ever if the value of kp is small and ki is large the phase margin is less than 30^{o} Moreover if Kp value is above 0.3 and Ki has small value phase margin value shoots above 80^{o}.
- Kp value is responsible to for improving transient response of the system. i.e., Rise time, peak time, maximum overshoot, maximum undershoot, settling time.
- Ki value is responsible for improving steady state performance i.e., steady state error.
- The value of Kp and Ki that seems to give results in optimum range as specified by the chief engineer seems to lie in range 0.2 to 0.3 for each value.
- If the value of Kp is greater than Ki by large margin, then we can see zero overshoot and zero undershoot which indicates an improvement in transient state response of the system reinforcing the fact that kp value is responsible for improving the transient response of the system.
- The system becomes unstable for higher values of decision parameters
- High correlation between maximum overshoot and maximum undershoot can be seen hence both display similar characteristics.
- if the value of Rise time decreases the value of maximum overshoot seems to be increasing.
- **Phase Margin** and **Rise time** seems to follow a trend where if we try to reduce the Rise time the value of phase margin decreases and tries to bring down phase margin below its optimum range.

Section 5: Optimization process

Optimization Approach (NSGA-II Optimizer)

Preliminaries:

To optimize the process, following changes were made to **evaluateControlSystem** to make a new function called **optimizeControlSystem**.

1. All the 'NaN' values were filtered out
2. The value obtained for gain margin is converted to decibels using the conversion factor $20 \log_{10} G_M$. As the EA toolkit provided assumes that all the objectives must be minimized, however the gain margin needs maximization therefor it is multiplied by -1.
3. Phase Margin needs to lie within a range rather than being maximized or minimized and this range is 30^o to 60^o so the absolute deviation of phase margin from the midpoint of the range is minimized. If the value of phase margin is greater than 45 then the value is subtracted by 45 to find the deviation from the mid-point and then this deviation can be minimized, similarly if the value is less than 45 then the value is subtracted from 45 to find its deviation from the mid-point so that this deviation can be minimized.
4. The maximum and minimum value of each criterion is capped to avoid any the operators relying on aggregating objectives to produce garbage output. However, Experimentation was done with several limiting values if the limiting values were kept too close to desired objectives most of the data was scrapped on the other hand if the capping was done leniently the optimization didn't reach the desired goal. The values of kp and ki that produces values of objectives outside the range specified above are skipped and the loop transfers to next parameter values.

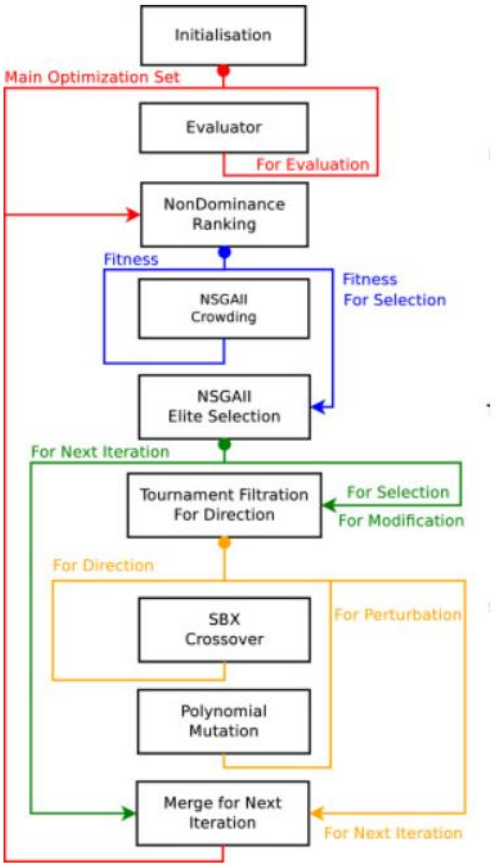
The Optimization engine:

The Steps involved in building the optimization engine are given below

1. Initializing the population: The favored sampling plan from lab is taken as the initial population for the optimizer. This population is then passed to **optimizeControlSystem** Function mentioned above.
2. Calculating Fitness: A loop of 250 iterations is initiated and inside this loop the first step is calculation of fitness of candidate design. To calculate the fitness two steps are taken
    - Non-dominated sorting: An individual A dominates another individual B if no objectives obtained by A is inferior to B and at least one objective of A is superior to B. This is done using **rank_nds** function.
    - Crowding Distance: Crowding distance is obtained by sorting the solutions in increasing order of objective function values. The distance value is the average distance between two neighboring solutions.

3. Performing Selection for Variation. Binary tournament selection is done in order to choose the candidates to be passed to the next generation. The candidates that are selected are passed to next generation this is done with the help of a k-way tournament. Tournaments takes place till the final selection. The parameter that decides the selection of a candidate is called selection pressure, which is a probabilistic measure of a candidate's selection. Rate of convergence of the algorithm is decided by selection pressure. This is done using **btwr** function.
4. Performing Variation: Two types of variations are carried out in this process.
    - Simulated Binary Crossover: It is a real parameter re-combination operator in which two parents are selected and returns two children. This is done using **sbx** function.
    - Poly Mutation: In this mutation one parent is taken and one child is returned. This is done using **polymut** function.

The populations are then combined and sent to Selection-for survival operator.



5.Performing selection for Survival: The combined population is passed through reducerNSGA_II function which gives the selected population based on survival of the fittest.

6. The next step is to monitor the convergence of the algorithm if the this is done using **Hypervolume** function. It takes in two inputs one is the objective and other is the reference values desired after convergence if the second parameter is not given it automatically takes nadir point as the reference point

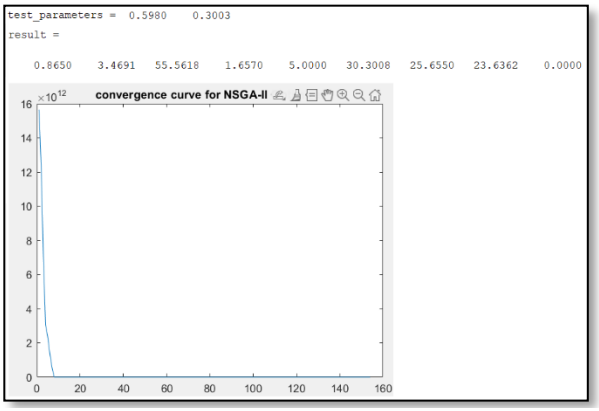
Preference Based Optimization

In this approach population is ranked based on preference. These preferences are the goals of the design problem. Each time the rank is calculated specific goals and priorities are fed to the the operator and it performs non-dominated sorting to optimize the parameters based on preferences. There are three scenarios in this problem.

- In first scenario priority is given only to closed loop pole
- In second scenario closed loop pole is kept at highest priority and stability criteria are kept next to it
- In Third Scenario priorities are given as per the demands of the chief engineer.

Section 6: Optimization Results

Multiple approaches have been tried and tested and their results are discussed below.



In first approach the NSGA Optimizer has been applied as per the instructions given in the LAB B and the results are shown in Figure 11. This optimization reached convergence in 160 iterations and the value of Test parameters obtained are kp=0.5980, Ki=0.3003. The desired parameters were not up to the mark for gain margin, maximum overshoot, maximum undershoot ,and settling time.

Figure 11 Results for NSGA-II Optimizer without filtering the parameters.

In Second approach the parameters are filtered in such a way that the objectives are already very close to the requirement. This was done on experimental basis and the results are shown in Figure 12. Convergence was reached in 140 iterations. The value of Kp=0.2372 & Ki=0.2732. all the objectives were met except for rise time which is 2.72 and maximum overshoot which is 15.9524. These are moderate constraints so overall the results can be excepted.

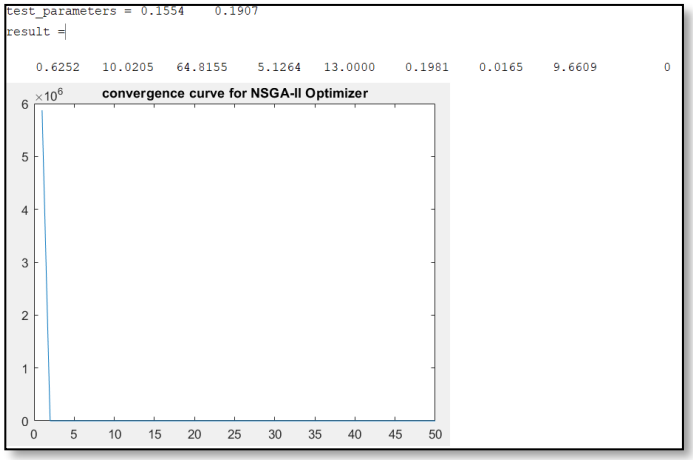


Figure 12 Result of NSGA-II Optimizer with preference given to closed loop pole

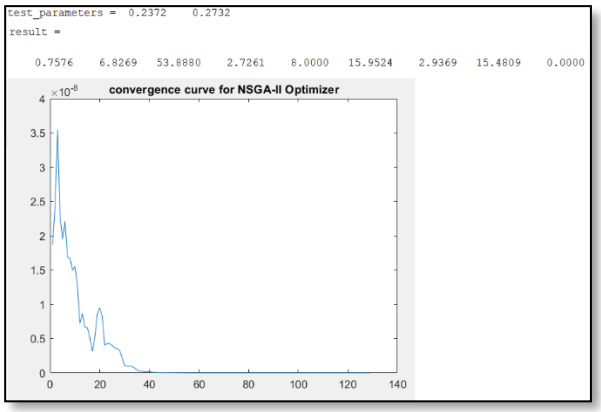


Figure 13 Result of NSGA-II Optimizer with aggressive filtering

The result for optimizer with preference given to closed loop pole are ok as all the other parameters reached desired objective except phase margin which is a high priority objective. It took 50 iterations to reach to this point and the value of Kp=0.1554 and Ki=0.1907.

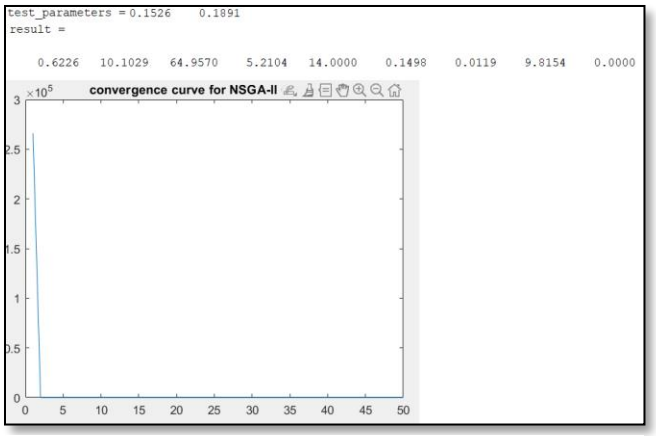


Figure 14 Result of NSGA-II Optimizer with preference given to closed loop pole given highest preference and stability criteria next to it

The results for NSGA-II optimizer with preference given to closed loop pole given highest preference and stability criteria preferred next to it . Convergence was not achieved till the 50th iteration. The value obtained for Kp=0.1526 and Ki=0.1891 and it can be observed that not that good as the phase margin was not as per the desired objectives similarly rise time also missed the mark with quiet a substantial amount and these are high and moderate priority criteria.

The result obtained with optimizer where preference were given according to chief engineers preferences are also not convincing enough as phase margin which is a high priority criteria and rise time which is a moderate priority criteria are not as desired

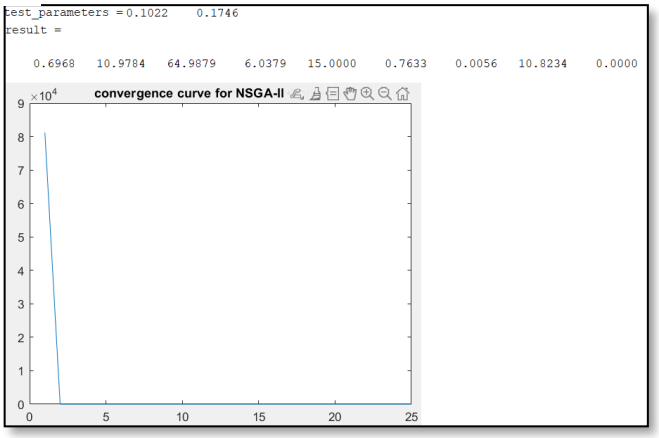
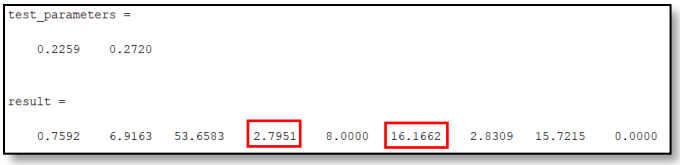


Figure 15 Result of NSGA-II Optimizer with preference given as per chief Engineer.

None of the Optimizers fulfilled the desired criteria of chief Engineer. The closest results achieved by any of the optimizers is shown below:



These results are not convincing but acceptable as only 2 parameters i.e., Rise time and Maximum overshoot which are Moderate priority criteria's have shown deviation from desired Goal.

Section 7: Recommendations

Based upon the learnings from knowledge discovery and Optimization Process the following recommendations can be made.

1. Various optimization approaches have been tried and tested but none appeared to satisfy Chief Engineer's preference for control system. However, one approach managed to satisfy all the hard constraints and high priority constraints, but it failed to fulfill two moderate priority constraints. Therefore, with some consideration the value of decision parameter obtained using this approach are aggregable and may be used further.
2. Analysis of objective values done during the knowledge discovery phase helps us in deriving following conclusion.
    - **Phase Margin** and **Rise time** seems to follow a trend where if we try to reduce the Rise time the value of phase margin decreases and tries to bring down phase margin below its optimum range.
    - On the contrary if the value of **Rise time** decreases the value of **maximum overshoot** seems to be increasing. There needs to be a **trade-off** in one amongst the two objectives.
    - On the other hand, value of largest closed loop pole and settling time seems to share a harmonious relationship as the value of pole decreases the value of settling time also decreases.
    - The system becomes unstable for high values of decision parameters i.e., Kp & Ki.
    - The region of optimality is seeming to be lying at a value of Kp and Ki lying in the range between 0.2 to 0.3.
3. In all the attempts to optimize the problems the optimizers seem to converge. This is due to the fact that in all the cases' objective function was not given to the **hypervolume** function, so the solutions seem to converge at **Nadir point**.
4. The best results for the control problem were obtained for the value of optimization parameters lying in the following range.

    Kp = 0.220 ~ 0.230 and Ki = 0.270~0.280

5. Although results were not obtained as per the desire, but all the major criteria were fulfilled and for the criteria's that doesn't satisfy it is evident from the study that there should be some tradeoffs between these criteria's.

Open Ended Solution: The best results that were obtained were by aggressive filtering of optimization parameters. This was done by taking a sobol set of 20000 points and then these values are evaluated using **optimizeControlSystem** function. This gives us 20000 solutions now these solutions are filtered in such a way that all the objectives lie within a certain range of desired values of objectives. A set of parameters corresponding to these values was made and then these parameters were assigned as initial population

Section 8: Conclusion

The problem helps in understanding the detailed procedure of decision making in design problem by developing a PI controller for a plant. This problem is analogous to the problem presented by client which requires implementation of a suitable Controller for the propulsion system also considering its effect on dynamic performance of the vehicle.

A propulsion control system helps in optimally distributing torque amongst the wheels of a car. The dynamic performance of a vehicle is highly dependent on propulsion control system as it is responsible for traction, braking, torque vectoring, driving efficiency and vehicle stability. As all these factors are dependent on the design of controller this makes the problem a Multi objective optimization problem. And hence all the strategies applied in designing the controller of plant can easily be implemented in designing the controller of the car.

Apart from using Genetic algorithm for this problem two other methods that can be used for optimization of this problem are:

Goal Programming: It can be considered as an extension of linear programming that can handle multiple normally conflicting objective measures. These measures are assigned with a target to be achieved. It uses an achievement function which minimizes the deviation from a set of target values. The deviation can either be a vector or weighted sum depending upon variation of Goal used.

MOGA (Multi-Objective Genetic Algorithm) : It is a guided random search method . It optimizes one variable at a time and can be used for optimizing multiple variables. Pareto fronts are used to illustrate the solution of MOGA. Pareto optimal set can be defined as a set of non-dominated solution frontier.

References

[1] Y. Z. a. E. W. T. N. Chen, "“A Fuzzy QFD Program Modelling Approach Using the Method of," *International Journal of Production Research,* vol. 24, no. 46, pp. 6823-6840, 2008.

[2] plos.figshare.com, "plos.figshare.com," [Online].

[3] M. J. S. a. E. K. Antonsson2, "Arrow’s Theorem and Engineering Design Decision Making," *Springer,* vol. 11, pp. 218-228, 1999.

[4] G. A. Hazelrigg^, "A Framework for Decision-Based," *Journal of Mechanical Design ,* vol. 4, no. 120, pp. 653-658, 1998.

[5] F. L. D. A. Cristina Renzi, "A review on decision-making methods in engineering design for the automotive industry," *Journal of Engineering Design ,* vol. 28, no. 2, pp. 118-143, 2015.

[6] A. S. Nurcan Sarikaya Basturk, "Performance comparison of population-based optimization algorithms for air traffic control," *Aircraft Engineering and Aerospace Technology,* vol. 92, no. 6, 2020.

[7] M. R. a. J. Montgomery, "Candidate Set Strategies for Ant Colony Optimisation".

[8] Y. Choi, "Comparison of Factorial and Latin Hypercube Sampling," *energies* .

[9] J. A. C. K. M. J. C. J.Fleming, "Liger: A cross-platform open-source integrated optimization and decision-making environment," *Applied Soft Computing,* vol. 98, 2021.

Appendix

Code for Lab A

```
clear
clc
%% Retrieving Value from first Task

Task_A_1;
A=X_fullfact;
B=X_rlh;
C=X_lhsdesign;
D=X_sobol;

%% plotting scatter plot

figure('Name','Scatter plot for different design');
subplot 221
plotmatrix(A(:,1),A(:,2));
title('for full factorial design')

subplot 222
plotmatrix(B(:,1),B(:,2));
title('for Latin hypercube design with provided function');

subplot 223
plotmatrix(C(:,1),C(:,2));
title('for Latin hypercube design using matlab inbuilt function');

subplot 224
plotmatrix(D(:,1),D(:,2));
title('for sobol sampling ');

%% Using evaluateControlSystem function on different sampling plans to see their results

Z_fullfact=evaluateControlSystem(X_fullfact);
Z_rlh=evaluateControlSystem(X_rlh);
Z_Sobol=evaluateControlSystem(X_sobol);

%% Plotting Parallel Co-ordinates

figure('Name','parallel co-ordinates plot for different design');
subplot 311
parallelcoords(Z_fullfact);
title('for full factorial design')

subplot 312
parallelcoords(Z_rlh);
title('for Latin hypercube design with provided function');

subplot 313
parallelcoords(Z_Sobol);
title('for Latin hypercube design using matlab inbuilt function');
 figure;
gplotmatrix(Z_Sobol)
```

Code for Lab B Task_B_1

```
%% Variable Naming Convention
% CP---> Combined population
% sp---> survived population


%% clearing the workspace and clearing the command window
clear
clc

%% Loading the Sampling plan for initialisation

P=struct2array(load("more_random_population.mat"));
% Task_A_1;
% P=X_sobol;
% ty=rescale(X_sobol,0.1,0.9)
% val=optimizeControlSystem(ty);

Convergence_data=[];
%% Initialising the population

 for iteration=1:250

Z=optimizeControlSystem(P);
P=[];
% Creating a new array of selected values of kp & ki after optimization
P(:,1:2)=Z(:,10:11);
% seperating the optimized values from the kp and ki values
Z=Z(:,1:9);

%% Calculating fitness

% Non-dominated sorting
rank=rank_nds(Z);

% Crowding Distance
distance=crowding(Z,rank);


% Getting the fitness value
maximum=max(rank);
fitness=maximum-rank(:,1);

%% Performing selection-for-variation
selected=btwr(fitness);
selected=unique(selected);

%% Perform Variation

% creating an array called parent of selected parameters
len=length(selected);
for i=1:len
    a=selected(i);
    if a>length(P)
        continue
```

```
    end
    parents(i,1)=P(a,1);
    parents(i,2)=P(a,2);
end

% creating bounds for inputting to different variation operator
kp_min=min(parents(:,1));
kp_max=max(parents(:,1));
ki_min=min(parents(:,2));
ki_max=max(parents(:,2));
bounds=[kp_min,ki_min;kp_max,ki_max];

% Simulated Binary Crossover
children_sbx=sbx(parents,bounds);

% polynomial mutation
children_polymut=polymut(parents,bounds);

%% performing selection-for-survival

%concatinating the two populations i.e. parent and children population
cp=cat(1,parents,children_sbx,children_polymut);

% Evaluating results for the population
cp_Z=optimizeControlSystem(cp);
cp=[];
cp=cp_Z(:,10:11);
cp_Z=cp_Z(:,1:9);

%finding the rank of combined population
rank_of_cp=rank_nds(cp_Z);

% finding the fitness of combined pipulation
max_rank_cp=max(rank_of_cp);
fitness_of_cp=max_rank_cp-rank_of_cp(:,1);

% crowding/distance for use in reducerNSGA_II operator
distance_cp=crowding(cp_Z,rank_of_cp);

% selection for survival operator
indices_sp =reducerNSGA_II(cp,rank_of_cp,distance_cp);

l=length(indices_sp);
for i=1:l
    b=indices_sp(i,1);
      Convergence_set(i,:)=cp_Z(b,:);
      passing_parameters(i,:)=cp(b,:);
  end


  convergence=Hypervolume_MEX(Convergence_set)
  P=[];
  P=passing_parameters;
  sprintf('iteration %d', iteration)
  Convergence_data(iteration,1)=convergence;
  if convergence==0
      break
```

```
112.  end
113.
114.   end
115.  test_parameters=mean(passing_parameters)
116.  result=evaluateControlSystem(test_parameters)
117.  plot(Convergence_data)
118.  title('convergence curve for NSGA-II Optimizer')
119.
```

Code for optimizeControlSystem function

```
1.  %
2.  % Simulation model for a PI controller.
3.  %
4.  % Z = evaluateControlSystem(X);
5.  %
6.  % Input:  X - a sampling plan
7.  %             (one design per row, one design variable per column)
8.  % Output: Z - performance evaluations
9.  %             (one design per row, one criterion per column;
10. %              criteria are...
11. %              1: maximum closed-loop pole magnitude
12. %              2: gain margin
13. %              3: phase margin
14. %              4: 10-90% rise time
15. %              5. peak time
16. %              6. overshoot (% points)
17. %              7. undershoot (% points)
18. %              8. 2% settling time
19. %              9. steady-state error (% points))
20. %
21. function Z = optimizeControlSystem(X);
22.
23. [noInds, noVar] = size(X);
24.
25. Z =[];
26.
27. warning off
28.
29. for ind = 1:noInds
30.   % Extract the candidate controller gains.
31.   kp=X(ind,1);
32.   ki=X(ind,2);
33.
34.   % Form the open-loop transfer function.
35.   olNum = [(kp+ki)*(1-exp(-1)) ...
36.            -kp*(1-exp(-1))];
37.
38.   olDen = [1 ...
39.            -1-exp(-1) ...
40.            exp(-1) ...
41.            0 ...
42.            0];
43.
44.   % Form the closed-loop transfer function.
```

```
45.   clNum=[(1-exp(-1))*(kp+ki) ...
46.          -(kp/(kp+ki))*(1-exp(-1))*(kp+ki)];
47.   clDen=[1 ...
48.          -(1+exp(-1)) ...
49.          exp(-1) ...
50.          (1-exp(-1))*(kp+ki) ...
51.    -(kp/(kp+ki))*(1-exp(-1))*(kp+ki)];
52.
53.   % Get stability measure.
54.   sPoles = roots(clDen);
55.   clStable = max(abs(sPoles));
56.
57.   olTF = tf(olNum, olDen, 1.0);
58.
59.   [gainMargin, phaseMargin, wcGP, wcPM] = margin(olTF);
60.
61.   % Do a unit step response.
62.   timeData = [0:1:100];
63.   outputData = dstep(clNum, clDen, timeData);
64.
65.   % Collect results where possible (stable).
66.   if clStable < 1
67.     riseTime = getRiseTime(timeData, outputData, outputData(end));
68.     settleTime = getSettlingTime(timeData, outputData, 0.02, ...
69.                                                  outputData(end));
70.   else
71.     riseTime = getRiseTime(timeData, outputData, 1.0);
72.     settleTime = getSettlingTime(timeData, outputData, 0.02, 1.0);
73.   end
74.   [overshoot, overTime, undershoot, underTime] = getShoots(timeData, outputData);
75.   ssError = getSSError(outputData);
76.
77.
78.
79.   % Assign to output variable.
80.   if clStable>1 || isnan(clStable) || isnan(gainMargin) || isnan(phaseMargin) ||
    isnan(riseTime) || isnan(overTime) || isnan(overshoot) || isnan(undershoot) ||
    isnan(settleTime) || isnan(ssError)
81.       continue
82.   end
83.   % code for optimizing phaseMargin
84.   if phaseMargin>45
85.       phaseMargin=phaseMargin-45;
86.   elseif phaseMargin<45
87.       phaseMargin=45-phaseMargin;
88.   end
89.   gainMargin=-1*20*log10(gainMargin);
90.   if clStable>1  || gainMargin>-5 || phaseMargin>15 || riseTime>3 || 100*overshoot>20 ||
    ssError>1
91.       continue
92.   end
93.
94.
95.   Z(end+1,1) = clStable;
96.   Z(end,2) = gainMargin;
97.   Z(end,3) = phaseMargin;
98.   Z(end,4) = riseTime;
```

```
99.   Z(end,5) = overTime;
100.    Z(end,6) = 100*overshoot;
101.    Z(end,7) = 100*undershoot;
102.    Z(end,8) = settleTime;
103.    Z(end,9) = 100*ssError;
104.    Z(end,10)= kp;
105.    Z(end,11)= ki;
106.    warning on
107.
108.  end
109.
```

Code for Task_B_2

```
1.  %% Variable Naming Convention
2.  % CP---> Combined population
3.  % sp---> survived population
4.
5.
6.  %% clearing the workspace and clearing the command window
7.  clear
8.  clc
9.  %% Data for Preferability
10. Task_A_1
11.
12.
13. %% Loading the Sampling plan for initialisation
14.
15. % P=struct2array(load("more_random_population.mat"));
16. Task_A_1;
17. P=X_sobol;
18.
19. Convergence_data=[];
20. %% Initialising the population
21. goal_vector_1=[0.7 -inf -inf -inf -inf -inf -inf -inf -inf];
22. goal_vector_2=[0.7 6 50 -inf -inf -inf -inf -inf -inf];
23. goal_vector_3=[0.7 6 50 2 10 10 8 20 1];
24. priority_vector_1=[1 0 0 0 0 0 0 0 0];
25. priority_vector_2=[2 1 1 0 0 0 0 0 0];
26. priority_vector_3=[3 2 2 1 0 1 0 0 1];
27.  for iteration=1:150
28.
29. Z=optimizeControlSystem(P);
30. P=[];
31. % Creating a new array of selected values of kp & ki after optimization
32. P(:,1:2)=Z(:,10:11);
33. % seperating the optimized values from the kp and ki values
34. Z=Z(:,1:9);
35.
36.
37.
38. %% Assigning goal_vector and Priority_vector before further processing
39.
40. if iteration<150
```

```
    goal_vector=goal_vector_1;
    priority_vector=priority_vector_1;
% if iteration<50
%     goal_vector=goal_vector_2;
%     priority_vector=priority_vector_2;
% if iteration<150
%     goal_vector=goal_vector_3;
%     priority_vector=priority_vector_3;
end

%% Calculating fitness

% Non-dominated sorting
rank=rank_prf(Z,goal_vector,priority_vector);

% Crowding Distance
distance=crowding(Z,rank);


% Getting the fitness value
maximum=max(rank);
fitness=maximum-rank(:,1);

%% Performing selection-for-variation
selected=btwr(fitness);
selected=unique(selected);

%% Perform Variation

% creating an array called parent of selected parameters
len=length(selected);
for i=1:len
    a=selected(i);
    if a>length(P)
        continue
    end
    parents(i,1)=P(a,1);
    parents(i,2)=P(a,2);
end

% creating bounds for inputting to different variation operator
kp_min=min(parents(:,1));
kp_max=max(parents(:,1));
ki_min=min(parents(:,2));
ki_max=max(parents(:,2));
bounds=[kp_min,ki_min;kp_max,ki_max];

% Simulated Binary Crossover
children_sbx=sbx(parents,bounds);

% polynomial mutation
children_polymut=polymut(parents,bounds);

%% performing selection-for-survival

%concatinating the two populations i.e. parent and children population
cp=cat(1,parents,children_sbx,children_polymut);
```

```
98.
99. % Evaluating results for the population
100.  cp_Z=optimizeControlSystem(cp);
101.  cp=[];
102.  cp=cp_Z(:,10:11);
103.  cp_Z=cp_Z(:,1:9);
104.
105.  %finding the rank of combined population
106.  rank_of_cp=rank_prf(cp_Z,goal_vector,priority_vector);;
107.
108.  % finding the fitness of combined pipulation
109.  max_rank_cp=max(rank_of_cp);
110.  fitness_of_cp=max_rank_cp-rank_of_cp(:,1);
111.
112.  % crowding/distance for use in reducerNSGA_II operator
113.  distance_cp=crowding(cp_Z,rank_of_cp);
114.
115.  % selection for survival operator
116.  indices_sp =reducerNSGA_II(cp,rank_of_cp,distance_cp);
117.
118.  l=length(indices_sp);
119.  for i=1:l
120.      b=indices_sp(i,1);
121.      Convergence_set(i,:)=cp_Z(b,:);
122.      passing_parameters(i,:)=cp(b,:);
123.  end
124.
125.
126.  convergence=Hypervolume_MEX(Convergence_set)
127.  P=[];
128.  P=passing_parameters;
129.  sprintf('iteration %d', iteration)
130.  Convergence_data(iteration,1)=convergence;
131.  if convergence==0
132.      break
133.  end
134.
135.   end
136.  test_parameters=mean(passing_parameters)
137.  result=evaluateControlSystem(test_parameters)
138.  plot(Convergence_data)
139.  title('convergence curve for NSGA-II Optimizer')
140.
141.
```