CHAPTER 3

EVOLUTIONARY ALGORITHMS

3.1 INTRODUCTION

Darwin’s theory of natural selection is considered as the fundamental unifying theory of life. Darwinian evolution is basically a robust search and optimization mechanism. The problems that biological species have solved are typified by chaos, chance, temporality and nonlinear interactivities. These are also characteristics of problems that have proven to be especially intractable to classic methods of optimization. Linear Programming, gradient decent and other related methods have difficulty in handling the intricacies of the real world problems. The process of evolution, however, can be applied to problems where heuristics are not available or generally lead to unsatisfactory results.

EAs are biologically-inspired optimization algorithms, imitating the process of natural evolution, and are becoming important optimization tools for several real-world applications. EAs are robust optimization methods. EAs do not require gradients of the objective function, they can handle noisy objective functions, and they may avoid premature convergence to local minima. The artificial creatures in EAs, known as individuals, are typically represented by fixed length strings or vectors. Each individual encodes a single possible solution to the problem under consideration.
The EA is started with an initial population of size $N$ comprising random individuals. Every individual is then assigned a fitness value based on the objective function and constraints of the problem. For minimization problem, population members with minimum fitness values represent better solutions to the problem than individuals with higher fitness values. Following this initial phase, the main iterative cycle of the algorithm begins. Using mutation (perturbation) and recombination operators, the N individuals in the current population produce $\mu$ offspring. The $\mu$ offspring are assigned fitness values. A new population of N individuals is formed from the N individuals in the current population and the $\mu$ offspring. This new population becomes the current population and the iterative cycle is repeated. At some point in the cycle, evolutionary selection is applied. That is, the Darwinian strategy of the survival of the fittest is employed and individuals compete against each other. This is achieved by selection based on fitness values, with fitter individuals more likely to be selected.

Figure 3.1 provides a simple flowchart illustrating the basic concept of EA (Fogel et al 1966). The process starts with the sampled randomly population of candidate solutions. From the population set, parent selection is made to retain a subset of the population. New solutions (offspring) are created by applying random variation to the existing parent solutions. This variation can come in the form of single-parent or multi-parent operators. The algorithm terminates, when some termination criterion has been satisfied, such as prescribed maximum number of generations, or a suitable error tolerance.
Three popular EAs, namely the MPSO, CMAES and SaDE algorithms are chosen to solve optimization problems in this thesis. These are discussed in this chapter.

### 3.2 MODIFIED PARTICLE SWARM OPTIMIZATION (MPSO)

Kennedy and Eberhart (1995) proposed the PSO algorithm conceptually based on social behaviour of organisms such as herbs, schools and flocks. PSO is a pseudo-optimization method (heuristic) inspired by the collective intelligence of swarms of biological populations. PSO is a zero-order, non-calculus-based method (no gradients are needed), can solve discontinuous, multimodal, non-convex problems. It includes some probabilistic features in the motion of particles. The system initially has a population of random solutions. Each potential solution, called particle, is given a random velocity and is flown through the problem space. The particles have memory and each particle keeps track of previous best position and corresponding fitness. The previous best value is called $p_{best}$. Thus, $p_{best}$ is related to a particular particle. It also has another value called $g_{best}$, which is the best value of all the particles $p_{best}$ in the swarm. The basic concept of PSO technique lies in accelerating the particle towards its $p_{best}$ and the $g_{best}$ locations at each time step. Acceleration has random weights for both $p_{best}$ and $g_{best}$ locations.
3.2.1 MPSO Algorithm

The step-by-step algorithm of MPSO is given below.

Step 1: Initialize a population ($N$) of particles with random positions and velocities of $d$ dimensions in the problem space.

Step 2: For each particle, evaluate the desired optimization fitness function.

Step 3: Compare particle’s fitness evaluation with particles pbest. If current value is better than pbest ($p_{id}$), then set pbest value is equal to the current value and the pbest location is equal to the current location in $d$ dimensional space.

Step 3: Compare fitness evaluation with the population’s overall previous best. If the current value is better than gbest ($p_{gd}$), then reset gbest to the current particles array index and value.

Step 4: Change the velocity and position of the particle according to Equation (3.1) and Equation (3.2) respectively. $v_{id}$ and $x_{id}$ represent the velocity and position of $i^{th}$ particle with $d$ dimensions respectively and rand1 and rand2 are two uniform random functions.

\[ v_{id} = \omega v_{id} + c_1 \cdot \text{rand1} \cdot (p_{id} - x_{id}) + c_2 \cdot \text{rand2} \cdot (p_{gd} - x_{id}) \]  
\[ x_{id} = x_{id} + v_{id} \]  

Step 5: Repeat from Step 2 until a stopping criterion is met, this is usually a sufficiently good fitness or a maximum number of iterations/function evaluations.
The $\omega$ is called the inertia weight that controls the exploration and exploitation of the search space because it dynamically adjusts velocity. $V_{\text{max}}$ is the maximum allowable velocity for the particles. $c_1$ and $c_2$ termed as cognition and social components respectively are the acceleration constants.

### 3.3 SELF-ADAPTIVE DIFFERENTIAL EVOLUTION (SaDE)

DE is an efficient and powerful population-based stochastic search technique that has been used for global optimization in many real problems. DE was invented by Storn and Price (1997). But, the efficiency of DE crucially depends on choosing an appropriate trial vector generation strategy and their associated control parameter values. This is usually done by a trial-and-error scheme which requires high computational costs. The performance of the original DE algorithm is highly dependent on the strategies and parameter settings. Also, during different evolution stages, different strategies with different parameter settings can be more effective than others. Self-adaptation has been found to be highly beneficial for adjusting control parameters during evolutionary process, especially when done without any user interaction.

#### 3.3.1 SaDE Algorithm

SaDE with four DE strategies and the control parameter $CR$ self-adapted (Qin et al 2009) are discussed. The SaDE algorithm automatically adapts the trial vector generation strategies and the crossover rate parameter $CR$ during evolution. SaDE algorithm is described in the following section.

#### 3.3.1.1 Initialization

The initial population is generated randomly and it should better cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum parameter bounds.
3.3.1.2 Trial vector generation strategy adaptation

The mutation operator is applied to each individual or target vector \( X_{i,G} \) at the generation \( G \) to produce the mutant vector. After the mutation phase, crossover operation is applied to each pair of the target vector \( X_{i,G} \) and its corresponding mutant vector to generate a trial vector \( U_{i,G} \). For each individual in the current population, one strategy will be chosen according to a probability learnt from its previous experience of generating promising solutions and applied to perform the mutation operation. If one of the strategies performs in the previous generations by generating promising solutions; the probability of that strategy being used in the current generation will increase.

The strategy candidate pool consists of the following four strategies as given by Qin et al (2009). The four strategies are given in Equations (3.3) – (3.6).

1) \( \text{DE/rand/1/bin (ST1)} \)

\[
\begin{cases} 
    x_{r1,j} + F \cdot \left( x_{r2,j} - x_{r3,j} \right), & \text{if } \text{rand}[0,1) < CR \text{ or } j = f_{\text{rand}} \\
    x_{i,j}, & \text{otherwise}
\end{cases}
\]  

(3.3)

2) \( \text{DE/rand-to-best/2/bin (ST2)} \)

\[
\begin{cases} 
    x_{i,j} + F \cdot \left( x_{\text{best},j} - x_{i,j} \right) + \text{diff}, & \text{if rand}[0,1) < CR \text{ or } j = f_{\text{rand}} \\
    x_{i,j}, & \text{otherwise}
\end{cases}
\]

where

\[
\text{diff} = F \cdot \left( x_{r1,j} - x_{r2,j} \right) + F \cdot \left( x_{r3,j} - x_{r4,j} \right)
\]  

(3.4)
3) **DE/rand-to-best/2/bin** (ST3)

\[
u_{i,j} = \begin{cases} 
    x_{r1,j} + \text{diff}, & \text{if } \text{rand}[0,1) < \text{CR} \text{ or } j \neq j_{\text{rand}} \\
    x_{i,j}, & \text{otherwise}
\end{cases}
\]

\[
\text{diff} = F\left(x_{r2,j} - x_{r3,j}\right) + F\left(x_{r4,j} - x_{r5,j}\right)
\]  

4) **DE/current-to-rand/1** (ST4)

\[
U_{i,G} = X_{i,G} + K \left(x_{r1,G} - X_{i,G}\right) + F\left(x_{r2,G} - X_{r3,G}\right)
\]  

Stochastic universal selection method is used to select one trial vector generation strategy for each target vector in the current population. The probabilities of the strategies are updated only after an initial learning period (LP) generation which is set by the user. The probabilities are initialized to \(\frac{1}{K}\), i.e., all strategies have the equal probability to be chosen. After the initial LP generations, the probabilities of choosing different strategies will be updated at each subsequent generation by,

\[
P_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}}
\]

where

\[
S_{k,G} = \frac{\sum_{k=1}^{G-1} S_{k,g} + nS_{k,g}}{\sum_{k=1}^{G-1} nS_{k,g} + \sum_{k=1}^{G-1} n_{f_{k,g}} + \varepsilon}
\]

where \(nS_{k,g}\) is the number of trial vectors successfully entering the next generation generated by \(k^{th}\) strategy and \(n_{f_{k,g}}\) is the number of trial vectors discarded, generated by \(k^{th}\) strategy and \(\varepsilon = 0.01\). In the SaDE algorithm, the \(CR\) value is made to adapt.
3.3.1.3 Parameter adaptation

In SaDE algorithm the population size \( (N) \) is a user-specified parameter because it highly relies on the complexity of a given problem. Between other two parameters, \( CR \) is usually more sensitive to problems with different characteristics, while \( F \) is closely related to the convergence speed. In the SaDE algorithm, the \( F \) parameter is approximated between the range 0 and 2 by a normal distribution with mean value 0.5 and standard deviation 0.3. A set of values are randomly sampled from such normal distribution and applied to each target vector in the current population. \( CR \) is normally distributed in a range with mean \( CR_m \) and standard deviation 0.1. Initially, \( CR_m \) is set at 0.5 and different \( CR \) values conforming to the normal distribution are generated for each individual in the current population. These \( CR \) are modified after five generations and a new set of \( CR \) values is generated under the same normal distribution.

During every generation, the \( CR \) values associated with trial vectors successfully entering the next generation are recorded. After the specified \( LP \) number of generations the \( CR_m \) value is recalculated according to all the recorded \( CR \) values corresponding to successful trial vectors during \( LP \) period. With this new normal distribution’s mean and standard deviation 0.1, the above procedure is repeated. The iterative procedure is terminated when a predefined maximum number of iterations have been completed. The detail of parameter adaptation is given by Qin et al (2009).

3.4 CONSTRAINT HANDLING METHOD

The presence of constraints in EAs reduces the feasible region and complicates the search process. The major issue in constraint optimization process is how to deal with the infeasible solutions throughout the process.
The simplest and the earliest method of involving infeasible individuals in the search process, is the static penalty method. However, they usually require different parameters to be defined by the user to control the amount of penalty added when multiple constraints are violated. To overcome this difficulty, Self adaptive Penalty (SP) method has been recently suggested where information gathered from the search process is used to control the amount of penalty added to infeasible individuals (Coello Coello 2002).

3.4.1 Self-adaptive Penalty (SP)

Tessema and Yen (2006) proposed self-adaptive penalty method, in which two types of penalties are added to each infeasible individual to identify the best infeasible individual in the current population. The individual selection is based on a value determined by the overall constraint violation and objective values. Thus, there is a chance for an individual with lower overall constraint violation and higher fitness to get selected over a feasible individual with lower fitness. The number of feasible individuals currently presents in the combined population including both the parent and the offspring population controls the amount of the added penalties. This algorithm requires no parameter tuning. The final fitness value based on which the population members are ranked as given below by the Equations (3.8)-(3.10).

\[
F(X) = d(X) + p(X) \tag{3.8}
\]

where \(d(X)\) is the distance value and \(p(X)\) is the penalty value.

The distance value is given by

\[
d(X) = \begin{cases} 
  v(X), & \text{if } r_i = 0 \\
  \sqrt{f''(X)^2 + v(X)^2}, & \text{otherwise}
\end{cases} \tag{3.9}
\]
where \( r_f = \frac{\text{Number of feasible individuals}}{\text{population size}} \)

\( v(X) \) is the overall constraint violation given by 
\[
\sum_{i=1}^{m} \frac{w_t_i (G_i(X))}{\sum_{i=1}^{m} w_t_i},
\]

where \( w_t_i = \frac{1}{G_{\text{max}_i}} \) is a weight parameter, \( G_{\text{max}_i} \) is the maximum violation of constraint \( G_i(X). f''(X) = \frac{f(X) - f_{\text{min}}}{f_{\text{max}} - f_{\text{min}}} \) where, \( f_{\text{max}} \) and \( f_{\text{min}} \) are the maximum and minimum values of the objective function \( f(X) \) in the current combined population.

The penalty value is defined as
\[
p(X) = (1 - r_f) M(X) + r_f N(X) \tag{3.10}
\]

\[
M(X) = \begin{cases} 
0, & \text{if } r_f = 0 \\
v(X), & \text{otherwise}
\end{cases}
\]

\[
N(X) = \begin{cases} 
0, & \text{if } X \text{ is feasible} \\
f''(X), & \text{if } X \text{ is infeasible}
\end{cases}
\]

\( F(X) \) in Equation (3.8) is used to rank the individuals in the combined population. The top \( N \) individuals form the next generation.

### 3.5 COVARIANCE MATRIX ADAPTATION EVOLUTIONARY STRATEGY (CMAES)

CMAES was proposed by Nikolaus Hansen and Ostermeier in 1996. Evolutionary search algorithms like GA, DE and CMAES algorithms differ in selection, offspring generation and replacement mechanisms. CMAES is a class of continuous EA that generates new population members
from a probability distribution that is constructed during the optimization process. CMAES algorithm is derived from the concept of self-adaptation in evolution strategies, which adapts the covariance matrix of a multivariate normal search distribution. One of the key concepts of this algorithm involves the self-adaptation of learning of correlations between parameters and the use of the correlations to accelerate the convergence of the algorithm. Because of the learning process, the CMAES algorithm performs the search independent of the coordinate system, reliably adapts topologies of arbitrary functions, and significantly improves convergence rate especially on non-separable and/or badly scaled objective functions. This algorithm outperforms all other similar classes of learning algorithms on the benchmark multimodal functions (Hansen 2006). The adaptation mechanism of CMAES consists of two parts, 1) the adaptation of the covariance matrix $C$ and 2) the adaptation of the global step size $\sigma$. The covariance matrix $C$ is adapted by the evolution path $P_e$ and vector difference between the $\mu$ best individuals in the current and previous generation. The CMAES algorithm describes how $\sigma$ and $C$ are adapted. The detailed CMAES algorithm is explained in following steps.

### 3.5.1 CMAES algorithm

**Step 1:** Set the parameters cumulation for distribution ($c_c$), cumlation for step size ($c_\sigma$), change rate for the covariance matrix ($c_{cov}$) and distance value ($d$) as given by the Equations (3.11)-(3.14) to their default values.

\[
\begin{align*}
    c_\sigma &= \frac{\mu_{eff} + 2}{n + \mu_{eff} + 3} 
\end{align*}
\]
\[ d = 1 + 2 \max \left( 0, \frac{\sqrt{\mu_{\text{eff}} - 1}}{n + 1} - 1 \right) + c_0 \]  

(3.12)

Co-variance matrix adaptation is performed as per Equations (3.13) and (3.14).

\[ c_c = \frac{4}{n + 4} \]  

(3.13)

\[ c_{\text{cov}} = \frac{1}{\mu} \left( \frac{2}{n + \sqrt{2}} \right)^2 + \left( 1 - \frac{1}{\mu} \right) \min \left( 1, \frac{2\mu - 1}{(n + 2)^2 + \mu} \right) \]  

(3.14)

Step 2: Set the evolution path \( P_\sigma^{(0)} = 0 \), \( P_c^{(0)} = 0 \) and covariance matrix \( C(0) = I \). Choose step size \( \sigma(0) \) and frame the stopping criteria.

Step 3: Generate an initial random solution.

Step 4: The offspring at \( g + 1 \) generation \( x_k^{(g+1)} \) are sampled from a Gaussian distribution using covariance matrix and global step size at generation \( g \) is given by Equation (3.15).

\[ x_k^{(g+1)} = Z_k, \quad Z_k = N\left( \mu^{(g)}, \sigma^{(g)} C^{(g)} \right) \quad k = 1, \ldots, \lambda \]  

(3.15)

where \( \mu^{(g)} = \sum_{i=1}^{\mu} x_i^{(g)} \) with \( \mu \) being the selected best individuals from the population,

\( x_k^{(g+1)} \) is \( k \)-th offspring (search point) from generation \( g + 1 \), \( N(x, C) \) denotes a normally distributed random vector with mean \( x \) and covariance matrix \( C \), \( \lambda \) is the population size, \( \mu \) parent number.
Step 5: Adaptation of global step size $\sigma^{(g+1)}$ is based on an evolution path $p^{(g+1)}$ as per Equation (3.16).

$$p^{(g+1)} = (1 - c_\sigma) \cdot p^{(g)} + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}} \cdot (C^{(g)})^{-\frac{1}{2}} \left( \frac{\langle x^{(g+2)} \rangle - \langle x^{(g)} \rangle}{\sigma^{(g)}} \right)$$  (3.16)

The global step size $\sigma^{(g+1)}$ is determined by Equation (3.17).

$$\sigma^{(g+1)} = \sigma^{(g)} \exp \left( \frac{c_\sigma \| p^{(g+1)} \|}{d \left( E[\|N(0, I)\|] \right)} - 1 \right)$$ (3.17)

The damping factor $d$ can be adapted if maximum iteration number is small.

Step 6: The covariance matrix, $C$ adaptation is based on the evolution path $p^{(g+1)}_c$ given by Equation (3.18) and (3.19).

$$p^{(g+1)}_c = (1 - c_c) \cdot p^{(g)}_c + \sqrt{c_c (2 - c_c)} \cdot \frac{\sqrt{\mu}}{\sigma^{(g)}} \left( \frac{\langle x^{(g+1)} \rangle - \langle x^{(g)} \rangle}{\sigma^{(g)}} \right)$$  (3.18)

$$C^{(g+1)} = (1 - c_{\text{cov}}) \cdot C^{(g)} + c_{\text{cov}} \cdot \left\{ \frac{1}{\mu} p^{(g+1)}_c (p^{(g+1)}_c)^T + \left( 1 - \frac{1}{\mu} \sum_{i=1}^{\mu} \frac{1}{\sigma^{(g)}} (x^{(g+1)}_i - \langle x^{(g)} \rangle \langle x^{(g+1)} \rangle - \langle x^{(g)} \rangle^T \right) \right\}$$  (3.19)

The strategy parameter $c_{\text{cov}} \in [0, 1]$ determines the rate of change of the covariance matrix $C$.

Step 7: Repeat Steps 4-6 until the stopping criteria are satisfied.
The above steps are illustrated by the flowchart in Figure 3.2. The key points of CMAES algorithm are:

- **Estimation principle:** The CMAES algorithm estimates the distribution parameters from a set of selected steps. These steps avoid premature convergence and support explorative search behaviour.

- **Step size control:** Methods to estimate or adapt the covariance matrix do not achieve good overall step lengths. In the CMAES algorithm, the adaptation of the covariance matrix is complemented with step size control. The adjustment of the step size is based on a different adaptation principle.

- **Population size, adaptation, and change rates:** Choosing the population size $\lambda$ is always a compromise. Small $\lambda$ leads to faster convergence and large $\lambda$ helps to avoid local optima. To achieve a fast learning scheme for a covariance matrix
  
  i. The population size $\lambda$ must be comparatively small.

  ii. An adaptation procedure must be established, where parameters are updated rather than estimated from scratch in every generation.

Based on the above key points, the CMAES algorithm can improve the performance on ill-conditioned and/or non-separable problems by orders of magnitude, leaving the performance on simple problems unchanged (Hansen 2006, Baskar et al 2005).
3.6 CONCLUSION

The EAs like CMAES and SaDE are applied successfully for solving benchmark nonlinear optimization problems. In this thesis, for the optimal design of robust PID, FOPID and MPC controllers, the CMAES and SaDE algorithm are employed.