H-infinity Optimal Actuator location for Generalized plants with Full Information

by

Purushottam Sinha

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Kirsten Morris

Waterloo, Ontario, Canada, 2017

I hereby declare that I am the sole author of this report. This is a true copy of the report, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

The main focus of this project is the algorithmic development of H_∞ optimal actuator location problem. Initially, the theoretical concepts useful in understanding the problem and algorithm development is discussed. The idea of H_∞ norm is introduced with the algorithms used to compute it for appropriate matrix functions. The concept of generalized plant , its notions of stability and the least attenuation problem is discussed. The key theorems in developing the algorithm for least attenuation are stated discussing their implication useful from the point of developing an algorithm.

A Pattern Search Optimization algorithm, combining features of Generalized Pattern Search(GPS) and Hooke and Jeeves Coordinate search(HJCS) is discussed in detail. The features of it different from Standard GPS has been highlighted. With this, MATLAB's built in routines $fminsearch$(Nelder Mead), $Patternsearch$(In GPS setting) and $fmincon$ have also been used to check their performance. The structure of the solver developed to implement all these algorithms together is presented. The example of 1-D heat equation has been used to implement the solver. The modeling of this problem in the form of generalized plants using FEM with hat functions basis is described. This example fits in the specific class of generalized plants for which the closed form result for least attenuation is given in [9]. This has been used to get the exact plots of variation of least attenuation with actuator location and check the performance of the solver. The contour plots have been presented with the quiver plot of trajectory of the Pattern Search Algorithm. The performance of optimization algorithms in different cases have been presented. The advantages of deploying some important theoretical results related to the attenuation problem in the Pattern Search Algorithm developed has been shown comparing its performance with MATLAB routines. Furthermore, a significant comparison has been made amongst the MATLAB subroutines.

Contents

I Motivation

In engineering applications, the controller design for a physical system is done according to the target objectives. A common controller design objective is to minimize the effect of unwanted external inputs to the output of a system which is of importance. A broad category of controller design area which deals with a similar objective is the H_∞ controller synthesis. In H_∞ control the target is to design a controller which minimizes the maximum gain possible due to unwanted external input to the concerned output. The ability of the controller to achieve this objective is highly dependent on the actuator location as will also be seen later. The optimal positioning of the actuator allows to attain the required objective with least amount of controlling effort spent, which is true in general and not just limited to this problem. This demands a way to estimate the best actuator location for a controlled system. This location is called H_∞ optimal actuator location. The method of solving this problem has been discussed in this report.

II H_∞ norm

H_∞ norm is a norm defined for matrix valued functions which are analytic and bounded in the right-half complex plane. Lets denote the set of all such matrix valued functions by MH_∞. For eg:

$$G(s) = \begin{bmatrix} \frac{1}{(s+1)^2+4} & \frac{1}{(s+2)^2+5} \\ \frac{1}{(s+3)^2+4} & \frac{1}{(s+4)^2+5} \end{bmatrix}$$

where $s \in \mathbb{C}$. Here $G(s) \in MH_\infty$

Definition II.1. *Consider G(s) $\in$ MH_∞ , the H_∞ norm for $G(s)$ is defined as*

$$||G||_\infty = \sup_\omega \max_i \sigma_i(G(j\omega))$$

where, $j = \sqrt{-1}$, $\sigma_i(G(j\omega))$ represents the singular value of the matrix $G(j\omega)$.

Consider a standard state space representation of a Linear Time Invariant (LTI) system

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Eu \end{aligned} \tag{1}$$

Here x denotes system states, u denotes the control input to the system and y denotes the observation or output of the system. On taking the Laplace transformation and eliminating the state variable $x(s)$, results in an equation relating u to y, which is

$$y(s) = (C(sI - A)^{-1}B + E)u(s)$$

The matrix function $G(s) = C(sI - A)^{-1}B + E$ is called the transfer function of the above LTI system which relates the input signal Laplace transform to the output signal Laplace transform as

$$y(s) = G(s)u(s)$$

The transfer functions $G(s) \in MH_\infty$ are called stable transfer function matrices. Here is a theorem which gives a very result about to the H_∞ norm of a stable transfer function.

Theorem II.1. *[1] Suppose the transfer function of (1) above $G(s) \in MH_\infty$, then the value*

$$||G||_\infty = \sup_{||u||_2=1} ||y||_2$$

$$where\ ||u||_2 = \sqrt{\int_0^\infty u(t)^T u(t)} = 1,\ ||v||_2 = \sqrt{\int_0^\infty v(t)^T v(t)}$$

The proof is skipped here. The idea which has to be extracted here is that the maximum gain possible in L_2 norm of the output signal is the H_∞ norm of the transfer function. The condition of G(s) being analytic and bounded should be noted for the above result.

There are several controller design problems which naturally simplifies down to a problem on H_∞ norm of a matrix function. Here is a simple instance below. Suppose there is a system with the state-space description

$$\dot{x} = Ax + Bu + B_1 d$$
$$y = Cx + Du$$

As in the state space description (1), x represents the state of the system, u represents control input and y represents the observation output. A new variable introduced here is d which denotes the disturbance input to the system. Now, suppose the task is to design a controller H with $u = Hx$ such that the peak gain in y due

to disturbance d (which can possibly be of different frequencies) is the least. On substituting $u = Hx$, and taking laplace transform of the system of equations, the transfer function equation is

$$y(s) = (C + DH)(sI - A - BH)^{-1}B_1 d(s)$$

The way to solve this problem is to find the controller H such that it minimizes the H_∞ norm of that transfer function.

An other common instance is the criteria of robust stability on the controller design. In this case a Plant model P (i.e with some state space description) is known with some uncertainty. The objective is to design a controller which unconditionally stabilizes the Plant with its uncertainty. This leads to a condition on the H_∞ norm of a particular matrix function depending on the kind of uncertainty in Plant P. Apart from this, some other criteria like reducing sensitivity of a system, or reducing sensitivity on a particular spectrum of input signal ultimately leads to a condition on the H_∞ norm of a matrix function.

Now the question arises, given a matrix function $G(s) \in MH_\infty$, how is its H_∞ norm calculated. One can see the difficulty involved in the problem as its the maximization of a singular value over the whole imaginary axis. There is no known closed form function to calculate H_∞ norm of a general transfer function. Hence, we rely on iterative methods to calculate this norm in general which also makes it computationally expensive. There is a result which proves to be the key for the iterative algorithms. The result is stated without proof here.

Theorem II.2. *[2] Consider the transfer function of the standard state space description of a control system*

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Eu \end{aligned} \tag{2}$$

which is $G(s) = C(sI - A)^{-1}B + E$. Now define the matrices $R = E^TE - \gamma^2 I, S = EE^T - \gamma^2 I$ and

$$Z_\gamma = \begin{bmatrix} A - BR^{-1}E^TC & -BR^{-1}B^T \\ \gamma^2 C^T S^{-1} C & -A^T + C^T E R^{-1} B^T \end{bmatrix}$$

Assume that A has no imaginary axis eigenvalues and $\gamma > ||E||$. Then γ is a singular value of $G(j\omega)$ if and only if $j\omega$ is an eigenvalue of Z_γ.

The above theorem is the key to calculating the H_∞ norm for valid matrices. Another important result which will not be rigorously proved here is that the function

$\sigma_{max}G(j\omega)$ is a continuous function of ω for $G(s) \in MH_\infty$. These two above results are pivots to developing the iterative algorithms. There are two iterative algorithms discussed below.

Bisection Algorithm

One inference which the Theorem $II.2$ has is that the problem of optimizing over the whole imaginary axis $j\omega$ can be reduced to an optimization problem in the finite range of real numbers (i.e the range in which the norm is lying). Suppose for a transfer function $G(s) \in MH_\infty$, there is a value $\gamma \in \mathbb{R}$ such that $\gamma_0 < \gamma < ||G||_\infty$, where $\gamma_0 = \sigma_{max}G(j\omega_0)$ for some $\omega_0 \in \mathbb{R}$, then

- if $\gamma < ||G||_\infty$ then the matrix Z_γ constructed above will have some purely imaginary eigenvalue.
- If $\gamma > ||G||_\infty$, then there will be no purely imaginary eigenvalue of Z_γ.

The bisection algorithm described below to calculate the $||G||_\infty$ norm is exactly based on this idea. Note that the bisection algorithm is not using the extra information which Theorem $II.2$ is giving which will be discussed in the next algorithm.

- Start with an upper bound $\gamma_{max} > ||G||_\infty$ and a lower bound $\gamma_{min} = \sigma_{max}G(j\omega_0)$ for some $\omega_0 \in \mathbb{R}$
- while $(\gamma_{max} - \gamma_{min}) \leq tol$
  - gamma = $(\gamma_{max} + \gamma_{min}\ )/2$
  - construct Z_γ and check if there is any imaginary eigenvalue of Z_γ
  - if there is any imaginary eigenvalue of Z_γ, then $\gamma_{min} = \gamma$, else $\gamma_{max} = \gamma$
- $H_\infty = (\gamma_{max} + \gamma_{min})/2$;

The tolerance value 'tol' would depend on the accuracy desired in the result. Here, in this algorithm the transfer function is not used but only the constituent matrices. There is an algorithm which makes the use of transfer function and the stronger statement of Theorem $II.2$

Transfer function based Algorithm

A strong statement made in *Theorem* $II.2$ is that if for the transfer function $G(s)$, $\sigma_i G(j\omega) = r$ for some i, then Z_r would be having an imaginary eigenvalue $j\omega$. This algorithm makes use of this strong statement.

- STEP 1: Choose a value ω_0 to start the algorithm
- STEP 2: Initialization $\gamma_{min} = max(\sigma_{max}G(0), \sigma_{max}G(j\omega_0), \sigma_{max}E)$
- STEP 3: Compute $Z_{\gamma_{min}}$ and its imaginary eigenvalues. Let the imaginary eigenvalues be $\{j\omega_1, j\omega_2, \ldots j\omega_n\}$ where $\{\omega_1, \omega_2 \ldots, \omega_n\}$ are in sorted order.
- STEP 4: Compute the midpoint $\{\omega_{12}, \omega_{23}, \ldots \omega_{n-1,n}\}$of each interval $(\omega_1, \omega_2), (\omega_2, \omega_3), \cdots, (\omega_{n-1}, \omega_n)$.
- STEP 5: Compute $\gamma_{min} = max(\sigma_{max}G(j\omega_{12}), \sigma_{max}G(j\omega_{23}), \cdots, \sigma_{max}G(j\omega_{n-1,n}), \gamma_{min})$
- STEP 6: Set $\gamma_{max} = \gamma_{min} + tolerance$
- STEP 7: Compute imaginary eigenvalues of $Z_{\gamma_{max}}$. If there are any compute all of them and start from step 4 with $\gamma_{min} = \gamma_{max}$. If there are no imaginary eigenvalues compute $\gamma_{ans} = \frac{\gamma_{max}+\gamma_{min}}{2}$

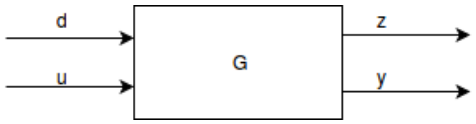
  return γ_{ans}

III Generalized Plants

The concept of **Generalized Plant** is an abstraction of a general system which rigorously characterizes the signals associated with it. Here is a brief description of the idea of Generalized plant.

1. Signal d is an uncontrolled signal input to the plant, for instance some known disturbance.
2. Signal u is a controlled signal input to the plant.
3. Signal z is the performance output which depends on the objective of the controller.
4. Signal y is the measured output fed to the controller

Here is an image of the open loop Generalized Plant indicating signal flows



Here is the state-space representation of a generalized plant.

$$\begin{aligned} \dot{x} &= Ax + B_1 d + B_2 u \\ z &= C_1 x + D_{11} d + D_{12} u \\ y &= C_2 x + D_{21} d + D_{22} u \end{aligned}$$

The x in the state space equation represents the internal states of the Plant. The concise way to denote this same state space representation is given below.

$$\begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{12} & D_{22} \end{bmatrix}$$

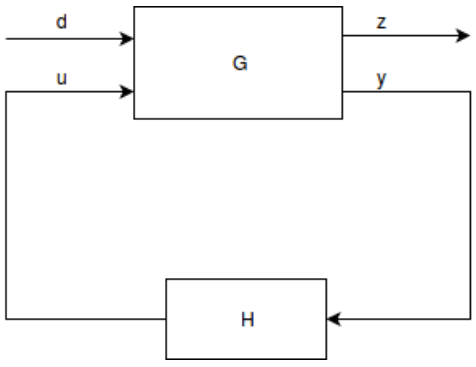
Here is the transfer function representation of the generalized plant.

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} d \\ u \end{bmatrix}$$

where G is the transfer function of the plant.
where every subsystem G_{ij} is denoted by (A, B_i, C_j, D_{ij}). The way to denote a generalized plant consistently used in this report is, Plant$(A, B_1, B_2, C_1, D_{11}, D_{12}, C_2, D_{12}, D_{22})$.
Now, the complete description of the general framework i.e generalized plant with the controller equation in the form of transfer function is

$$\begin{aligned} \begin{bmatrix} z \\ y \end{bmatrix} &= G \begin{bmatrix} d \\ u \end{bmatrix} \\ u &= Hy \end{aligned} \tag{3}$$

where H is the controller transfer function. Here is the closed loop diagram of the whole system.



Generalized plant with full information
A generalized plant with full information means the complete information about the known disturbance and the state is available to the controller. Mathematically, this uniquely determines the signal y being fed to the controller which is $\begin{bmatrix} x \\ d \end{bmatrix}$. Basically the matrices C_2, D_{21} and D_{22} are uniquely determined and these are given below.

$$C_2 = \begin{bmatrix} I \\ 0 \end{bmatrix}, D_{21} = \begin{bmatrix} 0 \\ I \end{bmatrix}, D_{22} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Now, introducing generality in plant description brings in some sophistication in controller design analysis. A definition and a result regarding the stability of generalized plant is stated here.

Definition III.1. *Lower Linear Fractional Transformation : Let G be the transfer function representation as given above. Let Δ be another matrix of appropriate dimension such that $lim_{s\to\infty} det(I - G_{22}(s)\Delta(s)) \neq 0$. Then, the Lower Linear Fractional Transformation is defined as*

$$F_L(G, \Delta) = G_{11} + G_{12}\Delta(I - G_{22}\Delta)^{-1}G_{21}$$

Here all the matrices G_{ij} are assumed to follow the required dimension criteria so that all the operations above are compatible.
Now in (3), the transfer function from the uncontrolled input **d** to the performance output **z** is

$$z = F_L(G, H)d$$

The basic requirement of a controller meant for any specific objective is that it should ultimately result in the plant being internally stable. Here is the result which states a useful property of the stabilizing controller and the existence condition of a stabilizing controller.

Definition III.2. *A pair of matrices $A(n \times n)$ and $B(n \times m)$ is stabilizable if there exists a matrix K such that $(A - BK)$ is Hurwitz i.e all the eigenvalues of (A-BK) is negative.*

Definition III.3. *A pair of matrices $A(n \times n)$ and $C(m \times n)$ is detectable if there exists a matrix F such that $(A - FC)$ is Hurwitz.*

Theorem III.1. *[3] A controller H internally stabilizes a generalized plant G if and only if it internally stabilizes $-G_{22}$ in standard negative feedback configuration. Furthermore, a stabilizing controller to G exists if and only if (A, B_2) is stabilizable and (A, C_2) is detectable.*

Implication: A useful insight which this result gives is that a lower order system can be analyzed to do the stability analysis of the generalized plant. The point useful for this report is the condition which the theorem gives for the existence of a stabilizing controller. One can appreciate some of the conditions given in the upcoming theorems knowing the above.

IV H_∞ attenuation

Suppose there is a generalized plant $\mathrm{P}(A, B_1, B_2, C_1, D_{11}, D_{12}, C_2, D_{12}, D_{22})$. Now, does a controller with transfer function H exist such that $||F_L(G, H)||_\infty < \gamma$? This is called H_∞ attenuation problem. If there does exist a controller H such that $||F_L(G, H)||_\infty < \gamma$, then the controller H is said to provide attenuation γ. An inherent complication in this problem is the unknown H, because of which the transfer function $||F_L(G, H)||_\infty$ is not known and hence the H_∞ norm cannot be trivially calculated or checked to be less than γ.

There are some key theorems on Algebraic Riccati Equations(AREs) and relating achievable attenuation γ in a generalized plant with the solution of an ARE constructed using γ and the constituent matrices of the plant. These theorems form the basis of developing the algorithm for checking the existence and even finding the controller providing attenuation γ. These theorems are stated without proofs with their utility in the algorithm explained. Firstly, there are some important assumptions about the constituent matrices which are stated below. The reasons behind these assumptions are not explained here and can be looked in [8]

Assumptions[7]

1. (A, B_2) is stabilizable and (A, C_2) is detectable.
2. (A, B_1) is stabilizable and (A, C_1) is detectable.
3. $D_{12}\begin{bmatrix} D_{12}^T & C_1 \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix}$
   $\begin{bmatrix} D_{21} & B_1 \end{bmatrix} D_{21}^T = \begin{bmatrix} I \\ 0 \end{bmatrix}$
4. $D_{11} = D_{22} = O$

Now, here are the Theorems.

This first theorem is a result on ARE.

Theorem IV.1. *[4] Consider the ARE with unknown P , where A, R and Q are known real square matrices with Q and R being symmetric*

$$A^T P + PA + PRP + Q = 0$$

Now, consider the Hamiltonian matrix

$$Z = \begin{bmatrix} A & R \\ -Q & -A^T \end{bmatrix}$$

Suppose $\begin{bmatrix} X \\ Y \end{bmatrix}$ *represents generalized eigenvectors of Z corresponding to the negative eigenvalues. If the above ARE has a stabilizing solution i.e there exists P which solves it and A+RP is Hurwitz, then the matrix Z has no purely imaginary eigenvalues and X is a full rank square matrix i.e*

$$Range \begin{bmatrix} X \\ Y \end{bmatrix} \cap \ Range \begin{bmatrix} 0 \\ I \end{bmatrix} = 0$$

and the solution P is a real symmetric matrix.

Implication: This theorem comes out to be useful in seeing the existence of a stabilizing solution of an ARE and indeed calculating it using the negative eigenspace of constructed Hamiltonian. The way this theorem helps in disturbance attenuation problem will be seen later.

Theorem IV.2. *[5]Assuming that* $D_{12}^T \begin{bmatrix} D_{12} & C_1 \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix}$ *and (A,* B_2*) is stabilizable and (A,* C_1*) are detectable, then if there exists a controller with transfer function H such that* $||F_L(G, H)||_\infty < \gamma$*, then there exists a static feedback controller K such that* $||F_L(G, K)||_\infty < \gamma$*.*

Implication: This non trivial result implies that if there is any controller which provides attenuation γ, then there would exist a constant state feedback controller which provides this attenuation. This not only shows that a very simple proportional control will get that attenuation but also that there is nothing related to disturbance which is needed to be included in the feedback.

Now comes the last and the main theorem which connects everything.

Theorem IV.3. *[6] For a given plant, assume that* $D_{12}^T \begin{bmatrix} D_{12} & C_1 \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix}$ *and (A, B_2) is stabilizable and (A, C_1) are detectable. There exists a stabilizing controller H for the full information problem with* $||F_L(G,H)||_\infty < \gamma$ *if and only if there exists a symmetric solution* $P \geq 0$ *of the ARE*

$$A^T P + PA + P(\frac{1}{\gamma^2} B_1 B_1' - B_2 B_2')P + C_1^T C_1 = 0$$

such that $A + (\frac{1}{\gamma^2} B_1 B_1^T - B_2 B_2^T)P$ is Hurwitz.

Implication: The very useful point which this theorem makes is the existence of a positive semi-definite stabilizing solution of the given ARE is necessary and sufficient to conclude that there is a controller which provides attenuation γ .

Interconnection: Here is a short description explaining their interconnection. Treat $R = (\frac{1}{\gamma^2} B_1 B_1' - B_2 B_2')$ and $Q = C_1^T C_1$ in Theorem *IV*.3. Now, Theorem *IV*.1 can be used with the ARE in Theorem *IV*.3 to check the existence and even finding the feedback stabilizing controller which can provide attenuation γ. Theorem *IV*.1 also gives the way to find the solution of the ARE. Although, very important to note, it doesn't say anything about the positive semi-definiteness of P so this will have to be checked separately. This does increase the computational expense as it requires the explicit calculation of the solution and then checking its positive semi-definiteness.

Here is the algorithmic description of the idea.

1. construct the matrices

$$Q = C_1 C_1^T$$
$$R = \frac{1}{\gamma^2} B_1 B_1' - B_2 B_2'$$

2. construct the Hamiltonian using the above matrices

$$Z = \begin{bmatrix} A & R \\ -Q & -A^T \end{bmatrix}$$

3. Check if any of the eigenvalues of Z is imaginary.

4. If there is an imaginary eigenvalue of Z, γ attenuation is **not possible** for the plant.

5. Else if there is no imaginary eigenvalue of Z, do the following.

   (a) find the matrix $\begin{bmatrix} X \\ Y \end{bmatrix}$ which is the negative eigenspace of Z with both X and Y to be square matrix.

   (b) If rank(X) = n,

   - find P = YX^{-1}.
   - If P is +ve semidefinite then γ attenuation is possible
     else its not possible

So its the very last step in the algorithm where an attenuation to be possible is clear. The algorithm for being able to check any attenuation comes out to be of huge importance in developing the algorithm for the main problem as will be seen later. Now, one can move towards solving the main problem

V The Main Problem

The H_∞ optimal actuator location problem mathematically is two optimization problems one over another. Though, understanding it physically is comparatively more intricate. The understanding of the idea of full information, performance output in the context of generalized plant presented in one of the sections above is assumed.

Problem Statement : There is a generalized plant with parametrized B_2 given as, Plant$(A, B_1, B_2(r), C_1, D_{11}, D_{12}, C_2, D_{12}, D_{22})$, with full information. There is some known performance output of the system. All the constituent matrices of generalized plants are constant (with all standard assumptions true) and known except B_2. B_2 is a function of r i.e the actuator location. The aim is to determine the actuator location for which the attenuation achievable is the least i.e H_∞ *optimal actuator location.*

Description: Given an actuator location r, the matrix B_2, denoted as B_{2_r} is known and hence the plant $P_r = Plant(A, B_1, B_{2_r}, C_1, D_{11}, D_{12}, C_2, D_{12}, D_{22})$ is known. From the H_∞ disturbance attenuation problem discussed above, the Plant P_r has some least disturbance attenuation say γ_r. Now, the location r^* for which γ_{r^*} is the least is the H_∞ *optimal actuator location* which has to be determined.

VI Solving The Problem

A clear complexity involved in solving this optimization problem is the unavailability of a general closed form function for the least attenuation of a generalized plant with everything known. Moreover, each function evaluation is computationally expensive. On the positive side, there are specific results for this problem which can be exploited to minimize the number of function evaluations required to solve the problem. This brings in the need of building optimization algorithm meant specifically for this problem. There is one algorithm which has been coded entirely for solving this problem. This algorithm is a mixture of Generalized Pattern Search (GPS) and Hooke and Jeeves coordinate search(HJCS)[10] and is described below. The other algorithms used are MATLAB's routines which are GPS, fminsearch(Nelder-Mead) and fmincon. The algorithms being implemented through MATLAB routines are not described here and can be seen in [12].

Patternsearch (a mix of GPS and HJCS) for Project
The direct search algorithms broadly consist of a combination of a local search and a step which ensures a faster coverage of the domain, "a domain coverer". There can be several variants of the local search and domain coverage step suitable for different problems. The Patternsearch algorithm coded for this project is using a combination of features of GPS and HJCS. Before going on with the description of this algorithm, here is a definition and a result which is important to understand a step in the algorithm.

Definition VI.1. *Positive Spanning Basis: A positive spanning basis* $\{e_1, e_2, e_3 \dots e_k\}$ *of* R^n *means for any* $x \in R^n$, $\exists\{c_i \geq 0 \ \forall \ i\}$ *such that*

$$x = \sum_{i=1}^{i=k} c_i e_i$$

Result: Let e_i represent ith column of the identity matrix of dimension n. It can be shown that the vectors

$$e_1, e_2, e_3, e_4, e_5....., e_n, e_{n+1} = -ones(n, 1)$$

form the positive spanning basis of R^n. Here is the proof.

Suppose there is a point in $R^n, x_0 = [a_1; a_2; a_3, a_4....., a_n]$. There are two cases possible

1. $\min_i a_i \geq 0$: Here trivially $x_0 = \sum_{i=1}^{i=n} a_i e_i + 0 e_{n+1}$.

2. $\min_i a_i = a_{i_0} < 0$: Let $d_i = a_i - a_{i_0}$. Note that $d_i \geq 0 \ \forall \ i$. It can be seen that

$$x_0 = \sum_{i=0}^{i=n} d_i a_i - a_{i_0} e_{n+1}$$

Hence proved. The notation B_0 is used to denote the above basis in this report.

Utility of the above result: Suppose there is some point x_0 in the domain of a given optimization problem (minimization). Assuming the gradient to be existing and its value at this step to be $-\nabla f(x_0)$. Now, suppose there is some positive spanning basis B_p of the domain. Its not difficult to see that there will exist atleast one basis vector $b_i \ \in \ B_p$ such that $b_i \cdot -\nabla f(x_0) \ > 0$. This ensures that there is atleast one direction in B_p in which the function is decreasing. This idea is exploited in the local or exploratory search part of the algorithm.

Algorithm Subparts

This whole algorithm is a combination of three functions. The complete flow of steps in the algorithm is described after the description of these functions.

1. **Initializer**: This function initializes the variable to start off the optimization. It initializes the location and a Plant object with all the matrices.

2. **Pattern Search Function**

   This function implements the core algorithm which is an iterative process until there is no further improvement which it can do. Here are the sub-parts of this function.

   (a) **Exploratory search or Polling** : This is the local search step of the algorithm. The method of local search used here is drawn from the GPS, wherein there is a positive spanning basis used to conduct the local search. A better point is looked for opportunistically around the current point with a given step size to save as many function evaluations as possible greedily. The set of directions chosen for search has been chosen to be the positive spanning basis B_0 specified above. The advantages of using this basis is that its having the least positive spanning basis size and its easy to implement. If there is no better point found then it simply returns the current point. The step size decreases if a better point is not found out with the current step size, until the minimum step size is reached, but note that the step size is reduced outside this function

in the implementation. The minimum step size is the acceptable tolerance in the optimal location. The initial step size has been chosen to be 5 times the minimum step size. The idea used is that initial step size shouldn't either be too big or too small. Although, there is a space of more rigorous choice of initial step here. Below is the algorithm

x_1 = exploratory search (x_0, B_0, step)
$x_1 = x_0$
for $b \in B_0$
    $x_{candidate} = x_0$ + step * b
    if $x_{candidate}$ in domain
        check if $x_{candidate}$ is better
        if yes, $x_1 = x_{candidate}$ and break ;
        if no, continue
return x_1

NOTE: The Theorem $IV.3$ gives the condition which can be used to check if a point is better or not. This is of huge importance as far as the performance of overall algorithm is considered. This helps in preventing the very expensive function evaluation at a point which is not better.

(b) **Patternmove** : This step is drawn from the idea of patternmove in HJCS. This step is done only after a successful exploratory search. The objective of this step is to get a bigger step in the direction in which function tends to be decreasing. The function has the information about the points before and after the successful exploratory search happened just before it. A step is taken directly in the the same direction and of the same step size to look for a better point. Further, very importantly, this step keeps continuing until it is getting successful. Its importance is explained after the Patternmove algorithm described below.

function x_2 = patternmove(x_0, x_1)
set $x_2 = x_1$
$x_{2_0} = x_1 + (x_1 - x_0)$
if x_{2_0} is in domain
    check if x_{2_0} is better
    if yes
        set $x_2 = x_{2_0}$

until step - minimum step > 0
do x_{2_1} = exploratory(x_{2_0}, B_0, step)
if x_{2_1} better than x_{2_0}
set $x_2 = x_{2_1}$ and break this loop
else step = step/2
return x_2

There are two important things about Patternmove here

i. First, an exploratory search is also and only conducted if the point x_{2_0} turns out be better than x_1. This is a bit different from patternmove of HJCS wherein the search is conducted around x_{2_0} irrespective of it being better or not.

ii. Second, one may mistakenly feel that terminating Patternmove right after checking the point x_{2_0} to be better or not and then conducting the exploratory search entirely outside this will be the same thing as being done above and apparently will also reduce complications in implementation. As mentioned and will be seen (when the entire algorithm will be described), this step is being conducted until it keeps getting successful. So conducting an exploratory search within it brings in the possibility of making even a bigger move in the next iteration of patternmove.

3. **Global Search** : This step is done after the *Pattern Search* function returns its optima. It picks prescribed number of random set of points in the domain and checks if the those points are better than than the optimal point found out by *Pattern Search*. If the *Global Search* is successful, the iterative process of *Pattern Search* and *Global Search* again starts from the new point obtained by *Global Search*. In general, this step is not expected to be of utility unless the pattern search algorithm ended up giving a bad minima i.e a huge portion of the domain having values lesser than the minima obtained through patternsearch function. The thing to note is that it just checks and doesn't compute the function values which keeps it from being a very computationally expensive step. Also, this function is used only after an optima given by Pattern Search.

NOTE: The whole algorithm together is being called as the **Pattern Search Algorithm** which is a combination of initializer, a **Pattern Search Function** and a Global search function.
Now, here is the sequence of steps or a pseudocode of the whole algorithm connecting all the pieces.

1. initialize the actuator location x and the Plant P with all the matrices given in the Problem and B_2 using the initialized actuator location x. Set a value for maximum number of function evaluations.

2. The $Patternsearch(x)$
   STEP 1 : set $x_0 = x$
   STEP 2 : x_1 = exploratory search(x_0, step)
   STEP 3 : if $x_1 = x_0$
   step = step/2
   if step $<$ *minimum step*
   terminate the *Pattern Search* and return x_1
   else if $x_1 \neq x_0$
   STEP 5 : do $x_2 = patternmove(x_0, x_1)$
   STEP 6 : if $x_2 \neq x_1$
   $x_0 = x_1$
   $x_1 = x_2$
   go to STEP 5
   else
   go to STEP 2

3. $x_2 = Globalsearch(x_1)$.
   if $x_2 \neq x_1$
   Go to (b) with $x = x_2$
   else
   terminate the whole algorithm and the optimal point is x_2

These all steps are done only till a certain number of function evaluations value set. $STEP$ 6 in Patternsearch function above is where one can see the continuous steps of patternmove if it is being successful and hence brings in the benefit of doing local search within.

Highlighting Differences from GPS/Direction Direct Search [11]
GPS is the pattern search algorithm which has a combination of Global Search and a Local search(Exploratory as said here or Polling) in each iteration. Though, there are some authors who say the Global search step to be optional, here it has been seen as a required step of GPS and comparisons have been made on this ground. The Pattern Search implemented in this project will be called as Project Pattern Search (PPS). The positive spanning basis is assumed to be minimal in both algorithms.

- In GPS, the iteration starts with a set of function evaluations, called "Global Search", which if successful directly starts the next iteration with the next

Global Search around the new point. If unsuccessful then there is a local search conducted around the current point. In PPS, the first step in each iteration is the local search around the current point.

- After a successful local search, GPS starts a new iteration doing another global search with the new point. PPS does a "Pattern move" after this with the idea of moving on in the same direction, the step of which can get bigger and bigger depending on the repated success of it.

The idea is to keep the number of function evaluations as less as possible with an efficient movement throughout the domain. Doing a Global Search, that too in each iteration, can possibly be very costly for an optimization problem which is so sensitive to the number of function evaluations. Meanwhile, the coverage of domain is ensured with the Pattern Move step which is likely to be efficient. The initial target was to implement both these algorithms project specifically but could not be done. MATLAB's built in routines have been used to implement some other pattern search and gradient based algorithms. Although, It would not be fair to compare their performance with PPS since it has been developed specifically for the project and has got some routines which are directly useful for this problem. But, one can gain some insight on the parameters crucial for good performance and also performance comparison can be made within these built-in routines.

The MATLAB algorithms implemented in this project are "fminsearch" (Nelder Mead), "Patternsearch" (implementing it in GPS specific way described above) and "fmincon". The performance of all these algorithms will be described in the upcoming sections. The implementation of all the above algorithms required a careful organization of routines. For instance, to be able to implement MATLAB algorithms, there has to be a function set up which on entering the location gives the least attenuation value. These features have to be added while maintaining the user friendly nature of the solver. This led to development of many subroutines which motivates a brief description of the structure of the whole solver.

Solver Description

1. **Mainfunction file, 'mainfunction.m'** : This is the head function which contains all the machinery inside to calculate the optimal actuator location for a given problem. The problem is specified to this function by passing all the generalized plant matrices except B_2, a function which calculates B_2 on passing a location variable and the domain which specifies the region in which actuator is going to lie.

2. **Plant object creator file, Plant.m**: This file defines the object Plant using all the required constituent matrices. There are three methods for this Plant object created in this file. The properties of this Plant object are all the constituent matrices $A, B_1, B_2, C_1, D_{11}, D_{12}, C_2, D_{12}$ and D_{22} .

   The methods are enumerated below with their functions.

   (a) constructor : This function constructs the object Plant on passing all the constituent matrices.

   (b) gammacheck: This function is used to check if an attenuation γ is achievable for the given Plant.

   (c) hinfcalculator: This function calculates the least attenuation for the given plant using the MATLAB's routine "hinfsyn".

   (d) myhinfcalculator: This function calculates the least attenuation based on the theorems in Section 3 and in general turns out to be pretty faster than using MATLAB's hinfsyn.

   Creating this separate object Plant helps in conveniently performing several steps within the optimization algorithm or other small routines and can infact be used independent of everything for doing small checks, for instance just checking if a given Plant has got a particular attenuation.

3. **Initializer file, 'intitializer.m'** This function initializes the actuator location and a Plant object with B_2 corresponding to the initialized location

   NOTE : The *domain* , B_2 $function$ and the initialized *Plant object* have been declared global to avoid the requirement of passing them and its usefulness will be clearer later on.

4. **Algortihm file, 'purusminsearch.m'** : This is the function file which contains the whole implementation of the Pattern Search optimization Algorithm described above. It returns the optimal actuator location and the least attenuation value for the given problem taking starting location as input. It requires the environment of the $mainfunction$ to run. This function file has been created separately accounting the possibility of adding some functionalities outside the

   (a) **Patternsearch function file, 'purunewpatternsearch.m' :** This function file implements the whole Pattern Search algorithm. It calls the exploratory search and patternmove functions iteratively. There are two sub-function files created for Patternsearch function

i. **Exploratory search function file, 'exploratorysearch.m'**

ii. **Patternmove function file, 'patternmove.m'**

(b) **Global Search function file, 'globalsearch.m'**

5. **MATLAB's algorithm compatible maker, 'hinfcalculator.m' :** This function is created just to be able to implement MATLAB's optimization algorithms. MATLAB's algorithms in general require a function to be optimized to be passed. Due to no closed form function for the current problem available it requires to set it up.

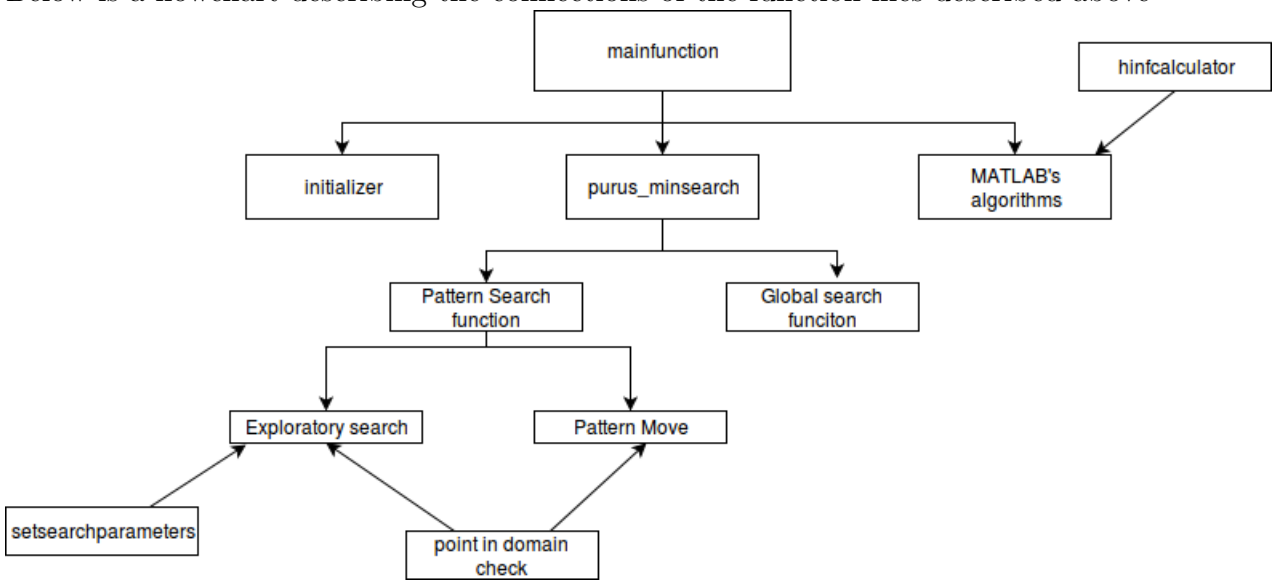
Additional small subroutine functions

1. **search parameters setter, 'setsearchparameters.m'** : This function sets the search parameters for the exploratory search. The parameters are the direction set, initial step and minimum step size values.

2. **point in domain checker, 'ispointindomain.m'** : This function checks if a location is within the domain.

3. **Quiver plotter, quiverplot.m : This plots the search trajectory the Pattern Search algorithm in 2D and 3D domain case**

Below is a flowchart describing the connections of the function files described above



VII Example: 1-D Heat Equation problem

The 1-D heat equation has been modeled in the form of generalized plant. The problem of H_∞ optimal actuator location is then solved for that plant model using the solver described above. The problem is solved for different possibilities of external parameters which here is number of actuators and the external disturbance mode. The comparison of Algorithms has then been made for different cases

Physical situation and Governing PDE
There is a very thin rod of unit length kept in the domain [0,1]. $T(x,t)$ represents the temperature of the rod at position x and time t. There is some external disturbance $d(x,t) = b_1(x)v(t)$ in the temperature which is spread throughout the rod which cannot be controlled. Note that the disturbance function taken here is such that the distribution of the disturbance in the domain is the same. There is an actuator(or may be multiple) which is being used to control the temperature. The form of the governing PDE for this problem is

$$\frac{\partial T}{\partial t} = \sigma \frac{\partial^2 T}{\partial x^2} + b_1(x)v(t) + bact_r(x)u(t);$$

where $bact_r(x)$ represents the influence of the actuator on the temperature when kept at a location r in the domain. $u(t)$ represents the control input by the actuator. Here in this example the actuator shape is assumed to be

$$bact_r(x) = \begin{cases} 0 & |x-r| > \epsilon \\ 1 & |x-r| < \epsilon \end{cases}$$

for some ϵ. The temperature at the ends are assumed to be fixed to zero i.e T(0, t) = T(1, t) = 0

Modelling the problem into State Space form
Finite Element Methods(FEM) is used to model the problem in the state space form. The basis used here for both solution function space and the test functions space are Hat functions. Let $\psi(x)$ be a test function. The variational representation of the PDE above is

$$\int_0^1 \frac{\partial T}{\partial t}\psi(x)dx = \int_0^1 (\sigma \frac{\partial^2 T}{\partial x^2}\psi(x) + b_1(x)v(t)\psi(x) + bact_r(x)u(t)\psi(x))dx$$

Using Dirichlet boundary condition i.e T(0) = T(1) = $\psi(0) = \psi(1) = 0$, the first term in the RHS can be simplified by doing integration by parts to get a simpler variational form

$$\int_0^1 \frac{\partial T}{\partial t}\psi(x)dx = -\int_0^1 \sigma\frac{\partial T}{\partial x}\psi'(x)dx + v(t)\int_0^1 b_1(x)\psi(x)dx + u(t)\int_0^1 bact_r(x)\psi(x)dx$$

The above identity has to be true for every test function and so for every basis function of test functions. These basis functions here are the hat functions which will be represented here as ϕ_i, where ϕ_i is

$$\phi_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & x_{i-1} \le x \le x_i \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & x_i \le x \le x_{i+1} \\ 0 & else \end{cases}$$

Replacing a general test function by the basis function ϕ_i in the variational form

$$\int_0^1 \frac{\partial T}{\partial t}\phi_i(x)dx = -\int_0^1 \sigma\frac{\partial T}{\partial x}\phi_i'(x)dx + v(t)\int_0^1 b_1(x)\phi_i(x)dx + u(t)\int_0^1 bact_r(x)\phi_i(x)dx$$

Now,

$$LetT = \sum_{i=1}^{i=n} c_i\phi_i$$

$$\implies \frac{\partial T}{\partial t} = \sum_{i=1}^{i=n} \dot{c_i}\phi_i \; and \; \frac{\partial T}{\partial x} = \sum_{i=1}^{i=n} c_i\phi_i'$$

Putting this back in the above variational representation

$$\int_0^1 (\sum_{i=1}^{i=n} \dot{c_i}\phi_i)\phi_j dx = -\sigma\int_0^1 (\sum_{i=1}^{i=n} c_i\phi_i')\phi_j' dx + v(t)\int_0^1 b_1(x)\phi_j dx + u(t)\int_0^1 bact_r(x)\phi_j dx$$

Now the above equation is true for all ϕ_j. So there are n equations and n variables. The system of equations is

$$M\dot{c} = -\sigma Sc + B_1 d + B_2 u \tag{4}$$

where

$$M = \begin{bmatrix} \int_0^1 \phi_1\phi_1 dx & \int_0^1 \phi_1\phi_2 dx & \int_0^1 \phi_1\phi_3 dx & \cdot & \cdot & \cdot & \int_0^1 \phi_1\phi_n dx \\ \int_0^1 \phi_2\phi_1 dx & \int_0^1 \phi_2\phi_2 dx & \int_0^1 \phi_2\phi_3 dx & \cdot & \cdot & \cdot & \int_0^1 \phi_2\phi_n dx \\ \int_0^1 \phi_3\phi_1 dx & \int_0^1 \phi_3\phi_2 dx & \cdot & \cdot & \cdot & \cdot & \int_0^1 \phi_3\phi_n dx \\ \int_0^1 \phi_4\phi_1 dx & \cdot & \cdot & \cdot & \cdot & \cdot & \int_0^1 \phi_4\phi_n dx \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \int_0^1 \phi_{n-1}\phi_1 dx & \cdot & \cdot & \cdot & \cdot & \cdot & \int_0^1 \phi_{n-1}\phi_n dx \\ \int_0^1 \phi_n\phi_1 dx & \int_0^1 \phi_n\phi_2 dx & \cdot & \cdot & \cdot & \cdot & \int_0^1 \phi_n\phi_n dx \end{bmatrix}$$

$$S = \begin{bmatrix} \int_0^1 \phi'_1\phi'_1 dx & \int_0^1 \phi'_1\phi'_2 dx & \int_0^1 \phi'_1\phi'_3 dx & \cdot & \cdot & \cdot & \int_0^1 \phi'_1\phi'_n dx \\ \int_0^1 \phi'_2\phi'_1 dx & \int_0^1 \phi'_2\phi'_2 dx & \int_0^1 \phi'_2\phi'_3 dx & \cdot & \cdot & \cdot & \int_0^1 \phi'_2\phi'_n dx \\ \int_0^1 \phi'_3\phi'_1 dx & \int_0^1 \phi'_3\phi'_2 dx & \cdot & \cdot & \cdot & \cdot & \int_0^1 \phi'_3\phi'_n dx \\ \int_0^1 \phi'_4\phi'_1 dx & \cdot & \cdot & \cdot & \cdot & \cdot & \int_0^1 \phi'_4\phi'_n dx \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \int_0^1 \phi'_{n-1}\phi'_1 dx & \cdot & \cdot & \cdot & \cdot & \cdot & \int_0^1 \phi'_{n-1}\phi'_n dx \\ \int_0^1 \phi'_n\phi'_1 dx & \int_0^1 \phi'_n\phi'_2 dx & \cdot & \cdot & \cdot & \cdot & \int_0^1 \phi'_n\phi'_n dx \end{bmatrix}$$

$$c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ \cdot \\ \cdot \\ c_{n-1} \\ c_n \end{bmatrix} \quad B_1 = \begin{bmatrix} \int_0^1 b_1\phi_1 \\ \int_0^1 b_1\phi_2 \\ \int_0^1 b_1\phi_3 \\ \int_0^1 b_1\phi_4 \\ \cdot \\ \cdot \\ \int_0^1 b_1\phi_{n-1} \\ \int_0^1 b_1\phi_n \end{bmatrix} \quad B_2 = \begin{bmatrix} \int_0^1 b_{act}\phi_1 \\ \int_0^1 b_{act}\phi_2 \\ \int_0^1 b_{act}\phi_3 \\ \int_0^1 b_{act}\phi_4 \\ \cdot \\ \cdot \\ \int_0^1 b_{act}\phi_{n-1} \\ \int_0^1 b_{act}\phi_n \end{bmatrix}$$

For the hat functions as the basis functions the Matrices M and S are

$$M = \frac{\Delta x}{6} \begin{bmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 & \dots \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 & \dots \\ \dots & & & & & & & \\ \dots & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & \dots & 1 & 4 \end{bmatrix}$$

$$S = -\frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & \dots & \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & \dots & \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & \dots & \\ \dots & & & & & & & & \\ \dots & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & \dots & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & \dots & \dots & -1 & 2 \end{bmatrix}$$

where Δx represents the finite element size.
Problem Statement: For the state space description of the 1-D rod heat equation given by (4), suppose the performance output z is $\begin{bmatrix} c \\ u \end{bmatrix}$. It is assumed that the full information about the disturbance v and states c is known. Further disturbance distrbution $b_1(x)$ i.e B_1 and the actuator function $bact_r(x)$ i.e $B_2(r)$ is given to us. The objective is to estimate the H_∞ optimal actuator location for the given performance output. The actuator function $bact_r(x)$ has been described in the problem introduction.

Solving the Problem:
First of all, the generalized plant representation of the system is

$$\dot{c} = -\sigma M^{-1} S c + M^{-1} B_1 d + M^{-1} B_2 u$$
$$z = \begin{bmatrix} I \\ 0 \end{bmatrix} c + \begin{bmatrix} 0 \\ I \end{bmatrix} u$$
$$y = \begin{bmatrix} I \\ 0 \end{bmatrix} c + \begin{bmatrix} 0 \\ I \end{bmatrix} d$$

The problem is set to be solved using the solver developed. There are different cases for which this problem has been solved. These cases differ in the disturbance function and the number of actuators used. The cases discussed here are

1. Using 1 actuator (B_2 with one column) with first mode disturbance i.e $b_1(x) = sin(\pi x)$, second mode disturbance i.e $b_1(x) = sin(2\pi x)$ and third mode disturbance i.e $b_1(x) = sin(3\pi x)$.

2. Using 2 actuators (B_2 with two columns) with first, second and third mode disturbance.

3. Using 3 actuators with first mode disturbance

Solver Results Verification

This beautiful paper [9] gives the expression for the closed form function of the least attenuation for a specific class of generalized plants. Our current problem of 1-D temperature flow fits into that specific class and hence the result can be used for verification of the result. Infact, this result not only has been used to verify the solver results but also to get the exact plots of the variation of the least attenuation with respect to actuator locations which helps in getting insight of the performance of the algorithms.

Parameters set for Algorithms

A set of parameters set for a particular algorithm has been used for all the test cases. The parameters have been set in a way which keeps the comparison on a suitable ground. The parameter settings are described below

1. Project Pattern Search
   Minimum step size = .01 (The size of the finite element)
   Initial step size = .05 (Chosen to be 5 times the minimum step size)
   Step size contraction = .5
   No step expansion

2. MATLAB's Patternsearch
   MATLAB's patternsearch in itself provides several parameters to be set. The parameters have been set in a way to replicate a standardized GPS discussed above.
   Mesh Size Tolerance = .01
   Mesh Contraction Factor = .5
   Initial Mesh ize = .05
   Polling Order = Consecutive (i.e the same order maintained irrespective of successful Poll search)
   Polling Method = n+1 basis
   Search Function = 'Latinhypercube'

3. MATLAB's fminsearch (Nelder-Mead)
   Setting parameters for MATLAB's Nelder-Mead proved to be a bit tricky. The important thing about MATLAB's Nelder Mead implementation is that it stops only when all the parameter values given are satisfied. A very low value of any of the function tolerance or variable tolerance can increase the number of function evaluations in a huge way. A high value of those parameters can lead to inaccurate results. The parameters set are
   Function Tolerance = 1

Simplex sides tolerance = .01

4. MATLAB's fmincon
   This was implemented to check the performance of a gradient based algorithm. The tuning of parameters for it was difficult. There are only two parameters which was set other than the default parameters
   Finite Difference Step Size = .01
   Minimum change in variables for finite difference = .01

IMPORTANT NOTE:
It will not be fair to compare the performance of the PPS algorithm with the MATLAB's routines and make some conclusion on this basis. This is because of the important functionality in PPS of being able to check the function value at a point to be less than a particular function value, through the results and algorithm discussed in section IV. This facility is not available to MATLAB's routines which would be computing the function value in its procedure. Although, one can compare the MATLAB's subroutines within themselves. Further, one can appreciate the importance of the number of function evaluations done. Also, the other purpose which it can serve is to see if a particular algorithm is being able to solve the problem. For instance, it would be interesting to see if $fmincon$ which is a gradient based algorithm is able to solve the problem. This is because firstly, there is no closed form of the function and further the problem of H_∞ optimal actuator location in itself doesn't have a result specifying the smoothness.

Results
Now, the results for each case has been described in detail. In each case the simulation has been done for four starting points. The performance parameters of each algorithm has been tabulated together. The exact plots(due to [9]) of least attenuation have been shown which can be used to verify the accuracy of the solver results. The optimal location rigorously searched through the domain (within the accuracy of .01) using the closed form has been mentioned. In case of 2-D(2 actuators) and 3-D(3-actuators) domain of optimization the quiver plot of the trajectory of Project Pattern Search Algorithm has also been shown. The quiver plots is helpful to appreciate some big systematic step sizes taken due to Patternmove step of PPS and also getting the insight of the work of the exploratory search algorithm.

1. Number of actuators = 1 , Disturbance mode = 1

   optimal location = .5

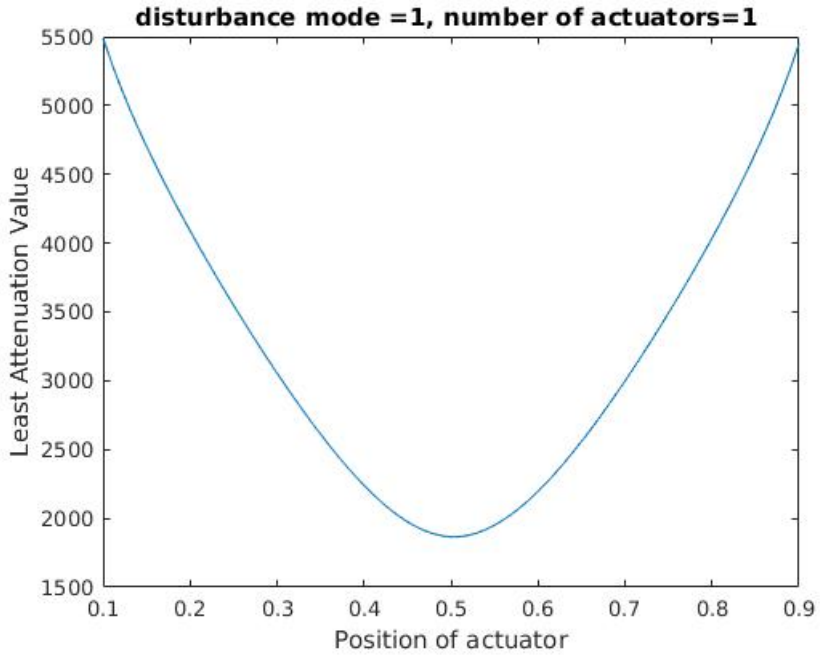


Fig: Variation of Least attenuation with actuator location

Performance of Algorithms- 1 Actuator with first mode disturbance

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.4898	1365.4595	11.0000	11.2332
MATLAB Patternsearch	0.5013	1368.2881	15.0000	18.0953
MATLAB Nelder-Mead	0.4959	1379.9252	18.0000	21.4779
MATLAB fmincon	0.4987	1367.8536	39.0000	47.4811

Table 1: Initial Point:0.177

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.4998	1359.7988	2.0000	3.2086
MATLAB Patternsearch	0.5098	1362.1513	21.0000	24.8932
MATLAB Nelder-Mead	0.5004	1361.3508	17.0000	18.4410
MATLAB fmincon	0.4810	1377.1824	43.0000	50.1881

Table 2: Initial Point:0.549

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.4809	1376.3604	10.0000	8.4628
MATLAB Patternsearch	0.4901	1362.7637	23.0000	27.0640
MATLAB Nelder-Mead	0.4998	1359.5667	26.0000	30.7616
MATLAB fmincon	0.4934	1379.0684	43.0000	51.3453

Table 3: Initial Point:0.964

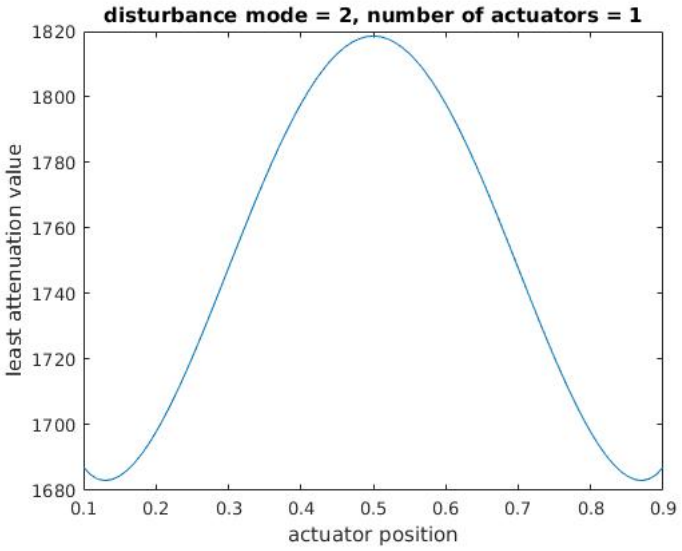
	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.5082	1372.8002	8.0000	5.7355
MATLAB Patternsearch	0.4892	1369.9327	15.0000	17.2546
MATLAB Nelder-Mead	0.5096	1363.4011	19.0000	21.2959
MATLAB fmincon	0.4955	1380.6158	60.0000	73.0040

Table 4: Initial Point:0.582

Brief Performance Summary: Accuracy of results have been consistent in all the algorithms. The Patternsearch function has consistently taken less time to solve due to significantly less number of function evaluations whereas the gradient based algorithm 'fmincon' took the highest time with very high number of function evaluations.

2. Number of actuators = 1 , Disturbance mode = 2

   optimal locations = .13, .88



Performance of Algorithms- 1 Actuator with 2nd mode disturbance

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.1297	1683.0480	6.0000	4.7538
MATLAB Patternsearch	0.8762	1682.4469	12.0000	13.6509
MATLAB Nelder-Mead	0.1326	1682.6651	14.0000	14.4552
MATLAB fmincon	0.1238	1682.4326	86.0000	98.5994

Table 5: Initial Point:0.241

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.1338	1682.5165	9.0000	6.3860
MATLAB Patternsearch	0.1268	1682.9982	17.0000	17.7900
MATLAB Nelder-Mead	0.1280	1683.0731	16.0000	16.5969
MATLAB fmincon	0.1238	1682.4326	76.0000	81.0176

Table 6: Initial Point:0.0434

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.1263	1682.9358	8.0000	7.1743
MATLAB Patternsearch	0.8663	1682.4878	11.0000	11.5983
MATLAB Nelder-Mead	0.1255	1682.8187	16.0000	15.9917
MATLAB fmincon	0.1238	1682.4326	100.0000	107.8479

Table 7: Initial Point:0.386

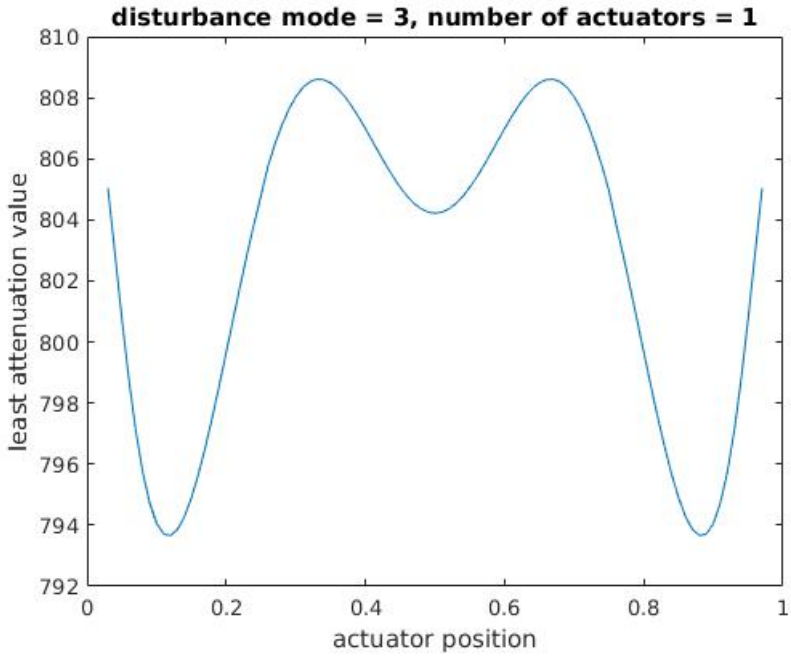
	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.1278	1683.0649	6.0000	5.5042
MATLAB Patternsearch	0.1267	1682.9798	22.0000	25.6240
MATLAB Nelder-Mead	0.1222	1682.9831	14.0000	15.2794
MATLAB fmincon	0.1238	1682.4326	71.0000	80.0356

Table 8: Initial Point:0.326

Performance Summary : The results obtained here almost follows the same trend as in the previous one. The Nelder Mead and Pattern Search almost performing similar here as well like the previous case .

3. Number of actuators : 1 Disturbance mode = 3

   Optimal location = (.12, .88), γ_{min} = 793.6.



Performance of Algorithms- 1 Actuator with 3rd mode disturbance

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.1212	793.5740	3.0000	2.8529
MATLAB Patternsearch	0.8993	793.9823	15.0000	15.8578
MATLAB Nelder-Mead	0.1013	793.9046	5.0000	5.1260
MATLAB fmincon	0.1163	793.5390	107.0000	124.9265

Table 9: Initial Point:0.0965

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.8895	793.6746	4.0000	3.6965
MATLAB Patternsearch	0.1181	793.6717	16.0000	17.8914
MATLAB Nelder-Mead	0.8832	793.5821	14.0000	16.7811
MATLAB fmincon	0.8886	793.5776	104.0000	124.7356

Table 10: Initial Point:0.939

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.1383	794.2500	9.0000	6.5887
MATLAB Patternsearch	0.8898	793.7024	14.0000	15.7371
MATLAB Nelder-Mead	**0.5069**	804.1252	6.0000	6.5307
MATLAB fmincon	0.8787	793.5663	106.0000	129.3479

Table 11: Initial Point:0.489

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	0.1210	793.5892	15.0000	13.0014
MATLAB Patternsearch	0.1278	793.8248	10.0000	11.0795
MATLAB Nelder-Mead	**0.4974**	804.1112	10.0000	12.4680
MATLAB fmincon	**0.4955**	804.1231	37.0000	41.0730

Table 12: Initial Point:0.577

Performance Summary : This case is a bit tricky due to multiple unequal local minimas. The wrong local minima has been highligted in the table. The Nelder-Mead and fmincon gave the wrong result on two occasions and 1 occasion respectively. Also, this time the MATLAB's patternsearch on average clearly took more time but with accurate results in each case compared to Nelder Mead. The other performance parameters nearly followed the same trends.

4. Disturbance mode : 1 , Number of actuators = 2

   Now, bringing in two actuators makes it a 2-D optimization problem. Local Minima: (.36, .64) and (.64, .36).

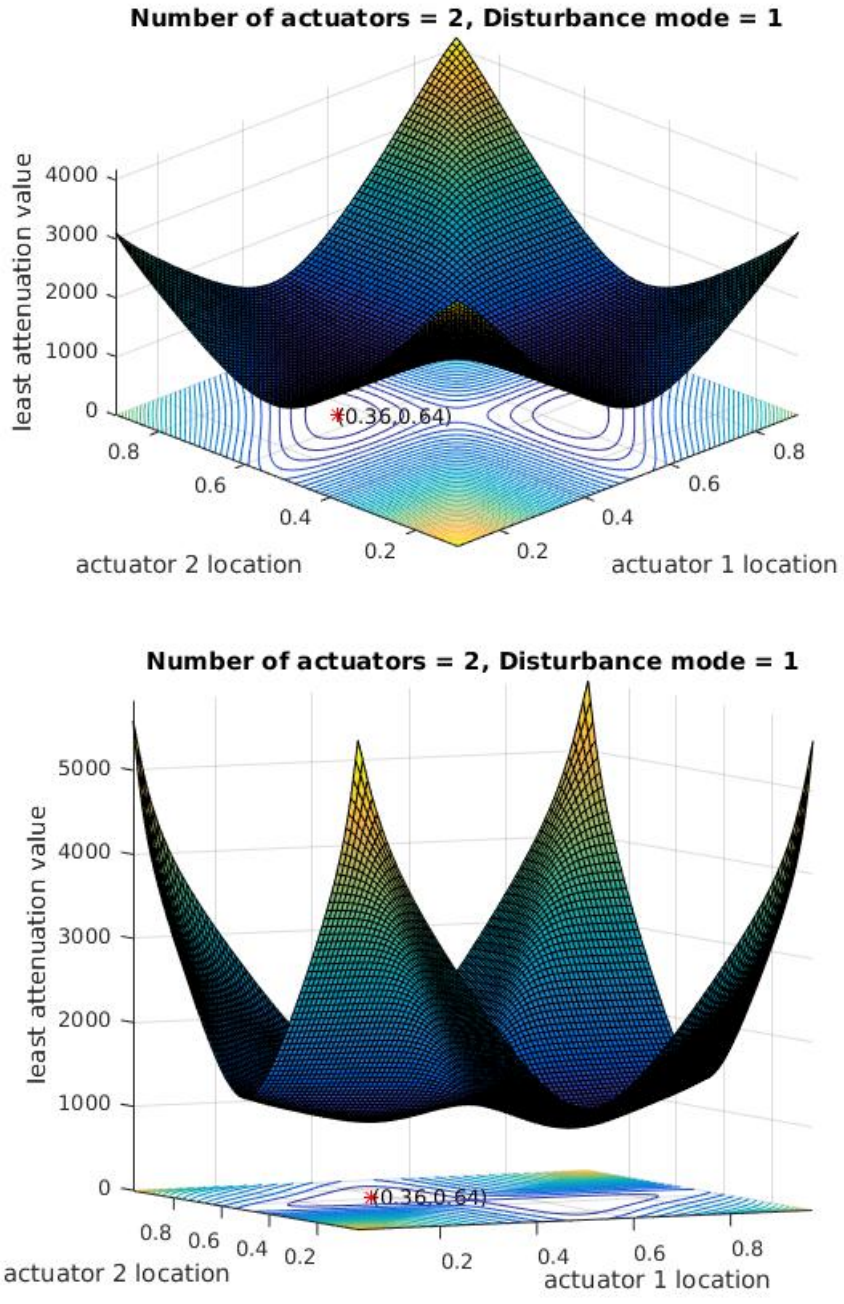


Fig : plot of the actual variation with the contour plot below

Performance of Algorithms- 2 Actuators with first mode disturbance

Algorithm	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.6396, 0.3523)	905.8859	9.0000	8.8396
MATLAB Patternsearch	(0.3531, 0.6373)	908.8079	38.0000	41.8543
MATLAB Nelder-Mead	(0.5399, 0.1164)	1185.1262	32.0000	35.9109
MATLAB fmincon	(0.6287, 0.3570)	908.3191	123.0000	133.6971

Table 13: Initial Point : (.63, .097)

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.3509, 0.6288)	901.8534	10.0000	10.3693
MATLAB Patternsearch	(0.6566, 0.3819)	911.8253	42.0000	42.2061
MATLAB Nelder-Mead	(0.3604 , 0.6282)	904.7453	53.0000	51.6024
MATLAB fmincon	(0.3703, 0.6391)	903.9698	62.0000	62.0078

Table 14: Initial Point:(0.666 0.894)

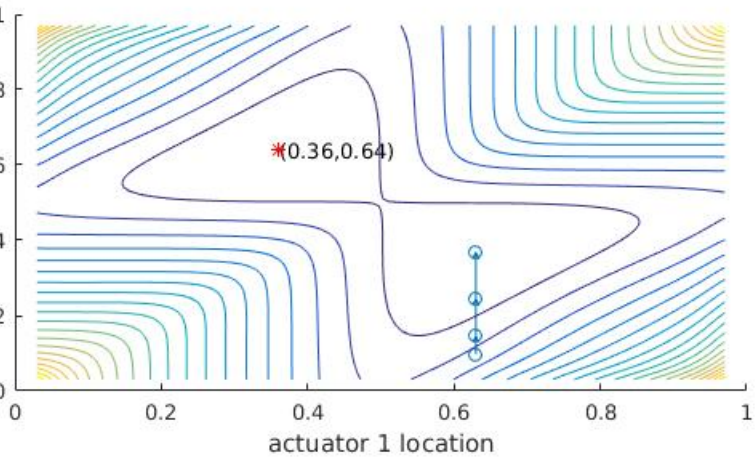
Algorithm	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.6100, 0.3317)	914.0815	10.0000	6.4921
MATLAB Patternsearch	(0.6376, 0.3517)	903.2342	42.0000	42.6662
MATLAB Nelder-Mead	(0.6285, 0.3615)	900.7136	65.0000	71.5872
MATLAB fmincon	(0.3515, 0.6287)	899.4619	138.0000	161.5411

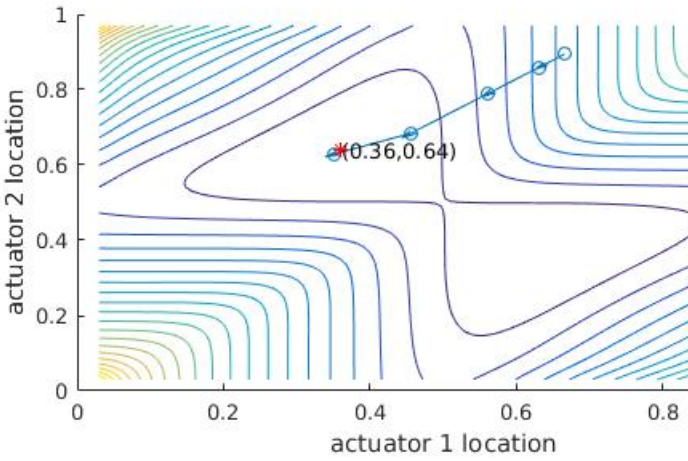
Table 15: Initial Point: (.805, .576)

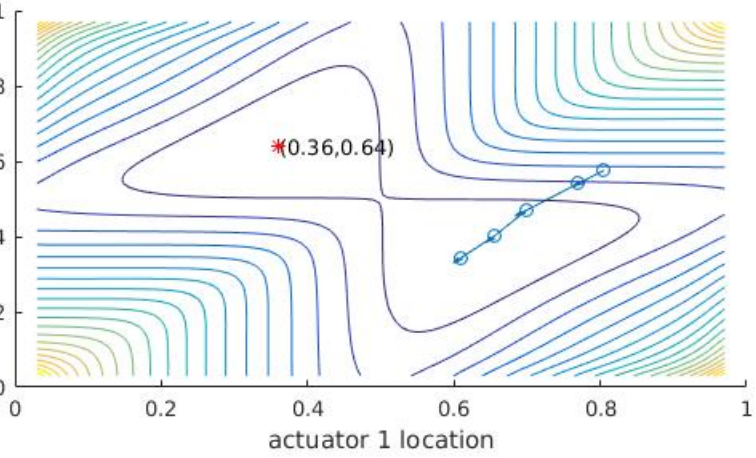
Algorithm	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.3515, 0.6286)	899.9824	12.0000	8.1304
MATLAB Patternsearch	(0.3694, 0.6305)	911.7375	42.0000	48.7847
MATLAB Nelder-Mead	(0.3615, 0.6389)	899.1703	67.0000	69.9822
MATLAB fmincon	(0.6287, 0.3515)	899.3980	157.0000	171.9311

Table 16: Initial Point = (.518,.943)

Performance Summary : The MATLAB's patternsearch clearly is better in performance compared to Nelder-Mead in this case. The increase in dimension is leading to more function evaluations in Nelder Mead and hence making it slow. Also the performance of fmincon has improved in terms of the ratio of time taken with other patternsearch methods compared to the 1-dimensional case.







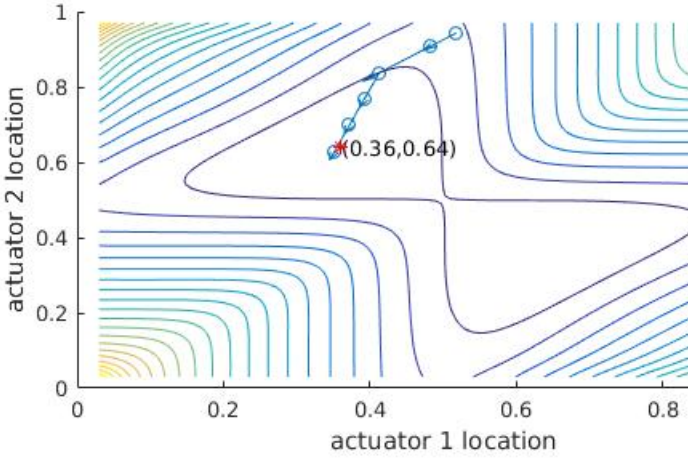


Fig: Contour Plot and PPS Quiver Plot , disturbance mode = 1

This gives a good idea of how the search algorithm is behaving in terms of efficiency.

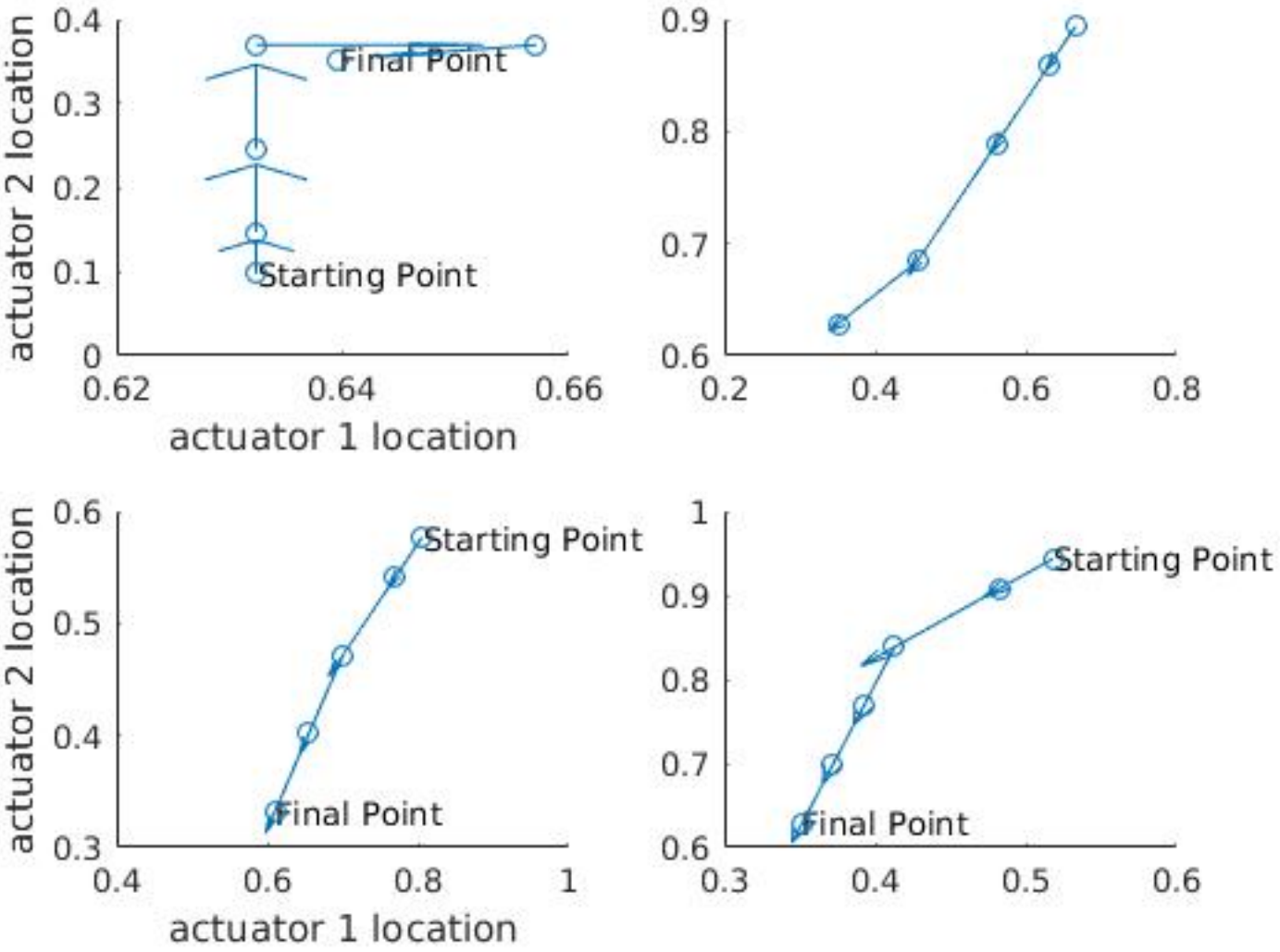
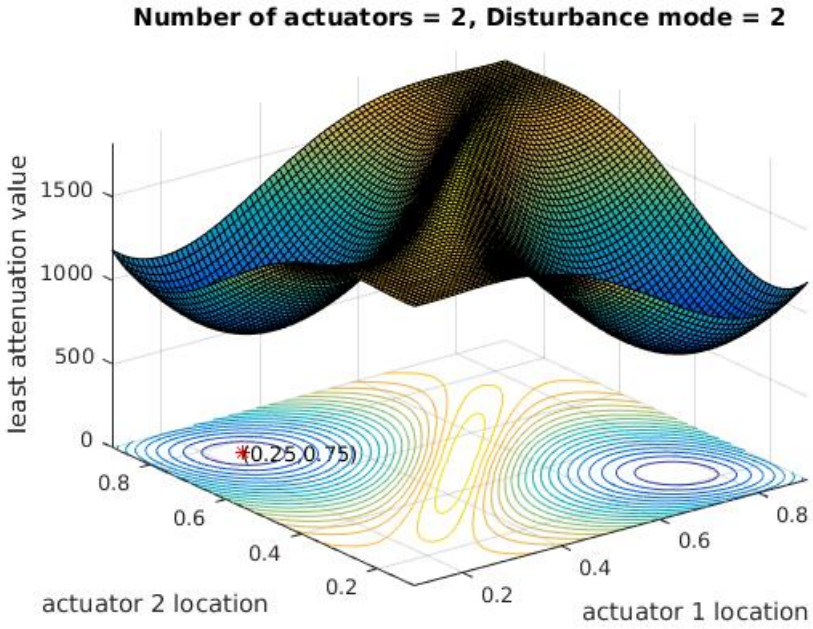
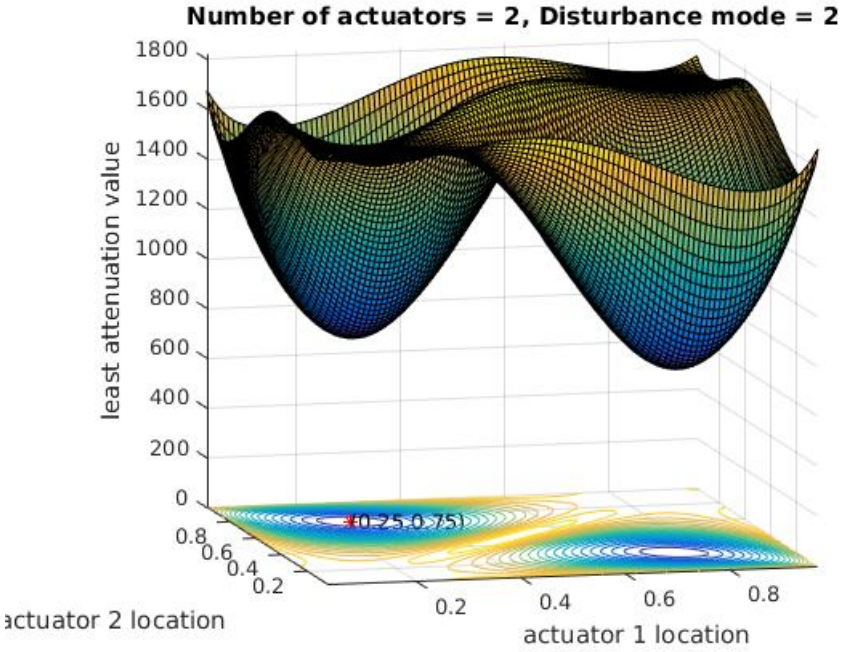


Fig: Corresponding PPS Quiver Plots

The quiver plot here helpful to see the variation in the step sizes also how efficient the search has been.

5. Disturbance mode = 2 , number of actuators = 2
   Optimal locations = (.25;.75) and (.75;.25).





Performance of Algorithms- 2 Actuator with 2nd mode disturbance

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.7569, 0.2340)	733.8441	16.0000	12.7035
MATLAB Patternsearch	(0.2325, 0.7580)	732.3525	57.0000	57.1768
MATLAB Nelder-Mead	(0.7513, 0.2425)	733.3779	50.0000	51.0978
MATLAB fmincon	(0.7475, 0.2457)	732.5882	70.0000	82.8462

Table 17: Initial Point 1:[0.728 0.576]

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.7579, 0.2435)	730.3839	14.0000	13.1648
MATLAB Patternsearch	(0.2418, 0.7664)	733.8085	38.0000	43.2345
MATLAB Nelder-Mead	(0.7381, 0.2532)	730.6285	48.0000	63.0703
MATLAB fmincon	(0.7547, 0.2515)	734.4159	101.0000	117.4730

Table 18: Initial Point 2:[0.734 0.430]

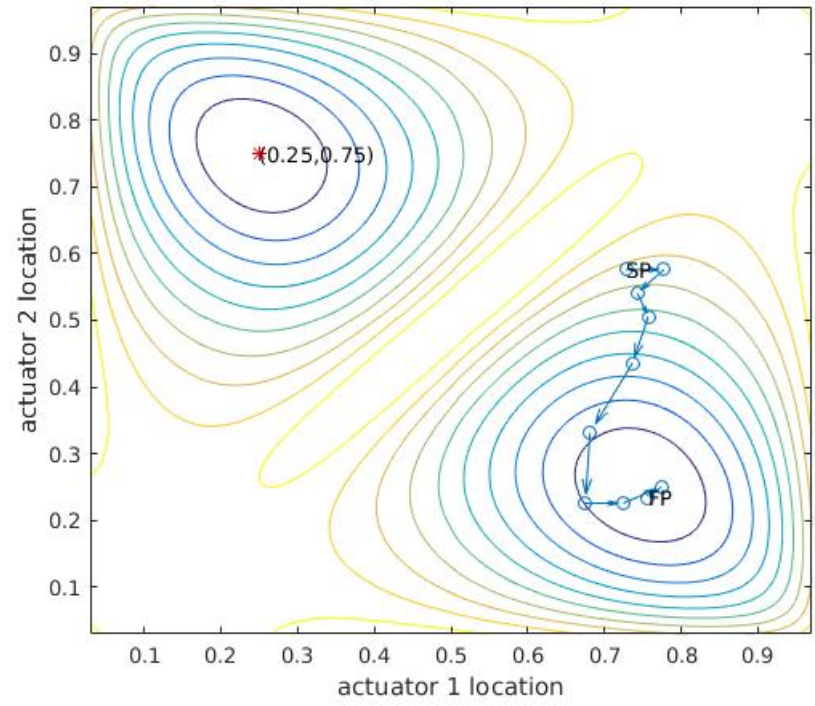
	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.2331, 0.7491)	734.4014	14.0000	14.1724
MATLAB Patternsearch	(0.2521, 0.7466)	728.9361	50.0000	55.5868
MATLAB Nelder-Mead	(0.2425, 0.7564)	729.2743	45.0000	52.5779
MATLAB fmincon	(0.2456, 0.7475)	732.1549	114.0000	126.2268

Table 19: Initial Point 3:[0.403 0.548]

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.7771 , 0.2329)	737.8534	17.0000	15.6100
MATLAB Patternsearch	(0.2513 , 0.7533)	736.1637	42.0000	46.9707
MATLAB Nelder-Mead	(0.7577 , 0.2427)	727.9388	57.0000	68.6455
MATLAB fmincon	(0.7420 ,0.2518)	735.6211	64.0000	72.1520

Table 20: Initial Point 4:[0.442 0.393]

Performance Summary : The most important thing to notice here is the significant improvement on average in the performance of fmincon. This can possibly be because of more elliptical nature of contours in a bigger range here compared to the previous example. Due to the same reason the relative performance of Nelder Mead as compared to MATLAB's Patternsearch has also improved from previous case since it inculcates the idea of gradient algorithm better in pattern search form.



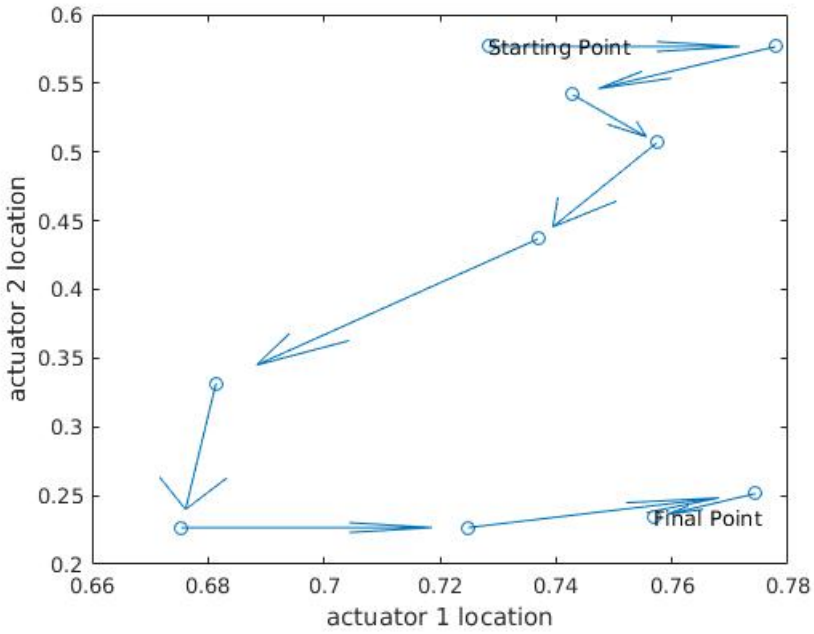


fig: Quiver Plot, Initial Point 1

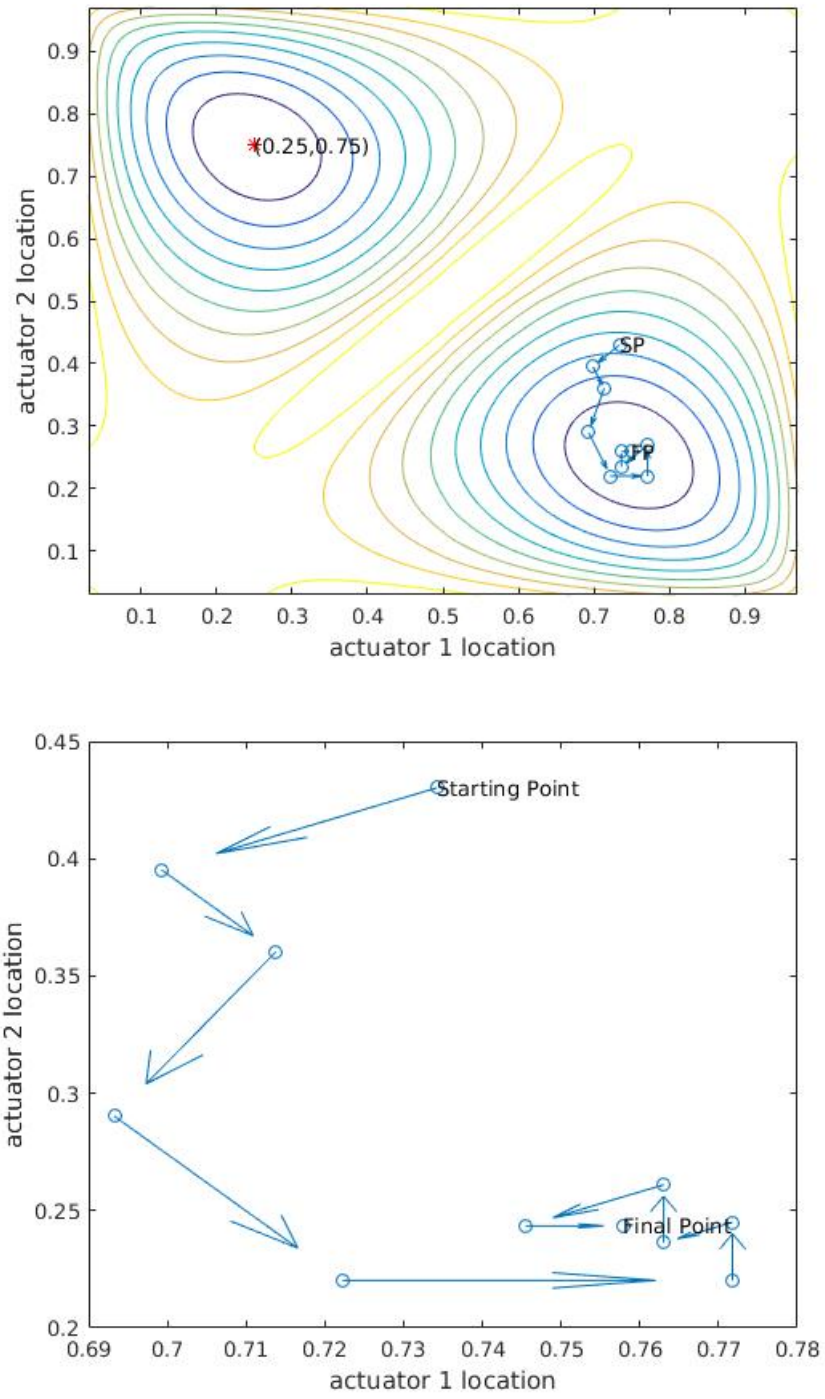


fig: Quiver Plot, Initial Point 2

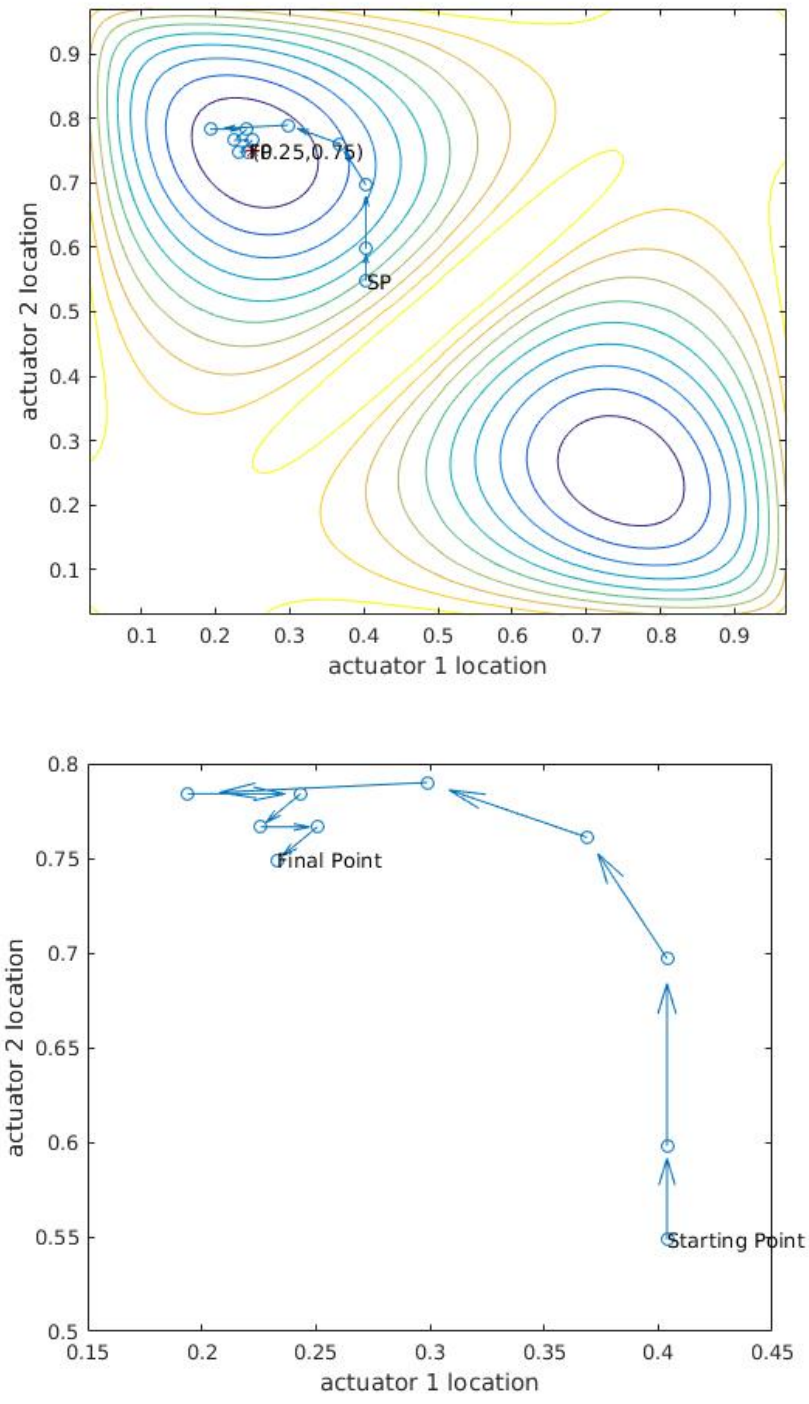


fig: Quiver Plot, Initial Point 3

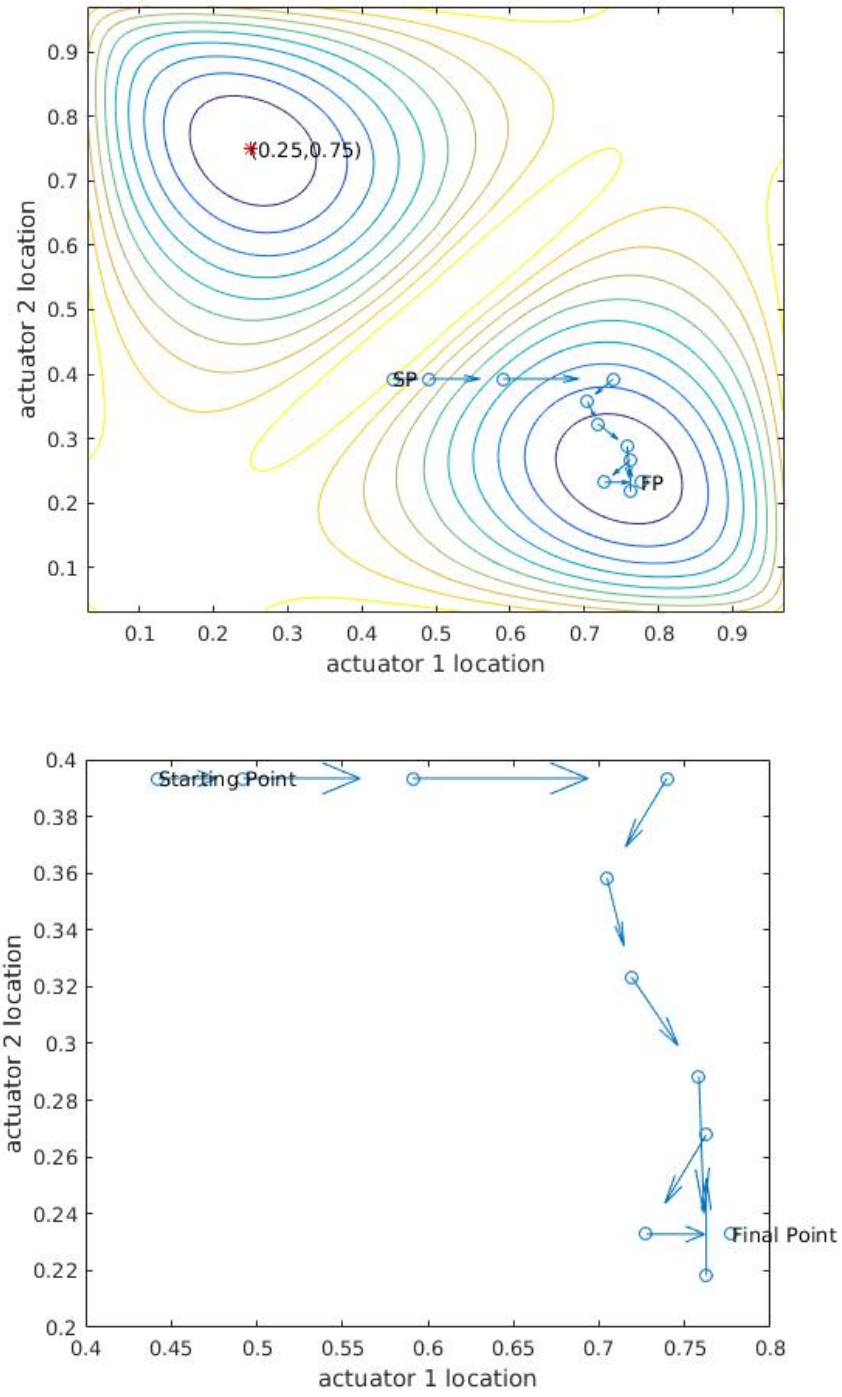
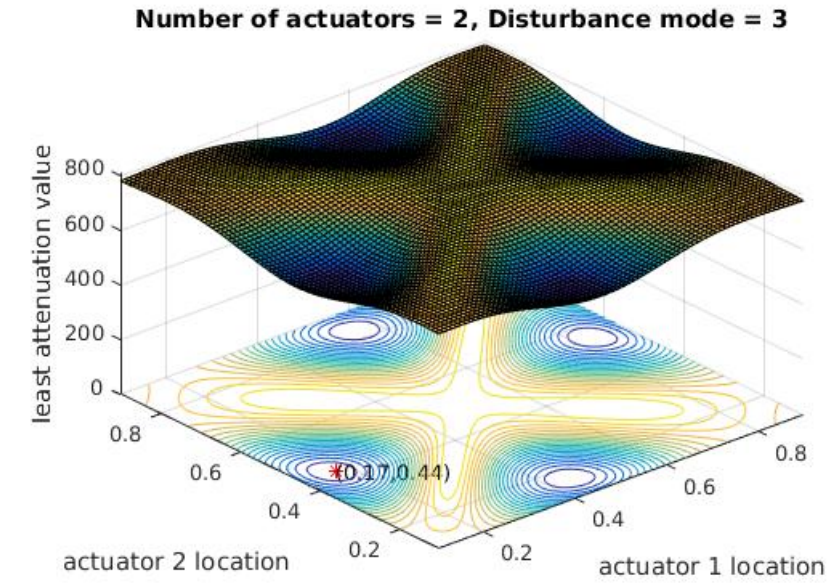
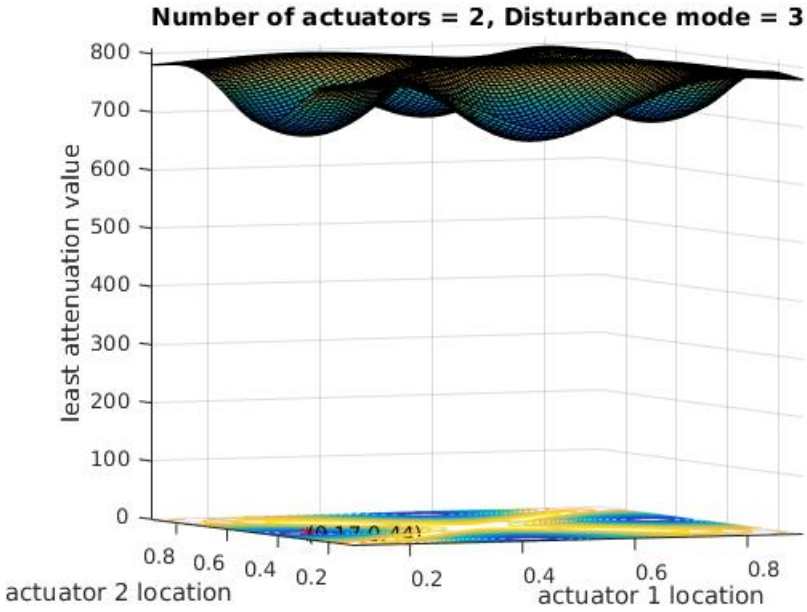


fig: Quiver Plot, Initial Point 4

6. disturbance mode = 3 , number of actuators = 2





Performance of Algorithms- 2 Actuator with 3rd mode disturbance

Algorithms	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.8391 , 0.5532)	683.5120	9.0000	7.9044
MATLAB Patternsearch	(0.1695 , 0.4462)	683.8662	54.0000	59.4004
MATLAB Nelder-Mead	(0.8348 , 0.5601)	682.7842	31.0000	38.1222
MATLAB fmincon	(0.8359 , 0.5498)	682.3078	106.0000	117.4604

Table 21: Initial Point 1:[0.627 0.539]

Algorithms	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.5577 , 0.8307)	683.6276	10.0000	8.5432
MATLAB Patternsearch	(0.4423 , 0.1638)	682.1351	42.0000	47.4825
MATLAB Nelder-Mead	(0.5498 , 0.8265)	682.4701	27.0000	26.3630
MATLAB fmincon	(0.5562 , 0.8294)	683.5206	88.0000	90.3925

Table 22: Initial Point 2:[0.533 0.955]

Algorithms	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.4510, 0.1638)	682.3118	14.0000	12.6227
MATLAB Patternsearch	(0.1546, 0.4373)	683.3469	30.0000	32.0690
MATLAB Nelder-Mead	(0.4510, 0.1628)	682.4018	56.0000	59.8260
MATLAB fmincon	(0.1545, 0.4413)	682.8396	79.0000	88.9140

Table 23: Initial Point 3:[0.741 0.0704]

Algorithms	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.5534, 0.8386)	683.2862	10.0000	9.5185
MATLAB Patternsearch	(0.5513, 0.8266)	682.5427	38.0000	42.0402
MATLAB Nelder-Mead	(0.5575, 0.8385)	683.0286	23.0000	24.0477
MATLAB fmincon	(0.5561, 0.8359)	682.5084	81.0000	92.5809

Table 24: Initial Point 4:[0.49 0.973]

Performance Summary : The performance of fmincon and Nelder-Mead showed even more relative improvement. This can be because of more number of local minimas increasing its vicinity to a random point with the elliptic contours. The accuracy was not affected since all the minimas are equal.

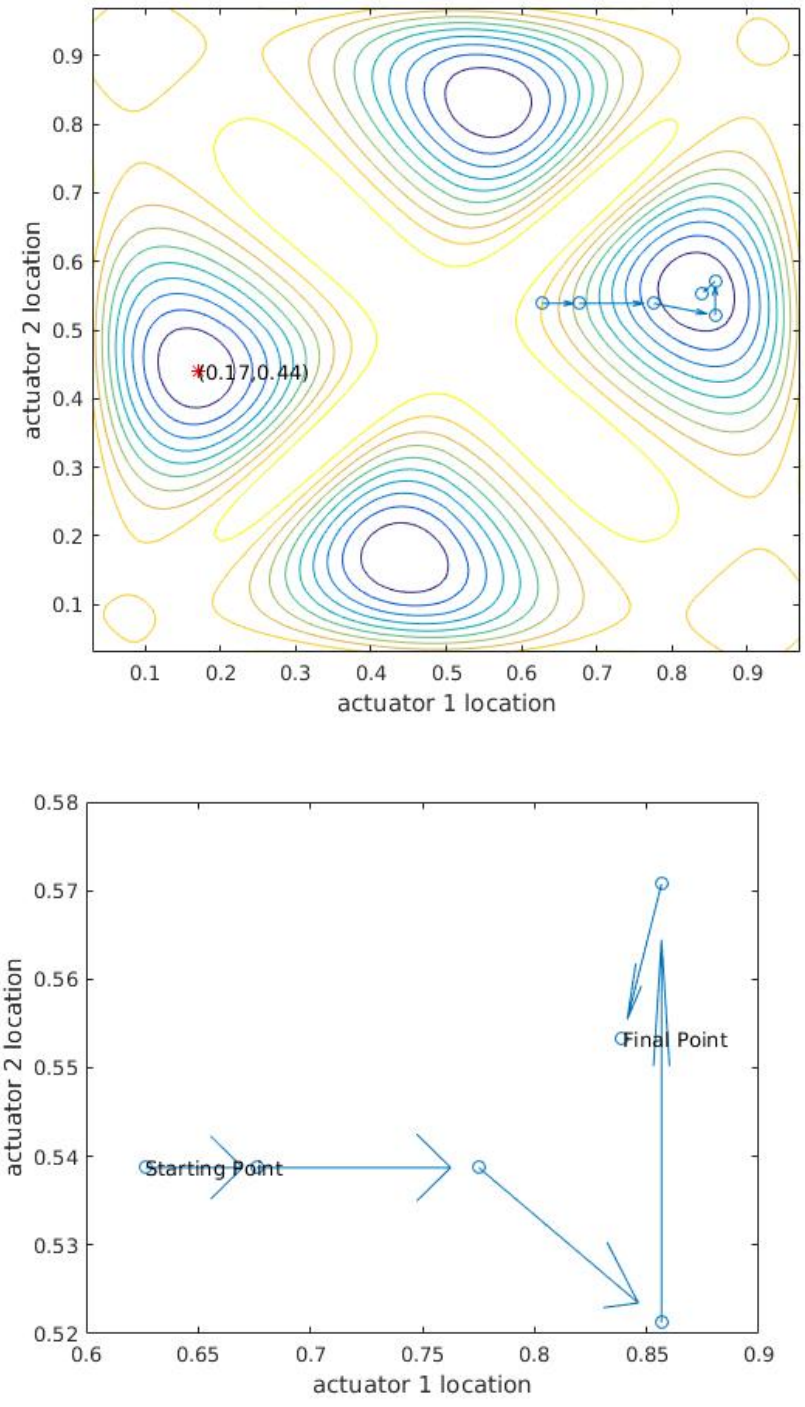


fig: Quiver Plot, Initial Point 1

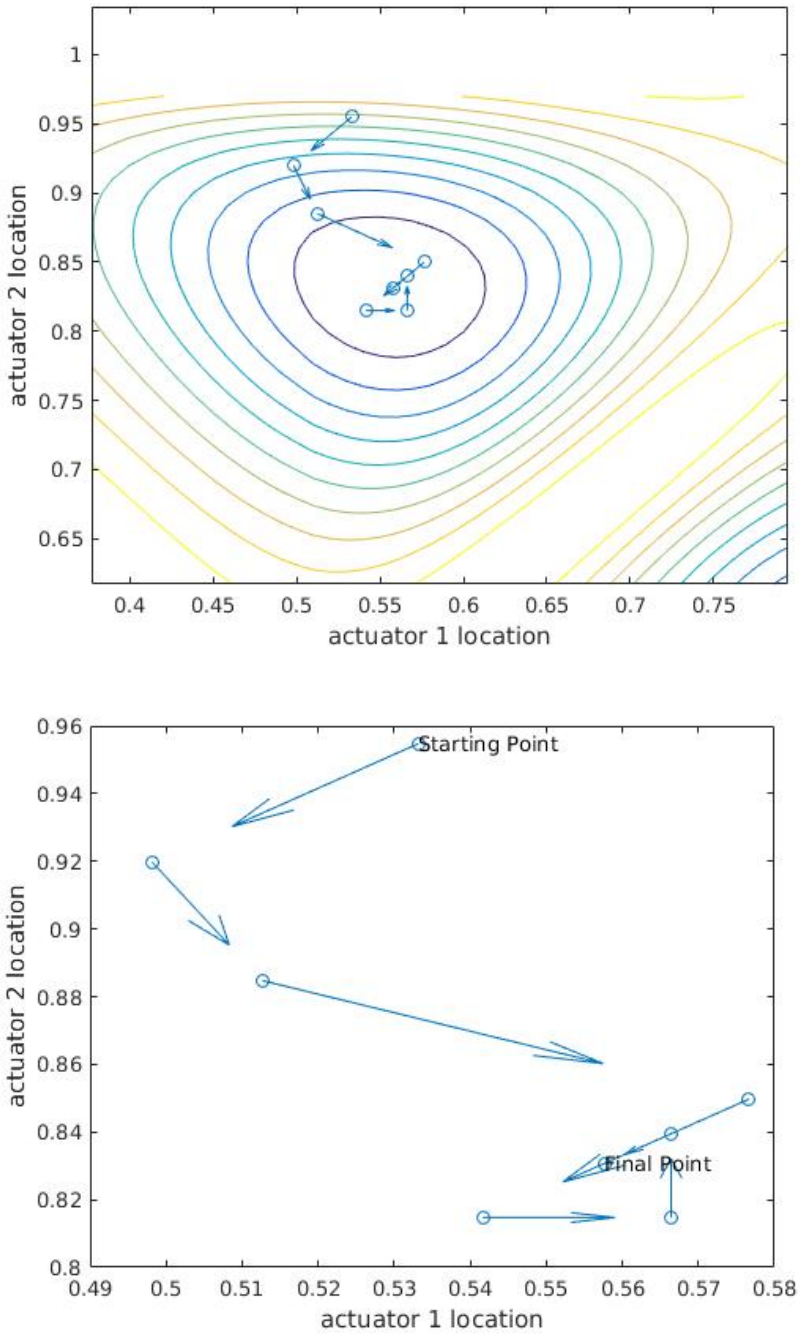


fig: Quiver Plot, Initial Point 2

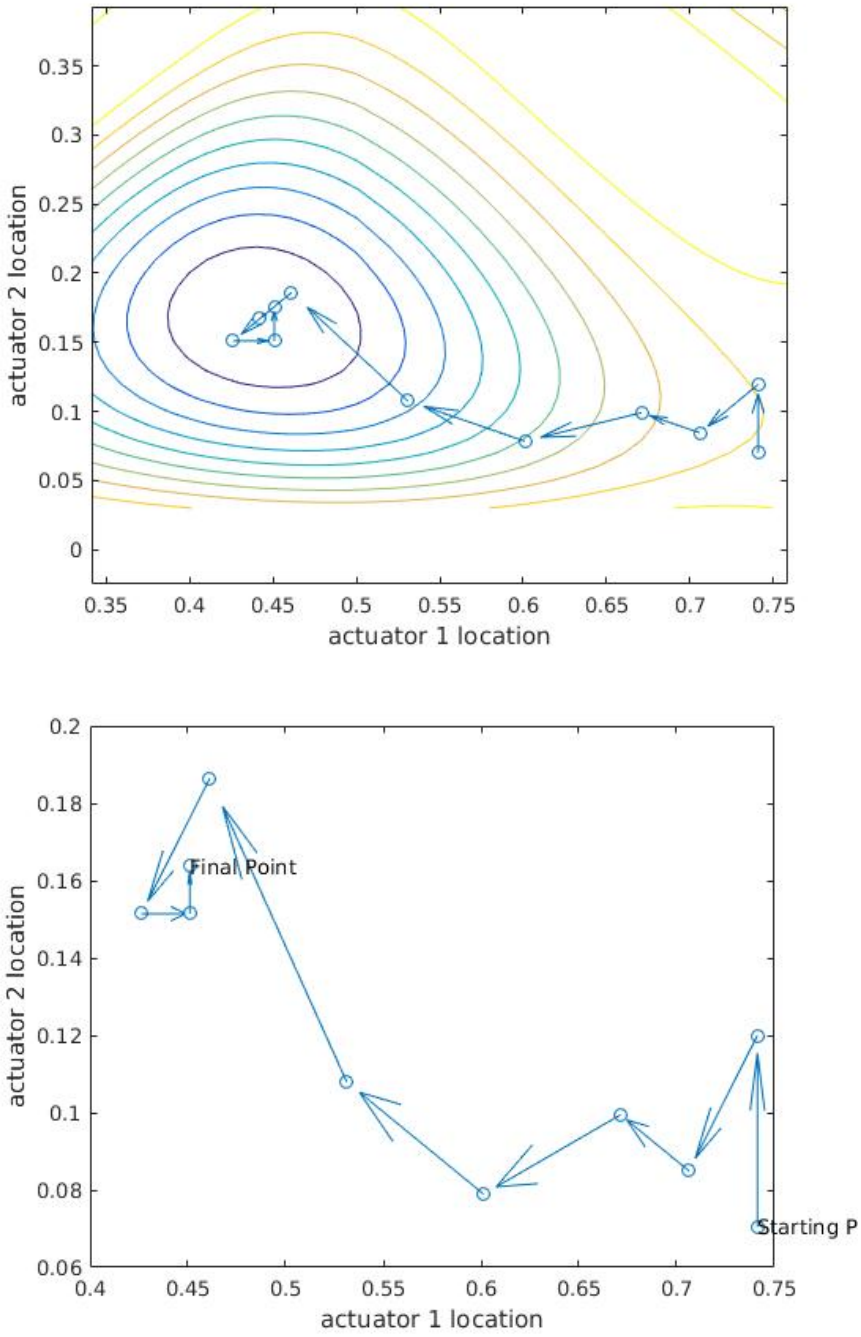


fig: Quiver Plot, Initial Point 3

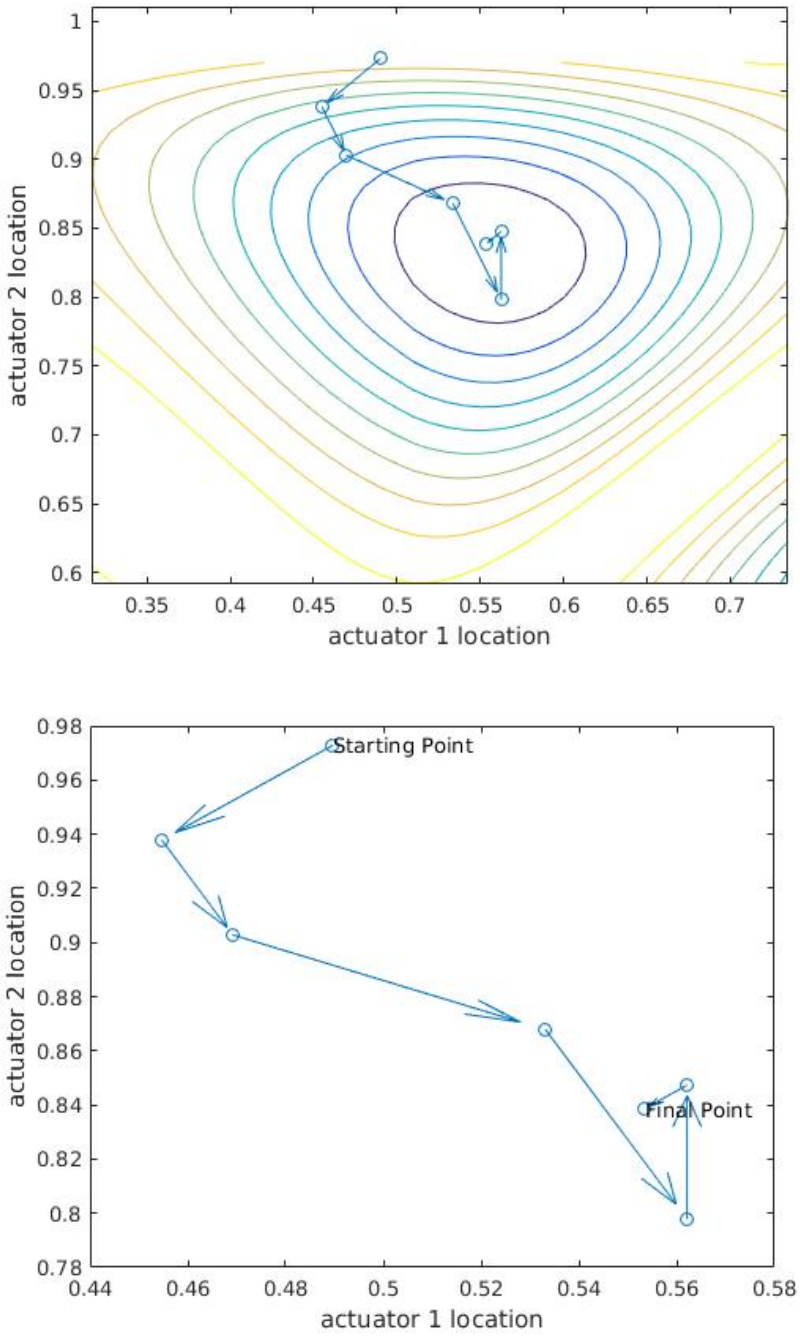


fig: Quiver Plot, Initial Point 4

7. Disturbance mode = 1 , Number of actuators = 3

   Since there are several minimas possible, so it cannot be exhaustively listed. The γ_{min} value is 738.88 from closed form rigorous search.

Performance of Algorithms- 3 Actuators with 1st mode disturbance

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.6986 0.5030 0.3145)	742.9264	22.0000	21.1322
MATLAB Patternsearch	(0.5195 0.7088 0.2934)	741.8082	78.0000	80.4833
MATLAB Nelder-Mead	(0.6955 0.3018 0.4905)	737.9412	86.0000	87.0701
MATLAB fmincon	(0.4819 0.2921 0.6881)	736.7003	121.0000	119.9274

Table 25: Initial Point:[0.818 0.709 0.743]

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.6999 0.3108 0.5089)	739.9220	15.0000	11.1220
MATLAB Patternsearch	(0.4981 0.2911 0.6794)	742.0988	71.0000	72.7436
MATLAB Nelder-Mead	(0.6984 0.3022 0.5098)	733.6735	89.0000	91.3772
MATLAB fmincon	(0.6866 0.2956 0.4900)	740.7381	123.0000	116.2032

Table 26: Initial Point:[0.993 0.357 0.753]

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.4992 0.6785 0.2920)	736.3181	11.0000	9.6818
MATLAB Patternsearch	(0.5102 0.6949 0.3221)	739.5257	54.0000	57.9752
MATLAB Nelder-Mead	(0.5097 0.6797 0.2736)	747.0353	47.0000	51.9260
MATLAB fmincon	(0.4983 0.6878 0.3020)	735.8468	119.0000	120.3229

Table 27: Initial Point:[0.539 0.768 0.233]

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.6987, 0.3215 , 0.5278)	740.2035	22.0000	20.2866
MATLAB Patternsearch	(0.4985 , 0.2829 , 0.6679)	743.8631	69.0000	77.8280
MATLAB Nelder-Mead	(0.3615 , 0.0733 , 0.6384)	**869.9228**	58.0000	70.6796
MATLAB fmincon	(0.3096 , 0.4905 , 0.6803)	742.0402	84.0000	98.5903

Table 28: Initial Point:[0.325 0.0836 0.513]

Performance Summary : The important observation in this case is relative improvement in performance of fmincon which can be because of the increase in dimension of optimization space. Apart from this Nelder Mead failed to produced correct result in one instance. To primarily check the relative improvement of fmincon a test case with 4 actuators is also run.

8. Disturbance mode = 1, Number of actuators = 4

   From closed form rigorous search minima comes out to be $\gamma_m in = 643.4$

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.4488 0.7277 0.5705 0.2832)	643.3564	31.0000	25.9805
MATLAB Patternsearch	(0.2631 0.7360 0.4123 0.5879)	646.9877	194.0000	184.6373
MATLAB Nelder-Mead	(0.1876 0.6580 0.4305 0.1161)	**778.0484**	84.0000	91.0171
MATLAB fmincon	(0.2637 0.7174 0.5683 0.4149)	645.0324	170.0000	180.1866

Table 29: Initial Point:[0.127 0.913 0.632 0.0975]

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.5987 0.7376 0.4519 0.2998)	646.6444	31.0000	27.0262
MATLAB Patternsearch	(0.4097 0.2732 0.7138 0.5492)	646.3857	101.0000	105.1364
MATLAB Nelder-Mead	(0.6980 0.5103 0.3228 0.1173)	698.6541	109.0000	119.9425
MATLAB fmincon	(0.7168 0.5643 0.4207 0.2688)	645.9019	147.0000	155.5011

Table 30: Initial Point:[0.933 0.973 0.192 0.139]

	final position	γ_{min}	Func. Count	Time
Project Patternsearch	(0.7169 0.4506 0.6547 0.3028)	658.6823	32.0000	29.4935
MATLAB Patternsearch	(0.2835 0.4387 0.7271 0.5802)	643.5616	132.0000	132.8168
MATLAB Nelder-Mead	(0.4110 0.1080 0.6392 0.0214)	**844.4908**	105.0000	114.7140
MATLAB fmincon	(0.7465 0.2798 0.5884 0.4399)	645.0452	127.0000	134.6103

Table 31: Initial Point:[0.989 0.0669 0.939 0.0182]

Performance summary : The performance of $fmincon$ has got even more closer to the patternsearch algorithms and in some case even beating them by a minor margin. Nelder-Mead algorithm showed bad performance in two instances here.

Overall Performance Summary of Algorithms

1. The PPS algorithm overall performed the best due to the least number of function evaluations done.

2. The overall performance of Patternsearch algorithm turned out to be better than Nelder-Mead and gradient based methods, with Nelder-Mead very occasionally turning out to be better. The Nelder-Mead algorithm also struggled with accuracy in a few instances which became more frequent with increase in the dimension of optimization.

3. Contrary to what was expected, the gradient based method fmincon produced correct results in all cases with very rare failures. Further, the relative performance of gradient based method improved with the increase in the domain of optimization. A shortcoming of gradient method was the requirement of a lot of function evaluations which makes it not the best alternative for low dimension $H_\infty problems$.

VIII Conclusion and Future Work

The problem of H_∞ optimal actuator location is a computationally expensive problem. The most determining factor about the performance of a solver for it is the number of function evaluations done in getting the final result. The Theorems described in Section 4 together give a way to check if attenuation at any location is less than a particular value. This has been extensively used in the pattern search algorithm proposed in the project. This greatly reduces the number of function evaluations in the exploratory search step as those points in the domain where the attenuation is already not less than the current point is not even evaluated. The reduction in number of function evaluations can be directly seen comparing it with MATLAB's pattern search algorithms which have the identical search parameters except having the tool to check an attenuation before calculating it. The Pattern search algorithm implemented in the project successfully estimated the optimal locations being fast. Another important observation is the performance of the gradient based algorithm. The result is correct in almost all occasions which is a useful result for future research. The success of gradient algorithm can be attributed to the smooth nature of the cases discussed. Seeing the results of the algorithms it can be inferred that the Patternsearch algorithms suits better to the nature of the problem if it is a low dimensional problem. Although, if the problem is high dimensional then it would be interesting to look at the performance of gradient based algorithm whose relative performance with respect to pattern search algorithms improves with increase in dimension.

One future work which naturally follows from here is the comparison of GPS and HJCS algorithms developed specifically for this problem with the performance of patternsearch which is using the combination of both. Also, an interesting problem encountered is related to the algorithm for checking an attenuation γ in section 4. In the algorithm, to check an attenutation, the very last step requires checking the positive semidefiniteness of solution of ARE P because it is forced by Theorem *IV*.3 but not concluded from Theorem *IV*.1. This requires some extra computational work of calculating the solution and then checking its positive semidefiniteness . An important thing to notice is that the Q for the ARE constructed to check the attenuation in Theorem *IV*.3 is positive semi-definite which has not been assumed in the Theorem *IV*.1 . It will be interesting to research the possibility of positive semidefiniteness automatically implied or a certain bound in γ value as a function of norms of the matrices which guarantees positive semidefiniteness of the solution. The another scope of work here is the performance of a gradient based optimization algorithm implemented specifically for this problem compared to the Patternsearch

algorithms. This is because of the improving performance of gradient methods with increase in dimension and deterioration in the performance of patternsearch algorithms.

Bibliography

[1] Kirsten Morris, *Introduction to Feedback Control*, Harcourt/Academic Press 2001, pp. 65-66

[2] Kirsten Morris, *Introduction to Feedback Control*, Harcourt/Academic Press 2001, pp. 66-67

[3] Kirsten Morris, *Introduction to Feedback Control*, Harcourt/Academic Press 2001, pp. 252-254

[4] Kirsten Morris, *Introduction to Feedback Control*, Harcourt/Academic Press 2001, pp. 272-275

[5] Kirsten Morris, *Introduction to Feedback Control*, Harcourt/Academic Press 2001, pp. 280-282

[6] Kirsten Morris, *Introduction to Feedback Control*, Harcourt/Academic Press 2001, pp. 282-285

[7] Kirsten Morris, *Introduction to Feedback Control*, Harcourt/Academic Press 2001, pp. 278-279

[8] Kirsten Morris, *Introduction to Feedback Control*, Harcourt/Academic Press 2001, pp. 259-261, pp. 278-279

[9] Lidstrom, Carolina; Rantzer, Anders; Morris, Kirsten; *H-infinity optimal control for infinite-dimensional systems with strictly negative generator*, 2016 IEEE 55th Conference on Decision and Control, CDC 2016

[10] http://www.sce.carleton.ca/faculty/chinneck/po/Chapter17.pdf

[11] Kasinathan, Dhanaraja; Morris, Kirsten; *H-infinity optimal actuator location*, 2013 IEEE Transactions on Automatic control, Vol. 58

[12] Jorge Nocedal, Stephen J. Wright *Numerical Optimization*, Second Edition 2006