UNIVERSITY OF SHEFFIELD

ACS 6124: Decision Systems

Part II: Decision Systems for Engineering Design

AWAB SYED
6/1/2021

Name: Awabullah M. Syed
Student Number: 2002 000 33
Module Assessor: George Konstantopoulos

Executing Summary:

This report aims to explore the optimal tuning parameter for a PI Controller via a *Multi-Objective Optimization* approach amalgamated with the deduced optimal sampling plan and the *NSGA-II* Genetic algorithm as a decision-making tool for the engineering design problem. The problem was *Initialized* via Matlab to deduce a [0.9 0.4] and [0.01 0.02] as the maximum and minimum range of decision variable, K_p and K_I, respectively. The procedure involved progressively incorporating goals and priority of the performance criteria provided by the *Chief of Engineer*. The candidate designs obtained from the 250 iterations (generation) of the *NSGA-II* algorithm were extensively analyzed to determine the trade-off and harmonious relationship between performance criteria. Several potential designs were extracted (Table 3), none of the candidate designs met the desired *Rise Time* and *Gain Margin* criteria. The *Chief of Engineering* will require to reconsider the *Rise Time* while taking the trade-off between the performance criteria into considerations as well as potentially considering *Gain Margin* as a *Hard* constraint. *Design 1* with the $[K_p\ K_I]$ gain of [0.2148 0.2384] was selected as an optimal candidate based upon it effectively meeting all performance criteria except those explained above. *Design 4*, [0.1106 0.2128] was considered as the second-best option. The approach used to deduce the tuning parameter of the PI controller can also be extended to examine the suitable controller for the propulsion engine via considering the Simulink model as a '*black-box'* system. Lastly, several examples are also provided where the *Multi-objective optimization technique* is implemented as a decision-making tool in the automotive industry for vehicle design along with potential decision systems tool that could potentially be utilized for vehicle design problem though, this greatly depends on the type of problem at hand.

Contents

Multi-Objective Optimization for Eng. Design

Section 1: Multi-Objective Optimization for Engineering Design

Introduction

Now more than ever with complex engineering systems, it is vital to analytically support the decision making and critically evaluate the design parameter via *Algorithmic Approach*. For example, the optimal tuning parameter of the controller for the propulsion engine to ensure optimal performance of the system with respect to the set specifications. The classical approach where the decision is based on expert opinion incorporated with the *Trial-and-Error* method and prototyping to run multiple simulations hinders the selection of the optimal design while also being costly. Furthermore, the classical approach also requires extensive and careful evaluation of the design parameters which greatly consumes already limited resources and expenditure while also immensely increases the workload and pressure on the already stretched engineering team. Not to mention the experimental/human/random errors are always involved in the physical experiments while the computation approach is not affected by random errors.

From business and decision-making acumen, it is indisputable to move away from the classical approach and towards the modern approach of implementing the *Multi-Objective Optimization* method, an offline decision-making procedure implemented via a cheap-to-evaluate 'surrogate' model that emulates the response and considers the complex engineering system as a '*black-box*' (Alexander Forrester, 2008). Although, there are several methodologies to solving nonlinear multi-objective optimization even for the propulsion engine including defining a *scalarizing* function (combing all objectives to form a single function) as illustrated by (Sunith Bandaru, 2016) which is a numerical optimization technique. However, implementing *scalarizing* function approach results in a compromise between all objective but more importantly, "*requires prior knowledge about the expected solution*" which at times is difficult to attain. Hence, the state-of-the-art population-based *Multi-Objective Evolutionary Algorithm* with the *Pareto-optimal front* approach will be ideal to derive the optimal controller for propulsion engine as well as for other parts of the nuclear-powered vehicle.

This report will tune the Proportional-Integral, *PI Controller* and examine the relationship between the design variables, K_p & K_I gains and the performance criteria. Additionally, this report will also explore ways to visualize the result and look for the trade-off between performance criteria, in an attempt to optimize the gains to meet the desired specification via implementing the *NSGA-II* algorithm to obtain decision variables in the stipulated design space.

Literature Examples for Usage in Vehicle Design

Population-based algorithms for *Multi-Objective Optimization of Engineering Design* e.g., *NSGA-II* (Multi-Objective Genetic Algorithm) will be ideal to thoroughly evaluate complex engineered product i.e., powertrain (Purshouse). This approach which is immensely used in the automotive industry to examine various trade-offs and objectives particularly in battery designing will further equip the design engineer to use resources efficiently and identify possible design solutions via understanding the correlation between design variables and performance criteria (Dandurand, April 2013, Vol. 135). The *Multi-Objective Optimization, MOO,* approach will also enable to obtain optimal solution and achieve an effective trade-off between objectives that at times are conflicting with no one solution that can optimize all objectives. Since *MOO* incorporates multiple objectives while providing the opportunity to classify constraints with respect to priority level.

Multi-Objective Optimization involving Evolutionary Algorithms has been extensively used in the automotive industry for decision-making, ranging from evaluating electric vehicle battery and storage modelling to determining optimal wheel size concerning specified objectives and constraints. Following are a few brief examples where multi-objective optimization approach was implemented for decision-making in the automotive industry:

Developing an electric vehicle with two main goals and three adjustable parameters.

Goals / Performance Criteria	**Decision Variables**
Maximum Acceleration	Wheel size
Maximum Range	Electric motor power
	Battery capacity

This is an ideal vehicle design problem where goals are conflicting since to increase range, requires a substantially large battery, adding weight and thus, reducing acceleration. Modifying the design parameters will have a great impact on the acceleration and the range however, selecting the optimal parameter will require a significantly large amount of trial and error. This is where the decision-making approach incorporated with the *Multi-Objective Genetic Algorithm*, *MOGA* plays an important role. MOGA will provide a cheap-to-evaluate process which will consider various possible candidate solutions and implement Genetic Algorithm e.g., *NSGA-II* to provide optimal parameter while considering and prioritizing the objectives/goals. This will also enable to easily visualize the correlation between the parameters and goals while also provide a trade-off between conflicting objectives and deliver the best of both worlds, acceleration and range. Although, the actual method is not described in-depth since the engineering team will approach each optimization problem depending on its need but one of the approaches that can be utilized is the *Pareto* method through identifying *Pareto* front and *Pareto* optimal solutions amalgamating it with a Genetic-inspired Algorithm that produces 'offspring' of better solutions with respect to fitness i.e., Non-dominant sorting. This is handled by our advanced engineering team to provide innovative and cutting-edge solutions (Copado-Mendez, 2012).

Battery Hardware design problem for a hybrid electric vehicle

The article published in the *Journal of Mechanical Design* by the author, (Dandurand, April 2013, Vol. 135) illustrates the implementation of *Pareto front* optimization to tackle one of the widely researched areas of the automotive industry, packaging of electric power technology in hybrid vehicle. This is one of the major area that is being extensively explored in an attempt to move towards environmental friendly electric vehicles due to multiple conflicting objectives.

Goals / Performance Criteria	**Decision Variables**
Minimize weight	Battery cell shape
Minimize cost	Cell size
Minimize size	Cell layout
	Spacing between c

Potential goals and decision variables to optimize battery performance of the hybrid vehicle are shown above, though other objectives and decision variables also contribute, including cell temperature deviation and heat flow coefficient in the vehicle, however, ignored in this example.

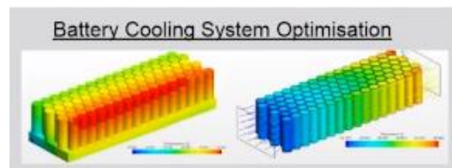

Figure 1 - Battery Cooling System Optimization: Temperature Deviation (Lotus, 2021)

During an online seminar arranged by *ATCx*, renowned Lotus Engineering (Lotus, 2021) represented the battery optimization result shown in Figure 1, obtained through the implementation of *Multi-Objective Mass Optimization* to deduce ideal design variables along with MATLAB Simulink for visualization. By implementing this approach for decision-making, Lotus Engineering was able to easily and effectively evaluate a complex battery system with interacting components. Implementing a Multi-objective approach "*not only allowed deduce optimal design variables with respect to performance criteria and priority level but also enabled to obtain Modular Battery & Motor Design"* which is pertinent to all vehicle types including hybrid/electric cars, trucks, and buses. This approach of multi-objective optimization, MOO also enabled to consider other important factors that play an indirect role such as sourcing (Economies of Scale), assembly time (reduce complexity) and quality (durability) of the battery system. Hence, to design an optimal nuclear-powered vehicle, it is crucial to implement the cutting-edge MOO decision-making approach to determine suitable controller gains or to optimize powertrain, where multiple objectives are in conflict.

<u>Other examples include</u>: Cost optimization design for Powertrain mount system for electric vehicles (Nguyen Huy, 2014)

Classes of population-based optimizers for Multi-Objective Optimization Process

There are three main classes of population-based optimizers for Multi-Objective Optimization, *Pareto-Based Method, Decomposition-Based Method* and *Indicator-Based Method.*

Pareto-Based Methods

This method utilizes the Pareto-dominance approach shown in Figure 2 to evaluate the quality of the population referred to as *fitness*. One of the main obstacles of implementing this approach arises when optimizing more than three objectives and limitations develop (Fonseca, 1995). According to (Deb), this method is severely criticized for its inability to scale *"gracefully"* to a high number of objectives which is further supported by (Teytaud) and (Hadka, 2012).

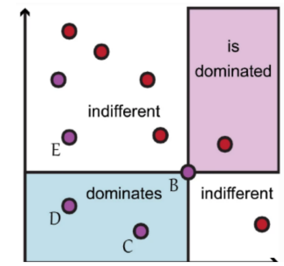

Figure 2 - Pareto-Dominance (Shahin Rostami, 2020)

Decomposition-Based Methods

Rather than using Pareto-dominance, the decomposition approach depends on *scalarizing function* to combine multiple objectives into a single optimization function as described in the *Introduction* section. The article published in *Springer Berlin*, (Giagkiozis, 2013) points out that this approach may cause slight issues due to the "*selection of the weighting vectors controls the distribution of solutions on the Pareto front"*.

Indicator-Based Methods

This approach of solving multi-objective optimization problem is largely based on measuring the quality of the solution set and is normally incorporated with *Pareto-based* methods. This approach is used to indicate the strengths and weaknesses of different algorithms. The most popular *Indication-based methods* being the *Hypervolume* which is used to evaluate the PI controller parameters in this report (Zitzler, 2004).

Section 2: Problem Formulation

$$G_D(z)G_0(z)G(z0 = \frac{(1-e^{-1})(K_P+K_I)z-K_P}{z^2(z-e^{-1})(z-1)} \qquad Eq.(1)$$

$$\min_x z(x,\boldsymbol{z}) \qquad Eq.(2)$$

$subject\ to$:

$$\begin{aligned} z_1 - 1 &\leq 0 \\ -z_2 + 6 &\leq 0 \\ -z_3 + 30 &\leq 0 \\ z_3 - 60 &\leq 0 \\ z_4 - 2 &\leq 0 \\ z_5 - 10 &\leq 0 \\ z_6 - 10 &\leq 0 \\ z_7 - 10 &\leq 0 \\ z_8 - 20 &\leq 0 \\ z_9 - 1 &\leq 0 \end{aligned}$$

Performance Criteria $\boldsymbol{z}$	**Direction**	**Goal**	**Priority**	**Decision Variables** $\boldsymbol{x}$	**Notions**
Largest closed-loop gain	Minimize	< 1	Hard Constraint	K_P & K_I	K_P & $K_I \in \mathbb{X}$ $z_i \rightarrow \mathbb{R}$
Gain Margin	Minimize	$-6\ dB$	High		
Phase Margin	Range	≥ 30 & $\leq 60°$	High		
Rise Time	Minimize	$2\ sec$	Moderate		
Peak Time	Minimize	$10\ sec$	Low		
Maximum Overshoot	Minimize	10%	Moderate		
Maximum Undershoot	Minimize	8%	Low		
Settling Time	Minimize	$20\ sec$	Low		
Steady-State Error	Minimize	1%	Moderate		

Table 1 - Performance Criteria | Design Variables | Goals (Objectives) | Priorities

NOTE: while evaluating decision variables, it was ensured that the candidate design resulted in a stable closed-loop response with $\left[K_{P_{MIN}}\ K_{I_{MIN}}\ ;\ K_{P_{MAX}}\ K_{I_{MAX}}\right] = [0.001\ \ 0.001\ ; 0.9\ \ 0.4]$.

Equation 1 depicts the open-loop transfer function which is used along with the *'evaluateControlSystem'* and '*optimizeControlSystem'* MATLAB functions to tune the gains, K_P & K_I of a PI controller such that a feedback control system is an optimal solution concerning the set of requirements. Equation 2 illustrates the problem in formal mathematical terms, where $\boldsymbol{z}$ represents the performance criteria (vectorial), x indicates the decision variables with $K_P > 0$ & $K_I > 0$. The optimal decision variable is the one that minimizes the performance criteria while considering the trade-offs and priorities. In other words, a candidate solution, $x \epsilon\ D \subset \mathbb{X}$ in the design space which optimizes set criteria.

Section 3: Sampling Plan

Numerous sampling plans are available to construct the design space, ranging from random to Sobol, which produces different quality of space-filling based on the problem at hand.

Depending on the number of variables and the active role each plays, it may be necessary to spend some additional time at the initial phase of the implementation of the multi-objective optimization problem to perform the *screening* process to potentially improve the accuracy of the optimal result and avoid the potential *Curse of Dimensionality*. However, the *screening* process is not been implemented while tuning the gains of the PI controller since every variable "*have a considerable impact on the objective"* (Alexander Forrester, 2008) and much focus was put on the actual modelling and analyzing the trade-offs between the performance criteria. Additionally, there are only two decision variables, K_P and K_I which further reduces the impact of implementing the *screening* process.

Stratification [Full Factorial] Sampling Plan

Full factorial approach partitions space into uniformly distributed smaller hypercubes with the design space given by Equation 3.

$$\mathbb{X} = \prod_i \bigcup_{l_i=0}^{N_i-1} [x_i^{\,l_i}, x_i^{\,l_i+1}] \qquad Eq.(3)$$

The full-factorial approach fails to evaluate each hypercube thoroughly since the design variable, x_2 remains the same horizontally while design variable, x_1 remains constant vertically as illustrated by Figure 3.

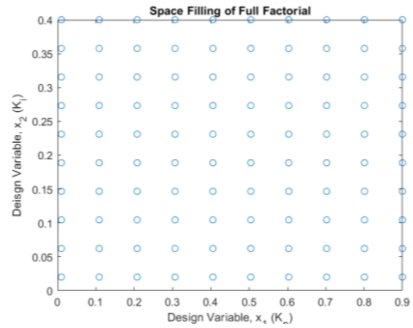

Figure 3 - Full Factorial Space Filling

Pertubed Full Factorial

Figure 3 shows that the implementation of the *Full Factorial* approach with a uniform distribution did not take advantage of the full design space hence, the hypercube was perturbed randomly (25 design variable pair were perturbed) as illustrated in Figure 4.

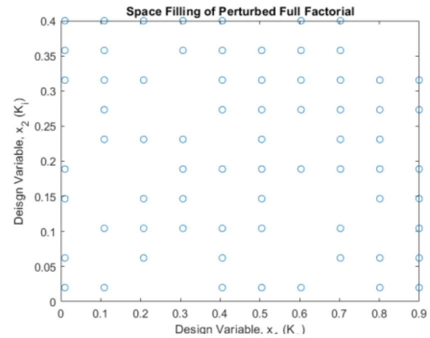

Figure 4 - Perturbed Full Factorial by 25

Latin Hypercube Sampling Plan

Random Latin Hypercube | Optimum (Morrison Latin Hypercube]

Figure 5 shows the *Random Latin Hypercube, RLH* design space which indicates multidimensional stratification and does not guarantee the space to be ‘space-filling. Hence *Morrison-Mitchell Latin Hypercube* approach was implemented to obtain Figure 6. This approach initializes RLH and then mutate through random perturbation yielding in ‘parents’. This is a brief process/description of the *Morrison-Mitchell Optimal Sampling Plan*, a detailed process that can be extracted from (Max D. Morris, 1995).

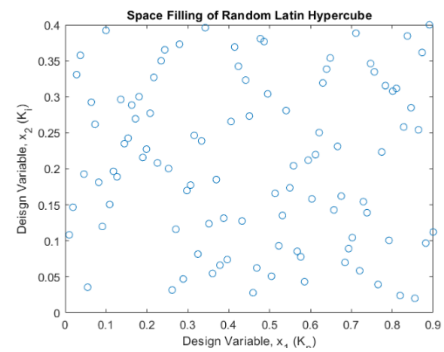

Figure 5 - Random Latin Hypercube

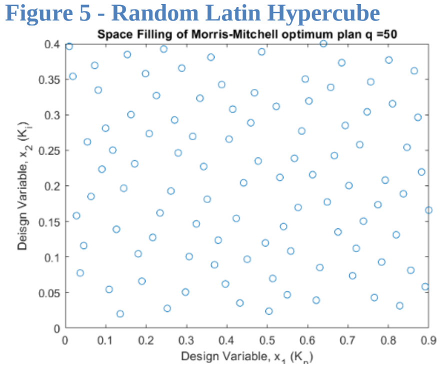

Figure 6 - Morrison Michell Optimal Plan

Sobol Sampling Plan

In Additional to the Full Factorial and Latin hypercube, the Sobol sampling plan was also implemented to deduce the design space illustrated in Figure 7 and *Morrison-Mitchell* criteria, phi metric - Equation 7 was determined.

NOTE: Relevant Matlab file can be extracted via QR code on the front page.

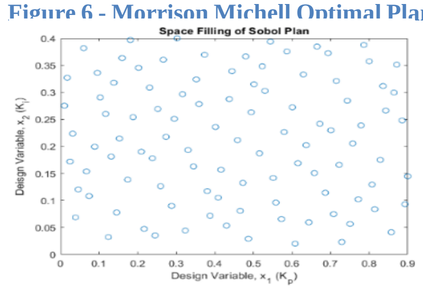

Figure 7 - Sobol Sampling Plan

Comparing Sampling Plans

$$\Phi_q = \left(\sum_{j=1}^{\mu} \frac{J_j}{d_j{}^q} \right)^{1/q} \quad where: q = 1 \quad J_j \to Min \quad d_j \to Max \qquad Eq.(4)$$

$$d(x_s, x_t) = \left[\sum\nolimits_{l=1}^{\mu} (x_s - x_t)^2 \right]^{1/2} \qquad Eq.(5)$$

The sampling plan with the smallest *Morrison-Mitchell* criterion, Phi metric, Φ_q is deemed optimal. It is apparent from the design space shown in Figure 3, Figure 4 and Figure 5 that the *Full Factorial*, *Perturbed Factorial* and *RLH* did not take full advantage of the design space which was also evident from the Phi metric, Φ_q as the values were significantly larger than the *Sobol* and M*orris* sampling approach. The difference between *Sobol* sampling and *Morris* was quite insignificant as evident from Figure 6 and Figure 7. However, the *Morris* sampling plan had the lowest Phi metric, Φ_q and was deduced the optimal sampling plan. This is mainly due to the fact that *Morris* is an extensive approach that already incorporates Phi metric while processing the design space and also begins with *RLH*. There are several other approaches to constructing design space including *'Space-filling Subets'* however, it was deemed outside the scope of this report. Therefore, *Morrison Latin Hypercube* was considered the optimal design space and is taken forward to describe the correlation between design variables and performance criteria.

Figure 8 shows the scatter plot for the *Optimum Morrison* sampling plan with respect to the performance criteria shown in Table 1. Each column of the scatter plot indicates the performance of each criterion. The plots in Figure 8 are symmetrical about the diagonal and can be used to deduce the decisiveness of each performance criteria. For example, the bar chart in column 1 and row 1 indicates that the candidate design with the largest pole approximately at 1 with K_p and K_i of 0.8 and 0.4, respectively. Scatter plots are extensively used to select optimal design variable pair and is also be used to determine trade-off in *Section 7*.

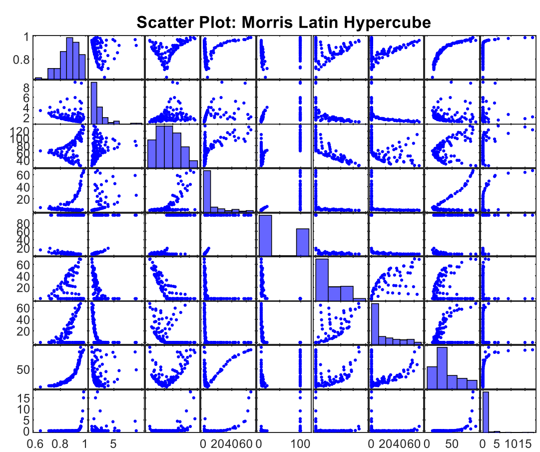

Figure 8 - Scatter Plot of Morrison Latin Hypercube with respect to performance criteria

Section 4: Knowledge Discovery

Morrison Latin Hypercube was selected due to its optimal space-filling (processing time ignored) and was taken forward to examine the correlation between the design variables and performance criteria.

Chernoff Face Plot Key	
Performance Criteria	**Facial Feature**
Magnitude of pole	Size of the face
Gain Margin	Forehead/jaw arc
Phase Margin	Shape of forehead
Rise time	Shape of jaw
Peak Time	Width between eyes
Max Overshoot	Vertical position of eyes
Max Undershoot	Height of eyes
Settling Time	Width of eyes
Steady-state error	Angle of eyes

Table 2 - Chernoff Face Plot feature identifier

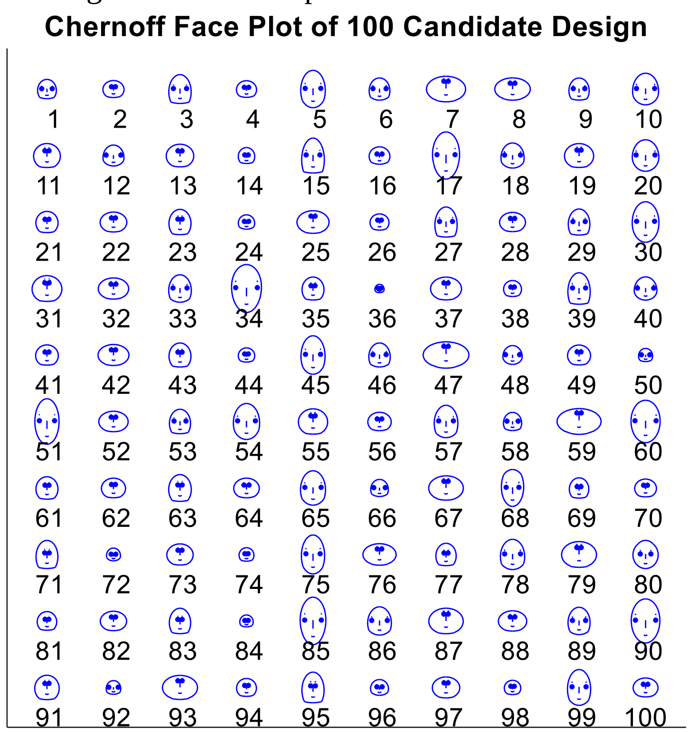

Figure 9 - Chernoff Face Plot of 100 Candidate Designs indicating nine performance criteria (MATLAB , n.d.).

The *Chernoff Face* plot is shown in Figure 9 shown 100 candidate designs with facial features depicting the relevant performance criteria. This plot is used to evaluate candidate designs with respect to the performance criteria. Candidate design, 36 has a significantly smaller *'Magnitude of largest pole in the closed-loop transfer function'* as indicated by the size of the face, however, has substantially higher *Settling Time.*

This plot was used to examine the candidate designs and it was concluded that implementing *Principal Component Analysis, PCA* dimension reduction approach via eigenvalues (orthogonal transformation) won't be suitable as it greatly affected few performance criteria. Meanwhile, the '*column'* method was used to map performance criteria onto [0,1] interval. This allowed obtaining a trade-off between not having enough information and information overload. However, *Chernoff Face Plot* does not explicitly depicts the relationship between the design variables and the performance criteria and is restricted in terms of accuracy and precision as it significantly banks on the visual perception of the reader to measure attributes/performance (Ward, 2008). Therefore, the *Parallel Coordinates* was implemented in Matlab to obtain the plot shown in Figure 10 which enabled to identify the relation between the candidate design performance and performance criteria shown in Table 2 as well as recognize any trade-off / conflicting objectives.

100 candidate designs via *Morrison Latin Hypercube* sampling plan were passed through the black box, *'evaluateControlSystem'*, to obtain the performance and a *Parallel Coordinate* plot was acquired. Each line in the plot represents a pair of design variable (candidate design) and indicates the performance. By observing the plot, potentially conflicting objectives were deduced. This includes the *Rise Time* & the *Phase Margin* and *Rise Time & Overshoot.* Therefore, the parallel coordinate plot allowed to gain an insight into the relationship between the pair of design variable and the performance criteria. This is further examined in *Section 6* and *Section 7*. Although, *'d3JS'*[1] library could have been used to further explore and better visualize the data but due to the simplicity of the *black-box* system, this was not considered, however, will be when evaluating the actual Simulink model of the propulsion engine.

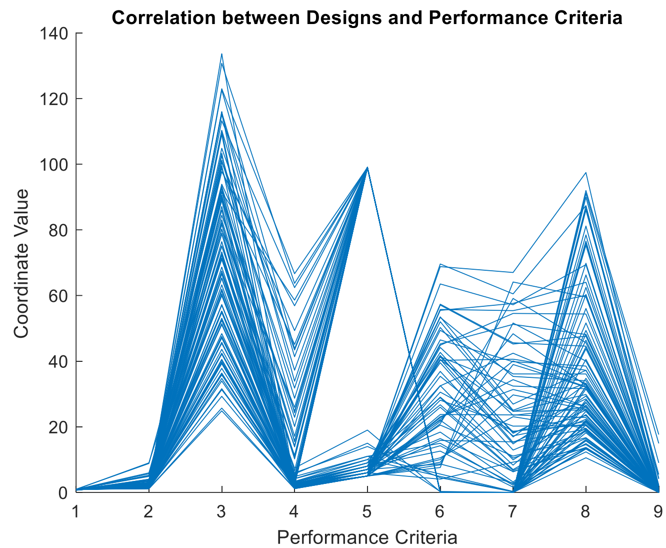

Figure 10 - Parallel Coordinate plot indicating the relationship between performance criteria and candidate designs

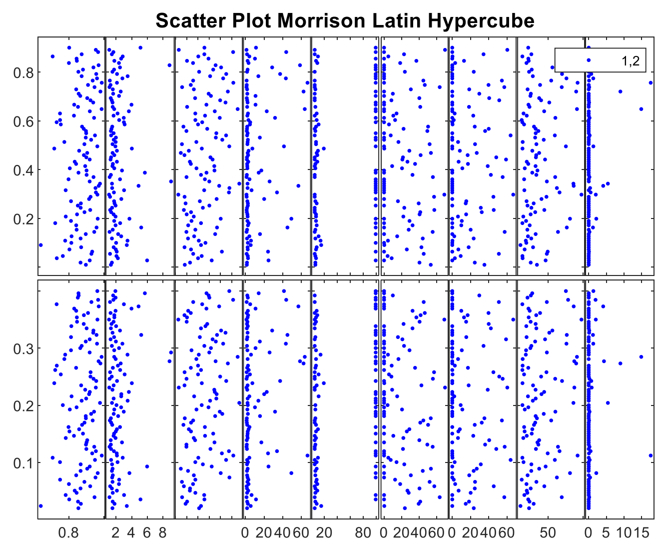

Figure 11 - Scatter Plot of the candidate designs obtained via Morrison Latin Hypercube design space

Figure 11 shows the scatter plot of the candidate designs of the *Morrison Latin Hypercube* design space, with each column representing the performance criteria shown in Table 2. The plot is divided into two horizontal sections, the upper horizontal shows the stability region of the design variable, K_p while the lower section represents, K_I. The K_P and K_I gains has the upper and lower range of [0.9 0.4] and [0.01 0.02], respectively.

[1] JavaScript Data Visualization Library

Section 5: Optimization Process

To construct a *Multi-Objective Problem,* a bio-inspired population-based optimizer genetic algorithm, *NSGA-II* [an improved NSGA which is criticized for its computational complexity (Bagchi, 1999)] was implemented which implies *Darwin's* survival of the fitness principle to deduce the *'fittest'* solution (Cunningham, n.d.).

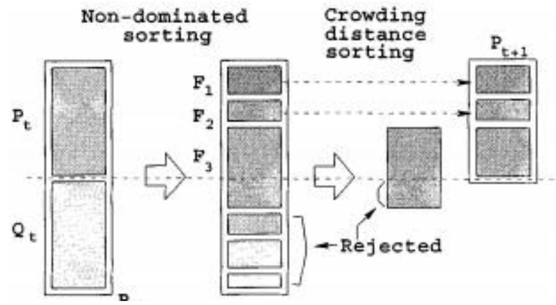

Figure 12 - Overall / Aerial view of the NSGA-II Algorithm (Deb)

NSGA-II Algorithm

The main loop of the algorithm is shown in Figure 12, where R_t is the combined (parent & offspring) population i.e., $P_t \cup Q_t$ with individual (candidate design) differing in the gene (design variable, K_p and K_I). The target indicators (*fitness value*) are determined for each candidate in the R_t via *Non-domianted sorting* resulting in various *Fronts*, F_1, F_2 (larger the *fitness value,* better the solution, smaller the front) with an example shown in Equation 6. The new population, P_{t+1} is chosen by selecting optimal individuals until the size of the original population is reached and the rest *Rejected*. However, if several members have the same *fitness value* but only certain can be selected due to limited space, *crowding distance sorting* is utilized which allows further nitpicking from the population, choosing the highest *crowding distance* population with some being *Rejected* as shown in Figure 12.Once the new population, P_{t+1} is formed, crossover and mutation (through *'sbx' &' polymut'* Matlab file) takes place to create a new generation, Q_{t+1}. The Matlab implementation of *NSGA-II* is shown in *Appendix C.*

NOTE: *Elitism* is ensured by utilizing previous and existing population in R_t

$$A(x_1, y_1)\ dominates\ B(x_2, y_2) \quad iff: (x_1 \leq x_2\ \& y_1 \leq y_2)\ \&\ (x_1 < x_2\ or\ y_1 < y_2) \qquad Eq.(6)$$

Offspring (new generation) is created iteratively as shown below:

1. *Tournament Selection* – different members of the population are compared in a binary tournament. From the parent population, P_t, two individuals are chosen and compared using *Non-dominant ranking* and if *rank* is equal then *crowding* takes place. Stronger candidates are chosen to 'reproduce', this leads to:
2. *Crossover* – in their genes (design variables).
3. *Random Mutation* occurs as shown in Figure 13.

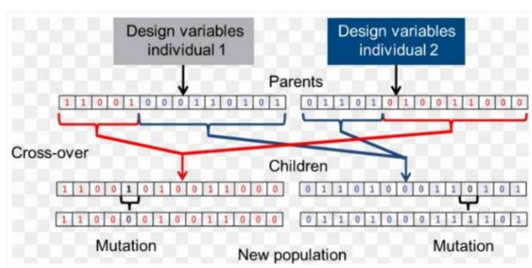

Figure 13 - Basic offspring (new generation) process (Dutta, 2019)

Incorporation Goals & Priorities

The goals and priorities of each performance criteria shown in Table 1 were implemented in Matlab by directly replacing the *Non-dominating sorting, 'rank_nds'* with the preferability operator, *'rank_prf'*, a basic example is shown in Equation 7 where v is dominated in g by u. A detailed mathematical form can be extracted from (Baldivieso-Monasterios, 2021). The preferability operator ranks performance criteria according to the priority i.e., hard and soft constraint and goals. The lowest priority of the performance criteria is denoted by zero and the performance criteria being minimized. A progressive approach was taken which allowed to gradually incorporate the preferences or the priority shown in Table 1, starting with hard constraint.

$$u \underset{g_{1,..p}}{\preccurlyeq} v \qquad Eq.(7)$$

To begin with, 50 iterations (generations) were computed with all the priority level set to zero except for the *Hard* constraint i.e., the *largest closed-loop pole* criteria which was set to one along with all low priority criteria goals set to $-\infty$ as illustrated in Figure 14. The next step was to compute additional 50 iterations with the *Hard* constraint set to one while others remained zero. The remaining 150 iterations were computed with the *Hard* constraint being further prioritised to two, while the *High* constraints were set to one and the rest shown in Figure 14 - *Step 3.*

This method enabled to incorporate goals and priorities progressively with the iterations (generations), allowing the generations to mutate accordingly.

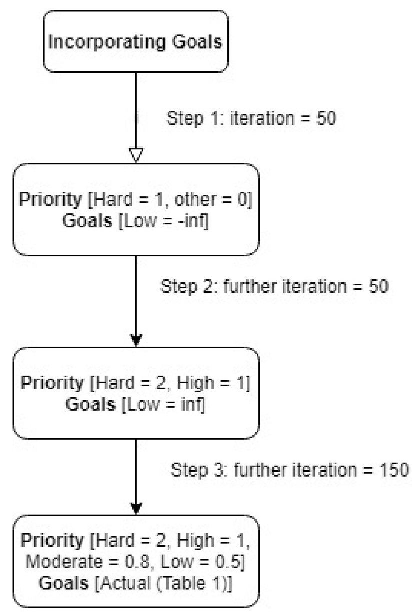

Figure 14 - Incorporating Goals & Priorities Procedure

Section 6: Optimization Results

Goals were incorporated in the three-stage process shown in Figure 14, during each stage the candidate designs were examined via *ParallelCoords* and *Scatter* plot to observe the behaviour concerning the performance criteria and identify any trade-offs and conflicts.

Step 1: First 50 Iterations

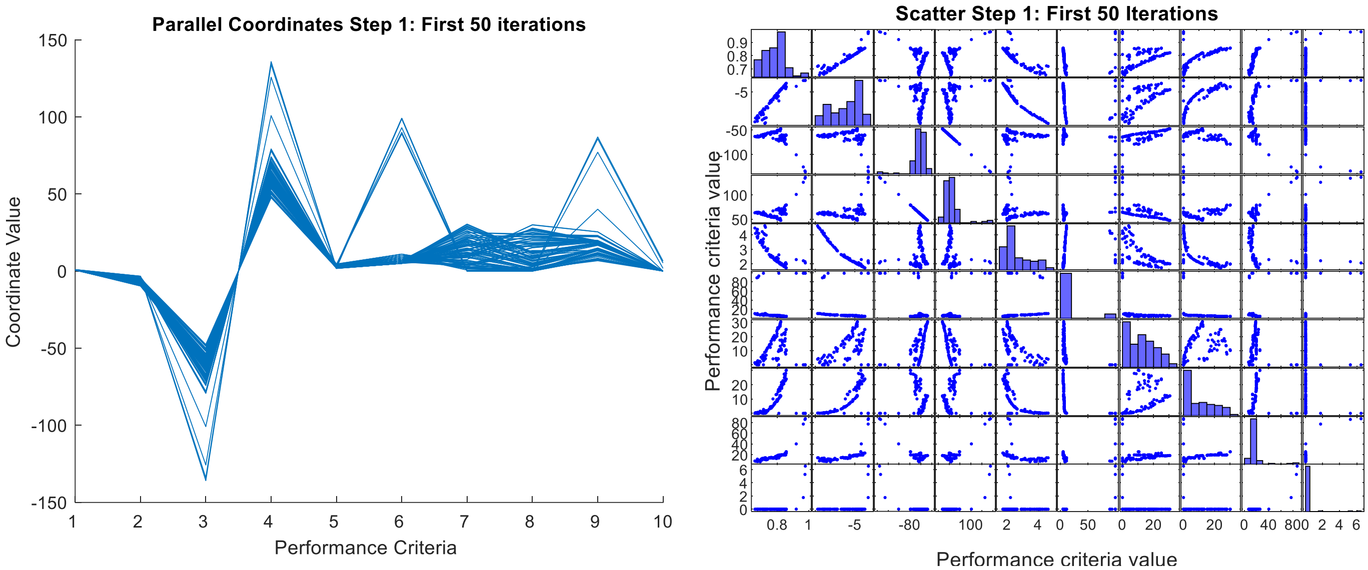

Figure 15 - Parallel Coordinate plot and Scatter plot with first 50 iterations (generations) - Priority [1 0 0 0 0 0 0 0 0 0] - Goal - [1 -6 -30 60 2 -inf 10 -inf -inf 1]

Step 2: Next 50 Iterations

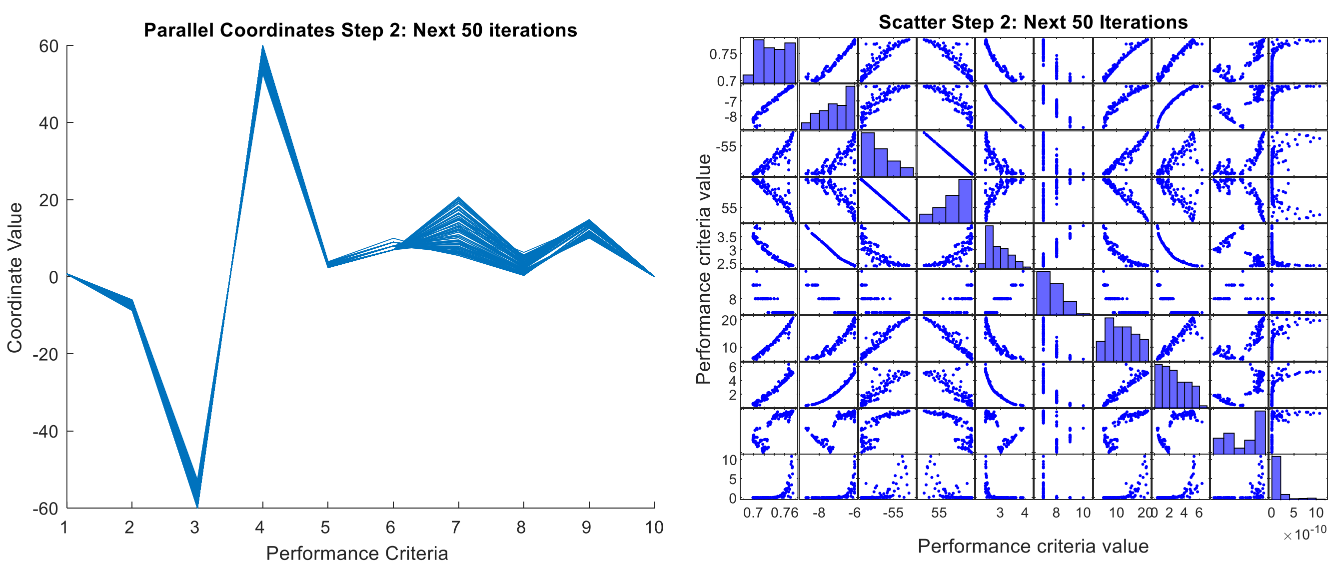

Figure 16 - Parallel Coordinate plot and Scatter plot with next 50 iterations (generations) - Priority [2 1 1 1 0 0 0 0 0 0] - Goal - [1 -6 -30 60 2 -inf 10 -inf -inf 1]

Step 3: Actual goals – Final 150 iterations

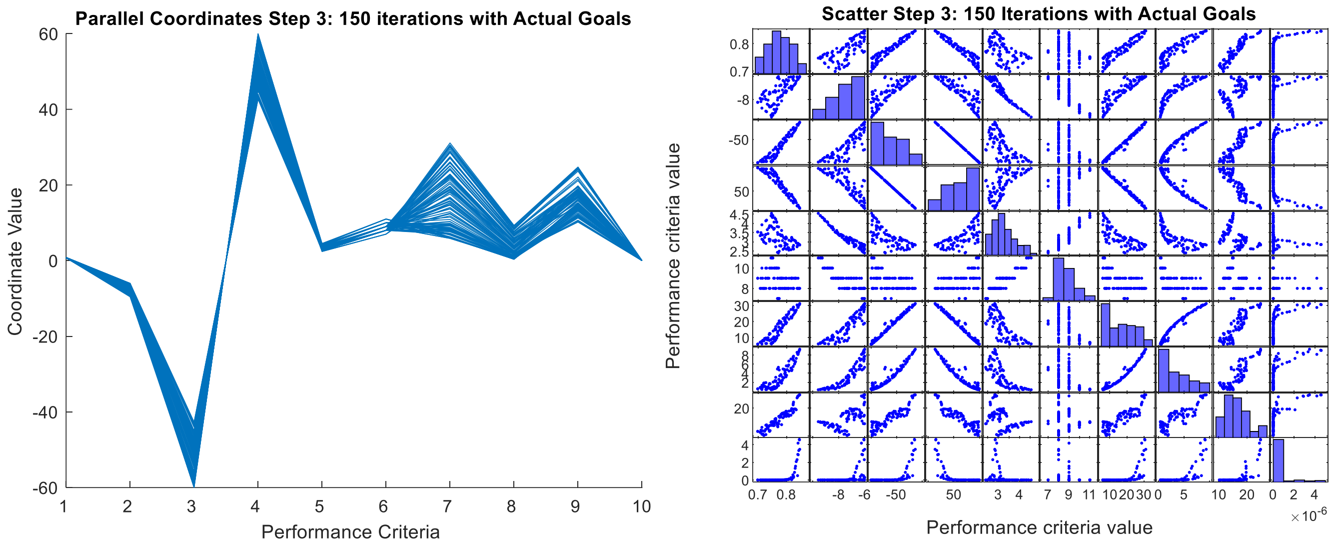

Figure 17 - Parallel Coordinate plot and Scatter plot with actual goals - [1 -6 -30 60 2 10 10 8 20 1] - Priority = [2 1 1 1 0.8 0.5 0.8 0.5 0.5 0.8]

The scatter plot in Figure 15 suggests that all design candidates effortlessly meets performance criteria, *'Largest closed-loop pole',* however, the majority of designs failed to meet other criteria and was very far off from the goals for certain criteria particularly *'Peak time'* and '*Maximum Overshoot'* as well as *'Settling Time'* and '*Steady-State-Error'*. This was predominantly due to the priority level set to zero for the rest of the criteria except the *'Largest pole'*. Hence, *Step 2* was necessary to ensure *High* priority goals are met after meeting the *Hard* constraint in *Step 1*.

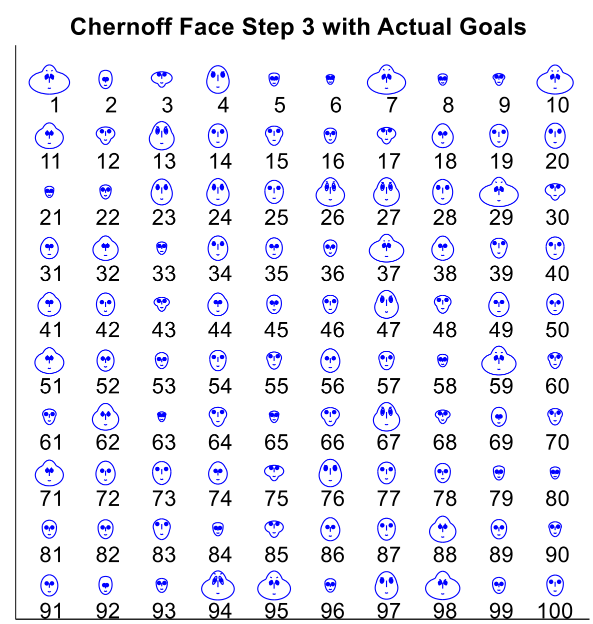

Figure 18 - Chernoof Face Plot – Key - Table 2

The scatter plot in Figure 16 illustrates that the *Hard* constraint is still within the desired specification but have enhanced the performance of *High* priority constraint including *Gain Margin* and *Phase Margin* which plays a significant role in ensuring the stability of the system. At the same time, the additional iterations also minutely improved other *Moderate* and *Low* priority performance criteria which were set to zero. Now it is ideal to consider other performance criteria that have *Moderate* and *Low* priority.

The remaining 150 iterations were carried out with the actual goal, $[1 - 6 - 30\ 60\ 2\ 10\ 10\ 8\ 20\ 1]$ and priority of $Hard = 2, High = 1, Moderate = 0.8, Low = 0.5$ i.e., [2 1 1 1 0.8 0.5 0.8 0.5 0.5 0.8].

The *Hypervolume* was utilized throughout the *NISA-II* implementation process to indicate the quality of the overall candidate design solutions/population. As a general rule, *'larger hypervolume value indicates better performance'* (Guerreiro).

Figure 17 indicates the potential trade-off and conflicting performance criteria especially between *Rise time* and Maximum *Overshoot.* This means increasing the performance of one criterion hampered the other criteria. The final candidate solutions were thoroughly examined to identify various trade-offs and to opt for the possible best candidate solution. A surprising observation was made while evaluating the *Rise time*, none of the candidate design met the goal of being less than two. This means that it may be necessary to relax the *Rise Time* criteria or adding it to the *High* priority list.

Furthermore, comparing the *Parallel Coordinate* plot of the *First* step with the *Third*, it is evident that as the priorities and the number of iterations increased, a smoother and better correlation with improved design candidate were being produced. The *Parallel Coordinate* shown in Figure 17 has a much slicker correlation between performance criteria than the one in Figure 15.

Figure 18 shows the *Chernoff* plot of the final candidate designs with the *features* key shown in Table 2. The vast majority of the design has the *Size of the face* significantly smaller than the candidate designs shown in Figure 9 with other performance criteria also being improved. This further support the value of adding the *Chief of Engineer*'s preferences into solving and optimizing the PI controller via implementing a *Multi-objective minimization* approach/algorithm.

Section 7: Recommendation

Firstly, it will be worth reconsidering the *Rise Time* goal of $< 2sec$ since none of the stable candidate design solutions was able to meet the criteria. From a controls engineering perspective, improving the *Rise Time* is simply achieved by increasing the K_p gains, however, crossing the maximum K_p range of 0.8 made the output of the black-box system unstable. Hence, it will be worth either relaxing the *Rise Time* criteria or exploring ways to improve the transient performance, possibly delving into *Model Predictive Control, MPC,* an advanced strategy widely used in the automotive industry for stringent performance requirement, supported by (Joseph Oncken, 2020) and (Stefano Di Cariano).

Secondly, since *Gain Margin* and *Phase Margin* plays an important role in determining the stability of the controller, it may be beneficial for the *Chief Engineer* to consider placing it in the *Hard* constraint priority list. This will further guarantee the robustness of the controller. The majority of the final candidate solutions after implementing the priorities have a *Gain Margin* within $2.0 - 3.5dB$ which lies at the lower-band of the industrially desired closed-loop system *Gain Margin*, $2 - 10\ dB$ (M. Sami Fadali, 2017).

Recommended Controller Parameters

Figure 8 and Figure 17 shows the performance of the candidate design before and after incorporating goals into the problem to obtain candidate designs, respectively. It is evident from the bar charts in Figure 8 that the majority of candidate designs not only just not met the performance criteria but were far away from it except for the '*Largest closed-loop pole*'. Incorporating goals via the process shown in Figure 14 amalgamated with implementing *NSGA-II* algorithm enabled selection and mutation of better candidate designs, resulting in superior final candidate designs shown in Figure 17. From these final candidate designs, after immense analysis, potential optimal solutions were extracted and are shown in Table 3.

	Design: $[K_p\ K_I]$			
Performance Criteria	**Design 1** [0.2146 0.2384]	**Design 2** [0.1307 0.2676]	**Design 3** [0.1013 0.2833]	**Design 4** [0.1106 0.2128]
Largest closed-loop pole: < 1	0.7090	0.8002	0.8282	0.7508
Gain Margin: $Max\ 6\ dB$	2.5004	2.3125	2.1699	2.9084
Phase Margin: $\geq 30°\& \leq 60°$	59.000	50.4674	46.558	58.822
Rise Time: $< 2\ seconds$	3.3600	3.1979	3.0857	3.99
Peak Time: $< 10\ seconds$	8.000	9.000	9.000	10
Max Overshoot: $< 10\%$	7.1776	19.3389	25.1198	6.97
Max Undershoot: $< 8\%$	0.843	3.5434	6.0227	0.4812
Settling Time: $20\ seconds$	11.000	18.8068	19.654	14.845
Steady-State-Error: $< 1\%$	0.000	2.543e-08	8.6e-7	7.49e-11

Table 3 - Potential Candidate Designs solutions

Several trade-offs and harmonious relationship between the performance criteria were observed (Table 4) during the analyses process to extract potential optimal designs.

Trade-off / Conflicting Criteria	**Harmonious Criteria**
↑ Rise Time | ↓ Maximum Overshoot	↑ Rise Time | ↑ Phase Margin
↑ Phase Margin | ↓ Maximum Overshoot	↑ Rise Time | ↑ Peak Time
↓ Settling Time | ↑ Phase Margin	
↑ Gain Margin | ↑ Rise Time	

Table 4 - General Trade-off and Harmonious relationship between Criteria

Numerous conflicting and harmonious relationship between performance criteria were observed with few major ones shown in Table 4. It is apparent that to meet the set *Rise Time* goal, it will be necessary to loosen the *Maximum Overshoot* criteria but at the same time, this will have an adverse effect on the *Phase Margin* since one increases the other. Hence, it may be worth increasing the desired *Rise Time* to $3.0\ or\ 4.0\ sec$, which will make *Design 1* and *Design 4* suitable choices.

Moreover, all candidate designs had the *Gain Margin* ranging from $1.9 - 3.2\ dB$, a general desirable range in the control systems are usually between $2 - 10\ dB$ as suggested by (M. Sami Fadali, 2017), (Bishop, 1967), (Ogata, 1970) and (Nise, 2012). Hence, it may be necessary to either reduce the goal of *Gain Margin* but is usually classified as important criteria therefore, it may be beneficial to consider *Gain Margin* and *Phase Margin* as *Hard* constraints. This will enable to add more emphasis on these two objectives while crossover/iterations occur.

Comparing the possible designs shown in Table 3, extracted from the final candidate solutions, it is apparent that the *Gain Margin* and *Rise Time* cannot be met even for the most optimal designs, *Design 1* and *Design 4* met most of the desired performance criteria except the common not met criteria, *Gain Margin* and *Rise Time.* Hence, *Design 2* and *Design 3* were eliminated, mainly due to the incredibly large *Maximum Overshoot.* While Design *1* has a *Phase Margin* close to the max range limit of $60°$ while *Design 4* has a *Peak Time* of $10s$ with *Rise Time* slightly greater.

Even though *Peak Time* has a *Low* priority, it is still desirable to meet the set goal, hence, *Design 1* with the gain of $[0.2146\ 0.2384]$ is recommended as the optimal candidate design with *Design 4* being the second-best but it will be worth reconsidering the *Rise Time* and having *Gain* and *Phase Margin* as a hard constraint.

Section 8: Conclusion

A multi-objective optimization approach amalgamated with *NSGA-II* bio-inspired algorithm to sort candidate design with respect to the *fitness* value and iterating new generation accordingly along with incorporating goals and priorities allowed to effectively observe the correlation between performance criteria, resulting in trade-offs and harmonious relationship shown in Table 4. This allowed extracting the optimal parameter of the PI controller and also gave various options to consider depending on the priority level. *Design 1,* $[0.2146\ 0.284]$ was deduced as the most reasonable while considering the priorities of the *Chief of Engineer* while *Design 4* being the second-best.

The same approach can be used to examine the suitable parameter of the controller for the propulsion engine. In addition to this, other methods and algorithms can also be used depending on the kind of problem presented. This could include exploring *NPGA*, *hybrid NSGA-III*, *NSGA-DO* and *SPEAR/R* algorithms (Adinovam H. M. pimenta, 30 November 2015). Population-based optimization techniques, PBOT, commonly used in the automotive industry for decision-making (Dao) to deduce optimal parameter while considering the trade-offs between performance criteria to effectively design a robust and stable controller for a complex system. Subsequently, the very same approach used in tuning the PI controller can also be used to deduce a suitable controller for the complex propulsion system via considering the Matlab Simulink model as a '*black-box'*. Implementing this approach will enable generating candidate solutions based on the priority and goals of the performance criteria and determine trade-offs, resulting in obtaining optimal parameter. Moreover, when designing a suitable controller for a complex propulsion engine, the *d3* library[2] along with *failure mode & effect analysis* and *fault tree analysis* will also be used for better visualization purposes as well as for analysis procedure (CIPS , n.d.) and (Hessing, n.d.).

Lastly, other significantly different decision-making system tools can also be utilized including *Fuzzy logic-based* and *Machine-Learning based* decision systems. *The fuzzy Logic* decision-making approach is generally used to evaluate systems with uncertainties incorporated in it and can also significantly reduce the computation time and the whole process, in general, supported by (H) and (Makrygiorgou). *Machine Learning* along with *Deep Neural Network* can also be employed to further support in producing quicker and better tuning parameters and with the potential to change/modify the tuning parameter in real-time based on the performance. Although this will increase complexity and will require additional computing power as well as needs further exploration for implementation (Bonaccorse, 2017). Equally, it will also be worth exploring other *Multidisciplinary Design Optimization Techniques* along with *Global Optimization Al*gorithms (Peri, 2008) as well as further extend the classical *Particle Swarm Optimization* techniques for multi-objective optimization as proposed by the *IEEE Congress on Evolutionary Computation* (Coello).

[2] JavaScript Data Visualization Library

Appendix: MATLAB Code

Appendix A: Sampling Plan

```
% Author : Awabullah Syed
% Date : 21th May 2021
% Description : Explore the design space for the problem - revealing the
%   relationships that exist between design variables and performance
%   criteria and communicating these relathiships clearly through
%   visualisation methods
clc; clear;
z = tf('z',ts);
% Gains (compare with stable and unstable system - if deemed necessary)
k = 2;  %No. of design variables
n = 100; % No. of desired points
Kp = [0.01];
Ki = [0.02];
KpMin = 0.01; KpMax = 0.9; %0.9
KiMin = 0.02; KiMax = 0.4;  % 0.4

K = [Kp Ki];
for i = 1:100
    Z(i,:) = evaluateControlSystem(K);
end
% Sobol Sampling
p = sobolset(k);
p.Skip = 1; %Skip first row
X_sobol = net(p,100); % Sobol Sampling

design_space = fullfactorial ([10 10],1); %Fulll Factorial [2 50]
design_space1 = rlh(n,k); % Latin

pert_fac = perturb(design_space,25); % Pertubed full fac by 25 numbers
% best_latin = bestlh(n,k, 100, 250); [save Morrison_Sampling.mat]
load Morrison_Sampling.mat % To avoid running everytime
space(:,1) = rescale(design_space(:,1),KpMin,KpMax); % Full Factorial
space(:,2) = rescale(design_space(:,2),KiMin,KiMax);

space1(:,1) = rescale(design_space1(:,1),KpMin,KpMax); % Latin
space1(:,2) = rescale(design_space1(:,2),KiMin,KiMax);

space2(:,1) = rescale(X_sobol(:,1),KpMin,KpMax);
space2(:,2) = rescale(X_sobol(:,2),KiMin,KiMax); % Sobol

space3(:,1) = rescale(pert_fac(:,1),KpMin,KpMax); %Perturbed Latin
space3(:,2) = rescale(pert_fac(:,2),KiMin,KiMax);
```

```
space4(:,1) = rescale(best_latin(:,1),KpMin,KpMax); % Morrison Latin
space4(:,2) = rescale(best_latin(:,2),KiMin,KiMax);

phiq_fac = mmphi(space,1,2); % Full factorial [Euclidean Distance]
phiq_latin = mmphi(space1,1,2); % Latin Hypercube design
phiq_sobol = mmphi(space2,1,2); % Sobol Sampling
phiq_pert = mmphi(space3,1,2); % Full Fac Perturbed
phiq_opt = mmphi(space4,1,2); % Best Optimal latin

disp(phiq_fac)
disp(phiq_latin)
disp(phiq_sobol)
disp(phiq_pert)
disp(phiq_opt);
%--------------------------------------Space Filling Plots---------------------------%
% Visually Compare Phi metric value
figure (1) % Full Factorial
plot(space(:,1),space(:,2),'o')
title('Space Filling of Full Factorial')
xlabel('Design Variable, x_1 (K_p)')
ylabel('Deisgn Variable, x_2 (K_i)')
figure (2)
plot(space3(:,1),space3(:,2),'o')
title('Space Filling of Perturbed Full Factorial')
xlabel('Design Variable, x_1 (K_p)')
ylabel('Deisgn Variable, x_2 (K_i)')
figure(3) % Latin
plot(space1(:,1),space1(:,2),'o')
title('Space Filling of Random Latin Hypercube')
xlabel('Design Variable, x_1 (K_p)')
ylabel('Deisgn Variable, x_2 (K_i)')
figure (4) % Optimal Morrison Latin Hyper
plot(space4(:,1),space4(:,2),'o')
title('Space Filling of Morris-Mitchell optimum plan q =50')
xlabel('Design Variable, x_1 (K_p)')
ylabel('Deisgn Variable, x_2 (K_i)')

figure (5) % Sobol
plot(space2(:,1),space2(:,2),'o')
title('Space Filling of Sobol Plan')
xlabel('Design Variable, x_1 (K_p)')
ylabel('Deisgn Variable, x_2 (K_i)')
%---------------------------------End of Sampling Plan-------------------------------%
```

Appendix B: Knowledge Discovery

```
%----------------------------Knowledge Discovery----------------------------%
figure (6) % Scatter Plot with Latin Hypercube sampling plan
Z1 = evaluateControlSystem(space1);
gplotmatrix(Z1)
title('Scatter Plot: Latin Hypercube Sampling Plan')
figure(7) % Scatter Plot with Full Factorial sampling plan
Z2 = evaluateControlSystem(space);
gplotmatrix(Z2)
title('Scatter Plot: Full Factorial Sampling Plan')
figure (8) % Scatter Plot with Sobol Sampling Plan
Z3 = evaluateControlSystem(space2);
gplotmatrix(Z3)
title('Scatter Plot: Sobol Sampling Plan')

figure (9) % Morrison Hypercube
Z4 = evaluateControlSystem(space2);
gplotmatrix(Z4)
title('Scatter Plot: Morris Latin Hypercube')

figure (10) % Star Plot with different sampling plan
glyphplot(Z4,'glyph','face','features',[1:9]);
title('Chernoff Face Plot of 100 Candidate Design')
print('-clipboard','-dmeta')

%-----------------------------------------Parallel Coordinates---------------------%
% Select the optimal sampling plan & then only used that to plot the
%   parallel coordinates plot
figure (11)
parallelcoords(Z4)
xlabel ('Performance Criteria')
title('Correlation between Designs and Performance Criteria')
print('-clipboard','-dmeta')

save Sobol_Space_Sampling.mat space2 % To be used in lab B
save Latin_Space_Sampling.mat space1 % To be used in Lab B
save Full_Space_Sampling.mat space % % To be used in Lab B
save Morrison_Latin_Space_Sampling.mat space4
% csvwrite('Z.csv',space4) % to be used for d3JS

%-----------------------------End of Knowledge Discovery--------------------------%
```

Appendix C: Optimization Process [NSGA-II Algorithm]

```
%-------------------Lab B -------------------------------------------%
% Author : Awabullah Syed
% Date : 28th May 2021
% DESCRIPTION: Identify a set of controller gains that will meet the
%   prefeerences of the Chief Engineer and consider trade-off in case the
%   preferences cannot be satisfied and resolve the design problem.

%-------------Initialize the population---------------------------------%
clc; clear;
mex sbx.c %Since toolbox written in C and wrapped using MATLAB
mex -DVARIANT=4 Hypervolume_MEX.c hv.c avl.c
mex rank_nds.c
mex rank_prf.c
mex crowdingNSGA_II.c
mex btwr.c
mex polymut.c

load ('Sobol_Sampling') % load  sampling
load ('Sobol_Space_Sampling')  %load space2 variable from lab A
load ('Morrison_Latin_Space_Sampling') % space4 variable, Morrison

 Z = optimizeControlSystem(space4);% Re-evaluate the design - post-processed

 %---------------------Calculating Fitness (NSGA-II)-----------------------%
% Non-dominated Sorting
% Goal = [1 0 0 0 -inf 0 0 -inf -inf 0]; % 5.2.1
Goal = [1 -6 -30 60 2 10 10 8 20 1];
% min_range = [0 -20 -30 0 0 0 0 0 1];
% max_range = [1 -6 60 2 10 10 8 20 1];
Level = [2 1 1 1 0.8 0.5 0.8 0.5 0.5 0.8]; % Step 1

% Goal Priorities
Hard = 2;
High = 1;
Moderate = 0.8;
Low = 0.5;
priority = [Hard High High High Moderate Low Moderate Low Low Moderate];
metric = Hypervolume_MEX(space4);

% [J,distinct_d] = jd(space2,2); %Euclidean p = 2
iteration = zeros(250,1);
for i = 1:150 % [N,M] = size(space2)
    pool = round(pop/2);
```

```
    nond_rank = rank_nds(Z); % non-dominated sorting
    fitness = max(nond_rank)-nond_rank +1;
    %Crowding Distance
    crowding_d = crowding(Z,nond_rank);
    % Selection for variation
    selectTourIndex = btwr([fitness,crowding_d]);
    selectTour = space4(selectTourIndex,:);
    % Z = optimizeControlSystem(parents);
    %----------------- performing variation-------------------%
    % Simulated binary crossover
    bounds = [0.1 0.2 ; 0.8 0.5];
    %bounds = [min(selectTour(:,1)),min(selectTour(:,2));...
        % max(selectTour(:,1)),max(selectTour(:,2))];
    offspring_sim = sbx(selectTour,bounds);
    % Polynomial mutation
    offspring_poly = polymut(offspring_sim,bounds); %Poly

    %----------------selection for survival-----------------%
    Z_offspring = optimizeControlSystem(offspring_sim);
    % Z_offspring_poly = optimizeControlSystem(offspring_poly);

    new_pop = [space4;offspring_poly]; % Polynomial mutation population
    % Selection for variation
[nond_rank_poly,fitness_poly,crowding_d_poly,Class] = evaluate(new_pop,Goal,Level);

    % Selection for Survival
    survivor = reducerNSGA_II(new_pop,nond_rank_poly,crowding_d_poly);

    iteration(i) = i;
    % use the survivor (after elitism) as new population
    space4 = new_pop(survivor,:);
   %  new_pop_indix = reducerNSGA_II(new_pop_sim,nond_rank_sim,
end

% Monitor
HV = Hypervolume_MEX(space4);

Z1 = optimizeControlSystem(space4);
% Z2 = evaluateControlSystem(space4);
figure(1) % Change the according to the step
parallelcoords(Z1)
title('Parallel Coordinates Step 3: 150 iterations with Actual Goals')
xlabel('Performance Criteria ')
print('-clipboard','-dmeta')

figure(2) % Scatter
```

```
gplotmatrix(Z1)
title('Scatter Step 3: 150 Iterations with Actual Goals')
ylabel('Performance criteria value')
xlabel('Performance criteria value')
print('-clipboard','-dmeta')

figure(3) % Chernoff Face plot
glyphplot(Z1,'glyph','face','features',[1:10]);
title('Chernoff Face Step 3 with Actual Goals ')
print('-clipboard','-dmeta')
%----------------------------------End of NSGA-II with Priority-------------------%
```

Bibliography

Adinovam H. M. pimenta, H. D. (30 November 2015). NSGA-DO: Non-Dominated Sorting Genetic Algorithm Distance Oriented . *IEEE International COnference on Fuzzy Systems (FUZZ-IEEE).*

Alexander Forrester, A. S. (2008). *Engineering Design Via Surrogate Modelling: A Practical Guide* . John Wiley & Sons, Incorporated .

Bagchi, T. P. (1999). *The Nondominated Sorting Genetic Algorithm: NSGA.* New York : Springer Boston . Retrieved from https://link.springer.com/chapter/10.1007/978-1-4615-5237-6_8

Baldivieso-Monasterios, P. R. (2021, April). *Interactive Decision Support - Part II: Decision Systems for Engineering Design* . Retrieved from The University of Sheffield .

Bishop, R. C. (1967). *Modern Control Systems* .

Bonaccorse, G. (2017). *Machine Learning Algorithms: Popular Algorithms for Data Science and Machine Learning* .

CIPS . (n.d.). *Failure Mode Effect Analysis FMEA.* Retrieved from cips : https://www.cips.org/knowledge/procurement-topics-and-skills/operations-management/failure-mode-effect-analysis/

Coello, C. A. (n.d.). MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. *IEEE Congress on Evolutionary Computation* .

Copado-Mendez, P. J. (2012). Multiobjective Optimization . *Computational Mechanics in Structural Engineering* .

Cunningham, C. (n.d.). *Survival of the fittest* . Retrieved from britannica: https://www.britannica.com/science/survival-of-the-fittest

Dandurand, B. (April 2013, Vol. 135). Equitable Multi-Objective Optimization Applied to the Deisgn of a Hybrid Electric Vehicle Battery . *Journal of Mechanical Deisgn* , 041004-1.

Dao, D.-N. (n.d.). New hybrid NSGA-III & SPEA/R to multi-objective optimization in a half-car dynamic model. *SAGE Journal* .

Deb, K. (n.d.). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II . *IEEE Transactions on Evolutionary Computation*, Volume:6, No2, April 2002.

Dutta, A. (2019, June 21). *Crossover in Genetic Algorithm*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/

Fonseca, C. a. (1995). Multiobjective Genetic Algorithms Made Easy: Selection Sharing and Mating Restriction,. *International Confernece on Genetic Algorithms in Engineering Systems: Innovations and Applications, IET* , pp. 45-52.

Giagkiozis, I. P. (2013). Generalized Decompostiion . *Evolutionary Multi-Criterion Optimization Lecture Notes in Computer Science* .

Guerreiro, A. P. (n.d.). *The Hypervolume Indicator: Problems and Algorithms.* University of Coimbra, Portugal .

H, J. Z. (n.d.). *Fuzzy Decision Support Systems* (Vol. Volume 162). Springer . Retrieved from https://link.springer.com/chapter/10.1007/978-3-642-58930-0_11

Hadka, D. a. (2012). Diagnostic Assessment of Search Controls and Failure Modes in Many - Objective Evolutionary Optimization . *Evolutionary Computation* .

Hessing, T. (n.d.). *Fault Tree Analysis*. Retrieved from sixsigmastudyguide : https://sixsigmastudyguide.com/fault-tree-analysis/

Joseph Oncken, B. C. (2020). Real-TIme Model Predictive Powertrain Control for a Connected Plug-In Hybrid Electric Vehicle . *IEEE Transactions on Vehicular Technology* , Volume: 69, Issue: 8, pp 8420 - 8432.

Konstantopoulos, G. (n.d.). *Part II - Decision Systems for Engineering Design Laboratory A : Sampling Plans and Knowledge discovery* . Sheffield, UK .

Lotus. (2021). ATCx Driving Innovation Webinar . Sheffield, UK : ATCx .

M. Sami Fadali, A. V. (2017). Phase Margin . *Electric Motor Control* .

Makrygiorgou, J. J. (n.d.). Fuzzy logic control of electric vehicles: Design and analysis concepts . *Ecological Vehicles and Renewable Energies (EVER)*.

MATLAB . (n.d.). *glyphplot*. Retrieved from mathworks : https://www.mathworks.com/help/stats/glyphplot.html

Max D. Morris, T. J. (1995). Exploratory designs for coputational experiments . *Journal of Statistical Planning and Inference* , Volume 43, Issue 3, pp 381 - 402.

Nguyen Huy, D.-N. D. (2014). New Hybrid Between NSGA-III with multi-objective particle swarm optimization to multi-objective robust optimization design for Powertrain mount system of electric vehicles. *SAGE Journals* , Volume: 12 Issue: 2 .

Nise, N. (2012). *Control Systems Engineering* .

Northwestern University . (n.d.). *Statistics Toolbox* . Retrieved from northwestern.edu: http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/stats/gplotmatrix.html

Ogata, K. (1970). *Modern Control Engineering* .

Peri, D. (2008). *Global Optimization Algorithms in Multidisciplinary Design Optimization* .

Purshouse, R. (n.d.). *Decision System for Engineering Design* . Sheffield, UK .

Shahin Rostami, F. N. (2020). On Algorithm Descriptions and Software Implementation for Multi-Objective Optimization: A Comparative Study . *SN Computer Science* .

Stefano Di Cariano, D. Y. (n.d.). *Model Predictive Powertrain Control: An Application to Idle Speed Regulation* . Springer. Retrieved from https://link.springer.com/chapter/10.1007/978-1-84996-071-7_12

Sunith Bandaru, A. H. (2016). Data Mining methods for knowledge discovery in multi-objective optimization: Part A - Survey . *ELSEVIER* , 140-145.

Teytaud, O. (n.d.). How entropy-theorrems can show that approximating high-dim Pareto-fronts is too hard. *Bridging the Gap between Theory and Practice-Workshop PPSN-BTP*.

Ward, M. (2008). *Multivariate data glyphs: Principles and practice.* Worcester Polytechnic Institute, 100 Institute Rd., Worcester, MA, 01609 USA .

Zitzler, E. a. (2004). Indicator-Based Selection in Multiobjective Search . *Parallel Problem Solving from Nature, Springer* , 832 - 842.