AIAA Guidance, Navigation, and Control Conference and Exhibit
5-8 August 2002, Monterey, California

AIAA 2002-4762

TUNING OF A RANDOM SEARCH ALGORITHM FOR CONTROLLER DESIGN

Rajeeva Kumar[*] And David C. Hyland[†]
Department of Aerospace Engineering
University of Michigan, Ann Arbor

Abstract

Fixed structure controllers display numerous local minima for quadratic performance measures and high dimensional plants. In fact, minimizing a quadratic function with nonlinear constraints is known to be an NP-hard problem. This implies a severe computational burden for higher dimensional problems. One class of multi -modal optimization approach that could overcome this problem is random search optimization. However, very little is known about how different parameters of such algorithms should be adjusted in order to achieve a desired convergence speed. This paper presents a systematic analysis of an Adaptive Random Search Algorithm similar to that of Pronzato. The analysis reveals the key parameters affecting convergence time and provides insight on ways to tune the algorithm for more rapid convergence. A new stopping criterion is also proposed that eliminates the need to estimate the optimum function value beforehand.

1. Introduction

Most controller synthesis problems require minimization of some kind of cost function. On one hand, these cost functions may not always be smooth and convex, thus giving rise to multi-modalness and non-differentiability. On the other hand, certain cost functions, e.g., fixed structure controllers with quadratic performance measures and high dimensional plants, intrinsically display numerous local minima. In fact, minimizing a quadratic function with nonlinear constraints is known to be an NP-hard problem[1] . This implies a severe computational burden for higher dimensional problems and thus may also prevent real-time implementations of many global optimization algorithms.

[*] Graduate Student Research Assistant
[†] Professor and Chair

Horst and Pardalos's handbook[1] contains an insightful summary of many of the common global optimization methods, both deterministic and stochastic.

Stochastic search algorithms for solving optimization problems have been applied successfully in many areas of science and engineering. For example, Schaetzer et al[2], used a stochastic search algorithm to successfully solve the vector optimization problem of a permanent magnet synchronous machine consisting of two objective functions and two design parameters. He et al[3] used stochastic search techniques for generation of transfer functions for the visualization of volumetric datasets. Cohen and Harvey[4] proposed a cluster search scheme for object location within an image. Solomatine[5] used random search algorithms for hydrological model calibration and for finding an optimal design for pipe network. Application of such methods in water distribution problems allows the solution of complex optimization problems without the necessity of formulating an analytical objective function.

There are many more applications. The point is that such algorithms are model-free, global in nature and can be implemented easily, even for complex problems mentioned here and scale well with the dimension of the problem. Such an approach is ever more feasible due to the continuing growth in availability of computational power. Stochastic search algorithms are clearly better suited for higher dimension problems and problems where analyticity of the cost function is not assured.
There are many randomized algorithms available in the literature[6,7]. In this paper, we present and analyze an algorithm that is very similar to the Adaptive Random Search technique first proposed by Bekey et.al.[8] and subsequently modified by Pronzato and Walter[9]. Here, we will develop a probabilistic model for the probability density of the number of iterations the algorithm takes in order to get within the acceptable performance level.

Most of the global optimization algorithms suffer from the disadvantage of not having any practical stopping criterion. The standard criterion based on gradient becoming sufficient small, does not necessarily work when faced with multiple local minima[1]. The main contribution of this paper lies in suggesting a stopping criterion, based on the probability model for the number of iterations the algorithm takes in order to get within an acceptable performance level. We will show the applicability of the simple model developed for the 1-dimensional case through finding the minimum of a negative sinc function. The same algorithm will then be used to design a model-following controller, for an unmanned air vehicle.

The paper is organized as follows. In section 2, we will state the problem formulation. In section 3, we will present the Adaptive Random Search Algorithm. In section 4, we develop an analytical model for predicting the probability distribution of the number of iterations it takes the algorithm to converge for the one-dimensional case. In section 5, we show the accuracy of the model by verifying it for a negative sinc function. We, then, discuss the tuning of the algorithm with respect to the different parameters involved. We also propose a new stopping criterion and show its effectiveness. Then, we use the algorithm to design a one parameter controller for a UAV. Section 6 concludes the paper with some recommendation for the future work.

2. Problem Formulation

The optimization is posed here as a minimization problem. Simply stated, the global minimization problem (P) is to find

$$J^* = \min_{x \in X} J(x), \qquad (1.1)$$

where $J(.)$ is the function to be minimized, $X \subset R^n$ is the feasible space of input parameters, assumed compact. Under these conditions, we know that the optimal solution value J^* exists and is attained, i.e. the set

$$X^* = \{x \in X : J(x) = J^*\}, \qquad (1.1A)$$

is nonempty[1]. As mentioned earlier, the global optimization problem (P) is inherently unsolvable in a finite number of steps. Thus, generally, we can not find a point in X^* in finite time. Usually, we consider the global optimization problem solved, if, we find a point in the level set

$$X_\varepsilon = \{x \in X : J(x) \le J^* + \varepsilon\} \qquad (1.2)$$

for some $\varepsilon > 0$.

In the next section we will outline the Adaptive Random Search (ARS) algorithm for solving such problems.

3. Adaptive Random Search Algorithm

The main idea in a probabilistic method is to generate, at each iteration k, a new parameter vector x^k until $J(x) < J(x^k)$, where x^k is the estimate of the optimal x at the beginning of iteration k. The last value of x is then accepted as x^{k+1}. Such random search methods can only be proved to converge in probability[1]. However, their implementation is very easy. Bekey et.al[8,11] proposed a random search global optimization algorithm, where the variance of the step-size distribution is periodically optimized. By searching over a wide variance range, the algorithm finds the step-size distribution that yields the best local improvement in the criterion function. The variance search is then followed by a specified number of iterations of local random search where the variance remains fixed. Periodic wide range searches are introduced to avoid the process stopping at local minima.

The ARS algorithm has many parameters and the performance of this algorithm may depend heavily on their practical implementation. Pronzato et.al[9] presented an implementation strategy that requires very little tuning. The scheme of ARS is as follows[9] :

ARS algorithm

Let $\theta^k = \left(\theta_1^k, \theta_2^k, ...\theta_N^k\right)^T$ be the parameter vector at iteration k and $J^k = J(\theta^k)$ the corresponding value of the performance criterion and Ω be the search space.

- Set constants f_1, f_2, f_3, f_4, f_5, γ.

- Input the range of parameters $\theta_{i_{\min}}, \theta_{i_{\max}}$ such that $\theta_{i_{\min}} \leq \theta_i \leq \theta_{i_{\max}}$ $\forall$ *i=1,2,….N.*
- Choose covariance $\Sigma = \left\{ {}^1\Sigma,\ {}^2\Sigma, \ldots.. {}^{f_1}\Sigma \right\}$, where ${}^i\Sigma = diag[\sigma_{i1}, \sigma_{i2}, \ldots. \sigma_{iN}]$.
- *Step* 0: Set $k = 0$, $l = 0$, $m = 0$ and choose the initial point according to the rule $\theta_i^{best} = \dfrac{\theta_{i_{\min}} + \theta_{i_{\max}}}{2}$.
- *Step* 1: $k = k + 1$, choose $\Sigma_c = \Sigma(k)$, $\theta_c = \theta^{best.}$
- *Step* 2: $l = l + 1$, generate one (or several) new trial points $y^l = \theta^c + r^l$, where the random displacement vector r^l is generated over the sample space Ω, according to a normal distribution with zero mean and a covariance Σ_c. Any non-admissible y^l has to be discarded and replaced by an admissible one.
- *Step* 3: Update $\theta^{best} = y^l$ and $\Sigma^{best} = \Sigma_c$, if $J(y^l) < J(\theta^{best})$.
- *Step* 4: Stop if minimum is found, else if $l \leq f_2$, go to *step* 2, else, if $k \leq f_1$, go to step1, else, go to *step* 5.
- *Step* 5: $m = m + 1$, generate one (or several) new trial points $y^m = \theta^{best} + r^m$, where the random displacement vector r^m is generated according to a normal distribution with zero mean and a covariance Σ^{best}. Any non-admissible y^m has to be discarded and replaced by an admissible one.
- *Step* 6: Update $\theta^{best} = y^m$, if $J(y^m) < J(\theta^{best})$.
- *Step* 7: Stop if minimum is found, else if $m \leq f_4$, go to *step* 5, else, if Σ^{best} is selected f_5 time consecutively from *step* 3 , stop; else go to *step*1

Pronzato suggested some typical values for various parameters. For example, $f_1 = 5$, $f_2 = f_3/k$, $f_3 = 100$, $f_4 = 100$, $f_5 = 5$, ${}^1\sigma = \theta_{\max} - \theta_{\min}$, ${}^i\sigma = \gamma^{(i-1)}\sigma$, $\forall$ i=2,…, f_1 , γ= 0.1.

When applied to many benchmark problems from the literature Pronzato et.al. showed that for their nominal parameter values, global convergence over a wide range of test problems occurs without the need to retune the parameters. Further, the convergence is faster or comparable to many other global optimization methods, which do require tuning. However there is no explicit convergence proof for the algorithm.

Our ARS scheme is very similar to this except that we use uniformly distributed random number instead of normally distributed numbers in *step2* and *step5* above for the sake of convenience. We also maintained the value of f_4 zero. The simulation results show that for moderately sized search domain with variance decreasing at each step, searches with very small variance increases the computational burden quite disproportionably, with little or no increase in the probability of convergence. With a uniform distribution, a change in variance simply implies a proportional change in the search domain (domain length = $2\sqrt{3\,\text{var}}$).

Here it is assumed that the user knows or has an estimate of the minimum value of the cost function *J* but does not know its location and the algorithm is run to find that location. In next section we will develop a simple probability model for *k*, the number of iterations the algorithm takes in order to converge within a given accuracy, for the one-dimensional case.

4. Mathematical Analysis of the ARS for One-Dimensional Case

Consider a one-dimensional cost function *J*(.) as shown in figure 1. The cost function is such that it has a uniquely defined isolated global minimum J^*. The accuracy level $\varepsilon > 0$ is given such that the set X_ε as defined in (1.2), is connected and non empty. Also let's denote the Lebesgue measure $\mu(X_\varepsilon)$, of X_ε by α. It is obvious that the algorithm presented in the previous section will get to the global optimum within a finite number of iterations, with probability one. By this we mean that, given any predefined accuracy level $\varepsilon > 0$, a sample point corresponding to a function value not higher than $J^* + \varepsilon$ will eventually be found. Actually this is a trivial result[1] enjoyed, in general, by all methods that are based upon sampling that assigns positive probability to every set with non-null Lebesgue measure.

In the first step, the N_1 (so, in ARS algorithm of previous section f_2=N_1 for this step) points are randomly selected from the uniform distribution over sample space Ω (shown as A_1B_1 in the figure). Let's denote the Lebesgue measure

$\mu(\Omega)$, of space Ω by $2L$. Without loss of any generality, let's assume that the global minimum point x^* lies exactly in the middle of the interval A_1B_1

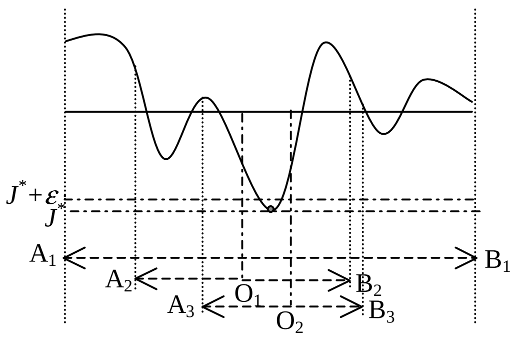

Figure 1 : ARS Algorithm Illustration

Now, the probability that the algorithm converges at m^{th} iteration, can be represented as

$$p(x_m \in X_\varepsilon) = \Pr\{x_m \in X_\varepsilon \text{ for the first time}\}$$
$$= \left(\frac{2L-\alpha}{2L}\right)^{m-1}\frac{\alpha}{2L}. \qquad (1.3)$$

Now, if $x_j \notin X_\varepsilon$, for j =1,…,N_1, we start the next set of N_2 sampling (so, in ARS algorithm of previous section f_2=N_2 for this step and so on) from the uniform distribution over Ω_1(shown as A_2B_2 in the figure) centered around the best point located in the previous N_1 samples. Let's denote the Lebesgue measure $\mu(\Omega_1)$, of space Ω_1 by $2\gamma L$. By best point we mean the point corresponding to the statistical minimum of previous N_1 points, i.e., the point $\bar{x} = \arg\min\{J(x); x = x_1,....,x_{N_1}\}$. For evaluating the above, we need to determine the probability density $p(\bar{x})$ of $\bar{x}$. In order to do this we need certain information about the function. Let us assume that during the previous N_1 trials the x_m –value generated fall in a region, D, such that

$$d(x_1, X_\varepsilon) \le d(x_2, X_\varepsilon) \Rightarrow J(x_1) \le J(x_2)\,,$$

where, $d(x_i, X_\varepsilon)$ = distance of the external point x_i from the boundary of X_ε.

This assumption essentially implies that the values of $J(x_m)$ for some set of m's increase in proportion to the distance of x_m's from the set X_ε.

Thus, the value of $\bar{x}$ is that x_m which is closest to the boundary of set X_ε. Then, it is trivial to show that the probability distribution of $\hat{x}$, the smallest distance from X_ε is given by

$$p(\hat{x}) = \frac{N_1}{2L-\alpha}\left(1-\frac{\hat{x}}{2L-\alpha}\right)^{N_1}, \qquad (1.4)$$

where, $\alpha = \mu(X_\varepsilon)$, is the Lebesgue measure of set X_ε .

For large N_1, (1.4) can be approximated as

$$p(\hat{x}) = \lambda\, e^{-\mu \hat{x}} \quad . \qquad (1.5)$$

Where, $\lambda = \dfrac{N_1}{2L-\alpha}$.

The probability of finding the minimum in the $(N_1 + m)^{\text{th}}$ step can be expressed as

$$P(x_{N_1+m} \in X_\varepsilon) = \Pr\{x_k \notin a \;\; \forall k \le N_1\} *$$
$$\Pr\{x_{k+n} \notin a \;\; \forall n < m\} \Pr\{x_m \in a\}$$

$$= \left(\frac{2L-\alpha}{2L}\right)^{N_1} F(m; L; \gamma; N_1; \alpha)\,,$$
$$\forall \;\; m = 1 \ldots N_2\,. \qquad (1.6)$$

It is easy to see that for a given $\hat{x}$, we have

$$\Pr(F/\hat{x}) = \begin{cases} 0; & \gamma L < \hat{x} \\ \left(1-\dfrac{\gamma L-\hat{x}}{2\gamma L}\right)^{m-1}\dfrac{\gamma L-\hat{x}}{2\gamma L}; & \gamma L \ge \hat{x},\ \gamma L-\hat{x} < \alpha\,. \\ \left(1-\dfrac{\alpha}{2\gamma L}\right)^{m-1}\dfrac{\alpha}{2\gamma L}; & \gamma L \ge \hat{x}\,,\ \gamma L-\hat{x} \ge \alpha \end{cases}$$
(1.7)

Hence,

$$F = I_1 + I_2\,, \qquad (1.8)$$

Where

$$I_1 = \int_{\max(\gamma L-\alpha,0)}^{\gamma L} d\hat{x}\; p(\hat{x})\left(1-\frac{\gamma L-\hat{x}}{2\gamma L}\right)^{m-1}\frac{\gamma L-\hat{x}}{2\gamma L}\,,$$

and

$$I_2 = \int_0^{\max(\gamma L-\alpha,0)} d\hat{x}\; p(\hat{x}) \frac{\alpha}{2\gamma L}\left(1-\frac{\alpha}{2\gamma L}\right)^{m-1} .$$

Carrying out the exact integration is possible but the result is very cumbersome. However, if N_1 is large enough, $p(\hat{x})$ can be replaced by the delta function $\delta(\hat{x}-2L/N_1)$. Later on we will show numerically that this does not affect the overall probability model very much.

This approximates the probability that the algorithm will result in an $x_m \in X_\varepsilon$, in $(N_1+m)^{th}$ iteration as by the expression :

$$p(x_{N_1+m} \in X_\varepsilon) \approx \left(1-\frac{\alpha}{2L}\right)^{N_1} F(m;L;\gamma;N_1;\alpha), \quad \forall\, m = 1, .., N_2 \quad . \quad (1.9)$$

Where

$$F = \begin{cases} 0; & \gamma N_1 < 2 \\ \left(1-\frac{\gamma-2/N_1}{2\gamma}\right)^{m-1} \frac{\gamma N_1 - 2}{2\gamma N_1}; & \gamma N_1 \ge 2, L(\gamma N_1 - 2) < \alpha N_1 . \\ \left(1-\frac{\alpha}{2\gamma L}\right)^{m-1} \frac{\alpha}{2\gamma L}; & \gamma N_1 \ge 2, L(\gamma N_1 - 2) \ge \alpha N_1 \end{cases} \quad (2.0)$$

If the algorithm does not converge in $N_1 + N_2$ iteration, using similar arguments as before we can find the probability that the algorithm will find an $x_m \in X_\varepsilon$ in $(N_1+N_2+m)^{th}$ iteration as

$$p(x_{\sum_{i=1}^{2} N_i + m} \in X_\varepsilon) \approx P_1^{not} F(m;\gamma L;\gamma;N_2,\alpha) \;, \quad \forall m = 1, \ldots, N_3 \quad . \; (2.1)$$

where

$$P_1^{not} = \left(1-\sum_{i=1}^{N_1+N_2} p(x_i \in X_\varepsilon)\right), \qquad (2.2)$$

Similarly we have the probability that the algorithm will find an $x_m \in X_\varepsilon$ in $(\sum_{i=1}^{q} N_i + m)^{th}$ iteration for q=3,4 , in case it has not converged before, as

$$p(x_{\sum_{i=1}^{q} N_i + m} \in X_\varepsilon) \approx P_{q-1}^{not} F(m;\gamma^{q-1}L;\gamma;N_q,\alpha), \quad \forall m = 1, \ldots, N_{q+1} \; . \quad (2.3)$$

Where

$$P_{q-1}^{not} = \left(1-\sum_{i=1}^{\sum_1^q N_j} p(x_i \in X_\varepsilon)\right), \qquad (2.4)$$

If the algorithm does not succeed in $\sum_{i=1}^{5} N_i$ iterations (so, in ARS algorithm of previous section f_1=5), it will start all over again and the probability that it will find an $x_m \in X_\varepsilon$ in next set of N_1 iteration will be given as

$$p(x_{\sum_{i=1}^{5} N_i + m} \in X_\varepsilon) \approx P_4^{not} \left(\frac{2L-\alpha}{2L}\right)^{m-1} \frac{\alpha}{2L} \quad \forall m = 1, \ldots, N_1 \qquad (2.5)$$

Where

$$P_4^{not} = \left(1-\sum_{i=1}^{N_1+N_2+N_3+N_4+N_5} p(x_i \in X_\varepsilon)\right), \qquad (2.6)$$

and so on.

Equations (1.3),(1.9),(2.1),(2.3) and (2.5) define the probability model for the algorithm to find an $x_m \in X_\varepsilon$ in m^th iteration. As mentioned earlier, these equations are obtained using delta function approximation for probability of (1.4). We computed the probability model with and without this approximation and the result is shown in figure2. The Figure shows that the approximation is very accurate.

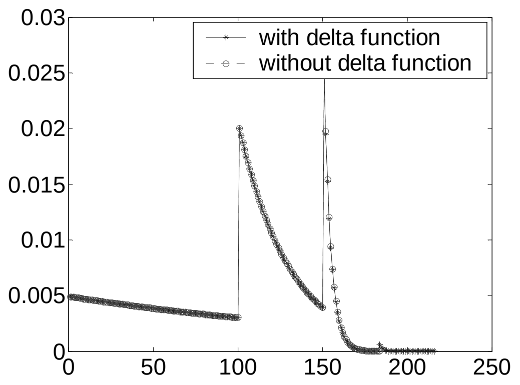

Figure 2: Comparison of the probability model with and without the delta function approximation

Now, it is easy to show that if γ is properly chosen ($\gamma > \max[2/N_1\ 2/N_2\ 2/N_3\ \ 2/N_4]$, to be precise) the cumulative probability will tend to unity for a certain value of m , implying that the algorithm succeeds in finding an $x_m \in X_\varepsilon$, with probability close to one. This argument can be turned around to find a stopping criterion for this algorithm. Recall that the algorithm as presented in previous section, requires the knowledge of the minimum value J^* for determining the stopping criterion. For certain applications, e.g. designing a model following controller, this assumption is valid as one knows the minimum value J^* , but in other cases this may be a serious restriction. However, using this probability model one can overcome this difficulty as presented in following stopping criterion.

Stopping Criterion :

In the absence of knowledge of the minimum value of the cost function J^* one can use the cumulative probability that $x_m \in X_\varepsilon$ is found by mth iteration as the basis for a stopping rule. Our proposed stopping criterion is as follows :

Let $p_f(m)$ be the probability of failure, i.e., the probability of never getting $x_m \in X_\varepsilon$ in m iterations. Then stop the algorithm when

$$p_f(m) \le \delta\ , \text{ for } \delta << 1\,. \qquad (2.7)$$

Where ,

$$p_f(m) = 1 - \sum_{i=1}^{m} p(x_i \in X_\varepsilon)\,. \qquad (2.8)$$

Now, as $\delta \to 0$, the point $\bar{x}$ corresponding to the statistical minimum, i.e $\bar{x} = \arg\min\{j(x); x = x_1,....,x_m\}$, will be in the acceptable set X_ε with probability one.

In the next section, we will show the accuracy of the probability model developed here through numerical examples.

5. Results

In this section, we will use the algorithm to find out the global minimum of the negative sinc function and use the result to verify the accuracy of the probability model developed here. We will also demonstrate our stopping criterion and the usefulness of the model in tuning some parameters of the algorithm for faster convergence. Also, we will use this algorithm to design a controller gain for longitudinal axis control law for a UAV.

Example1: Negative sinc Function:

The algorithm presented here is used to find the global minimum of a negative sinc function $f(x) = -\sin(x)/x$ over a range of [-500 500], with parameters γ=0.15, N_1=100, N_2=50, N_3=33, N_4=25, N_5=20. To get a statistical estimate of the probability of the number of iterations required to find a solution, we run the algorithm 10000 times with the tolerance ε = 0.01. Figure 3 presents the results. The ordinate is the number of iterations, m, the algorithm performs before getting an acceptable solution. The plot shows that with current value of the parameters the algorithm is very likely to converge between 150 and 200 iterations. This is also reflected in the histogram presented in Figure 4.

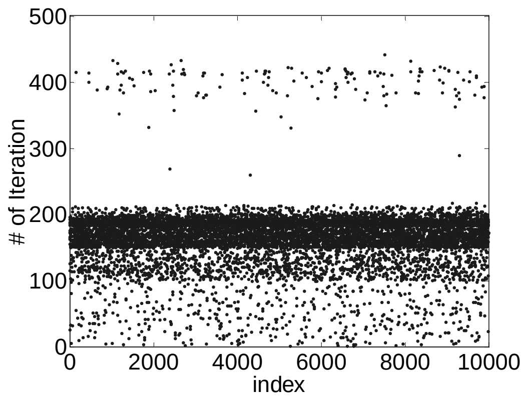

Figure 3: ARS run for 1000 times on the Negative sinc function

The y-axis has been divided by 10000 to get the probability mass function of the number of iterations. This is compared with the theoretical model developed in the previous section.. For the theoretical model, we have used α=0.49 which corresponds to ε = 0.01. As can be seen, the theoretical model matches very well with the experimental results.

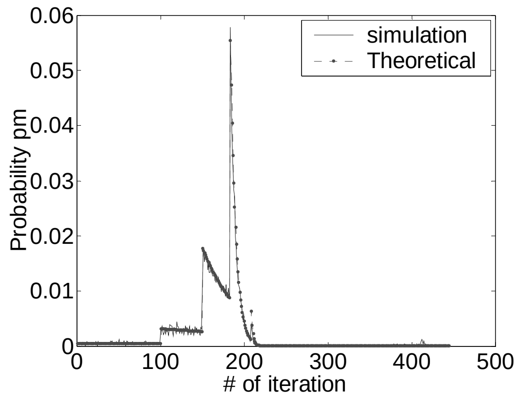

Figure 4 : Comparison of the theoretical model with the simulation

In the above example, if we choose the threshold δ=5x10^{-7}, the stopping criterion (2.7) gives the value of *m* as 445. This means that if we stop the algorithm after 445 iterations, the statistical minimum point will lie in the acceptable set X_ε with very high probability. This can be verified by figure 3.

Using the probability model developed here, one can optimize the parameters of the algorithm for faster convergence. For example one can find out the expected number of iterations as a function of reduction parameter γ, for a specified probability of failure level δ.

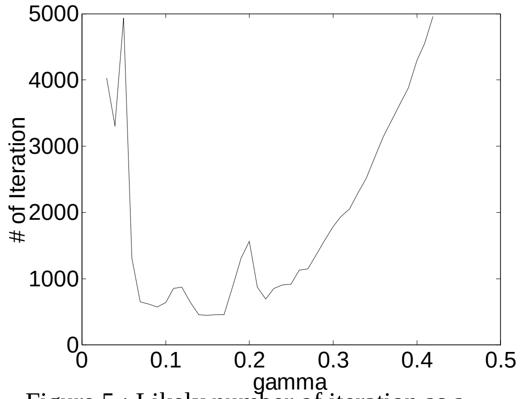

Figure 5 : Likely number of iteration as a function of γ for the algorithm to converge

Figure 5 shows the expected number of iterations the algorithm will take for a probability of failure level not more than 5x10^{-7}, as a function of γ. The other parameters are kept fixed as N1=100, N_2=50, N_3=33, N_4=25, N_5=20 and *L*=500. The figure indicates that γ=0.15 is the best value with these parameters. Similar studies can be undertaken for the other parameters too.

Example2 : Longitudinal Controller Design for a UAV

We designed a single parameter longitudinal control laws using the UAV model and the controller structure of Rajeeva and Hyland[10]. For the controller, the only free parameter is K_q, the pitch rate (q) feedback. The Angle of attack feedback is kept fixed at 2.0 rad/rad. The controller is designed as a model following controller for a standard second order pitch rate model having a zero at –0.7685 and damping and natural frequency as 0.547 and 1.38 rad/sec. Here the cost function is mean squared model following error. Hence, the minimum value, J^*, is zero. The cost function here has more than one local minimum. With tolerance ε set as 0.001, the algorithm gives K_q=1.107 /sec in 192 iterations, with the actual mean square model following error of 5x10^{-4}. Figure 6 compares the q-model response with that generated with the closed loop system. The plot shows that the match between the q-model and the closed loop q-response is very good.

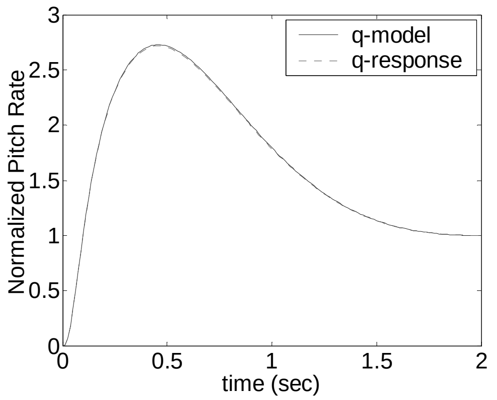

Figure 6 : Comparison of pitch rate model And the closed loop response

Suppose the cost function was such that we did not know the J^* value, then the proposed stopping criterion based on the probability model could be used.

Using the probability model developed here we can compute the expected number of iterations the algorithm would take for various value of α/2L with fixed value of N_1=100, N_2=50, N_3=33,

N_4=25, N_5=20, γ=0.1 as a function of the parameter of the cost function. In figure 7, it is plotted against the $\alpha/2L$ for probability of failure level, δ=5x10^{-7}. Now, if we can estimate the $\alpha/2L$ range, say, between 0.0005 and 0.0015, we can stop the algorithm at 400 iterations and pick the statistical minimum as the actual minimum, based on these 400 iterations. The estimate of $\alpha/2L$ may be based on previous experience.

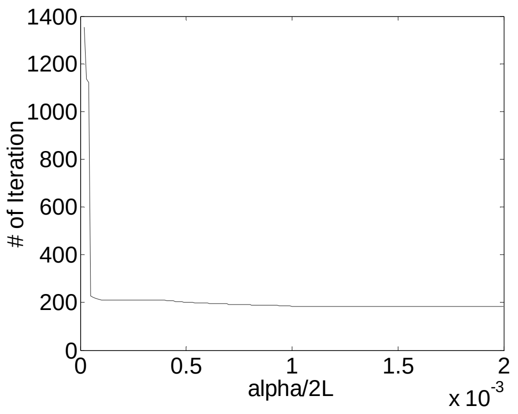

Figure7 : Number of iteration as function of $\alpha/2L$ with other parameter fixed

6. Conclusion

In this paper, we considered a version of the Adaptive Random Search Algorithm that is similar to the algorithm proposed by Pronzato. We developed a probabilistic model for the number of iterations the algorithm takes to find an acceptable solution in a single parameter system. We verified the model through numerical simulation. Then, using this model we proposed a novel stopping criterion for a global optimization problem. The criterion is particularly useful when the optimum value of the cost function is unknown. We, also, illustrated the usefulness of the probability model through examples.

The model developed here is valid for only single parameter case and where the acceptable set is connected. A potential extension would be to extend this model to multi-dimensional problems and a more general form of acceptable set.

References

1. Horst, R., ed., Pardalos, P.M., ed., *Handbook of Global Optimization*, Kluwer Academic Publishers.
2. Schaetzer, Ch., Binder, A. and Mueller, W., “A New Approach for Solving Vector Optimization Problems”, *IEEE Transactions on Magnetics*, Vol. 36, No 4, July 2000.
3. He, T., Hong, L., Kaufman, A., and Pfister H., “Generation of Transfer Functions with Stochastic Search Techniques”, *Visualization*, San Francisco, US, 1996, pp227-234.
4. Cohen, Harvey A., and Harvey, Alan L., “Stochastic Search Approach to Object Location”, *IEEE International Conference on System, Man, and Cybernetics*, San Antonio, US, Vol. 3. , pp2237-2241, 1994.
5. Solomatine, D. P. “Random Search Methods in Model Calibration and Pipe Network Design”, *Water Industry Systems: Modeling and Optimization Applications*, Vol.2, D.Savic, G. Walters(eds). Research Studies Press Ltd., Baldock, UK, 1999, pp317-332.
6. Boender, C.G.E., and Romeijn, H.E., “Stochastic Methods”, pp-829-869 in reference [1] above.
7. Schoen, Fabio, “Stochastic Techniques for Global Optimization : A Survey of Recent Advances”, *Journal of Global Optimization* 1:207-228, 1991.
8. Bakey, G.A. and Mastri, S.F., “ Random Search Techniques for Optimization of Nonlinear Systems with many parameters”, *Mathematics and Computers in Simulation* 25 (3) , 263-268, 1982.
9. Pronzato, L., Walter, E., Venor, A., Lebruchec, J.-F, “A General Purpose Global Optimizer: Implementation and Application,” *Mathematics in Computers and Simulation,* Vol. 26, pp. 412-422, 1984.
10. Kumar, R. and Hyland, David, “Control Law Design Using Repeated Trial”, *American Control Conference 2001*, pp-837-842, 2001.
11. Masri, S.F., Bekey, G.A. and Safford, F.B., “A Global Optimization Algorithm Using Adaptive Random Search,” *Applied Mathematics and Computation,* Vol. 7, pp. 353-375, 1980.