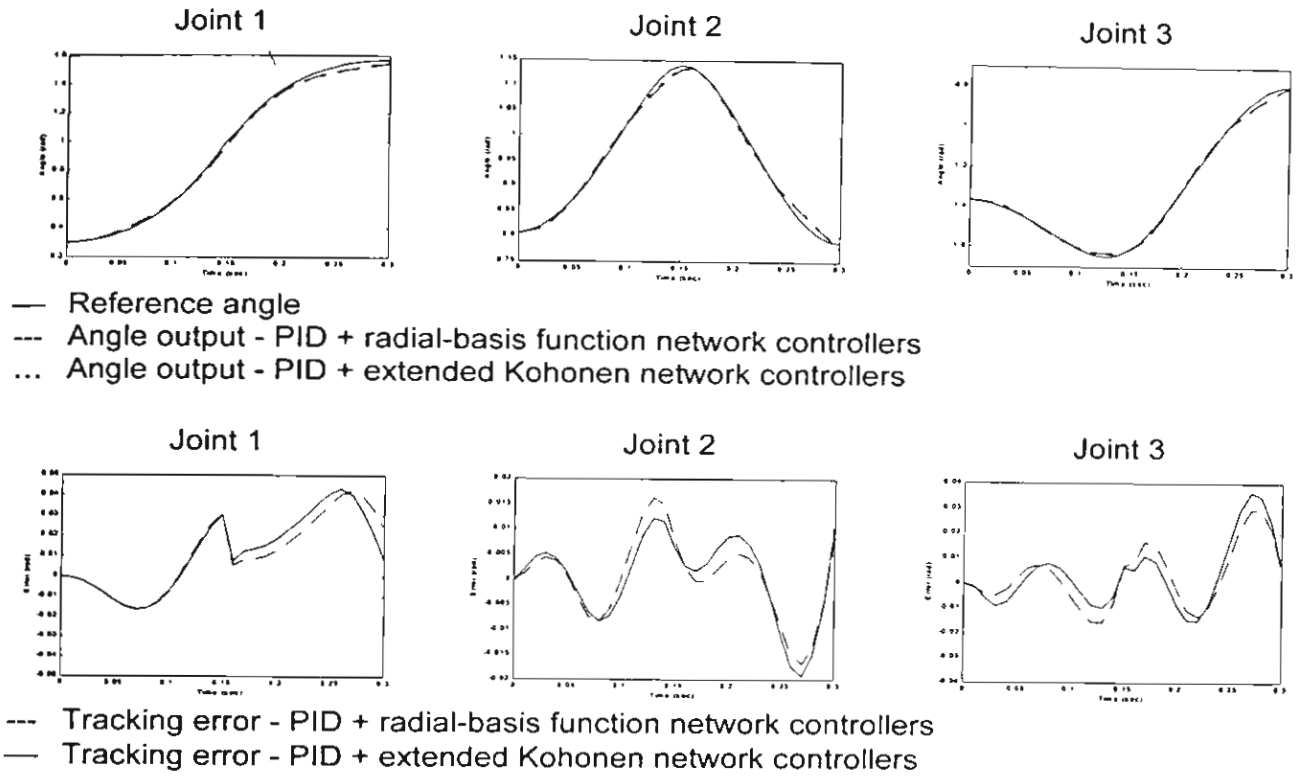


Figure 4.37 Angular positions from each joint and their tracking errors.

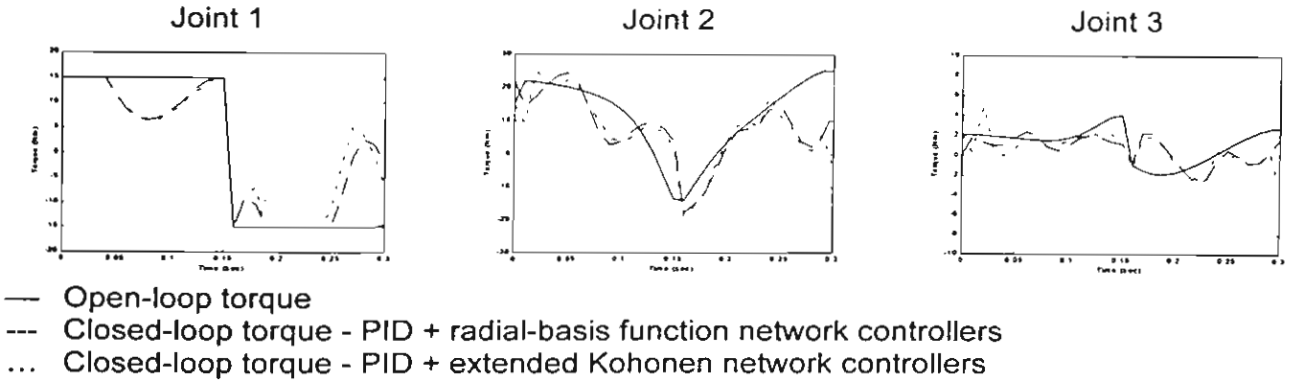


Figure 4.38 Open-loop and closed-loop torque on each joint.

B.4. Case 4 – 30 % Mass Loss on the Last Link

In the fourth test case, a 30 % loss of mass occurs on the last link. The mass of the last link has therefore changed from 1.0 kg to 0.7 kg. The simulation results for the cases of PID and radial-basis function network controllers with torque limits, and for the PID and extended Kohonen network controllers with torque limits are shown in Figures 4.39 and 4.40. The sum of the squared errors, the sum of the absolute errors and the maximum value of magnitude of the closed loop torque over the trajectory from each simulation are summarised in Table 4.20.

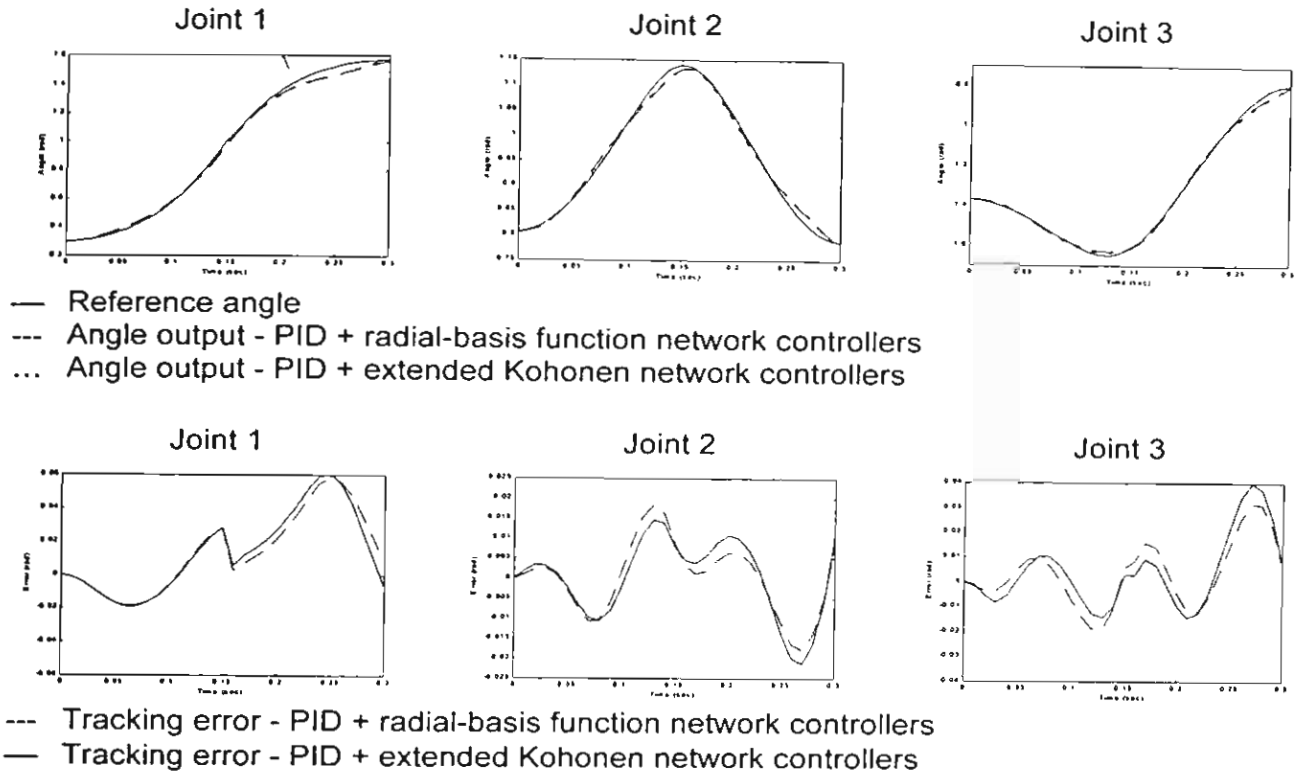


Figure 4.39 Angular positions from each joint and their tracking errors.

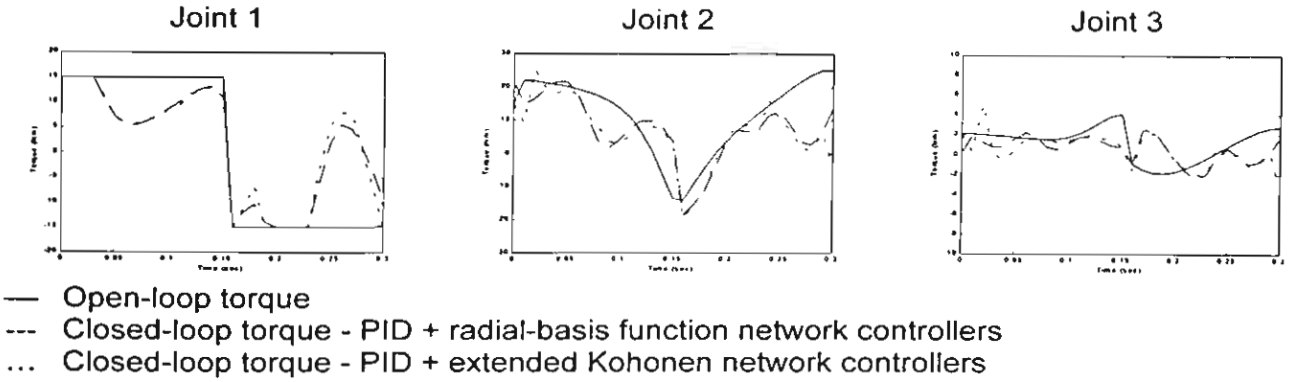


Figure 4.40 Open-loop and closed-loop torque on each joint.

Table 4.20 Summary of tracking errors and closed-loop torque.

Controller	Tracking Error		Magnitude of Closed-loop Torque		
	Squared	Absolute	Joint 1	Joint 2	Joint 3
PID + RBF network	0.0304	1.1929	15.0000	22.0439	2.5137
PID + Kohonen network	0.0332	1.2447	15.0000	25.0000	5.0000

B.5. Case 5 – 40 % Mass Loss on the Last Link

In the fifth test case, a 40 % loss of mass occurs on the last link. The mass of the last link has therefore changed from 1.0 kg to 0.6 kg. The simulation results for the cases of PID and radial-basis function network controllers with torque limits, and for the

PID and extended Kohonen network controllers with torque limits are shown in Figures 4.41 and 4.42. The sum of the squared errors, the sum of the absolute errors and the maximum value of magnitude of the closed loop torque over the trajectory from each simulation are summarised in Table 4.21.

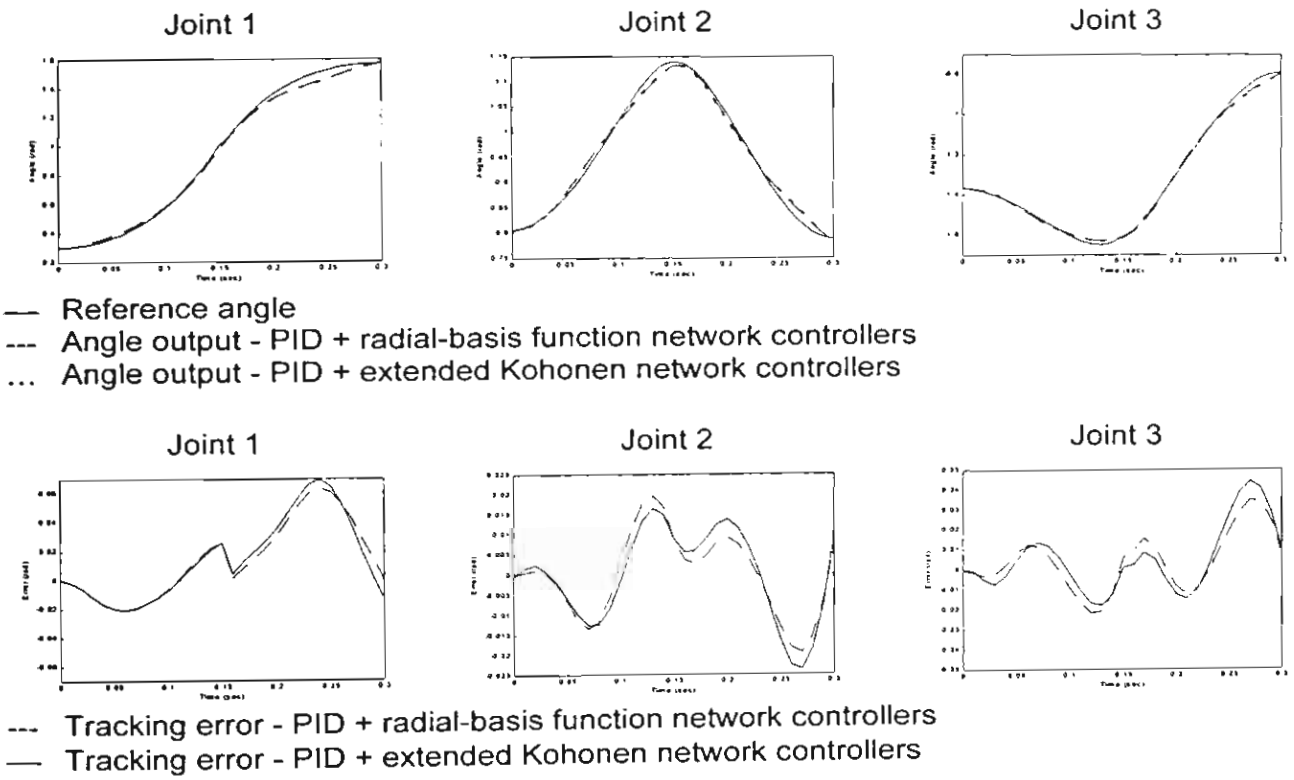


Figure 4.41 Angular positions from each joint and their tracking errors.

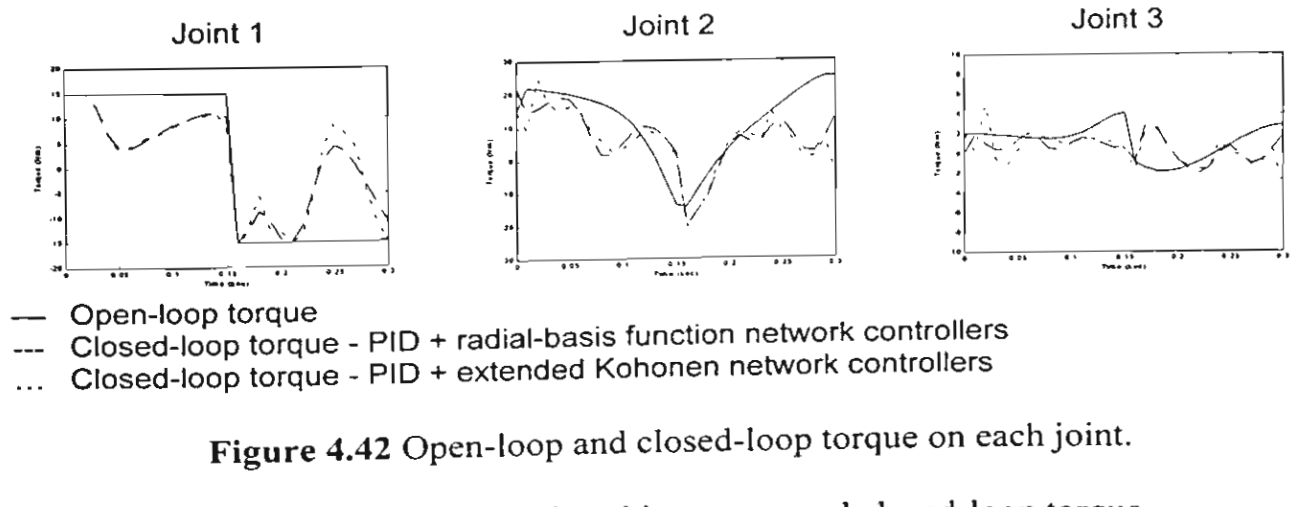


Figure 4.42 Open-loop and closed-loop torque on each joint.

Table 4.21 Summary of tracking errors and closed-loop torque.

Controller	Tracking Error		Magnitude of Closed-loop Torque		
	Squared	Absolute	Joint 1	Joint 2	Joint 3
PID + RBF network	0.0365	1.3096	15.0000	22.0439	2.7616
PID + Kohonen network	0.0422	1.3989	15.0000	25.0000	5.0000

B.6. Case 6 – 50 % Mass Loss on the Last Link

In the sixth test case, a 50 % loss of mass occurs on the last link. The mass of the last link has therefore changed from 1.0 kg to 0.5 kg. The simulation results for the cases of PID and radial-basis function network controllers with torque limits, and for the PID and extended Kohonen network controllers with torque limits are shown in Figures 4.43 and 4.44. The sum of the squared errors, the sum of the absolute errors and the maximum value of magnitude of the closed loop torque over the trajectory from each simulation are summarised in Table 4.22.

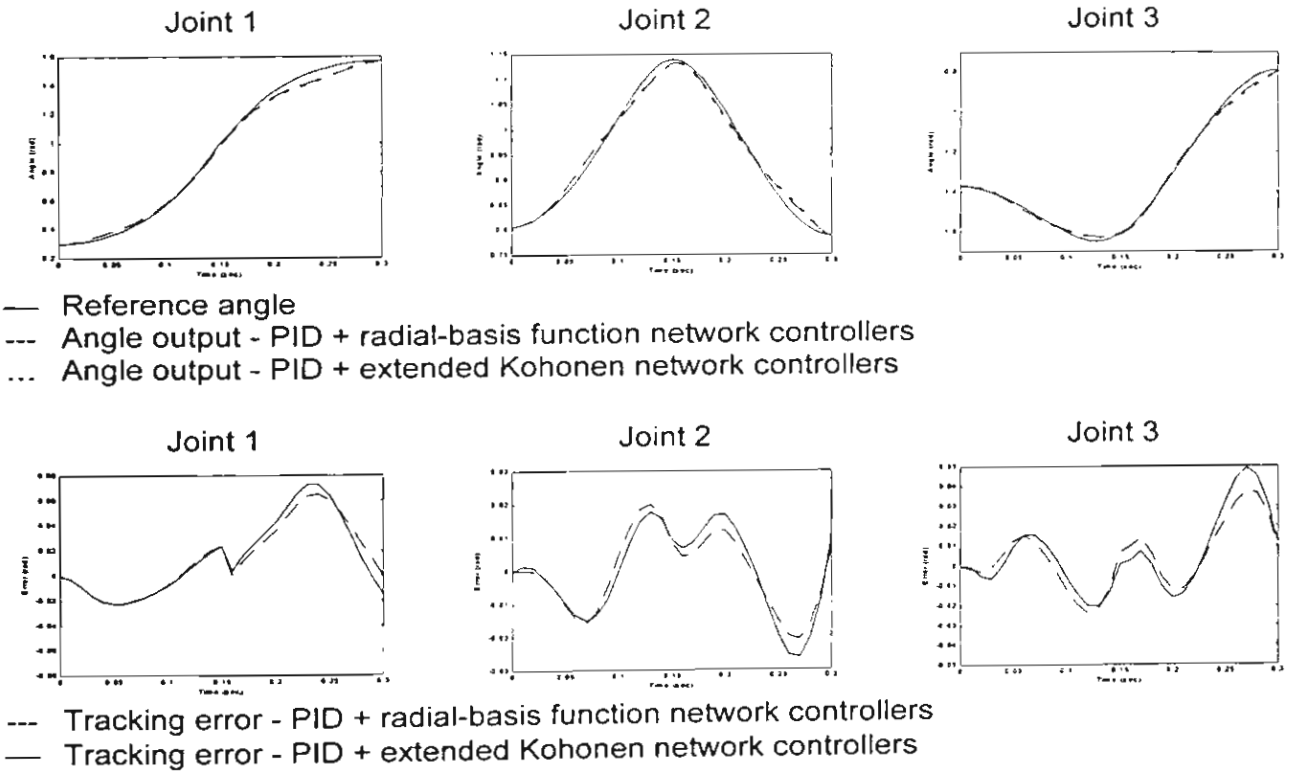


Figure 4.43 Angular positions from each joint and their tracking errors.

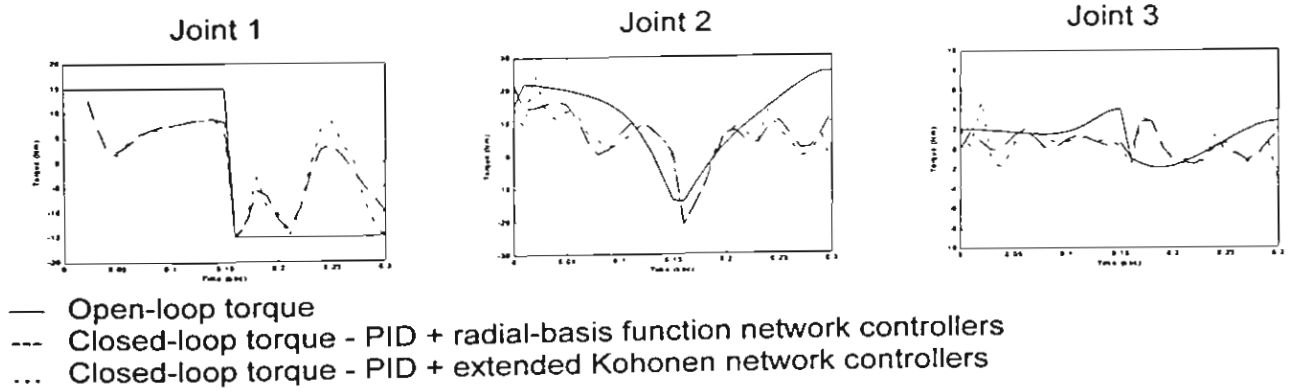


Figure 4.44 Open-loop and closed-loop torque on each joint.

Table 4.22 Summary of tracking errors and closed-loop torque.

Controller	Tracking Error		Magnitude of Closed-loop Torque		
	Squared	Absolute	Joint 1	Joint 2	Joint 3
PID + RBF network	0.0409	1.4249	15.0000	22.0439	3.1131
PID + Kohonen network	0.0502	1.5483	15.0000	25.0000	5.0000

B.7. Discussions on Results from Using the Radial-Basis Function Network and Extended Kohonen Network Controllers

From all six test cases, the command tracking performances of the robotic system where the radial-basis function networks and extended Kohonen networks are used as additional controllers are very much the same. This can be observed from the characteristics of the angular position and tracking error profiles. However, there are some differences between the characteristics of the closed-loop torque profile when the radial-basis function networks are used and that when the extended Kohonen networks are utilised. In addition, the summary of the sum of the squared and absolute tracking errors over the trajectory also indicates that the extended Kohonen network is slightly better than the radial-basis function network when it is used in the normal operating condition. In contrast, the radial-basis function network exhibits a slightly better compensation performance than the extended Kohonen network in the situation where modelling errors exist in the system. These differences can be explained as follows.

Regarding the differences in the closed-loop torque profile over the trajectory, although there are some differences between the closed-loop torque profiles from both cases of neural network controllers, both closed-loop torque profiles still remain close to the open-loop torque profile. This implies that the time-optimality requirement is met in both cases of neural network controllers. The differences in the closed-loop torque profile only confirm the nature of the robotic system in being a non-linear and multivariable system. In other words, there will be more than one set of control command sequences that can drive the system from the initial state to the final state in the state-space viewpoint.

Moving onto the summary on the tracking error results. It can be observed that when there is no mass loss from the robot arm both the sum of the squared and absolute tracking errors over the trajectory when the extended Kohonen network controllers are used are slightly better than those when the radial-basis function

networks are used. In contrast, once there are some modelling errors in the system, it can be seen that the tracking errors when the extended Kohonen network controllers are used are slightly higher than that when the radial-basis function networks are utilised. These results are caused by the differences in the network structure and the learning algorithm used. Recall that the output from a radial-basis function network is the weighted-sum of the signals from hidden neurons. This means that each control action of the radial-basis function network will be responsible by more than one neuron in the network. This also means that when a fault occurs in the control system, the fault tolerance load will be shared by a number of neurons. This makes the performance of the radial-basis function network remains high even when there is a large modelling error in the system. However, with the network structure like this, the learning process can also be a difficult one since the target output has to be achieved through the distribution of the adjustment of a number of connection weights in the output layer. It means that the control action produced by the radial-basis function network can be worse than that produced by the network which the target network output is achieved through an easier approach of adjusting one connection weight in the output layer. This will be the case here since the learning algorithm used to adjust the extended Kohonen network involves the adjustment of the connection weight of a neuron in the motor map which can be treated as a connection weight in the output layer. This leads to the command tracking performance of the extended Kohonen network being better than that of the radial-basis function network in the normal operating condition. Note that the mentioned normal operating condition is also the condition at which the neural network controllers are subjected to the process of learning. Following the same line of reasoning, since the control action produced by the extended Kohonen network is the direct result from the firing of a neuron in the state map, the fault tolerance load will only be directed to one neuron in the state map and another neuron in the motor map. Without load sharing as in the case of the radial-basis function network, it comes to no surprise that the fault tolerance performance of the extended Kohonen network would be worse than that of the radial-basis function network when modelling errors exist in the system.

The tracking errors of the system, expressed in terms of both the sum of the squared errors and the sum of the absolute errors, from all previously discussed simulation results are displayed in Figure 4.45.

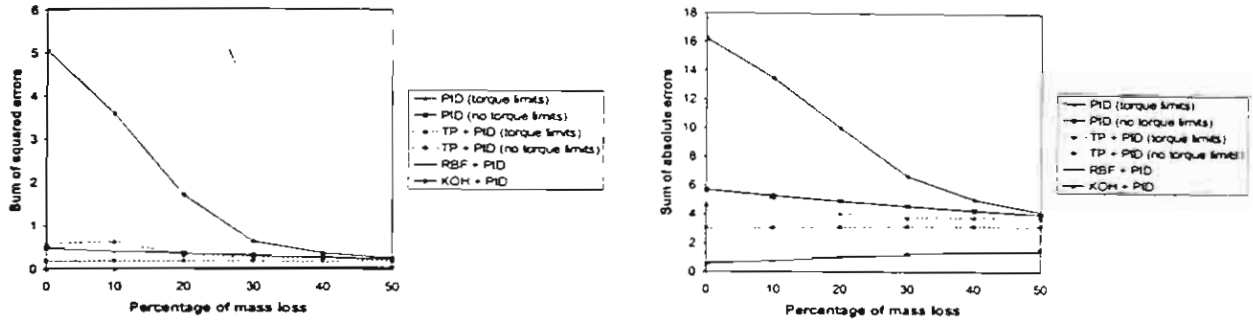


Figure 4.45 Tracking errors from all simulations involving modelling errors.

From Figure 4.45, it can be seen that trajectory pre-shaping can help to reduce tracking errors in the system where only PID controllers are used. However, both radial-basis function network and extended Kohonen network controllers are more effective than trajectory pre-shaping scheme in all cases. Note that the extended Kohonen network is more suitable to the time-optimal control task when the robot is operated under the normal condition while the radial-basis function network is more effective when there are modelling errors in the robotic system.

4.4. Time-Optimal Control Task II – Multi-Objective Optimisation Using a Genetic Algorithm Section

In practice, the maximum torque limits, which are used in the time-optimal trajectory calculation process for a closed-loop control, are usually less than the actual torque limits on the actuators. This safety precaution is done in order to allow some margins of error for possible discrepancies introduced to the system by modelling errors and controller dynamics (Shiller et al., 1996). This implies that for a given set of the actual torque limits of the actuators, there is a set of admissible torque limit combinations that can lead to a certain level of time-optimality within an acceptable range of tracking error. In addition, in certain applications such as welding or edge-deburring it is possible to modify the end-effector trajectory in Cartesian space without effecting the task requirement provided that the position and orientation of the work piece at which the end-effector has to remain in contact with can be modified accordingly. The

control task discussed in section 4.2 is an example which reflects such applications. By modifying the initial and final locations of the straight-line path, the task description in the application viewpoint would remain the same while the angular trajectory at which the robot joint has to follow would be different. Such change in the angular trajectory would lead to a variation in position tracking error. Combining with the issue on torque limits, this points to a design problem in robotic applications. The objective of such problem is to find a combination of torque limits from a set of admissible torque ranges and the initial and final position of the end-effector which leads to a trajectory which meets the time-optimality and tracking error constraints. A schematic diagram of the problem is illustrated in Figure 4.46.

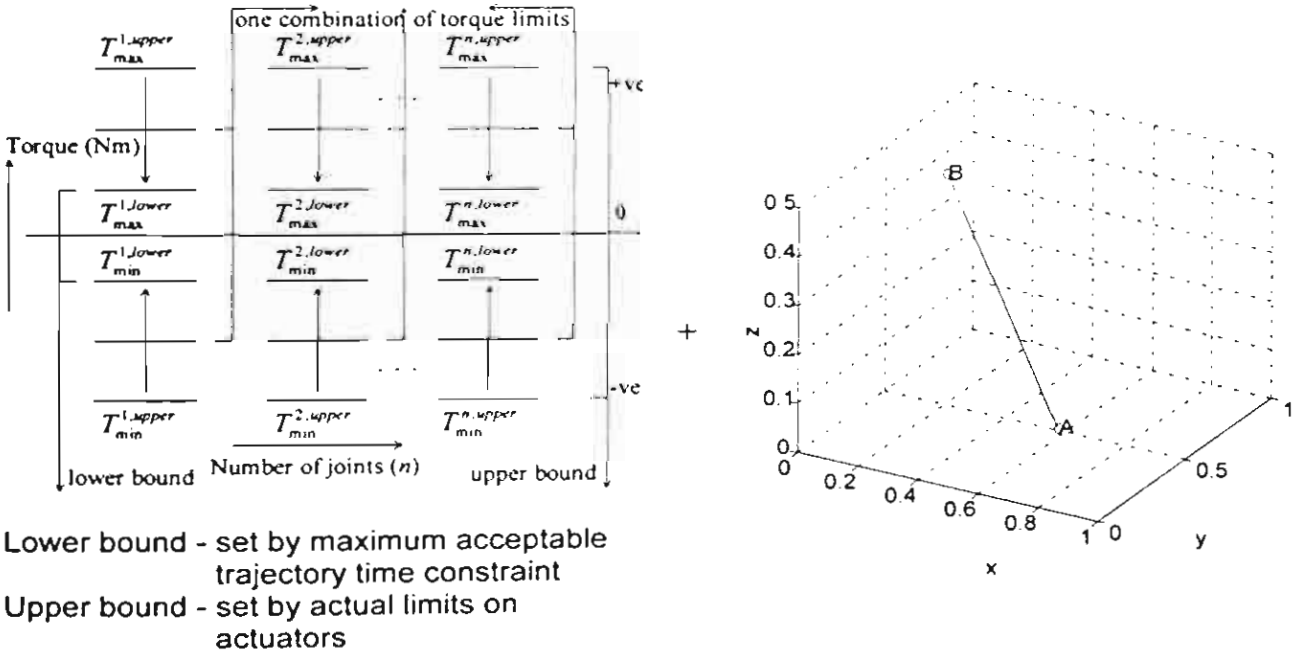


Figure 4.46 Schematic diagram illustrating torque limit combinations and end-effector positions.

With this arrangement, this problem will be a multi-objective optimisation problem since it would be highly unlikely to obtain a single trajectory that can minimise both the trajectory time and tracking error simultaneously. A multi-objective genetic algorithm (MOGA) will be used to solve the problem associated with the torque limit and end-effector position selection in this study. The problem formation and the genetic operators used are discussed as follows.

4.4.1. Decision Variables

A 3-dof robot with the task of tracking a straight-line path in Cartesian space presented earlier in section 4.2 is used to demonstrate this multi-objective optimisation problem. The decision variables of the problem consist of the torque limit combination and the initial and final position of the end-effector. Assuming that the magnitudes of the maximum and minimum torque limits are the same for each actuator, the torque limit part of the decision variables would consist of the magnitude of the torque limits of each joint. In this study, the range of the magnitudes of the torque limits on joints 1, 2 and 3 are set to 15-30, 25-40 and 5-20 Nm, respectively. The lower bounds of the limits (i.e. 15, 25, 5) are based on the maximum allowable trajectory time requirement of 0.3 seconds, while the upper bounds of the torque limits (i.e. 30, 40, 20) are set by the actual torque limits of the actuators.

Moving on to the part of decision variables which involves the positions of the end-effector. In order to create a fixed-length path in Cartesian space, two vectors are required: the position vector for the initial position of the end-effector and the direction vector pointing from the initial position toward the desired final position of the end-effector. This requirement can be achieved by setting up two search variables. The first variable will be the initial location of the end-effector while the second variable will be another point in the robot workspace at which a direction vector pointing from the initial position of the end-effector toward this point can be established. In this investigation the search range for the initial position of the end-effector is given by (0.721-0.751, 0.211-0.241, 0.078-0.108) in the x, y and z directions, respectively. In contrast, the search range for the location of the other point in the robot workspace is set to (-0.015-0.015, 0.839-0.869, 0.339-0.369) in the x, y and z directions, respectively. Note that the search ranges for these two points are in the vicinity of the initial and final positions of the straight-line path described earlier in section 4.2.

4.4.2. Objective Variables

There are two optimisation objective variables in this problem: the tracking error and trajectory time. The tracking error is expressed in terms of the sum of the mean

absolute value from the three joints, calculated over the whole trajectory. The tracking error objective function is given by

$$Tracking\ \ Error\ \ Objective = \sum_{i=1}^{3}\left(\frac{\sum_{j=1}^{N_f}\left|\theta_d^i(j)-\theta_i(j)\right|}{N_f}\right) \tag{4-17}$$

where $\theta_d^i(j)$ is the jth sample of the desired angular position of joint i, $\theta_i(j)$ is the jth sample of the actual angular position of joint i and N_f is the total number of samples of the time-optimal trajectory in discrete time. Moving onto the second objective variable - the trajectory time: the trajectory time is the optimal time obtained from the motion control algorithm described in section 3.3. Note that since the sampling period used in this simulation is 0.01 seconds, the trajectory time will always be in the form of $0.01m$, where m is a positive integer.

4.4.3. Chromosome Coding

Nine decision variables – the magnitudes of the torque limits from all three joints and the co-ordinates along three axes of the two points for identifying the straight-line path – are concatenated together and coded to form a chromosome. Two chromosome coding schemes are explored here: Gray and integer-based coding schemes. The torque ranges for all three joints are discretised using a search step of 0.5 Nm. This leaves 31 search points for the magnitude of the torque limits of each joint. In a similar way, the search ranges of the co-ordinates of the two points for dictating the location of the straight-line path are discretised using a search step of 0.001 m. This also leaves 31 search points for the co-ordinate in each axis. With the use of a Gray coding scheme, a Gray code of length 5 can be used to represent a decision variable. The total length of the chromosome in this case would be equal to 45. Note that there are certain search points obtained after decoding the chromosome which lie outside the required search space. These points are mapped back into the feasible region by changing the most significant bit of the Gray code section representing the particular decision variable that violates the feasibility constraint into zero. In contrast to the case of the Gray coding scheme, with the use of an integer-based coding system a

single gene can be used to represent a decision variable. Each gene can then take an allele value from a set which is composed of 31 integers ranging from 0 to 30. The chromosome length in this case would be equal to nine. A schematic diagram of the chromosome coding mechanisms is given in Figure 4.47.

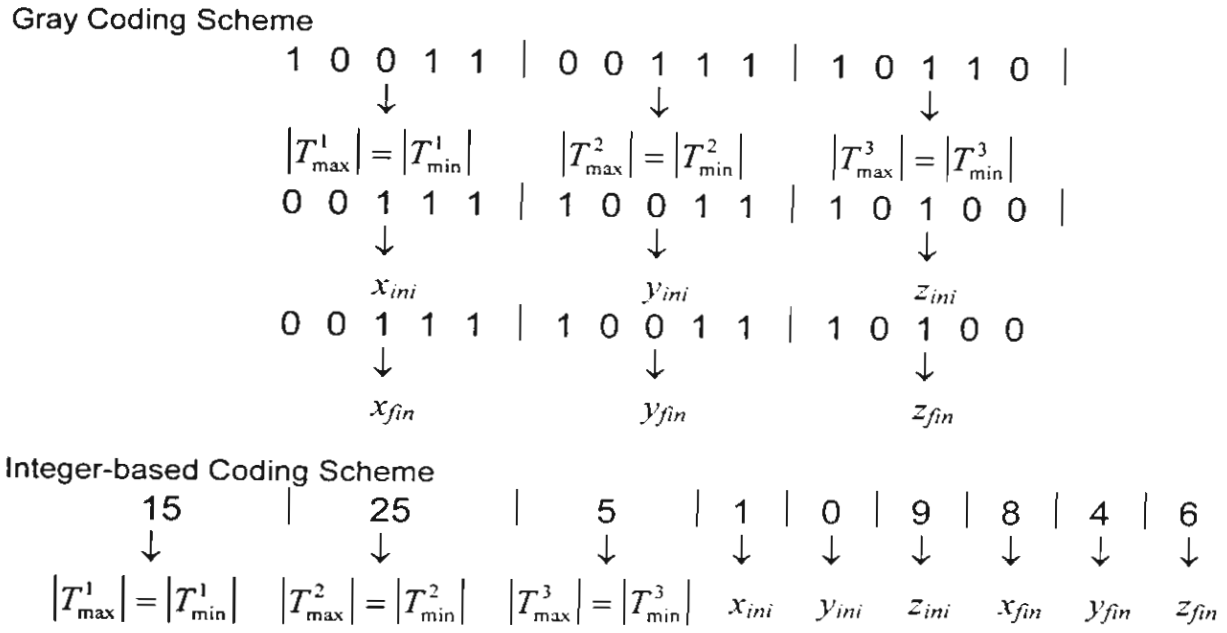


Figure 4.47 Schematic diagram of the chromosome coding mechanisms

4.4.4. Fitness Assignment and Fitness Sharing

The ranking method, as described by equation (3-32), is used to rank each individual in the population. Following that, a linear fitness interpolation is used to assign fitness to each individual. Fitness sharing, with the use of triangular sharing function, is then carried out in normalised objective space.

4.4.5. Selection Method

Stochastic universal sampling (Baker, 1989) is used in the fitness selection. The elitist strategy used is to select two individuals with the highest fitness and pass onto the next generation without crossover or mutation.

4.4.6. Crossover and Mutation Methods

The standard one-point crossover is used in the recombination. Two individuals are allowed to perform crossover if, and only if, they are within the mating restriction distance from each other. For simplicity, the mating restriction radius is set to equal to the sharing radius and the consideration on the distance between the two individuals is also done in normalised objective space. For the case of chromosome coding using a Gray code, a standard bit-flipped operation is used for the mutation. In contrast, the value 1 will be added to or subtracted from the allele value of the mutated gene to achieve mutation in the integer-based coding system. The parameter settings for the MOGA are summarised in Table 4.23.

Table 4.23 Parameter settings for the MOGA.

Parameter	Value
Chromosome length	
Gray code	45
Integer-based code	9
Crossover probability	0.8
Mutation probability	
Gray code	0.02
Integer-based code	0.1
Sharing and mating restriction radii	0.03
Population size	30
Number of elitist individuals	2
Number of generations	30

For the purpose of comparison, the random search technique is also used to find the Pareto optimal solutions in this study. Eschenauer et al. (1990) have explained that in the case of multi-objective optimisation, the random search method can generally be used to obtain a non-dominated solution set. In the random search technique, a set of random solutions is generated. Then, non-dominated solutions are picked from this solution set. This can be done by applying the ranking mechanism used in the MOGA to the initial random solutions and select solutions with rank 0. A number of multi-objective optimisation search techniques also use random search as an initial search procedure. For example, in the Monte Carlo method, after non-dominated solutions are found from the pool of initial solutions, one compromised solution is selected from this non-dominated solution set based on a min-max optimum criteria (Coello Coello and Christiansen, 1996). Since the MOGA is used in

this framework in order to obtain a non-dominated solution set, the random search will produce a good means for comparison in this aspect. However, since the genetic algorithm also uses randomly generated solutions as its initial search points, the random search has already been embedded into the genetic algorithm as the initial search procedure. This means that a comparison between the non-dominated solutions found from the initial population of the genetic algorithm and the non-dominated solutions obtained from the last generation of the genetic algorithm run would provide an adequate comparison in terms of the comparison with the random search method. Note that since the results from the random search method are represented by the results from the initial population of the genetic algorithm, the number of initial random solutions would be equal to the population size. The simulation results, with regards to this multi-objective optimisation problem, are discussed in the next section.

4.4.7. Simulation Results

Two case studies are investigated in the subsequent sections. The aim of the first case study is to find a set of torque limit combinations and straight-line paths which lead to trajectories with the sum of the mean absolute tracking errors ≤ 0.15708 radians (3 degrees per joint) and the trajectory time ≤ 0.27 seconds. The aim of the second case study is to find a set of torque limit combinations and straight-line paths which lead to trajectories with the sum of the mean absolute tracking errors ≤ 0.07854 radians (1.5 degrees per joint) and the trajectory time ≤ 0.30 seconds. The purpose of the first case study is to find solutions that concentrate more on optimising the trajectory time while the second case study emphasises on the tracking error optimisation. Within each case study, each simulation using a search technique is repeated five times with different initial guess solution sets. Hence, the displayed results will include the Pareto optimal solutions from each simulation run and the combined Pareto optimal solutions from all simulation runs. A brief description of the simulations within each case study is given in Table 4.24.

Table 4.24 Summary of the description of the simulations within each case study.

Section	Simulation Description
4.4.7.A.1	Simulation using random search (Case I)
4.4.7.A.2	Simulation using MOGA with a Gray-coding scheme (Case I)
4.4.7.A.3	Simulation using MOGA with an integer-based coding scheme (Case I)
4.4.7.A.4	Summary of simulation results from sections 4.4.7.A.1-3 (Case I)
4.4.7.B.1	Simulation using random search (Case II)
4.4.7.B.2	Simulation using MOGA with a Gray-coding scheme (Case II)
4.4.7.B.3	Simulation using MOGA with an integer-based coding scheme (Case II)
4.4.7.B.4	Summary of simulation results from sections 4.4.7.B.1-3 (Case II)

A. Case Study I

As mentioned earlier, the purpose of this case study is to find solutions which concentrate more on optimising the trajectory time. The results from the random search, the MOGA with a Gray-coding scheme and the MOGA with an integer-based coding scheme are as follows.

A.1. Random Search

The goal vector used to obtain non-dominated solutions from the initial random solutions is given by

$$\left[tracking\ error \quad trajectory\ time\right]^T = \left[0.15708 \quad 0.27\right]^T. \tag{4-18}$$

The Pareto optimal solutions and their objective values from five simulation runs are shown in Table 4.25.

Table 4.25 Pareto optimal solutions and their objective values from five simulation runs.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
1	26.5	39.5	11.5	0.723	0.222	0.092	0.014	0.855	0.366	0.11320	0.23
	24.0	34.5	16.0	0.721	0.229	0.084	0.003	0.858	0.346	0.10316	0.24
	21.5	36.5	16.0	0.748	0.221	0.081	-0.013	0.858	0.368	0.07753	0.25
	20.5	35.0	15.5	0.745	0.230	0.086	0.010	0.866	0.344	0.06561	0.26
2	27.0	38.5	20.0	0.724	0.220	0.080	-0.009	0.859	0.365	0.14410	0.22
	25.5	39.5	18.5	0.729	0.212	0.081	-0.008	0.857	0.369	0.10980	0.23
	24.5	34.5	16.5	0.733	0.212	0.078	-0.012	0.864	0.339	0.10335	0.24
	23.0	38.0	16.0	0.750	0.212	0.078	0.015	0.869	0.359	0.07003	0.25
	20.0	33.5	9.5	0.721	0.215	0.093	0.001	0.868	0.369	0.06623	0.26
	18.5	28.5	18.5	0.730	0.232	0.083	-0.007	0.844	0.354	0.05871	0.27

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

Table 4.25 Pareto optimal solutions and their objective values from five simulation runs (cont.).

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
3	28.0	39.5	12.0	0.739	0.221	0.082	-0.005	0.850	0.366	0.12512	0.22
	23.0	33.0	9.0	0.739	0.228	0.106	0.007	0.843	0.347	0.11207	0.24
	23.0	38.5	7.0	0.745	0.226	0.078	0.011	0.854	0.343	0.07316	0.25
	21.0	34.5	9.0	0.749	0.237	0.102	-0.006	0.850	0.343	0.05950	0.26
	19.0	34.5	14.0	0.730	0.214	0.106	0.006	0.858	0.364	0.05419	0.27
4	28.0	38.5	17.5	0.746	0.218	0.108	-0.014	0.846	0.363	0.12628	0.22
	26.0	36.0	20.0	0.731	0.234	0.101	-0.007	0.839	0.362	0.11665	0.23
	24.0	37.0	15.0	0.735	0.239	0.099	0.004	0.847	0.365	0.09527	0.24
	22.5	34.0	16.5	0.723	0.241	0.090	0.011	0.849	0.356	0.07671	0.25
	19.5	36.5	9.0	0.737	0.223	0.081	-0.005	0.851	0.339	0.07542	0.26
	18.5	34.5	13.5	0.732	0.212	0.106	0.014	0.851	0.349	0.05592	0.27
5	28.0	39.5	12.0	0.739	0.221	0.082	-0.005	0.850	0.366	0.12512	0.22
	25.5	39.5	18.5	0.729	0.212	0.081	-0.008	0.857	0.369	0.10980	0.23
	24.0	39.0	16.5	0.736	0.235	0.099	-0.003	0.852	0.368	0.09433	0.24
	23.0	38.0	16.0	0.750	0.212	0.078	0.015	0.869	0.359	0.07003	0.25
	21.0	34.5	9.0	0.749	0.237	0.102	-0.006	0.850	0.343	0.05950	0.26
	19.0	34.0	7.0	0.742	0.214	0.104	-0.015	0.865	0.348	0.05298	0.27

Table 4.26 Combined Pareto optimal solutions and their objective values.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
-	28.0	39.5	12.0	0.739	0.221	0.082	-0.005	0.850	0.366	0.12512	0.22
	25.5	39.5	18.5	0.729	0.212	0.081	-0.008	0.857	0.369	0.10980	0.23
	24.0	39.0	16.5	0.736	0.235	0.099	-0.003	0.852	0.368	0.09433	0.24
	23.0	38.0	16.0	0.750	0.212	0.078	0.015	0.869	0.359	0.07003	0.25
	21.0	34.5	9.0	0.749	0.237	0.102	-0.006	0.850	0.343	0.05950	0.26
	19.0	34.0	7.0	0.742	0.214	0.104	-0.015	0.865	0.348	0.05298	0.27

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

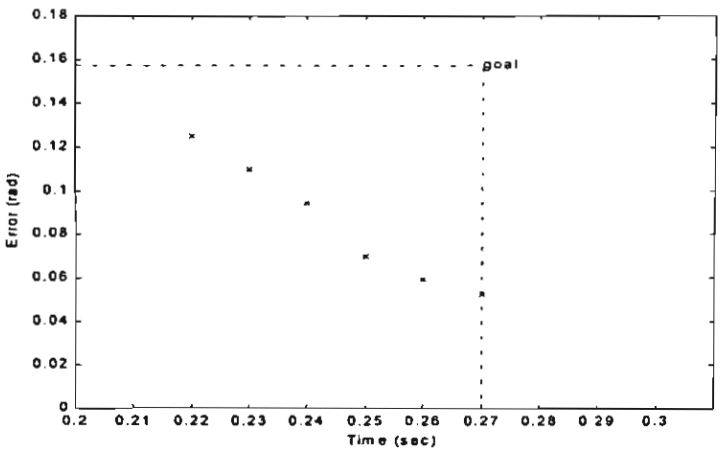


Figure 4.48 Trade-off surface of the Pareto front.

The combined Pareto optimal solutions and their objective values from all simulation runs are summarised in Table 4.26 while the trade-off surface of the Pareto front is given in Figure 4.48.

A.2. MOGA with a Gray Coding Scheme

The goal vector used to obtain the non-dominated solutions is the same as the one presented in the random search approach. The Pareto optimal solutions and their objective values from five simulation runs are shown in Table 4.27.

Table 4.27 Pareto optimal solutions and their objective values from five simulation runs.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
1	28.5	38.5	12.0	0.740	0.221	0.078	0.012	0.841	0.361	0.12481	0.22
	26.0	37.5	12.0	0.749	0.220	0.094	0.011	0.840	0.355	0.11127	0.23
	23.5	37.5	12.0	0.740	0.216	0.078	0.011	0.844	0.363	0.09043	0.24
	22.5	37.5	14.5	0.741	0.233	0.092	0.005	0.851	0.358	0.07112	0.25
	20.5	35.0	15.0	0.748	0.230	0.082	0.002	0.849	0.352	0.05886	0.26
	19.5	35.0	11.0	0.748	0.227	0.081	0.008	0.859	0.341	0.04156	0.27
2	26.0	38.0	9.0	0.750	0.214	0.088	-0.004	0.858	0.363	0.10090	0.23
	24.5	36.5	17.0	0.748	0.211	0.086	0.012	0.865	0.341	0.08945	0.24
	22.5	38.5	7.5	0.726	0.213	0.082	0.012	0.861	0.348	0.07355	0.25
	21.0	37.0	8.0	0.749	0.219	0.085	0.011	0.865	0.340	0.05544	0.26
	19.5	29.5	8.5	0.749	0.234	0.086	0.012	0.855	0.356	0.04902	0.27
3	28.0	39.5	11.5	0.745	0.214	0.079	-0.002	0.851	0.367	0.12442	0.22
	25.5	39.5	11.5	0.745	0.214	0.084	-0.001	0.850	0.367	0.10600	0.23
	24.0	39.5	13.0	0.740	0.224	0.090	0.006	0.847	0.343	0.09009	0.24
	23.5	39.5	12.5	0.739	0.235	0.078	0.007	0.867	0.339	0.06801	0.25
	20.5	39.5	17.5	0.750	0.218	0.081	-0.004	0.853	0.344	0.05835	0.26
	19.0	34.5	8.5	0.750	0.216	0.079	-0.007	0.854	0.350	0.04239	0.27
4	28.5	38.5	18.0	0.747	0.215	0.079	-0.010	0.858	0.346	0.11797	0.22
	26.5	36.5	18.5	0.750	0.219	0.078	-0.005	0.867	0.354	0.10205	0.23
	25.0	35.0	15.0	0.750	0.219	0.078	-0.005	0.867	0.354	0.08179	0.24
	23.0	33.0	7.5	0.751	0.226	0.081	0.006	0.857	0.346	0.07602	0.25
	21.0	35.0	9.0	0.751	0.240	0.083	0.003	0.863	0.348	0.05714	0.26
	19.0	39.0	17.0	0.744	0.219	0.082	0.001	0.862	0.354	0.04467	0.27
5	30.0	40.0	14.0	0.735	0.214	0.106	-0.005	0.849	0.343	0.14613	0.21
	29.0	39.0	17.5	0.751	0.232	0.078	0.015	0.849	0.354	0.11324	0.22
	26.0	36.0	9.5	0.751	0.229	0.078	-0.015	0.866	0.360	0.10753	0.23
	24.0	39.0	11.0	0.738	0.228	0.083	-0.005	0.846	0.352	0.08989	0.24
	22.0	39.0	16.5	0.731	0.220	0.099	-0.005	0.853	0.358	0.07633	0.25
	20.5	37.0	7.5	0.751	0.217	0.094	0.000	0.869	0.364	0.06043	0.26
	19.5	29.5	18.5	0.746	0.239	0.078	0.015	0.860	0.342	0.04392	0.27

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

The combined Pareto optimal solutions and their objective values from all simulation runs are summarised in Table 4.28 while the trade-off surface of the Pareto front is given in Figure 4.49.

Table 4.28 Combined Pareto optimal solutions and their objective values.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
-	30.0	40.0	14.0	0.735	0.214	0.106	-0.005	0.849	0.343	0.14613	0.21
	29.0	39.0	17.5	0.751	0.232	0.078	0.015	0.849	0.354	0.11324	0.22
	26.0	38.0	9.0	0.750	0.214	0.088	-0.004	0.858	0.363	0.10090	0.23
	25.0	35.0	15.0	0.750	0.219	0.078	-0.005	0.867	0.354	0.08179	0.24
	23.5	39.5	12.5	0.739	0.235	0.078	0.007	0.867	0.339	0.06801	0.25
	21.0	37.0	8.0	0.749	0.219	0.085	0.011	0.865	0.340	0.05544	0.26
	19.5	35.0	11.0	0.748	0.227	0.081	0.008	0.859	0.341	0.04156	0.27

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

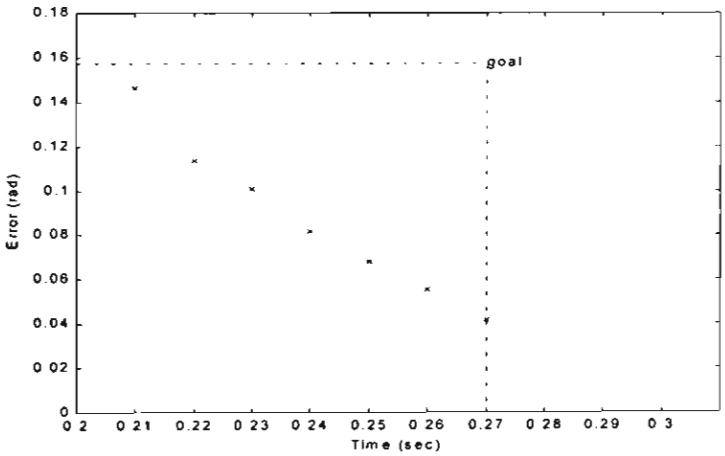


Figure 4.49 Trade-off surface of the Pareto front.

A.3. MOGA with an Integer-Based Coding Scheme

The goal vector used to obtain the non-dominated solutions is the same as the one presented in the random search approach. The Pareto optimal solutions and their objective values from five simulation runs are shown in Table 4.29. The combined Pareto optimal solutions and their objective values from all simulation runs are summarised in Table 4.30 while the trade-off surface of the Pareto front is given in Figure 4.50.

Table 4.29 Pareto optimal solutions and their objective values from five simulation runs.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
1	27.0	40.0	11.0	0.724	0.221	0.093	0.011	0.855	0.367	0.14643	0.22
	26.0	40.0	11.0	0.725	0.220	0.093	0.011	0.855	0.369	0.11077	0.23
	24.0	40.0	19.0	0.748	0.219	0.078	0.013	0.846	0.369	0.08789	0.24
	22.0	39.0	18.5	0.746	0.221	0.078	0.014	0.843	0.367	0.07072	0.25
	21.0	35.0	16.5	0.749	0.219	0.084	0.010	0.865	0.345	0.05415	0.26
	21.0	35.0	17.0	0.749	0.219	0.084	0.010	0.865	0.345	0.05415	0.26
	19.5	34.5	15.0	0.742	0.224	0.084	0.009	0.865	0.346	0.04475	0.27
2	28.0	39.5	20.0	0.724	0.220	0.079	-0.012	0.860	0.367	0.12774	0.22
	27.5	39.0	20.0	0.724	0.221	0.079	-0.011	0.859	0.366	0.10086	0.23
	24.5	35.5	11.0	0.750	0.212	0.078	0.015	0.859	0.340	0.08856	0.24
	23.5	37.5	11.0	0.751	0.232	0.078	0.015	0.858	0.340	0.06955	0.25
	23.5	37.5	17.5	0.751	0.232	0.078	0.015	0.858	0.340	0.06955	0.25
	20.0	35.0	10.5	0.737	0.220	0.082	0.009	0.841	0.353	0.06088	0.26
	19.5	34.5	12.0	0.724	0.236	0.078	0.013	0.869	0.359	0.04084	0.27
3	28.5	40.0	14.0	0.741	0.223	0.079	0.010	0.844	0.363	0.12255	0.22
	28.5	40.0	9.5	0.735	0.223	0.078	0.010	0.844	0.362	0.10552	0.23
	24.0	40.0	9.5	0.736	0.224	0.078	0.010	0.844	0.364	0.08860	0.24
	23.0	40.0	7.5	0.749	0.226	0.078	0.010	0.855	0.342	0.07036	0.25
	21.0	36.5	10.0	0.747	0.238	0.103	-0.005	0.851	0.349	0.05760	0.26
	19.5	36.0	13.5	0.750	0.236	0.101	-0.007	0.851	0.342	0.04385	0.27
4	29.5	40.0	18.0	0.744	0.218	0.108	-0.015	0.843	0.349	0.14403	0.21
	28.0	38.0	18.5	0.746	0.218	0.108	-0.013	0.848	0.353	0.12338	0.22
	26.5	40.0	15.5	0.733	0.237	0.088	-0.005	0.841	0.352	0.10994	0.23
	26.5	40.0	16.0	0.733	0.237	0.088	-0.005	0.841	0.352	0.10994	0.23
	24.0	38.0	13.5	0.733	0.239	0.097	0.006	0.846	0.341	0.09189	0.24
	22.5	34.0	17.0	0.723	0.227	0.080	-0.005	0.851	0.343	0.07423	0.25
	20.5	38.0	8.5	0.735	0.224	0.080	-0.004	0.852	0.340	0.06077	0.26
	20.5	38.0	9.0	0.735	0.224	0.080	-0.004	0.852	0.340	0.06077	0.26
	19.5	35.5	9.5	0.735	0.239	0.092	0.007	0.865	0.342	0.04257	0.27
5	29.5	39.5	12.5	0.722	0.224	0.079	0.008	0.847	0.341	0.12677	0.22
	27.0	37.0	18.0	0.751	0.240	0.078	0.011	0.847	0.342	0.10061	0.23
	24.0	39.5	16.5	0.737	0.235	0.082	-0.015	0.851	0.344	0.08914	0.24
	22.0	40.0	10.5	0.732	0.237	0.081	-0.014	0.852	0.367	0.07279	0.25
	20.5	30.5	18.0	0.748	0.230	0.100	-0.007	0.868	0.369	0.06260	0.26
	19.0	33.5	7.5	0.739	0.236	0.089	-0.006	0.862	0.345	0.04770	0.27

Table 4.30 Combined Pareto optimal solutions and their objective values.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
-	29.5	40.0	18.0	0.744	0.218	0.108	-0.015	0.843	0.349	0.14403	0.21
	28.5	40.0	14.0	0.741	0.223	0.079	0.010	0.844	0.363	0.12255	0.22
	27.0	37.0	18.0	0.751	0.240	0.078	0.011	0.847	0.342	0.10061	0.23
	24.0	40.0	19.0	0.748	0.219	0.078	0.013	0.846	0.369	0.08789	0.24
	23.5	37.5	17.5	0.751	0.232	0.078	0.015	0.858	0.340	0.06955	0.25
	21.0	35.0	16.5	0.749	0.219	0.084	0.010	0.865	0.345	0.05415	0.26
	19.5	34.5	12.0	0.724	0.236	0.078	0.013	0.869	0.359	0.04084	0.27

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

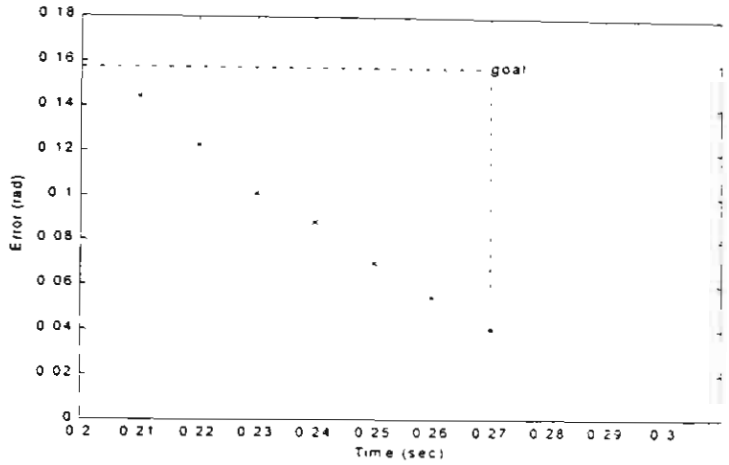


Figure 4.50 Trade-off surface of the Pareto front.

A.4. Summary of Simulation Results from Case Study I

Results from the random search, the MOGA with a Gray coding scheme, and the MOGA with an integer-based coding scheme are summarised in Table 4.31.

Table 4.31 Summary of results from the three approaches.

Approach	N_d	N_n
Random search	6	0
MOGA with a Gray coding scheme	7	3
MOGA with an integer-based coding scheme	7	4

N_d - Number of distinct solutions found

N_n - Number of solutions which are not dominated by solutions found by other techniques

B. Case Study II

As stated earlier, the purpose of this case study is to find solutions that concentrate more on optimising the tracking error objective. The results from the random search, the MOGA with a Gray coding scheme and the MOGA with an integer-based coding scheme are as follows.

B.1. Random Search

The goal vector used to obtain non-dominated solutions from the initial random solutions is given by

$$[tracking\ error \quad trajectory\ time]^T = [0.07854 \quad 0.30]^T . \qquad (4\text{-}19)$$

The Pareto optimal solutions and their objective values from five simulation runs are shown in Table 4.32. The combined Pareto optimal solutions and their objective values from all simulation runs are summarised in Table 4.33 while the trade-off surface of the Pareto front is given in Figure 4.51.

Table 4.32 Pareto optimal solutions and their objective values from five simulation runs.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
1	21.5	36.5	16.0	0.748	0.221	0.081	-0.013	0.858	0.368	0.07753	0.25
	20.5	35.0	15.5	0.745	0.230	0.086	0.010	0.866	0.344	0.06561	0.26
	17.5	27.5	15.0	0.732	0.222	0.083	0.003	0.861	0.363	0.04325	0.28
	16.5	28.5	8.5	0.742	0.213	0.084	0.010	0.862	0.340	0.02312	0.29
2	23.0	38.0	16.0	0.750	0.212	0.078	0.015	0.869	0.359	0.07003	0.25
	20.0	33.5	9.5	0.721	0.215	0.093	0.001	0.868	0.369	0.06623	0.26
	18.5	28.5	18.5	0.730	0.232	0.083	-0.007	0.844	0.354	0.05871	0.27
	17.0	37.0	7.0	0.722	0.216	0.100	-0.006	0.862	0.369	0.04744	0.28
	16.5	26.5	15.5	0.727	0.241	0.089	0.012	0.851	0.340	0.03249	0.29
	15.5	35.5	13.5	0.751	0.218	0.096	0.002	0.861	0.340	0.02811	0.30
3	23.0	38.5	7.0	0.745	0.226	0.078	0.011	0.854	0.343	0.07316	0.25
	21.0	34.5	9.0	0.749	0.237	0.102	-0.006	0.850	0.343	0.05950	0.26
	19.0	34.5	14.0	0.730	0.214	0.106	0.006	0.858	0.364	0.05419	0.27
	18.0	37.5	7.0	0.750	0.228	0.106	0.014	0.851	0.357	0.03750	0.28
	16.5	26.5	11.5	0.744	0.222	0.094	0.011	0.858	0.339	0.02661	0.29
	15.0	33.0	12.5	0.745	0.236	0.106	-0.014	0.862	0.365	0.02235	0.30
4	22.5	34.0	16.5	0.723	0.241	0.090	0.011	0.849	0.356	0.07671	0.25
	19.5	36.5	9.0	0.737	0.223	0.081	-0.005	0.851	0.339	0.07542	0.26
	18.5	34.5	13.5	0.732	0.212	0.106	0.014	0.851	0.349	0.05592	0.27
	17.0	33.0	8.5	0.725	0.225	0.105	-0.014	0.854	0.354	0.04114	0.28
	16.5	26.5	14.5	0.733	0.240	0.106	0.002	0.850	0.350	0.04070	0.29
	15.0	32.0	19.5	0.740	0.218	0.083	0.008	0.849	0.347	0.02224	0.30
5	22.0	39.5	9.0	0.731	0.233	0.089	-0.006	0.839	0.366	0.07506	0.25
	20.5	30.5	18.0	0.748	0.231	0.098	-0.015	0.854	0.340	0.06408	0.26
	19.0	34.0	7.0	0.742	0.214	0.104	-0.015	0.865	0.348	0.05298	0.27
	17.5	35.5	8.5	0.725	0.238	0.094	0.000	0.847	0.366	0.03582	0.28
	16.5	36.0	17.5	0.751	0.238	0.078	0.011	0.848	0.341	0.02458	0.29

Table 4.33 Combined Pareto optimal solutions and their objective values.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
-	23.0	38.0	16.0	0.750	0.212	0.078	0.015	0.869	0.359	0.07003	0.25
	21.0	34.5	9.0	0.749	0.237	0.102	-0.006	0.850	0.343	0.05950	0.26
	19.0	34.0	7.0	0.742	0.214	0.104	-0.015	0.865	0.348	0.05298	0.27
	17.5	35.5	8.5	0.725	0.238	0.094	0.000	0.847	0.366	0.03582	0.28
	16.5	28.5	8.5	0.742	0.213	0.084	0.010	0.862	0.340	0.02312	0.29
	15.0	32.0	19.5	0.740	0.218	0.083	0.008	0.849	0.347	0.02224	0.30

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

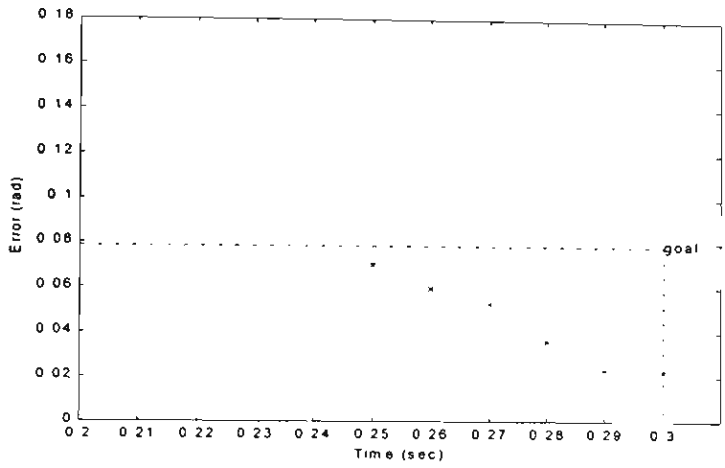


Figure 4.51 Trade-off surface of the Pareto front.

B.2. MOGA with a Gray Coding Scheme

The goal vector used to obtain the non-dominated solutions is the same as the one presented in the random search approach. The Pareto optimal solutions and their objective values from five simulation runs are shown in Table 4.34. The combined Pareto optimal solutions and their objective values from all simulation runs are summarised in Table 4.35 while the trade-off surface of the Pareto front is given in Figure 4.52.

Table 4.34 Pareto optimal solutions and their objective values from five simulation runs.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
1	22.0	37.0	14.0	0.729	0.232	0.088	-0.002	0.849	0.352	0.07582	0.25
	20.5	38.0	15.5	0.744	0.227	0.103	0.011	0.844	0.351	0.06070	0.26
	19.0	39.0	15.0	0.748	0.231	0.088	-0.009	0.843	0.349	0.04280	0.27
	17.5	27.5	15.0	0.733	0.220	0.080	-0.002	0.860	0.346	0.04127	0.28
	16.5	34.0	11.0	0.727	0.234	0.084	0.010	0.863	0.347	0.02182	0.29
2	22.5	36.0	10.0	0.736	0.233	0.079	-0.013	0.857	0.347	0.07310	0.25
	20.5	36.5	10.0	0.736	0.233	0.079	-0.013	0.857	0.347	0.05837	0.26
	19.5	33.5	11.5	0.721	0.238	0.078	-0.005	0.864	0.359	0.05047	0.27
	17.5	36.0	20.0	0.736	0.215	0.090	-0.003	0.864	0.366	0.03151	0.28
	16.5	30.0	15.0	0.723	0.240	0.082	0.002	0.860	0.339	0.02202	0.29
	15.5	34.5	16.0	0.744	0.237	0.107	0.009	0.861	0.344	0.01703	0.30
	15.5	34.5	16.0	0.744	0.238	0.107	0.009	0.862	0.343	0.01703	0.30

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

Table 4.34 Pareto optimal solutions and their objective values from five simulation runs (cont.).

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
3	22.5	38.0	15.0	0.733	0.226	0.079	0.000	0.852	0.341	0.07202	0.25
	20.5	39.0	18.0	0.750	0.233	0.078	-0.006	0.850	0.350	0.05799	0.26
	19.0	39.0	16.0	0.721	0.222	0.101	0.007	0.839	0.346	0.05390	0.27
	17.5	39.5	16.5	0.731	0.234	0.083	0.000	0.846	0.359	0.03104	0.28
	16.0	29.0	10.5	0.730	0.234	0.086	-0.003	0.843	0.355	0.02362	0.29
	15.0	36.0	19.0	0.734	0.227	0.108	0.000	0.851	0.351	0.01668	0.30
4	22.5	39.5	17.0	0.739	0.235	0.081	-0.010	0.866	0.359	0.07145	0.25
	21.0	32.5	6.5	0.749	0.237	0.104	-0.011	0.864	0.361	0.05850	0.26
	19.5	34.0	8.5	0.747	0.223	0.081	0.006	0.868	0.346	0.04235	0.27
	18.0	35.0	17.0	0.750	0.230	0.079	0.014	0.854	0.346	0.03210	0.28
	16.5	36.5	8.0	0.721	0.238	0.078	0.011	0.850	0.339	0.02311	0.29
	15.0	36.0	7.5	0.734	0.234	0.104	0.001	0.849	0.356	0.01710	0.30
5	22.0	39.5	10.5	0.732	0.234	0.080	-0.007	0.845	0.362	0.07252	0.25
	20.5	33.5	16.0	0.746	0.235	0.107	-0.010	0.861	0.353	0.06198	0.26
	19.0	38.0	16.0	0.739	0.239	0.091	-0.002	0.847	0.353	0.04591	0.27
	17.5	35.0	7.5	0.726	0.235	0.094	-0.015	0.847	0.344	0.03401	0.28
	16.5	32.0	16.0	0.751	0.234	0.104	-0.010	0.856	0.340	0.02226	0.29
	15.0	40.0	6.0	0.730	0.237	0.104	-0.015	0.862	0.367	0.01649	0.30

Table 4.35 Combined Pareto optimal solutions and their objective values.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
-	22.5	39.5	17.0	0.739	0.235	0.081	-0.010	0.866	0.359	0.07145	0.25
	20.5	39.0	18.0	0.750	0.233	0.078	-0.006	0.850	0.350	0.05799	0.26
	19.5	34.0	8.5	0.747	0.223	0.081	0.006	0.868	0.346	0.04235	0.27
	17.5	39.5	16.5	0.731	0.234	0.083	0.000	0.846	0.359	0.03104	0.28
	16.5	34.0	11.0	0.727	0.234	0.084	0.010	0.863	0.347	0.02182	0.29
	15.0	40.0	6.0	0.730	0.237	0.104	-0.015	0.862	0.367	0.01649	0.30

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

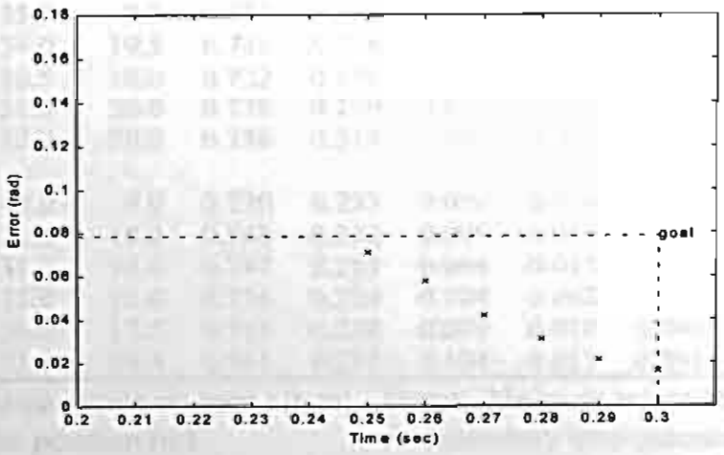


Figure 4.52 Trade-off surface of the Pareto front.

B.3. MOGA with an Integer-Based Coding Scheme

The goal vector used to obtain the non-dominated solutions is the same as the one presented in the random search approach. The Pareto optimal solutions and their objective values from five simulation runs are shown in Table 4.36. The combined Pareto optimal solutions and their objective values from all simulation runs are summarised in Table 4.37 while the trade-off surface of the Pareto front is given in Figure 4.53.

Table 4.36 Pareto optimal solutions and their objective values from five simulation runs.

Run	Decision Variables								Objectives		
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
1	22.5	36.5	13.0	0.728	0.235	0.093	0.005	0.852	0.361	0.07178	0.25
	21.0	35.0	14.5	0.747	0.228	0.087	0.010	0.864	0.347	0.06069	0.26
	20.0	35.0	17.0	0.747	0.231	0.087	0.013	0.866	0.345	0.04686	0.27
	17.5	39.5	8.0	0.725	0.222	0.079	0.003	0.860	0.365	0.03571	0.28
	16.5	29.5	8.5	0.742	0.213	0.082	0.012	0.863	0.339	0.01994	0.29
	15.0	34.5	10.0	0.737	0.222	0.092	-0.013	0.863	0.351	0.01760	0.30
2	23.0	38.5	16.5	0.751	0.213	0.079	0.015	0.869	0.361	0.06986	0.25
	21.0	35.5	12.5	0.746	0.232	0.081	0.015	0.848	0.339	0.05414	0.26
	18.5	28.5	19.0	0.731	0.232	0.081	-0.009	0.842	0.354	0.05310	0.27
	17.0	37.5	7.5	0.721	0.215	0.101	-0.008	0.847	0.360	0.03934	0.28
	16.0	37.5	16.0	0.736	0.222	0.098	-0.006	0.849	0.361	0.02360	0.29
	15.0	37.0	16.5	0.734	0.222	0.101	-0.003	0.851	0.358	0.02158	0.30
3	21.0	35.0	7.5	0.746	0.234	0.078	0.014	0.853	0.345	0.05371	0.26
	19.0	35.0	14.5	0.730	0.213	0.105	0.006	0.856	0.365	0.05329	0.27
	17.5	34.0	6.5	0.731	0.232	0.086	-0.004	0.848	0.360	0.03014	0.28
	16.5	26.5	10.5	0.744	0.222	0.093	0.011	0.858	0.342	0.02546	0.29
	15.0	33.5	14.5	0.746	0.235	0.106	-0.014	0.860	0.367	0.01898	0.30
4	22.5	34.5	16.0	0.735	0.221	0.081	0.012	0.850	0.355	0.07126	0.25
	21.0	31.0	8.0	0.735	0.219	0.089	-0.002	0.865	0.339	0.06395	0.26
	19.0	35.5	9.0	0.737	0.223	0.080	-0.005	0.849	0.341	0.04850	0.27
	18.0	34.0	19.5	0.748	0.238	0.097	0.014	0.868	0.355	0.03289	0.28
	16.5	26.5	18.0	0.732	0.236	0.097	0.012	0.854	0.354	0.02945	0.29
	15.0	31.5	20.0	0.738	0.219	0.083	0.007	0.850	0.347	0.01848	0.30
	15.0	32.0	20.0	0.738	0.219	0.083	0.007	0.850	0.347	0.01848	0.30
5	22.0	40.0	9.0	0.730	0.233	0.088	-0.006	0.839	0.367	0.07455	0.25
	20.5	30.5	18.0	0.748	0.232	0.099	-0.013	0.853	0.340	0.06348	0.26
	19.0	31.0	14.0	0.747	0.215	0.094	-0.015	0.867	0.368	0.05065	0.27
	17.5	35.0	13.0	0.736	0.239	0.104	0.002	0.848	0.365	0.03357	0.28
	16.5	36.0	17.5	0.751	0.238	0.079	0.010	0.848	0.342	0.02381	0.29
	15.0	25.0	19.5	0.741	0.237	0.104	-0.015	0.861	0.368	0.02379	0.30

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

Table 4.37 Combined Pareto optimal solutions and their objective values.

Run	Decision Variables									Objectives	
	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	Error	t
-	23.0	38.5	16.5	0.751	0.213	0.079	0.015	0.869	0.361	0.06986	0.25
	21.0	35.0	7.5	0.746	0.234	0.078	0.014	0.853	0.345	0.05371	0.26
	20.0	35.0	17.0	0.747	0.231	0.087	0.013	0.866	0.345	0.04686	0.27
	17.5	34.0	6.5	0.731	0.232	0.086	-0.004	0.848	0.360	0.03014	0.28
	16.5	29.5	8.5	0.742	0.213	0.082	0.012	0.863	0.339	0.01994	0.29
	15.0	34.5	10.0	0.737	0.222	0.092	-0.013	0.863	0.351	0.01760	0.30

T_i - Magnitude of torque limits on joint i (Nm) Error - Mean absolute tracking error (rad)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m) t - Trajectory time (second)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)

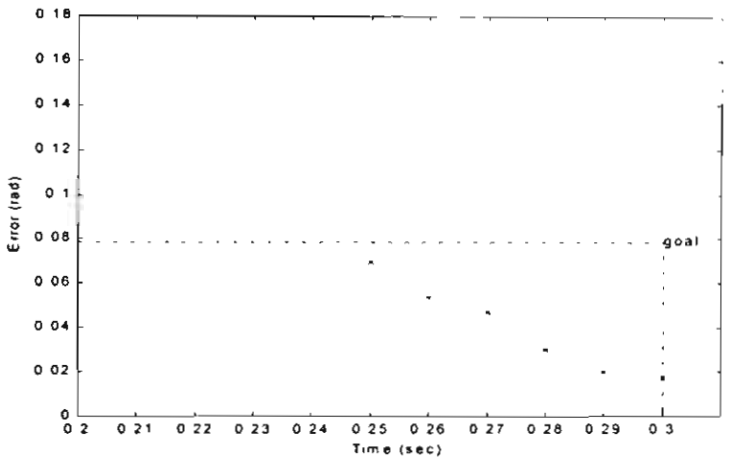


Figure 4.53 Trade-off surface of the Pareto front.

B.4. Summary of Simulation Results from Case Study II

Results from the random search, the MOGA with a Gray coding scheme and the MOGA with an integer-based coding scheme are summarised in Table 4.38.

Table 4.38 Summary of results from the three approaches.

Approach	N_d	N_n
Random search	6	0
MOGA with a Gray coding scheme	6	2
MOGA with an integer-based coding scheme	6	4

N_d - Number of distinct solutions found
N_n - Number of solutions which are not dominated by solutions found by other techniques

4.4.8. Discussions

Prior to any analysis on the simulation results can be carried out, a number of points are required to be made clear. The Pareto front results are used to represent two main

aims of the search; these are to find the range of variety in solutions and to locate the solutions which are close to the true Pareto optimal solutions of the problem. For this time-optimal control application, the exact range of variety in solutions is known. Such knowledge is gained by inspecting the non-dominated solutions and their corresponding objectives in the solution set itself. This statement will be made clearer later on in the discussions. Nonetheless, similar to the majority of engineering applications, the theoretical, or true, Pareto optimal solutions of the problem are not known. Of course, there will be a possibility that some of the Pareto optimal solutions found by one technique can be dominated by the solutions found by another technique. In order to compare the Pareto optimal solutions obtained from each technique objectively, both points of view on the variety in solutions found and the number of solutions found which cannot be dominated by the solutions obtained from other techniques needs to be considered.

First of all, consideration is placed on the simulation results from the first case study. Both the MOGA with a Gray coding scheme and the MOGA with an integer-based coding scheme can locate seven distinct solutions while the random search fails to locate a solution with the trajectory time of 0.21 seconds. For this case study, there can be only seven distinct solutions in the Pareto optimal solution set. This is because the solution that has a trajectory time of 0.21 seconds and still has the tracking error within the target value is obtained for magnitudes of torque limits which are close to the actual limits on the actuator torque. In addition, there are only seven distinct solutions which can occupy the trajectory time solution space from $t = 0.21$ seconds to $t = 0.27$ seconds with an increment of 0.01 seconds (the sampling period). As far as the variety of solutions found is concerned, both approaches of the MOGA are equally good in this respect. With a close inspection, it is noticeable that all solutions found by both approaches of the MOGA dominates all optimal solutions found by the random search. However, after comparing the results found by both approaches of the MOGA, it is found that the solutions with the trajectory times of 0.21, 0.23, 0.26 and 0.27 seconds found by the MOGA with a Gray coding scheme are dominated by the corresponding solutions found by the MOGA with an integer-based coding scheme. At the same time, the solutions found by the MOGA with an integer-based coding scheme which have trajectory times of 0.22, 0.24 and 0.25 seconds are dominated by the solutions obtained by the MOGA with a Gray coding scheme. In this respect, it

can be said that the search performances of the two MOGA approaches are very close to one another.

Moving onto the second case study: all three search techniques are capable of locating six distinct solutions. Note that for this case study, there can be a maximum of six distinct solutions in the Pareto optimal solution set. This is concluded from the results obtained from the first case study which indicates that the solution which has the minimum allowable trajectory time and also has the tracking error which is smaller than 0.07854 rad is the one with the trajectory time of 0.25 seconds. With the maximum allowable trajectory time being limited to 0.3 seconds by the search target and the sampling period is set to 0.01 seconds, there are only six distinct solutions with the trajectory times ranging from 0.25 to 0.30 seconds that can cover the whole Pareto front. The simulation results in this case study also reveals that all solutions found by the MOGA with an integer-based coding scheme dominates all solutions found by the random search. In contrast, the MOGA with a Gray coding scheme can only find five solutions which dominate the solutions located by the random search. The only solution found by the MOGA with a Gray coding scheme which is dominated by the solution found by the random search is the one with the trajectory time of 0.25 seconds. Among the solutions found by the two MOGA approaches, two solutions found by the MOGA with a Gray coding scheme dominates the solutions located by the MOGA with an integer-based coding scheme. These two solutions are the solutions with the trajectory times of 0.27 and 0.30 seconds. In contrast, the MOGA with an integer-based coding scheme can locate four distinct solutions that dominates the solutions found by the MOGA with a Gray coding scheme: the solutions with the trajectory times of 0.25, 0.26, 0.28 and 0.29 seconds. In overall, it can be noticed that the performance of the MOGA with an integer-based coding scheme is slightly higher than that of the MOGA with a Gray coding scheme.

In summary, it can be seen that the MOGA with an integer-based coding scheme has emerged as the most effective method in finding the Pareto front for this problem. This conclusion is supported by both viewpoints on the variety of solutions found and the number of found solutions which cannot be dominated by solutions obtained from the other techniques. Another important point, which can be observed from both case studies, is that nearly all of the solutions found by the random search method cannot dominate the solutions found by both approaches of the MOGA. Since

the solutions found by the random search method in this case are the non-dominated solutions of the initial population of the genetic algorithm, this indicates that successful evolution has been accomplished by the MOGA.

4.5. Conclusions

In this chapter, the task hybridisation framework presented in Chapter 3 is applied to a robotic application. In this hybrid framework, the overall application task is divided into a number of small tasks which subsequently benefit from different components in the framework. Particularly, this framework concentrates on the combination of neural networks and genetic algorithms in which the results obtained from the neural network module are used as the objective values in the genetic algorithm module. A 3-dof robotic system has been used to demonstrate this framework, where the neural networks are used as assistants to PID controllers in a non-linear de-coupled feedback control scheme. The performance of the robot is then measured and used as the objective values in a multi-objective optimisation problem.

The robotic application which is chosen to illustrate the effectiveness of the framework is a time-optimal control application. The task of tracking a straight-line path in Cartesian space is given to the robot in this case. The time-optimal joint trajectory time history is calculated by using the time-optimal control algorithm as described by Shiller and Lu (1992). Time-optimality is achieved by executing a bang-bang control, where the control torque signal in one joint is saturated and the control torque signal on other joints is adjusted accordingly such that the torque limits on each actuator are not violated. However, the trajectory time history obtained from the time-optimal control algorithm is calculated by using only the open-loop dynamics of the robot model. Previously, in order for this trajectory time history to be used as input to the position control loop, the time history had to be modified using a trajectory pre-shaping scheme (Shiller et al., 1996). In this investigation, the use of extended Kohonen networks which contain an additional lattice of output neurons as assistants to PID controllers has been proven to be an effective method in compensating for the closed-loop dynamics and modelling errors. This results in being able to use the trajectory time history as the input to the control loop directly without the use of trajectory pre-shaping scheme.

Subsequently, a genetic algorithm has been used to solve a multi-objective optimisation involving the selection of torque limits and an end-effector path subject to time-optimality and tracking error constraints. Two approaches of a multi-objective genetic algorithm (MOGA) have been used in this application: the MOGA with a Gray coding scheme and the MOGA with an integer-based coding scheme. The simulation results suggest that the integer-based chromosome is more suitable than the Gray chromosome at representing the decision variables. This makes the MOGA with an integer-based coding scheme emerge as the most effective method in finding the Pareto optimal solutions for this problem.

Chapter 5

Conclusions

5.1. Introduction

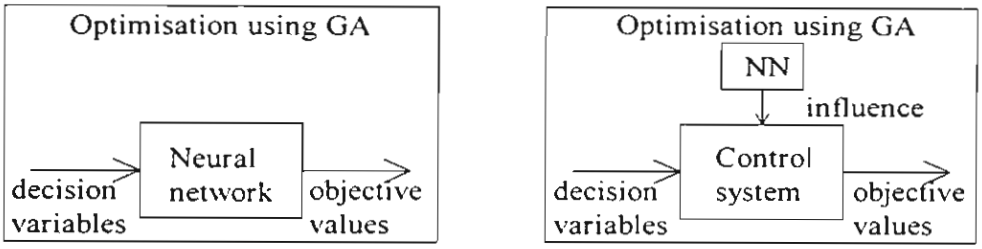
In this report, a neuro-genetic based hybrid framework has been presented. The framework has been fully explained and demonstrated in a robotic application. In particular, the developed framework is identified as a task hybridisation framework where the neural network has the role of being an additional controller in the control system while the genetic algorithm is used to solve a multi-objective optimisation problem associated with the control task. The developed framework has been successfully applied to a closed-loop time-optimal control application where the interested optimisation objectives are trajectory time and position tracking error.

In this final chapter, conclusions from the works presented in this report are briefly summarised. In addition, a comparison between the developed framework and the one reported in early literature, and possible further works are also discussed. The summary of conclusions and the framework comparison is given in section 5.2 while the recommended further work is explained in section 5.3.

5.2. Framework Comparison and Conclusions

In this report, a task hybridisation framework has been introduced. In this particular framework, the neural network component has the role of being an additional

controller in a control system. The purpose of using a neural network controller in this case is to compensate for modelling errors in the control system. In contrast, the genetic algorithm component is used to solve a multi-objective optimisation problem where the performances of the control system are optimisation objectives. This developed framework is comparable to one of the task hybridisation frameworks reported in early literature. In particular, this previously developed framework uses a neural network to model a relationship between decision variables and objective values in an optimisation using a genetic algorithm. A schematic diagram showing the structures of these two frameworks is given in Figure 5.1.



Early developed framework Task hybridisation framework

Figure 5.1 Comparison between an early developed framework and the task hybridisation framework.

From Figure 5.1, in the early developed framework the neural network component has a direct role in calculating the objective values for the optimisation process based upon its input. In contrast, the neural network component can only influence the outcome of the objective values in the task hybridisation framework illustrated in this report. In this case a specific function, based on the structure of a control system, is used to calculate the objective values instead.

In this report, a time-optimal control application is used to demonstrate the functionality of the task hybridisation framework. The conclusions drawn from implementing this task hybridisation framework in the time-optimal control application can be summarised as follows.

1. The use of neural network controllers for the purpose of minimising the effect of modelling error in a robotic system has proven to be more effective than using trajectory pre-shaping (Shiller et al., 1996) for the same purpose in time-optimal control. In addition, it is found that the extended Kohonen network is more suitable to the time-optimal control task when the robot is operated under the

normal condition while the radial-basis function network is more effective when there are modelling errors in the system. Moreover the use of the neural network controllers also enables the use of a time-optimal angular trajectory profile, obtained from the time-optimal control algorithm (Shiller and Lu, 1992), as the reference input to the closed-loop robotic system without the use of trajectory pre-shaping as required in Shiller et al. (1996).

2. A multi-objective genetic algorithm (MOGA) has successfully solved a multi-objective optimisation problem involving the selection of a torque limit combination and a pre-defined path for use as input to the time-optimal control algorithm where the objectives are trajectory time and tracking error. The MOGA has been proven to be more effective than a random search in obtaining Pareto optimal solutions. In addition, it is also found that an integer-based chromosome is more suitable than a Gray code chromosome at representing decision variables.

5.3. Recommended Further Works

In the previous section, the conclusions from the report have been discussed. Based on these conclusions, a possible further work can be explained as follows. In the report, the control problem discussed involves the time-optimal control of a robot that is given a task of tracking a straight-line path in Cartesian space. The scope of the work can be extended to include obstacle avoidance criteria where the use of a more sophisticated path-planning scheme is required. Obstacle avoidance constraints can be integrated into the problem investigated where the obstacles can be dynamic or static ones.

Appendix A

Lagrange-Euler Formation of the Dynamic Model of a Robot

Prior to the Lagrange-Euler formation of the dynamics equation of a robot arm, three key elements of the robot model have to be identified: homogeneous transformation matrices, pseudo-inertia matrices and centre of mass vectors. These three components are briefly explained as follows.

A.1. Homogeneous Transformation Matrices

A homogeneous transformation matrix is a matrix which is used to relate the spatial displacement of a link co-ordinate frame to another co-ordinate frame in a robotic system. Let ${}^{i-1}\mathbf{A}_i$ be the homogeneous transformation matrix which relates the co-ordinate frame of the ith link to the co-ordinate frame of the $(i\text{-}1)$th link. If the joint is a revolute joint, ${}^{i-1}\mathbf{A}_i$ is given by

$$ {}^{i-1}\mathbf{A}_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(A-1)} $$

where θ_i is the joint angle from the $\mathbf{x}_{i-1}$ axis to the $\mathbf{x}_i$ axis about the $\mathbf{z}_{i-1}$ axis (using the right hand rule), d_i is the distance from the origin of the $(i\text{-}1)$th co-ordinate frame to the intersection of the $\mathbf{z}_{i-1}$ axis with the $\mathbf{x}_i$ axis along the $\mathbf{z}_{i-1}$ axis, a_i is the offset distance from the intersection of the $\mathbf{z}_{i-1}$ axis with the $\mathbf{x}_i$ axis to the origin of the ith frame along the $\mathbf{x}_i$ axis (or the shortest distance between the $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ axes) and α_i is the offset angle from the $\mathbf{z}_{i-1}$ axis to the $\mathbf{z}_i$ axis about the $\mathbf{x}_i$ axis (using the right hand rule). On the other hand, if joint i is a prismatic joint, ${}^{i-1}\mathbf{A}_i$ is given by

$$ {}^{i-1}\mathbf{A}_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & 0 \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad \text{(A-2)} $$

For a revolute joint, d_i, a_i and α_i are the joint parameters and remain constant for a robot while θ_i is the joint variable that changes when link i rotates with respect to link i-1. For a prismatic joint, θ_i, a_i and α_i are the joint parameters and remain constant for a robot while d_i is the joint variable. To model an n-dof robot requires the knowledge of n homogeneous transformation matrices where each matrix relates the ith co-ordinate frame to the (i-1)th co-ordinate frame for i = 1, 2, ..., n. Note that

$$ {}^{j}\mathbf{A}_k = {}^{j}\mathbf{A}_{j+1}\,{}^{j+1}\mathbf{A}_{j+2}\cdots{}^{k-2}\mathbf{A}_{k-1}\,{}^{k-1}\mathbf{A}_k. \qquad \text{(A-3)} $$

A.2. Pseudo-Inertia Matrices

A pseudo-inertia matrix is a matrix which is composed of first moments, moments of inertia and products of inertia of a link about the three principal axes of a co-ordinate frame. In order to model an n-dof robot, n pseudo-inertia matrices are required in the formation of the dynamics equation. Each pseudo-inertia matrix is calculated for a link between the ith co-ordinate frame and the (i-1)th co-ordinate frame. The pseudo-inertia matrix of link i where link i is located between the (i-1)th co-ordinate frame and the ith co-ordinate frame is given by

$$ \mathbf{J}_i = \begin{bmatrix} \frac{-I_{xx}+I_{yy}+I_{zz}}{2} & I_{xy} & I_{xz} & m_i\bar{x}_i \\ I_{xy} & \frac{I_{xx}-I_{yy}+I_{zz}}{2} & I_{yz} & m_i\bar{y}_i \\ I_{xz} & I_{yz} & \frac{I_{xx}+I_{yy}-I_{zz}}{2} & m_i\bar{z}_i \\ m_i\bar{x}_i & m_i\bar{y}_i & m_i\bar{z}_i & m_i \end{bmatrix} \qquad \text{(A-4)} $$

where $\mathbf{J}_i$ is the pseudo-inertia matrix of link i, I_{xx}, I_{yy} and I_{zz} are the moments of inertia about the principal axes $\mathbf{x}_i$, $\mathbf{y}_i$ and $\mathbf{z}_i$ in the ith co-ordinate frame, I_{xy}, I_{yz} and I_{xz} are the products of inertia about the axes $\mathbf{x}_i\mathbf{y}_i$, $\mathbf{y}_i\mathbf{z}_i$ and $\mathbf{x}_i\mathbf{z}_i$ in the ith co-ordinate frame, m_i is the mass of link i, and $\bar{x}_i$, $\bar{y}_i$ and $\bar{z}_i$ form the centre of mass of link i expressed in the ith co-ordinate system.

A.3. Centre of Mass Vectors

A centre of mass vector is a vector describing the centre of mass of a robot link in a co-ordinate frame. Let ${}^{i}\bar{\mathbf{r}}_i$ be the centre of mass vector of link i in the ith co-ordinate frame. ${}^{i}\bar{\mathbf{r}}_i$ is given by

$$ {}^{i}\bar{\mathbf{r}}_i = \begin{bmatrix} \bar{x}_i & \bar{y}_i & \bar{z}_i & 1 \end{bmatrix}^T \tag{A-5} $$

where $\bar{x}_i$, $\bar{y}_i$ and $\bar{z}_i$ form the centre of mass of link i, expressed in the ith co-ordinate system.

With the full knowledge regarding homogeneous transformation matrices, pseudo-inertia matrices and centre of mass vectors, the dynamics equation of a robot arm can be formed where it is given by

$$ \mathbf{D}(\theta)\ddot{\theta} + \mathbf{h}(\theta,\dot{\theta}) + \mathbf{c}(\theta) = \mathbf{u}(t) \tag{A-6} $$

where $\mathbf{D}(\theta)$ is the $n \times n$ inertial acceleration-related matrix, $\mathbf{h}(\theta,\dot{\theta})$ is the $n \times 1$ centrifugal and Coriolis forces vector, $\mathbf{c}(\theta)$ is the $n \times 1$ gravity loading force vector, $\mathbf{u}(t)$ is the $n \times 1$ torque input vector, $\theta(t)$ is the $n \times 1$ angular position vector, $\dot{\theta}(t)$ is the $n \times 1$ angular velocity vector, $\ddot{\theta}(t)$ is the $n \times 1$ angular acceleration vector and n denotes the degree of freedom of the robot model. The explanation for the terms $\mathbf{D}(\theta)$, $\mathbf{h}(\theta,\dot{\theta})$ and $\mathbf{c}(\theta)$ is given as follows.

Firstly, a consideration is placed on the inertial acceleration-related matrix, $\mathbf{D}(\theta)$. Each element of $\mathbf{D}(\theta)$ is given by

$$ D_{ik} = \sum_{j=\max(i,k)}^{n} Tr(\mathbf{U}_{jk}\mathbf{J}_j\mathbf{U}_{ji}^T), \quad i,k = 1, 2, \ldots, n \tag{A-7} $$

where

$$ \mathbf{U}_{ij} = \begin{cases} {}^{0}\mathbf{A}_{j-1}\mathbf{Q}_j{}^{j-1}\mathbf{A}_i & \text{for } j \le i \\ 0 & \text{for } j > i \end{cases} \tag{A-8} $$

and

$$ \mathbf{Q}_j = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{for a revolute joint} \tag{A-9} $$

and $\mathbf{Q}_j = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ for a prismatic joint, (A-10)

$\mathbf{J}_j$ is the pseudo-inertia matrix of joint j, $Tr(.)$ denotes the trace operation and T indicates the transposition.

Moving onto the term describing centrifugal and Coriolis forces: $\mathbf{h}(\theta,\dot{\theta})$ is given by

$$\mathbf{h}(\theta,\dot{\theta}) = \begin{bmatrix} h_1 & h_2 & \cdots & h_n \end{bmatrix}^T \tag{A-11}$$

where $$h_i = \sum_{k=1}^{n}\sum_{m=1}^{n} h_{ikm}\dot{\theta}_k\dot{\theta}_m, \qquad i = 1, 2, \ldots, n \tag{A-12}$$

and $$h_{ikm} = \sum_{j=\max(i,k,m)}^{n} Tr(\mathbf{U}_{jkm}\mathbf{J}_j\mathbf{U}_{ji}^T), \quad i, k, m = 1, 2, \ldots, n. \tag{A-13}$$

Note that $$\mathbf{U}_{ijk} = \begin{cases} {}^0\mathbf{A}_{j-1}\mathbf{Q}_j{}^{j-1}\mathbf{A}_{k-1}\mathbf{Q}_k{}^{k-1}\mathbf{A}_i & \text{for } i \geq k \geq j \\ {}^0\mathbf{A}_{k-1}\mathbf{Q}_k{}^{k-1}\mathbf{A}_{j-1}\mathbf{Q}_j{}^{j-1}\mathbf{A}_i & \text{for } i \geq j \geq k \\ 0 & \text{for } i < j \text{ or } i < k \end{cases}. \tag{A-14}$$

Finally, the gravity loading force vector is given by

$$\mathbf{c}(\theta) = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix}^T \tag{A-15}$$

where $$c_i = \sum_{j=i}^{n}(-m_j\mathbf{g}\mathbf{U}_{ji}{}^j\bar{\mathbf{r}}_j), \qquad i = 1, 2, \ldots, n \tag{A-16}$$

and m_j is the mass of link j, ${}^j\bar{\mathbf{r}}_j$ is the centre of mass vector of link j in the jth co-ordinate frame and $\mathbf{g}$ is a gravity row vector expressed in the base co-ordinate system. For a level system, $\mathbf{g} = \begin{bmatrix} 0 & 0 & -|g| & 0 \end{bmatrix}$ and g is the gravitational constant.

Appendix B

Calculations of Moments and Products of Inertia

In Chapter 4, a robot model is used during the simulations. All links in the robots are modelled as a rigid rod in one-dimensional space. Moments of inertia and products of inertia of a rod are described as follows.

Within the robot model used during the simulations, moments of inertia of a rigid rod is only required for the case of rotations about the principal axes in a co-ordinate frame. In particular, the rod will always coincide with the **x** axis of the co-ordinate frame where the rotation axes being at one end of the rod itself. Hence, the moments of inertia of the rigid rod about the principal axes are given by

$$I_{xx} = 0,\ I_{yy} = \frac{1}{3}ml^2 \text{ and } I_{zz} = \frac{1}{3}ml^2 \qquad \text{(B-1)}$$

where m is the mass of the rod, l is the length of the rod and I_{xx}, I_{yy} and I_{zz} represent moments of inertia about the **x**, **y** and **z** axes respectively. Since the rod occupies one-dimensional space, particularly with the dimensions of the rod in the **y** and **z** axes of the co-ordinate frame always being equal to zero, the products of inertia (I_{xy}, I_{yz}, I_{xz}) will be equal to zero.

Appendix C

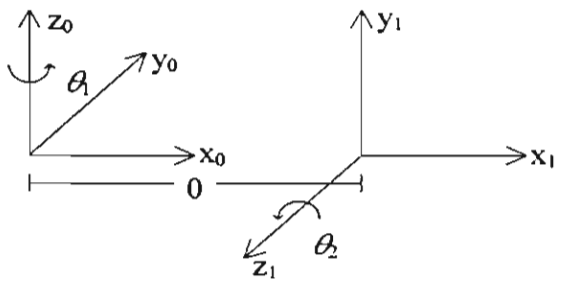
Robot Model Used in the Simulations

As mentioned earlier, the 3-dof robot illustrated in Figure 4.1 is used in the simulations presented in Chapter 4. Using the Denavit-Hartenberg convention (Fu et al., 1985), the parameters used during the co-ordinate frame transformations are given in Table C.1.

Table C.1 Parameters for co-ordinate frame transformations.

Joint Number	α_i	a_i	d_i
1	90°	0	0
2	0°	l_1	0
3	0°	l_2	0

Note that the definition of parameters α_i, a_i and d_i are given in Appendix A. Homogeneous transformation matrices, pseudo-inertia matrices and centre of mass vectors for this 3-dof robot are given alongside co-ordinate transformation diagrams in Figure C.1. In addition, the mass and length properties of each link in the robot model are given in Figure 4.1.



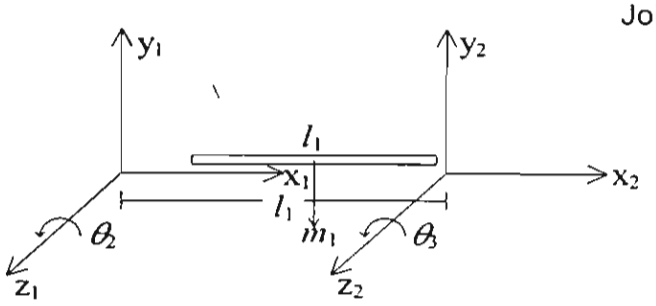
Joint 1

$$ {}^{0}\mathbf{A}_1 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ \mathbf{J}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

$$ {}^{1}\bar{\mathbf{r}}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T $$

Joint 2

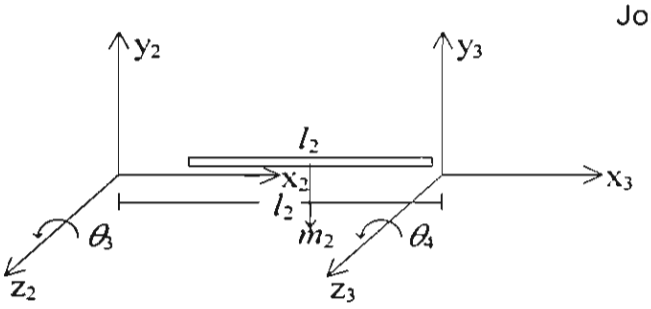


$$
{}^{1}\mathbf{A}_{2} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & l_1\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & l_1\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\mathbf{J}_{2} = \begin{bmatrix} \frac{1}{3}m_1 l_1^{\,2} & 0 & 0 & -\frac{1}{2}m_1 l_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2}m_1 l_1 & 0 & 0 & m_1 \end{bmatrix}
$$

$$
{}^{2}\bar{\mathbf{r}}_{2} = \begin{bmatrix} -\frac{1}{2}l_1 & 0 & 0 & 1 \end{bmatrix}^{T}
$$

Joint 3



$$
{}^{2}\mathbf{A}_{3} = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & l_2\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & l_2\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\mathbf{J}_{3} = \begin{bmatrix} \frac{1}{3}m_2 l_2^{\,2} & 0 & 0 & -\frac{1}{2}m_2 l_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2}m_2 l_2 & 0 & 0 & m_2 \end{bmatrix}
$$

$$
{}^{3}\bar{\mathbf{r}}_{3} = \begin{bmatrix} -\frac{1}{2}l_2 & 0 & 0 & 1 \end{bmatrix}^{T}
$$

Figure C.1 Co-ordinate transformation diagrams.

Appendix D

Publication of the Research Result

Chaiyaratana, N. and Zalzala, A. M. S. (2001). Time-optimal path planning and control using neural networks and a genetic algorithm. *2001 ASME International Mechanical Engineering Congress and Exposition*, New York, NY, to appear.

Proceedings of
2001 ASME International Mechanical Engineering Congress and Exposition
November 11-16, 2001, New York, NY

TIME-OPTIMAL PATH PLANNING AND CONTROL USING NEURAL NETWORKS AND A GENETIC ALGORITHM

Nachol Chaiyaratana
King Mongut's Institute of Technology North
Bangkok, R & D Center for Intelligent Systems,
Bangkok 10800, Thailand,
Tel: 66 (0)2 9132500 Ext. 8208,
Fax: 66 (0)2 5870029, E-mail: nchl@kmitnb.ac.th

Ali M. S. Zalzala
Heriot-Watt University,
Dept. of Computing and Electrical Engineering,
Edinburgh EH14 4AS, United Kingdom,
Tel: 44 (0)131 4513894, Fax: 44 (0)131 4513327,
E-mail: A.Zalzala@hw.ac.uk

ABSTRACT

This paper presents the use of neural networks and a genetic algorithm in time-optimal control of a closed-loop 3-dof robotic system. Extended Kohonen networks which contain an additional lattice of output neurons are used in conjunction with PID controllers in position control to minimise command tracking errors. The results indicate that the extended Kohonen network controller is more efficient than the trajectory pre-shaping scheme reported in early literature. Subsequently, a multi-objective genetic algorithm (MOGA) is used to solve an optimisation problem related to time-optimal control. This problem involves the selection of actuator torque limits and an end-effector path subject to time-optimality and tracking error constraints. Two chromosome coding schemes are explored in the investigation: Gray and integer-based coding schemes. The results suggest that the integer-based chromosome is more suitable at representing the decision variables. As a result of using both neural networks and a genetic algorithm in this application, an idea of a hybridisation between a neural network and a genetic algorithm at the task level for use in a control system is also effectively demonstrated.

Keywords: Genetic Algorithm, Neural Network, Robotics, Time-Optimal Control

NOMENCLATURE

dof	degree-of-freedom
KOH	Kohonen network
MOGA	multi-objective genetic algorithm
NN	neural network
PID	proportional-integral-derivative controller
RBF	radial-basis function network
SMAE	sum of mean absolute tracking errors
TP	trajectory pre-shaping
$\mathbf{c}(\theta)$	gravity loading force vector
$\mathbf{D}(\theta)$	inertial acceleration-related matrix
$\mathbf{h}(\theta,\dot{\theta})$	centrifugal and Coriolis forces vector
t	continuous time
$\mathbf{u}(t)$	torque input vector
$u^i_{ref}(t)$	reference control signal for the ith joint sub-system
α_{ij}	arbitrary scalar
$\theta(t)$	angular position vector
$\dot{\theta}(t)$	angular velocity vector
$\ddot{\theta}(t)$	angular acceleration vector
θ^i_d	desired angular position of joint i
$\dot{\theta}^i_d$	desired angular velocity of joint i
λ_i	arbitrary scalar

1. INTRODUCTION

Time-optimal control has been one of the major research interests in robotics during the past decade. Time-optimality can lead to an overall improvement in the level of productivity from a manufacturing viewpoint and an increase in the effectiveness of a task execution from an operational viewpoint. One particular aspect of research is the theory and application of time-optimal control of a robot arm along a pre-defined path. An algorithm that can lead to time-optimality of this kind was firstly developed by Bobrow et al. [1]. Over the years, this algorithm has undergone a number of refinements and one of the latest modifications has been described in Shiller and Lu [2]. In summary, a time-optimal motion of a robot arm along a pre-defined path is achieved when the motion is executed with

ther the maximum possible acceleration or deceleration along e path. This can be done when one of the actuators on the obot arm is always saturated and the other actuators adjust eir torque values so that their torque limits are not violated 3].

Although this time-optimal control algorithm has been roven to be a useful algorithm in a number of robotic pplications, the majority of the demonstrations have only been one in the open-loop control mode. This can hardly be the case r a practical use of motion control in a real-time mplementation where closed-loop control would be a more ommon practice. Shiller et al. [4] have pointed out that the ctuator dynamics and the delays caused by an on-line feedback ontroller would lead to a reduction in the efficiency of the lgorithm when closed-loop control is used. Three possible ethods have been used to solve this problem. The first method s based on a modification of the original time-optimal control roblem into a time-energy optimal control problem which can e regarded as a lagrangian constraint optimisation problem and an only be solved numerically [5]. A drawback of this method s that the modification also leads to an increase in the resulting rajectory time. The second method is based on the use of a implified friction model to compensate for the actuator ynamics and the implementation of a trajectory pre-shaping to ccount for the dynamics of the controller [4]. Finally, the third ethod covers the use of a neural network which is trained sing feedback error learning [6] as an additional controller in he control loop. The primary function of this neural network is o compensate for modelling errors and delays caused by the ain controller in the system. It has also been demonstrated that he compensation performance of the neural network controller s higher than that of the trajectory pre-shaper [7].

The work initiated by Chaiyaratana and Zalzala [7] will be ontinued in this paper where the investigation will cover the se of time-optimal control in a closed-loop 3-dof robotic ystem. Similar to the earlier work, the investigation will be arried out in a similar way to that described in Shiller et al. [4] except that the actuator dynamics are not considered. The eural network controllers will be used in conjunction with the standard controllers, which leads to the redundancy of the use of trajectory pre-shaper. In contrast to the earlier work where the feedback error learning is used, in this paper the neural etwork controllers will be trained using reinforcement learning. In addition to the continuation on the study of neural network capability in the compensation task, a further multi-objective optimisation problem associated with the use of time-optimal control is also considered. Note that this is an extension to the multi-objective problem addressed in Chaiyaratana and Zalzala [7]. The optimisation problem interested involves the selection of torque limit combination and the path planning process where the search objectives are expressed in terms of the position tracking error and trajectory time. An approach on multi-objective optimisation using a genetic algorithm, namely a multi-objective genetic algorithm (MOGA) [8] will be used to solve the mentioned problem. Since the neural network and genetic algorithm are used in the different part of the control application, in essence this indicates a task hybridisation between a neural network and a genetic algorithm.

This paper is presented as follows. The time-optimal control algorithm as described by Shiller and Lu [2] is briefly explained in section 2. In addition, the trajectory pre-shaping scheme is also explained in this section. In section 3, the overview of the time-optimal control problem is discussed. In section 4, the control structure of the robotic system and the neural network contribution is given. The improvement in the system performance gained by using neural network controllers and the comparison with the previous results reported in Chaiyaratana and Zalzala [7] is illustrated in section 5. The multi-objective optimisation problem associated with time-optimal control and the MOGA are explained in section 6. The optimisation results and the related discussions are given in section 7. Finally, the conclusions are drawn in section 8.

2. TIME-OPTIMAL CONTROL ALGORITHM AND TRAJECTORY PRE-SHAPING SCHEME

In summary, time-optimal control algorithm as described by Shiller and Lu [2] can be used to generate the time-optimal profiles of the reference joint position and the open-loop control torque signal provided that the physical properties of the robot arm are known and a pre-defined path of the robot arm in the workspace is available. In particular, the torque limits on the actuators within the robot are the key factors which have a major influence on the trajectory time obtained from the algorithm. As stated earlier, the time-optimal motion is achieved when one of the actuators on the robot arm is always saturated and the torque values of other actuators are within the bounds of the corresponding limits. This means that with the large values of the torque limits, the obtained trajectory time will be short. On the other hand, with the smaller values of the torque limits, the obtained trajectory time will be relatively larger. A schematic diagram describing input and output of the time-optimal control algorithm is given in Fig. 1.

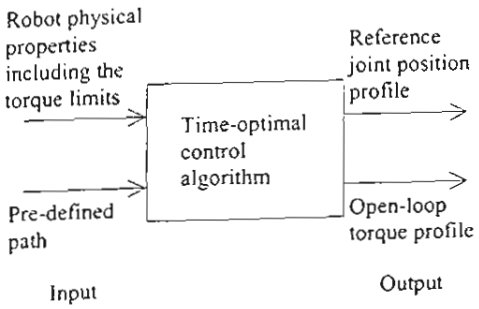


Figure 1. Schematic diagram of the time-optimal control algorithm.

In Fig. 1, the time-optimal control algorithm takes the robot physical properties and the information regarding the pre-defined robot's path as inputs. The outputs from the algorithm

s the reference joint position and the open-loop torque rofiles.

Nonetheless, the time-optimal control algorithm will roduce a result based on the open-loop dynamics of the stem. This means that a certain number of problems will arise hen using the reference joint position profile obtained from e algorithm as input to the closed-loop system [4, 5]. In order solve the problem, Shiller et al. [4] have introduced a method nown as trajectory pre-shaping which involves a modification f the reference joint position profile according to the dynamics f the closed-loop system. This modification involves adding e open-loop reference joint position profile with a factor iven by the open-loop torque profile which has been ansformed by the inverse model of the controller in the closed-oop system. This modified or "pre-shaped" reference joint osition profile is then used as input to the position feedback ystem in the usual way. Although some good results obtained y using trajectory pre-shaping have been reported in early iterature, it will be demonstrated in sections 4 and 5 that the se of neural networks to compensate for dynamics of the ontrollers and modelling errors helps to remove the need for rajectory pre-shaping.

3. OVERVIEW OF THE TIME-OPTIMAL CONTROL PROBLEM

The simulations which are used to demonstrate the unctions of a neural network and a genetic algorithm in the ime-optimal control application involve the use of a 3-dof obot in a position control task. This task requires the robot to rack a one-metre straight-line path; this is illustrated in Fig. 2.

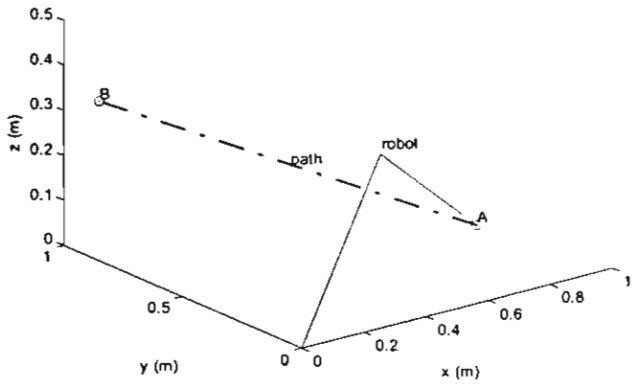


Figure 2. Robot and the straight-line path.

Referring to Fig. 2, point A (0.736, 0.226, 0.093) is the initial location of the robot end-effector and point B (0.0, 0.854, 0.354) is the final desired location of the robot end-effector on this path. The time-optimal control algorithm is then used to generate the trajectory time history, which is subsequently used as the input to the position control loop.

4. CONTROL STRUCTURE AND NEURAL NETWORK CONTRIBUTION

Firstly, consider the dynamic equation of motion for an n-dof robot which is given by

$$\mathbf{D}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{h}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \mathbf{c}(\boldsymbol{\theta}) = \mathbf{u}(t) \qquad (1)$$

where $\mathbf{D}(\boldsymbol{\theta})$ is the $n \times n$ inertial acceleration-related matrix, $\mathbf{h}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})$ is the $n \times 1$ centrifugal and Coriolis forces vector, $\mathbf{c}(\boldsymbol{\theta})$ is the $n \times 1$ gravity loading force vector, $\mathbf{u}(t)$ is the $n \times 1$ torque input vector, $\boldsymbol{\theta}(t)$ is the $n \times 1$ angular position vector, $\dot{\boldsymbol{\theta}}(t)$ is the $n \times 1$ angular velocity vector, $\ddot{\boldsymbol{\theta}}(t)$ is the $n \times 1$ angular acceleration vector and n is the degree of freedom of the robot model. Equation (1) indicates a non-linear relationship between the input torque and the joint angular parameters. The control strategy which is used in this study is the non-linear de-coupled feedback control. In this case, the control objective is to find a control signal $\mathbf{u}(t)$ such that the overall robotic system will be de-coupled into n linear second order systems. Freund [9] has suggested such a control signal which takes the form of

$$\mathbf{u}(t) = \mathbf{h}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \mathbf{c}(\boldsymbol{\theta}) - \mathbf{D}(\boldsymbol{\theta}) \begin{bmatrix} \alpha_{11}\dot{\theta}_1(t) + \alpha_{01}\theta_1(t) - \lambda_1 u^1_{ref}(t) \\ \vdots \\ \alpha_{1n}\dot{\theta}_n(t) + \alpha_{0n}\theta_n(t) - \lambda_n u^n_{ref}(t) \end{bmatrix} \qquad (2)$$

where α_{ij} and λ_i are arbitrary scalars. With the use of $\mathbf{u}(t)$ of this form, the overall dynamics of the system as described in Eq. (1) will transform into

$$\ddot{\theta}_i(t) + \alpha_{1i}\dot{\theta}_i(t) + \alpha_{0i}\theta_i(t) = \lambda_i u^i_{ref}(t)\,,\ i = 1, 2, \ldots, n \qquad (3)$$

which indicates the de-coupled input-output relationship of the system. Using this form of de-coupling and non-linear compensation, each de-coupled joint sub-system can be controlled using a standard PID controller. In addition, a neural network can be used as an additional controller in each joint control loop where it will have a role of compensating for the dynamics of the primary controller and the possible modelling errors. This arrangement is illustrated in Fig. 3.

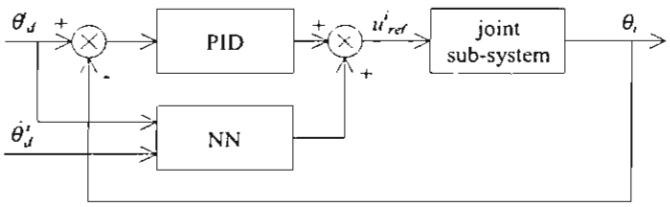


Figure 3. Neural network and PID controllers in each joint control loop.

However, with the control scheme as shown in Fig. 3, it is not possible to derive an exact desired neural network output training signal. Hence, an alternative training signal must therefore be acquired. One possible way for deriving an appropriate neural network output signal for use as an additional control signal is to use a reinforcement learning paradigm; this can be done as follows.

One procedure which can be used to accomplish reinforcement learning is a generate-and-test process. Basically, this process begins by generating a possible value of the neural network output. This newly generated neural network output is then combined with the control signal from the PID controller and subsequently applied to the model of joint sub-system where the predicted value of the command tracking error can be obtained. This command tracking error will be represented in the form of the reward value achieved by using the generated neural network output. In a similar manner, the unmodified value of the neural network output is also applied to the same model of the joint sub-system where another predicted value of the command tracking error and the associated reward value can also be obtained. If the change in the reward function – the difference between the two reward values – meets the criteria for adjusting the network connection weights, the network parameters will undergo an adaptation using an appropriate learning rule such as an error correction learning rule. After the adaptation, the network will send out the output signal which is being calculated using the updated settings of the network parameters where this output signal will be used as a part of the overall control signal for the actual robot. On the other hand, if the change in the reward function does not satisfy the criteria for the network adaptation, the network parameters will remain unchanged. The output from the network will then be calculated based on the unmodified settings of the network parameters. This process will continue until there are no changes in the network parameters. The schematic diagram of the reinforcement learning paradigm described above is illustrated in Fig. 4.

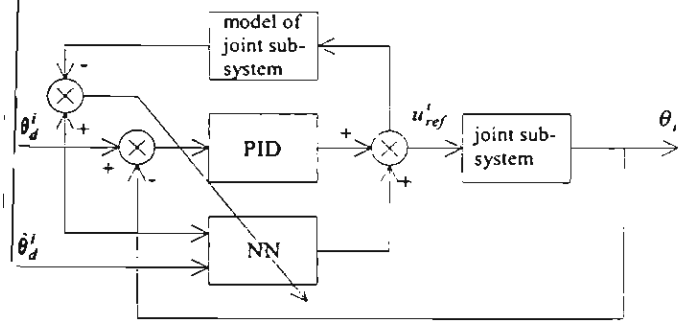


Figure 4. Model-based reinforcement learning within the control loop of joint *i*.

In this study, Kohonen networks with an additional lattice of output neurons or the extended Kohonen networks [10] are used to assist PID controllers in the position control loop. The model-based reinforcement learning is used to train the connection weights within the networks. Three neural network controllers, one for each joint sub-system, are trained and tested for use in position control of the 3-dof robot by using a combination between the time-optimal position and velocity trajectories as both the training and testing samples. Note that this time-optimal trajectory is obtained for a robot task of tracking a straight-line path in Cartesian space shown in Fig. 2 with the torque limits on joints 1, 2 and 3 of ±15, ±25 and ±5 Nm, respectively. The parameter settings for training neural networks are summarised in Table 1. The simulation results are displayed and discussed in the following section.

Table 1. Parameter settings for training neural networks.

Parameter	Value
Number of neurons in each network	98
Number of connection weights in each network	147
Number of input nodes in each network	2
Number of firing output nodes in each network	1
Number of training samples	30
Number of training epochs	400

5. RESULTS FROM USING NEURAL NETWORK CONTROLLERS AND DISCUSIONS

In this section, the simulation results from using the extended Kohonen network controllers will be discussed. In order to make the comparison, the results obtained using other techniques including the results achieved via the use of trajectory pre-shaping scheme and radial-basis function networks [7] will also be displayed alongside. Firstly, the simulation results for the case of PID controllers with trajectory pre-shaping and the case of PID and extended Kohonen network controllers are shown in Figs. 5, 6 and 7. In Figs. 5 and 6, the simulation results indicate that with the use of the extended Kohonen network controllers as the assistants to the PID controllers, a significant improvement in the control performance over that achievable by using trajectory pre-shaping mechanism can be observed. In Fig. 7, with the use of the extended Kohonen network controllers, the characteristics of the closed-loop torque profiles are similar to those of the open-loop control. This indicates that the time-optimality has been achieved within the torque constraints. Note that these trained extended Kohonen networks are used in the following parts including the following multi-objective optimisation problem without any further training.

Another advantage gained by using neural networks as assistants to PID controllers is the resistance to modelling errors which can occur during the robot operation. Many forms of error can be introduced to the robot system after the controllers have been designed. For example, a liquid spillage from the container attached to the last link of the robot arm can occur after an unexpected event, such as a collision. This kind of malfunction can lead to a loss of the overall mass in the last link of robot, which is a form of modelling error. This kind of modelling error is used in the following test cases which in turn are used to demonstrate the effectiveness of extended Kohonen network controllers in this kind of situation. In summary, in the following five test cases, a certain amount of mass in the last link, ranging from 10 % to 50 % of the overall mass, is lost during the operation. Note that the modelling error will only

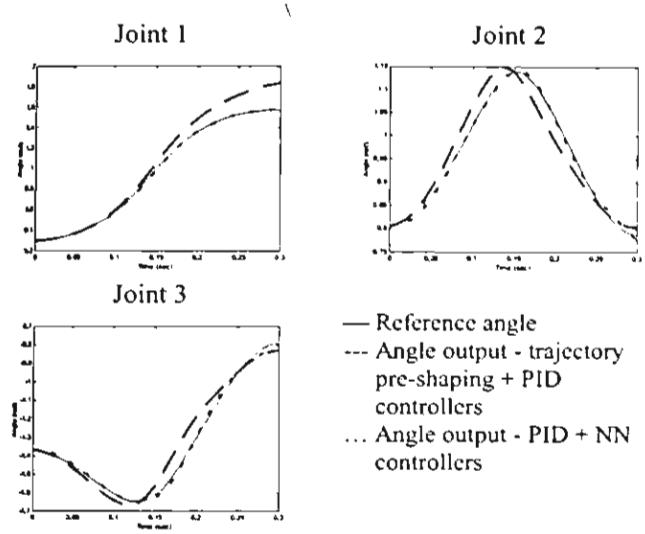


Figure 5. Angular position from each joint.

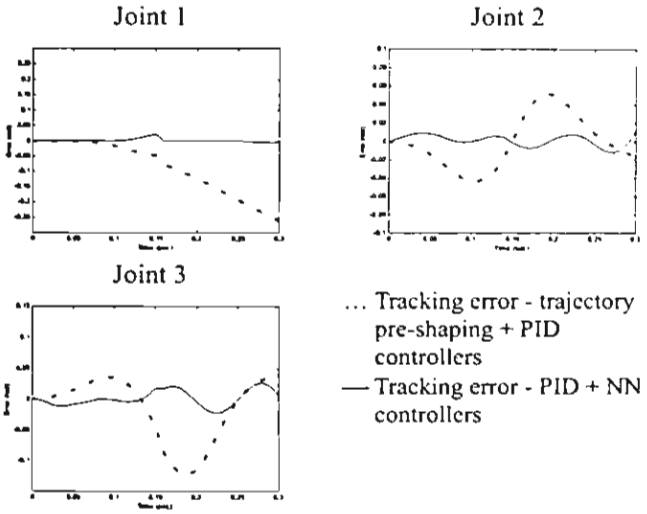


Figure 6. Tracking errors from each joint.

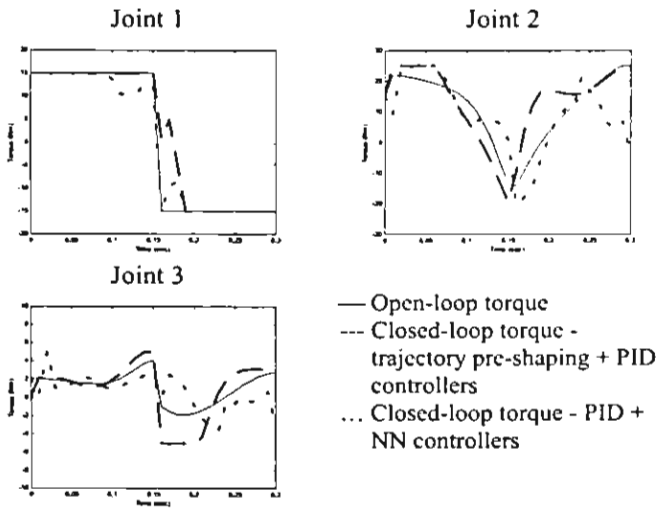


Figure 7. Open-loop and closed-loop torque on each joint.

effect the mass of the last link and not the length of the last link. This makes the robot trajectory unaffected by this modelling error.

Since the modelling error occurs after the control structure has been determined, the robot physical model which is used in the time-optimal trajectory generation and feed-forward compensator for de-coupling the robot dynamics will remain unchanged. Also the same torque limits on the actuators will be used in the time-optimal trajectory generation. However, the generated reference position trajectory and open-loop torque profile will no longer be time-optimal. Nevertheless, it is sufficed to say that although time-optimality cannot be maintained after the occurrence of the fault, the robot operation can still be described as time sub-optimal. A summary of simulation results, expressed in terms of the sum of the squared position tracking errors and the sum of the absolute errors, from all five test cases and the previous simulation with no mass loss is given in Table 2. Note that the simulation results from using trajectory pre-shaping and radial-basis function network controllers which are trained using feedback error learning [7] are also given for comparison purposes.

Table 2. Summary of tracking error results.

Loss	Squared Error (rad^2)			Absolute Error (rad)		
(%)	PID+KOH	PID+RBF	PID+TP	PID+KOH	PID+RBF	PID+TP
0	0.0074	0.0084	0.5894	0.5593	0.6669	4.6620
10	0.0128	0.0103	0.6350	0.8442	0.7607	5.1721
20	0.0220	0.0205	0.3191	1.0616	1.0206	4.0637
30	0.0332	0.0304	0.2754	1.2447	1.1929	3.7820
40	0.0422	0.0365	0.2656	1.3989	1.3096	3.8071
50	0.0502	0.0409	0.2509	1.5483	1.4249	3.7661

Again from Table 2, it is noticeable that in all test cases, the use of the neural networks as assistants to PID controllers has proven to be a more effective method in reducing tracking errors than the use of trajectory pre-shaping scheme. This indicates that neural network controllers are more suitable to the time-optimal control application both in the normal operating condition and in the event of the occurrence of modelling errors in the control system.

Although in all test cases, the PID and neural network controllers exhibit a very good performance, a significant increase in tracking errors can be observed as more mass is lost from the last link. However, this is to be expected since the neural network controllers are originally trained to cope with the robotic system which has been de-coupled into a set of de-coupled linear systems. As more mass is lost from the last link, the level of coupling in the overall system will increase. This will certainly lead to the deterioration in the performance of the neural network controllers. It can also be observed from the case where there is no mass loss from the robot arm that both the sum of the squared and the absolute tracking errors over the trajectory when the extended Kohonen networks are used are slightly better than those when the radial-basis function

etworks are used. In contrast, once there are some modelling rors in the system, it can be seen that the tracking errors when e extended Kohonen networks are used are slightly higher an that when the radial-basis function networks are utilised. hese results are caused by the differences in the network ructures and the learning algorithms used.

5. MULTI-OBJECTIVE OPTIMISATION USING A GENETIC ALGORITHM

In practice, the maximum torque limits, which are used in he time-optimal trajectory calculation process for a closed-loop ontrol, are usually less than the actual torque limits on the ctuators. This safety precaution is done in order to allow some nargins of error for possible discrepancies introduced to the ystem by modelling errors and controller dynamics [4]. This mplies that for a given set of the actual torque limits of the ctuators, there is a set of admissible torque limits combinations hat can lead to a certain level of time-optimality within an cceptable range of tracking error. In addition, in certain pplications such as welding or edge-deburring it is possible to nodify the end-effector trajectory in Cartesian space without effecting the task requirement provided that the position and orientation of the work piece at which the end-effector has to remain in contact with can be modified accordingly. The control ask discussed in section 3 is an example which reflects such pplications. By modifying the initial and final locations of the straight-line path, the task description in the application viewpoint would remain the same while the angular trajectory at which the robot joint has to follow would be different. Such change in the angular trajectory would lead to a variation in the position tracking error. Combining with the issue on torque limits, this points to a design problem in robotic applications. The objective of such problem is to find a combination of torque limits from a set of admissible torque ranges and the initial and final position of the end-effector which will lead to a trajectory which meets the time-optimality and tracking error constraints. This is a multi-objective optimisation problem since it would be highly unlikely to obtain a single trajectory that can minimise both the trajectory time and tracking error simultaneously. A multi-objective genetic algorithm (MOGA) will be used to solve the problem associated with the torque limit and end-effector position selection in this study. The problem formation and the genetic operators used are discussed as follows.

6.1. Decision Variables

A 3-dof robot with the task of tracking a straight-line path in Cartesian space presented earlier is used to demonstrate this multi-objective optimisation problem. The decision variables of the problem consist of the torque limit combination and the initial and final position of the end-effector. Assuming that the magnitudes of the maximum and minimum torque limits are the same for each actuator, the torque limit part of the decision variables would consist of the magnitude of the torque limits of each joint. In this study, the range of the magnitudes of the torque limits on joints 1, 2 and 3 are set to 15-30, 25-40 and 5-20 Nm, respectively. The lower bounds of the limits (i.e. 15, 25, 5) are based on the maximum allowable trajectory time requirement of 0.3 seconds, while the upper bounds of the torque limits (i.e. 30, 40, 20) are set by the actual torque limits of the actuators.

Moving on to the part of decision variables which involves the positions of the end-effector. In order to create a fixed-length path in Cartesian space, two vectors are required: the position vector for the initial position of the end-effector and the direction vector pointing from the initial position toward the desired final position of the end-effector. This requirement can be achieved by setting up two search variables. The first variable will be the initial location of the end-effector while the second variable will be another point in the robot workspace at which a direction vector pointing from the initial position of the end-effector toward this point can be established. In this investigation the search range for the initial position of the end-effector is given by (0.721-0.751, 0.211-0.241, 0.078-0.108) in the x, y and z directions, respectively. In contrast, the search range for the location of the other point in the robot workspace is set to (-0.015-0.015, 0.839-0.869, 0.339-0.369) in the x, y and z directions, respectively. Note that the search ranges for these two points are in the vicinity of the initial and final positions of the straight-line path described earlier in section 3.

6.2. Objective Variables

There are two optimisation objective variables in this problem: the tracking error and the trajectory time objectives. The tracking error objective is expressed in terms of the sum of the mean absolute errors over three joints, calculated over the whole trajectory. The trajectory time objective is the optimal trajectory time obtained from the time-optimal control algorithm. Note that the sampling period used in the simulation of this 3-dof robotic closed-loop system is 0.01 seconds. Hence, the trajectory time will always be in the form of $0.01m$ where m is a positive integer.

6.3. Chromosome Coding

Nine decision variables – the magnitudes of the torque limits from all three joints and the co-ordinates along three axes of the two points for identifying the straight-line path – are concatenated together and coded to form a chromosome. Two chromosome coding schemes are explored here: Gray and integer-based coding schemes. The torque ranges for all three joints are discretised using a search step of 0.5 Nm. This leaves 31 search points for the magnitude of the torque limits of each joint. In a similar way, the search ranges of the co-ordinates of the two points for dictating the location of the straight-line path are discretised using a search step of 0.001 m. This also leaves 31 search points for the co-ordinate in each axis. With the use of a Gray coding scheme, a Gray code of length 5 can be used to represent a decision variable. The total length of the chromosome in this case would be equal to 45. Note that there are certain search points obtained after decoding the

chromosome which lie outside the required search space. These points are mapped back into the feasible region by changing the most significant bit of the Gray code section representing the particular decision variable that violates the feasibility constraint into zero. In contrast to the case of the Gray coding scheme, with the use of an integer-based coding system a single gene can be used to represent a decision variable. Each gene can then take an allele value from a set which is composed of integers ranging from 0 to 30. The chromosome length in this case would be equal to nine.

6.4. Fitness Assignment and Fitness Sharing

The ranking method as described in Fonseca and Fleming [11] is used to rank each individual in the population. Following that, a linear fitness interpolation is used to assign fitness to each individual. Fitness sharing, with the use of triangular sharing function, is then carried out in normalised objective space.

6.5. Selection Method

Stochastic universal sampling [12] is used in the fitness selection. The elitist strategy used is to select two individuals with the highest fitness and pass onto the next generation without crossover or mutation.

6.6. Crossover and Mutation Methods

The standard one-point crossover is used in the recombination. Two individuals are allowed to perform crossover if, and only if, they are within the mating restriction distance from each other. For simplicity, the mating restriction radius is set to equal to the sharing radius and the consideration on the distance between the two individuals is also done in normalised objective space. For the case of chromosome coding using a Gray code, a standard bit-flipped operation is used for the mutation. In contrast, the value 1 will be added to or subtracted from the allele value of the mutated gene to achieve mutation in the integer-based coding system. The parameter settings for the MOGA are summarised in Table 3.

For the purpose of comparison, the random search technique is also used to find the Pareto optimal solutions in this study. Eschenauer et al. [13] have explained that in the case of a multi-objective optimisation, the random search method can generally be used to obtain a non-dominated solution set. In the random search technique, a set of random solutions is generated. Then non-dominated solutions are picked from this solution set. This can be done by applying the ranking mechanism used in the MOGA to the initial random solutions and selected solutions with the highest rank. Since a genetic algorithm also uses randomly generated solutions as its initial search points, the random search has already been embedded into the genetic algorithm as the initial search procedure. This means that a comparison between the non-dominated solutions found from the initial population of the genetic algorithm and the non-dominated solutions obtained from the last generation of the genetic algorithm run would provide an adequate comparison in terms of the comparison with the random search method. The description of the case studies explored, the simulation results and the discussions will be given in the next section.

Table 3. Parameter settings for the MOGA.

Parameter	Value
Chromosome length	
Gray code	45
Integer-based code	9
Crossover probability	0.8
Mutation probability	
Gray code	0.02
Integer-based code	0.1
Sharing and mating restriction radii	0.03
Population size	30
Number of elitist individuals	2
Number of generations	30

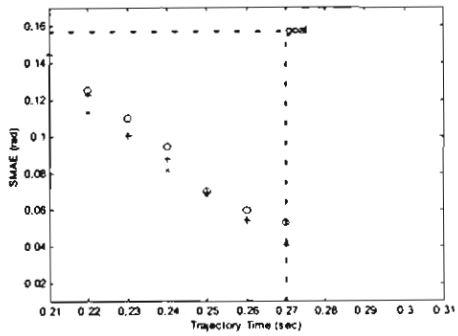
7. RESULTS FROM USING A GENETIC ALGORITHM AND DISCUSSIONS

Two case studies are investigated in this paper. The aim of the first case study is to find a set of torque limit combinations and straight-line paths which lead to trajectories with the sum of the mean absolute tracking errors ≤ 0.15708 radians (3 degrees per joint) and the trajectory time ≤ 0.27 seconds. The aim of the second case study is to find a set of torque limit combinations and straight-line paths which lead to trajectories with the sum of the mean absolute tracking errors ≤ 0.07854 radians (1.5 degrees per joint) and the trajectory time ≤ 0.30 seconds. The purpose of the first case study is to find solutions that concentrate more on optimising the trajectory time while the second case study emphasises on the tracking error optimisation. The simulation results for these two cases are summarised in Figs. 8 and 9. Note that the displayed results are the combination of Pareto optimal solutions obtained from five different simulation runs. In addition, the initial populations used in both approaches of the MOGA in each simulation run are generated such that the resulting decision variables are the same. In other words, the initial populations used in the two approaches are equivalent in terms of the decision variables obtained after decoding the chromosomes.

In overall, it can be seen from the results that the MOGA with an integer-based coding scheme has emerged as the most effective method in finding the Pareto front for this problem. This conclusion is supported by both viewpoints on the variety of solutions found and the number of found solutions which cannot be dominated by solutions obtained from the other techniques. Another important point, which can be observed from both case studies, is that nearly all of the solutions found by the random search method cannot dominate the solutions found by both approaches of the MOGA. Since the solutions found by the random search method in this case are the non-dominated solutions from the initial population of the genetic

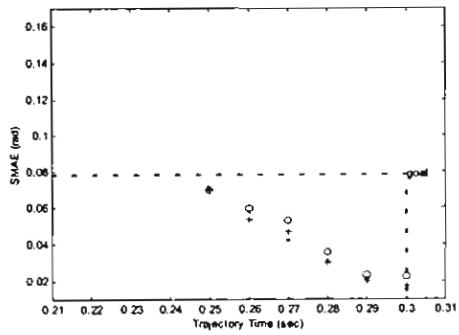
\

gorithm, this indicates that successful evolution has been complished by the MOGA.



x MOGA with a Gray coding scheme
+ MOGA with an integer-based coding scheme
o Random search

Figure 8. Pareto optimal solutions from case 1.



x MOGA with a Gray coding scheme
+ MOGA with an integer-based coding scheme
o Random search

Figure 9. Pareto optimal solutions from case 2.

8. CONCLUSIONS

In this paper, the robotic application which is chosen to illustrate the effectiveness in combining neural networks and a genetic algorithm at the application task level is a time-optimal control application. The task of tracking a straight-line path in Cartesian space is given to the robot in this case. The time-optimal joint trajectory time history is calculated by using the time-optimal control algorithm as described by Shiller and Lu [2]. Time-optimality is achieved by executing a bang-bang control, where the control torque signal in one joint is saturated and the control torque signal on other joints is adjusted accordingly such that the torque limits on each actuator are not violated. However, the trajectory time history obtained from the time-optimal control algorithm is calculated by using only the open-loop dynamics of the robot model. Previously, in order for this trajectory time history to be used as input to the position control loop, the time history had to be modified using trajectory pre-shaping scheme [4]. In this paper, the use of extended Kohonen networks which contain an additional lattice of output neurons as assistants to PID controllers has been proven to be an effective method in compensating for the closed-loop dynamics and modelling errors. This results in being able to use the trajectory time history as the input to the control loop directly without the use of trajectory pre-shaping scheme.

Subsequently, a genetic algorithm has been used to solve a multi-objective optimisation involving the selection of torque limits and an end-effector path subject to time-optimality and tracking error constraints. Two approaches of a multi-objective genetic algorithm (MOGA) have been used in this application, the MOGA with a Gray coding scheme and the MOGA with an integer-based coding scheme. The simulation results suggest that the integer-based chromosome is more suitable than the Gray chromosome at representing the decision variables. This makes the MOGA with an integer-based coding scheme emerge as the most effective method in finding the Pareto optimal solutions for this problem.

REFERENCES

[1] Bobrow, J. E., Dubowsky, S. and Gibson, J. S., 1985, "Time-Optimal Control of Robotic Manipulators along Specified Paths," *International Journal of Robotics Research*, 4(3), pp. 3-17.

[2] Shiller, Z. and Lu, H.-H., 1992, "Computation of Path Constrained Time Optimal Motions with Dynamic Singularities," *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 114(1), pp. 34-40.

[3] Sahar, G. and Hollerbach, J. M., 1986, "Planning of Minimum-Time Trajectories for Robot Arms," *International Journal of Robotics Research*, 5(3), pp. 90-100.

[4] Shiller, Z., Chang, H. and Wong, V., 1996, "The Practical Implementation of Time-Optimal Control for Robotic Manipulators," *Robotics and Computer-Integrated Manufacturing*, 12(1), pp. 29-39.

[5] Shiller, Z., 1996, "Time-Energy Optimal Control of Articulated Systems with Geometric Path Constraints," *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 118(1), pp. 139-143.

[6] Kawato, M., Uno, Y., Isobe, M. and Suzuki, R., 1988, "Hierarchical Neural Network Model for Voluntary Movement with Application to Robotics," *IEE Control Systems Magazine*, 8(2), pp. 8-16.

[7] Chaiyaratana, N. and Zalzala, A. M. S., 1999, "Hybridisation of Neural Networks and Genetic Algorithms for Time-Optimal Control," *The 1999 Congress on Evolutionary Computation (CEC'99)*, Washington, DC, Vol. 1, pp. 389-396.

[8] Fonseca, C. M. and Fleming, P. J., 1993, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," *Genetic Algorithms: Proceedings of the 5th International Conference*, Urbana-Champaign, IL, pp 416-423.

[9] Freund, E., 1982, "Fast Nonlinear Control with Arbitrary Pole-Placement for Industrial Robots and Manipulators," *International Journal of Robotics Research*, 1(1), pp. 65-78.

[10] Martinetz, T. M., Ritter, H. J. and Schulten, K. J., 1990, "Three-Dimensional Neural Net for Learning Visuomotor Coordination of a Robot Arm," *IEEE Transactions on Neural Networks*, 1(1), pp. 131-136.

[11] Fonseca, C. M. and Fleming, P. J., 1995, "Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction," *The 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, Sheffield, UK, pp. 45-52.

[12] Baker, J. E., 1989, "An Analysis of the Effects of Selection in Genetic Algorithms," Ph.D. Thesis, Vanderbilt University, Nashville, TN.

[13] Eschenauer, H., Koski, J. and Osyczka, A. (Eds.), 1990, "Multicriteria Design Optimization: Procedures and Applications," Springer-Verlag, Berlin, Germany.

References

Aleksander, I. and Morton, H. (1990). *An introduction to neural computing*. London, UK: Chapman and Hall.

Anderson, J. A. and Rosenfeld, E. (1998). *Talking nets: An oral history of neural networks*. Cambridge, MA: MIT Press.

Bäck, T., Hammel, U. and Schwefel, H. P. (1997). Evolutionary computation: Comments on history and current state. *IEEE Transactions on Evolutionary Computation*, *1*(1), 3-17.

Baker, J. E. (1989). *An analysis of the effects of selection in genetic algorithms*, Ph.D. Thesis, Computer Science Department, Vanderbilt University, Nashville, TN.

Barto, A. G. (1992). Reinforcement learning and adaptive critic methods. In D. A. White and D. A. Sofge (Eds.), *Handbook of Intelligent Control* (pp. 469-491). New York, NY: Van Nostrand-Reinhold.

Bobrow, J. E., Dubowsky, S. and Gibson, J. S. (1985). Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research*, *4*(3), 3-17.

Bolt, G. R. (1992). *Fault tolerance in artificial neural networks*, D.Phil. Thesis, York University, Ontario, Canada.

Broomhead, D. S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, *2*(3), 321-355.

Chaiyaratana, N. and Zalzala, A. M. S. (1997). Recent developments in evolutionary and genetic algorithms: Theory and applications. *The 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow, UK, 270-277.

Chaiyaratana, N. and Zalzala, A. M. S. (1999). Hybridisation of neural networks and genetic algorithms for time-optimal control. *The 1999 Congress on Evolutionary Computation (CEC'99)*, Washington, DC, *1*, 389-396.

Coello Coello, C. A. (1998). Two new approaches to multiobjective optimisation using genetic algorithms. *Adaptive Computing in Design and Manufacture: The Integration of Evolutionary and Adaptive Computing Technologies with Product/System Design and Realisation*, Totnes, UK, 151-160.

Coello Coello, C. A. (1999). An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. *The 1999 Congress on Evolutionary Computation (CEC'99)*, Washington, DC, *1*, 3-13.

Coello Coello, C. A. and Christiansen, A. D. (1996). *MOSES: A multiobjective optimization tool for engineering design*, Technical Report, Department of Computer Science, Tulane University, New Orleans, LA.

Dissanayake, M. W. M. G., Goh, C. J. and Phan-Thien, N. (1991). Time-optimal trajectories for robot manipulators. *Robotica*, *9*(2), 131-138.

Eschenauer, H., Koski, J. and Osyczka, A. (Eds.) (1990). *Multicriteria design optimization: Procedures and applications*. Berlin, Germany: Springer-Verlag.

Fiorini, P. and Shiller, Z. (1996). Time optimal trajectory planning in dynamic environments. *Proceedings. 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, MN, *2*, 1553-1558.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Genetic Algorithms: Proceedings of the 5th International Conference*, Urbana-Champaign, IL, 416-423.

Fonseca, C. M. and Fleming, P. J. (1995). Multiobjective genetic algorithms made easy: Selection, sharing and mating restriction. *The 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, Sheffield, UK, 45-52.

Foslien, W. and Samad, T. (1993). Fuzzy controller synthesis with neural network process models. *Proceedings. 1993 IEEE International Symposium on Intelligent Control*, Chicago, IL, 370-375.

Freund, E. (1982). Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators. *International Journal of Robotics Research*, *1*(1), 65-78.

Fu, K. S., Gonzalez, R. C. and Lee, C. S. G. (1987). *Robotics: Control, sensing, vision, and intelligence*. Singapore: McGraw-Hill.

Goh, C. J. and Teo, K. L. (1988). Control parameterization: A unified approach to optimal control problems with general constraints. *Automatica*, *24*(1), 3-18.

Goldberg, D. E. (1989). *Genetic algorithms: In search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Haykin, S. (1994). *Neural network: A comprehensive foundation*. New York, NY: Macmillan.

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. New York, NY: Wiley.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbour, MI: University of Michigan Press.

Hollerbach, J. M. (1984). Dynamic scaling of manipulator trajectories. *Transactions of the ASME. Journal of Dynamic Systems, Measurement, and Control*, *106*(1), 102-106.

Horn, J. and Nafpliotis, N. (1993). *Multiobjective optimization using the niched pareto genetic algorithm*, IlliGAL Report No. 93005, Department of Computer Science, Department of General Engineering, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL.

Kawato, M., Uno, Y., Isobe, M. and Suzuki, R. (1988). Hierarchical neural network model for voluntary movement with application to robotics. *IEE Control Systems Magazine*, *8*(2), 8-16.

Kim, B. K. and Shin, K. G. (1985). Suboptimal control of industrial manipulators with a weighted minimum time-fuel criterion. *IEEE Transactions on Automatic Control*, *30*(1), 1-10.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1), 59-69.

Kohonen, T. (1988). *Self-organization and associative memory* (3rd ed.). New York: NY: Springer-Verlag.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464-1480.

Koza, J. R. (1992). *Genetic programming: On the programming by computers by means of natural selection*. Cambridge, MA: MIT Press.

Kuntze, H.-B., Sajidman, M. and Jacubasch, A. (1995). A fuzzy-logic concept for highly fast and accurate position control of industrial robots. *Proceedings. 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, *1*, 1184-1190.

Lei, J. and Jiang, J. (1997). The model reference adaptive control based on the genetic algorithm. *Proceedings. 1997 IEEE International Conference on Neural Networks*, Houston, TX, *2*, 783-787.

Lippmann, R. P. (1989). Pattern classification using neural networks. *IEEE Communications Magazine*, *27*(11), 47-64.

Lo, Z.-P., Fujita, M. and Bavarian, B. (1991). Analysis of neighborhood interaction in Kohonen neural networks. *Proceedings. The Fifth International Parallel Processing Symposium*, Anaheim, CA, 247-249.

Martinetz, T. M., Ritter, H. J. and Schulten, K. J. (1990). Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Transactions on Neural Networks*, *1*(1), 131-136.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115-133.

Michalewicz, Z. (1994). *Genetic algorithms + Data structures = Evolution programs* (2nd ed.). Berlin, Germany: Springer-Verlag.

Minsky, M. L. (1954). *Theory of neural-analog reinforcement systems and its application to the brain-model problem*, Ph.D. Thesis, Princeton University, Princton, NJ.

Minsky, M. L. and Papert, S. A. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Moody, J. E. and Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, *1*(2), 281-294.

Morimoto, T., Torii, T. and Hashimoto, Y. (1995). Optimal control of physiological processes of plants in a green plant factory. *Control Engineering Practice*, *3*(4), 505-511.

Nakatsuji, T., Seki, S., Shibuya, S. and Kaku, T. (1994). Artificial intelligence approach for optimizing traffic signal timings on urban road network. *1994 Vehicle Navigation and Information Systems Conference Proceedings*, Yokohama, Japan, 199-202.

Pan, Q. Y., Huang, W. D., Song, R. G., Zhou, Y. H. and Zhang, G. L. (1998). The improvement of localized corrosion resistance in sensitized stainless steel by laser surface remelting. *Surface and Coatings Technology*, *102*(3), 245-255.

Pfeiffer, F. and Johanni, R. (1987). A concept for manipulator trajectory planning. *IEEE Journal of Robotics and Automation*, *3*(2), 115-123.

Rana, A. S. and Zalzala, A. M. S.(1996). Near time-optimal collision-free motion planning of robotic manipulators using an evolutionary algorithm. *Robotica*, *14*(6), 621-632.

Ritter, H. J., Martinetz, T. M. and Schulten, K. J. (1992). *Neural computation and self-organizing maps: An introduction*. Reading, MA: Addison-Wesley.

Rose, D. K. (1995). *Neural network models for the control of an inverted pendulum*, B.Eng. Thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386-408.

Rumelhart, D. E. and McClelland, J. L. (Eds.) (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*, Vol. 1. Cambridge, MA: MIT Press.

Sahar, G. and Hollerbach, J. M. (1986). Planning of minimum-time trajectories for robot arms. *International Journal of Robotics Research*, *5*(3), 90-100.

Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms*, Ph.D. Thesis, Vanderbilt University, Nashville, TN.

Sette, S., Boullart, L. and Van Langenhove, L. (1998). Using genetic algorithms to design a control strategy of an industrial process. *Control Engineering Practice*, *6* (4), 523-527.

Shiller, Z. (1996). Time-energy optimal control of articulated systems with geometric path constraints. *Transactions of the ASME. Journal of Dynamic Systems, Measurement, and Control*, *118*(1), 139-143.

Shiller, Z. (1997). Optimal robot motion planning and work-cell layout design. *Robotica*, *15*(1), 31-40.

Shiller, Z. and Chang, H. (1995). Trajectory preshaping for high-speed articulated systems. *Transactions of the ASME. Journal of Dynamic Systems, Measurement, and Control*, *117*(3), 304-310.

Shiller, Z., Chang, H. and Wong, V. (1996). The practical implementation of time-optimal control for robotic manipulators. *Robotics and Computer-Integrated Manufacturing*, *12*(1), 29-39.

Shiller, Z. and Dubowsky, S. (1989). Robot path planning with obstacles, actuator, gripper, and payload constraints. *International Journal of Robotics Research*, *8*(6), 3-18.

Shiller, Z. and Dubowsky, S. (1991). On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, *7*(6), 785-797.

Shiller, Z. and Lu, H.-H. (1992). Computation of path constrained time optimal motions with dynamic singularities. *Transactions of the ASME. Journal of Dynamic Systems, Measurement, and Control*, *114*(1), 34-40.

Shin, K. G. and McKay, N. D. (1984). Open-loop minimum-time control of mechanical manipulators and its application. *Proceedings of the 1984 American Control Conference*, San Diego, CA, 1231-1237.

Slotine, J.-J. and Yang, H. S. (1989). Improving the efficiency of time-optimal path-following algorithms. *IEEE Transactions on Robotics and Automation*, *5*(1), 118-124.

Sutton, R. S., Barto, A. G. and Williams, R. J. (1991). Reinforcement learning is direct adaptive optimal control. *Proceedings of the American Control Conference*, Boston, MA, *3*, 2143-2146.

Swiniarski, R. and Zaremba, M. B. (1990). A time-optimal neural network controller for robot manipulators. *Proceedings of the IASTED International Symposium. Modelling, Simulation and Optimization*, Montreal, Quebec, Canada, 173-176.

Trebi-Ollennu, A. and White, B. A. (1997). Multiobjective fuzzy genetic algorithm optimisation approach to nonlinear control system design. *IEE Proceedings. Control Theory and Applications*, *144*(2), 137-142.

Van Veldhuizen, D. A. and Lamont, G. B. (2000). Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, *8*(2), 125-147.

Wang, Q. and Zalzala, A. M. S. (1996). Genetic control of near time-optimal motion for an industrial robot arm. *Proceedings. 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, MN, *3*, 2592-2597.

Widrow, B. (1962). Generalization and information storage in networks of adaline 'neurons'. In M. C. Yovitz, G. T. Jacobi and G. D. Goldstein (Eds.), *Self-Organizing Systems* (pp. 435-461). Washington, DC: Sparta.

Widrow, B. and Hoff Jr., M. A. (1960). Adaptive switching circuits. *IRE WESCON Convention Record*, 96-104.

Woolley, I. And Kambhampati, C. (1997). Intelligent control plug ins for Connoisseur™. *IEE Colloquium on Industrial Applications of Intelligent Control*, London, UK, 2/1-4.

Yamamoto, M., Isshiki, Y. and Mohri, A. (1994). Collision free minimum time trajectory planning for manipulators using global search and gradient method. *IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robot and Systems. Advanced Robotic Systems and the Real World*, Munich, Germany, *3*, 2184-91.

Zhang, P. X., Zhang, Q., Wu, L. and Sui, Z. (1996). Optimization of compositions of MgO-B_2O_3-SiO_2 slags using artificial neural networks and genetic algorithm. *Zeitschrift fur Metallkunde*, *87*(1), 76-78.

Ziauddin, S. M. and Zalzala, A. M. S. (1995). Model-based compensation and comparison of neural network controllers for uncertainties of robotic arms. *IEE Proceedings. Control Theory and Applications*, *142*(5), 501-507.

Ziegler, J. G. and Nichols, N. B. (1942). Optimum settings for automatic controllers. *Transactions of the ASME*, *64*(8), 759-768.