Department
AUTOMATI
CONTRO
and SYSTEMS
ENGINEERING
UNIVERSITY of SHEFFIELD

GENETIC OPTIMISATION AND EXPERIMENTATION FOR THE PUMA 560 MANIPULATOR

Q. Wang and A. M. S. Zalzala

Robotics Research Group
Department of Automatic Control and Systems Engineering
The University of Sheffield, Mappin Street, Sheffield S1 3JD, United Kingdom

Research Report #567
April 1995

Tel : +44 (0)114 2825250
Fax : +44 (0)114 2731729
EMail : rrg@sheffield.ac.uk
RRG Robotics Research Group

GENETIC OPTIMISATION AND EXPERIMENTATION FOR THE PUMA 560 MANIPULATOR

Q. Wang and A. M. S. Zalzala

Robotics Research Group
Department of Automatic Control and Systems Engineering
The University of Sheffield, Mappin Street, Sheffield S1 3JD, United Kingdom
Tel: ++44 (0)114 2825136; Fax: ++44 (0)114 2731729; Email: rrg@sheffield.ac.uk

Abstract:. The search for minimum-time motion of an articulated mechanical arm by tessellating the joint space involves heavy computational burden. In this work, Genetic Algorithms (GAs) are used to tackle this problem while considering different optimisation criteria, namely, minimum motion time, constraints on torque commands and constraints on end-point velocities. In addition, the search algorithm considers all constraints imposed on the manipulator design, including bounds on motor torques. In addition to reporting the simulation results of a number of case studies, experiments are carried out using a transputer-based PUMA arm.

1. Introduction

A long standing issue in robot control has always been how fast can a manipulator execute a task from a start position to another end position. This is not quite straight forward to determine, considering the requirements for different control modules including task planning, motion planning, motion control, perception and intelligence. Although these issues are fascinating for academics from a research point of view, much is of interest to industry as it is directly linked to the productivity of the plant.

Trajectory planning of manipulators requires providing a time-history of motion for the arm to accomplish a specific task. However, there are infinite trajectories, in the joint space, for a robotic manipulator to move from one position to another, and a decision should be made on which trajectory to use according to some agreed criteria. In addition to other criterion, the motion may be optimised considering the travel time, energy consumption and/or environmental constraints. Nonetheless, the need to combine more than one criteria in the optimisation process may prove difficult due to the often conflicting natures of the considered criteria.

Fu et al. [1] discussed the basic concepts on how to plan the manipulator's trajectories in the joint space. Normally, each joint space trajectory is set up following a certain pattern, i.e. *Acceleration-Zero acceleration-Deceleration*, for simplicity. During the *Acceleration* period, the joint gains speed, preferably reaching its maximum (or near-maximum) limit. Then follows a cruising zone during which constant velocity (i.e. zero-acceleration) is maintained. Finally, during the deceleration period, the manipulator decreases its speed and finally rests at its goal position. Typically, a combination of 4-3-4 or 3-5-3 polynomials is used for planning a trajectory where the entire segment is divided into three sub-sections with four points, i.e. start, lift-off, set-down and finish positions.

Optimum trajectory planning was attempted by many researchers. Some of the early works on minimum-time motion used either a simplified dynamic model (e.g., Kahn and Roth [2]) or no dynamics at all (e.g., Lin et al. [3])). In the latter, an attempt was made to find a sequence of time intervals that minimises the total time spent between two points. Nonetheless, the constraints consisted of purely kinematic bounds on positions and their derivatives. Later, a joint-space tessellation and a graph search scheme was presented by Sahar and Hollebach [4], planning for optimal-time motion via an exhaustive search method. The full dynamic model of the manipulator and actuator torque limits were both taken into consideration in arriving at the time-optimal trajectory. However, only a two-joint arm was considered and the authors reported a vast increase in the search time when the tessellated grid increases in size. This approach in [4] appears to be the ultimate solution for this type of planning problem, but seems to be rendered inefficient by the excessive computations involved.

Genetic algorithms are stochastic algorithms mimicking the Darwinian theory of the strife for survival. As an optimisation method, a GA initially generates a finite set of solutions for the problem (i.e. an initial population), each represented by a string structure, followed by an iterative search procedure. During each iteration, random exchange of information among the set of solutions produces a new set of solutions (i.e. a new generation). This randomised, although structured, exchange mechanism exploits historical information to speculate on new search points with expected improved performance. The direction of the search is influenced only by the objective function associated with the individuals' fitness levels. GAs search for optimum solutions globally, thus avoiding being trapped in a local minimum, which is a common handicap in conventional methods [5]. Other benefits in using GAs includes their feasibility to be paralleled on a distributed multi-processor system [6], thus providing for one main requirement in a real-time implementation. For further discussion of the origins and different approaches for using GAs, the reader is referred to Michalewicz's [7] and Goldberg's [5] texts.

In robotics, GAs have mainly been used in path planning and decision-making on collision avoidance. Davidor [8] was one of the first to introduce the use of GAs in the motion planning of robots. In his pioneering work, he used GAs to minimise the Cartesian position error of redundant manipulators. However, due to the nature of the problem, the problem was purely solved based on kinematic analysis and no consideration of the dynamics of the manipulator was necessary.

The Robotics Research Group at the University of Sheffield has investigated the use of GAs for optimum motion of mobile vehicles [9], static arms [10], mobile arms [11] and multi-arms[1]. Although the previous work on static arms [10] considered only a two-joint robot, it was concluded that GAs lead to a tremendous reduction in the planning time. Thus, for a case study similar to that reported in [4] and while considering the dynamic model of the arm, searching a grid of size 10x10 using the GA required one twentieth of the planning time required by the exhaustive search routine.

In this paper, the research on GA motion planning for static arms is further extended for a 6-joint PUMA arm. In addition, the criteria for motion optimisation are minimum-time, bounded-torques and bounded end-point velocity. Further, experimentations are reported for applying the planned motion to the PUMA via a transputer-based system.

This paper is organised as follows. Section 2 briefly describes the PUMA transputer interface. In section 3, the problem of optimum-motion is illustrated for a simple single-ink system. The genetic motion planning procedure is illustrated in section 4, with the dynamic scaling scheme illustrated. Comprehensive simulation results are reported in section 5 for different optimisation criteria. PUMA experimentations are given in section 6, while conclusions are given in section 7.

2. The PUMA transputer interface

The experimentation is carried out on a new transputer-based PUMA control system, which subsequently deserves a brief description.

The controller of the PUMA-560 robot can be enhanced by the addition of an external computer system. Basically, there are two levels in the existing controller (see Figure 1). One is the lower level which consists of a digital servo board, the 6503 microprocessor, an analogue servo board and a power amplifier for each joint. In this lower level, a PD controller which samples motion data at a period of 0.875 ms, while the manipulator dynamics is resolved in the PD controller design. There is an independent loop for each of the arm's six joints. The higher level, which is also called the supervisory level, consists of a LSI-11 computer. This supervisory computer mainly functions as a management system using VAL, and ensures that new data for the lower level are sent every 28 ms. Kinematics transformation, path planning, error handling and man-machine interfacing etc. are all processed by this computer, which presents many limitations.

[1] S. Sun and A. Rana, unpublished works.

Several interface alternatives to the PUMA controller, with or without VAL, are possible and some have already been implemented by various researchers. These alternatives differ in terms of the extent to which the existing controller hardware is replaced and the capabilities of the external computer used (see Goldenberg [12]).

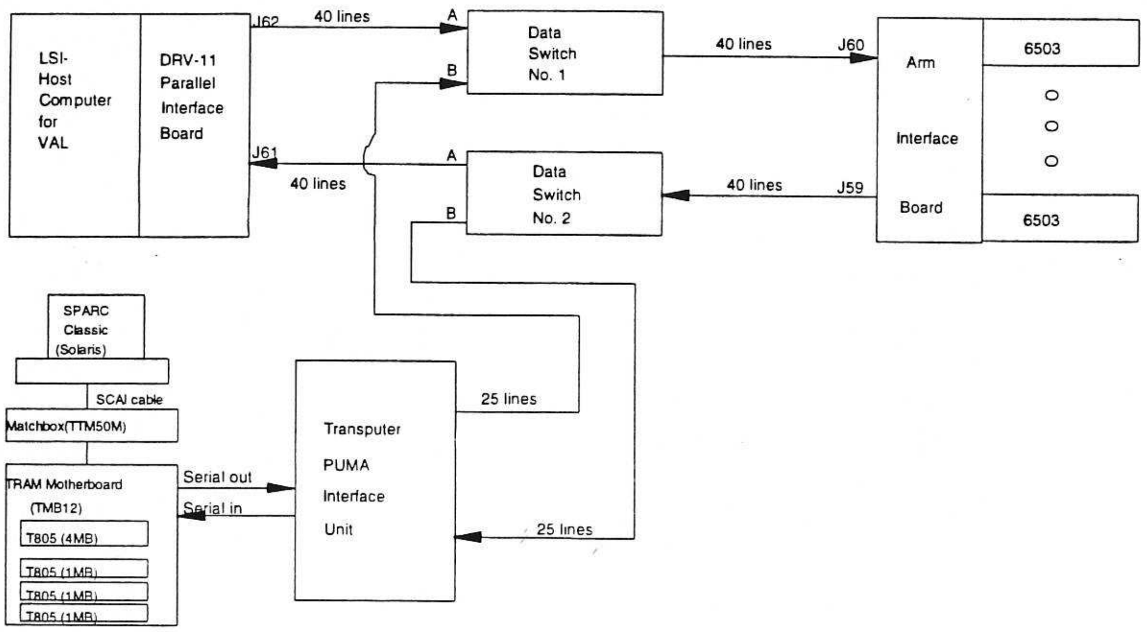


Figure 1 Interface hardware layout

The system described here uses the PUMA's Arm Interface Board to interface with, where the T805 transputer network is used to replace the LSI-11. A Sun-SPARC workstation is used to provide the man-machine interface to develop and store the control software system. The transputer system consists of a motherboard housing TRAM modules which are linked together by high speed (20MBaud) serial links. One of the TRAMs is the master which provides communications with the Sun while managing and controlling the slave TRAMs.

The interactions between the transputer and the out-side world (e.g. other peripherals) are achieved using the Inmos C011 link adapter. The link adapter is currently linked to the master TRAM at the same baud rate as the internal links between TRAMs. The function of a link adapter is to transfer serial signals to parallel ones, which are TTL compatible and can be manipulated according to the user's requirements.

One custom-built hardware required for the new control system is the PUMA-Transputer Interface Board (TIB). Transputers differ significantly from LSI-11 in the hardware used to interface to external devices. The LSI-11 uses a DEC product, the DRV-11 parallel interface board, to communicate with the PUMA's Arm Interface Board (AIB). In order to use the transputer system, the TIB must be compatible with the low-level control section AIB, as a DRV-11 is, to provide proper data and control signal transfer and buffering. Further details on the hardware design are documented [13] including technical drawings[2].

The programming language used to implement the control software is ANSI C running under the Inmos tool-sets, which offers dedicated functions especially for link communications and parallel processing purposes. The whole software system is written in this high level language, which enables it to be easily transported to other platforms for more convenient debugging and developments [13]. The software system is organised in a hierarchical way, where the upper levels can

[2] Q. Wang, unpublished progress report.

make use of the routines in the lower levels to implement more sophisticated functions.

3. Minimum-time Path Planning

This section tries to answer the question asked in the introduction section by demonstrating a simple example. A one joint manipulator, shown in Figure 2, is considered here, which, for a link length of 1 m and a mass of 12.5 kg, has the following mathematical model:

$$\tau = 12.5 * \ddot{\theta} \tag{1}$$

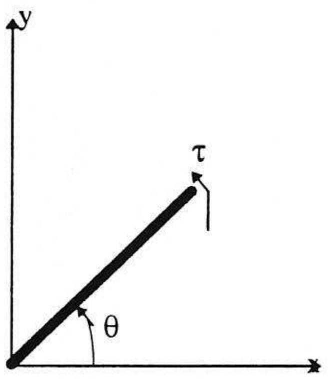


Figure 2 A one-joint arm

The effect of friction is ignored, and no gravitational effects are considered. The motor bound torques are assumed o be ±200 N.m. The physical range of motion is set from $\theta = 0$ to $\theta = \pi / 2$.

It must be emphasised that in order to move the manipulator in the quickest way, the upper-bound torque should be applied all the time. However, although the motion time will be the shortest the joint actuator can achieve, the motion is not quite practical as a predetermined final sate (e.g. zero velocity) is normally imposed by the user. Therefore, a bang-bang motion would be more appropriate to ensure the arm reaching a particular state at the endpoint.

In theory, if there are no friction and brakes, applying a reverse torque will be the only way to bring the manipulator to a rest when it has reached its final position. Thus, for this simple system, the lower-bound torque (i.e. -200 N.m) must be applied. The situation is shown in Figure 3 in the form of the motion profiles for both acceleration and velocity over a time period *T*. Hence, the acceleration can be calculated from equation (1) as a = 200/12.5 = 16 m/s^2.

Considering that the period *T* is shared equally by the acceleration and deceleration portions of motion, the motion equation is as

$$\frac{s}{2} = \frac{1}{2} a \left(\frac{t}{2} \right)^2 \tag{2}$$

where s is the total angular displacement, i.e. $\pi /2$. So the shortest travel time is

$$t = 2\sqrt{\frac{s}{a}} = 0.627 \text{ seconds}$$

Although the dynamics of a multi-joint arm would be much more complicated than this simple case, due to the coupling and coriolis effects, this bang-bang pattern of motion is necessary to achieve minimum-time motion, with one motor (at least) reaching its bounds at any point in time during motion.

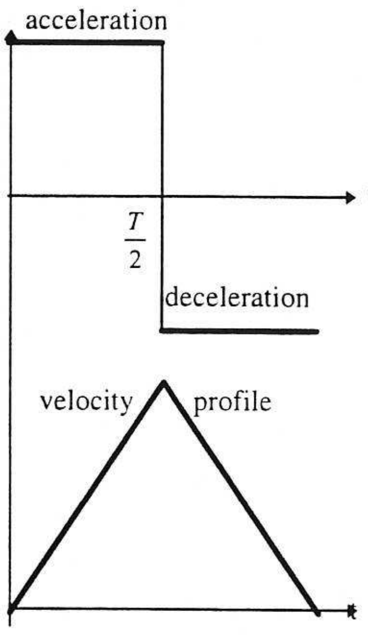


Figure 3 Acceleration and velocity profiles

4. Motion Planning Using GAs

One well established approach for robot motion planning employs a heuristic exhaustive technique to search the work space of the arm. The main idea of the algorithm is to tessellate joint space into a grid of possible motion nodes, where at each option node, given the position and velocity at the previous node, possible velocity values are constrained by the dynamics of the arm. The most comprehensive formulation is reported by Sahar and Hollerbach [4].

However, the complexity of this approach is $O(4^n)$, n being the grid size, which results in rather heavy computations. However, based on the results obtained earlier, GAs are used in this work for the motion planning of the PUMA's six joints in the configuration space.

4.1 Decoding the problem

Manipulator trajectories consist of finite sequence of positions (joint angles) and it is suitable to code these into a string of the format:

$$[\theta_{11},\theta_{21},\ldots,\theta_{n1};\theta_{12},\theta_{22},\ldots,\theta_{n2};\theta_{1m},\theta_{2m},\ldots,\theta_{nm}]$$

where θ_{ji} is the jth intermediate position node of the ith link, n is the number of intermediate position nodes and m is the number of joints.

For each joint, the motion is divided into a *nxn* grid. The joint angles are evenly divided while the time is not as will be discussed in a later section. The motion steps in a trajectory are generated using a relative transitional scheme as shown in Figure 4, where, from any one node, the arm's joint is restricted in moving to only six neighbouring nodes. In addition, the space was tessellated such that a transition has a higher probability of moving towards the end point.

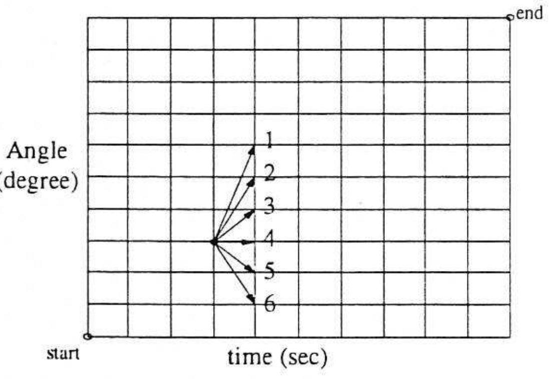


Figure 4 Relative transition scheme

To create the initial population, one trajectory is generated randomly using the relative transition scheme starting from the *start* position, while another starts from the *end* position. If the two meet together, the combined trajectory forms a valid path, as shown in Figure 5.

4.2 The Dynamic Scaling Scheme

A population of trajectories are generated by the method described in the previous sub-section, characterised as trajectory chromosomes. From these trajectories, it is possible to calculate the travel time for each of these trajectories using the dynamic scaling scheme [15], which can be summarised by the following steps.

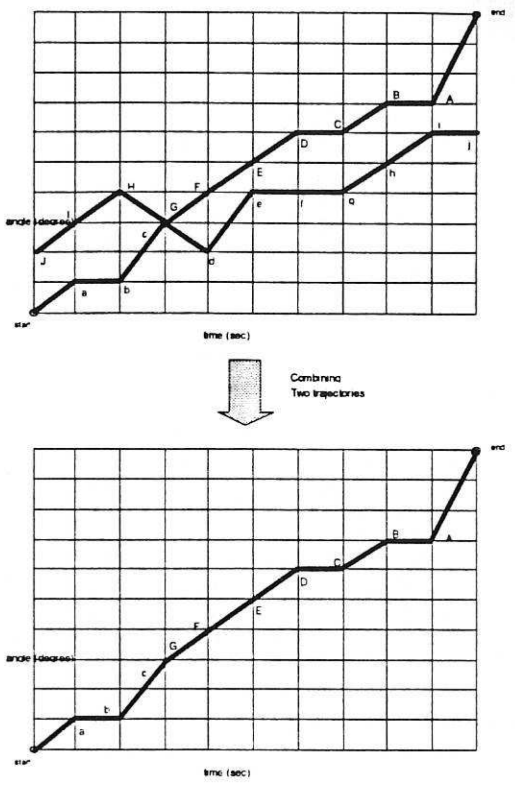


Figure 5 Generating a valid trajectory

1. For a trajectory, the average velocity and average acceleration of the *j*th segment can be calculated as follows:

$$\dot{\theta}_j = \frac{\Delta\theta_j}{h_j} \tag{3}$$

$$\ddot{\theta}_j = \frac{2}{h_j}\left(\frac{\Delta\theta_j}{h_j} - v\right) \tag{4}$$

where $\Delta\theta_j$ is the joint angular displacement of *j*th segment which is decided by the trajectory planning described in the former sub-section, h_j is some time interval chosen before the dynamic scaling. It is chosen in such a way to reflect what have already discussed in section 3 so as to guide the GA search. For the first part of the motion (i.e. acceleration), the interval is set to 0.01, while for the deceleration part of the motion, it is set to 0.5. In [4], the total time interval was chosen as 1. Also, v denotes the final velocity of the previous segment (note that it is different from the average speed as defined in (3)).

2. The calculation has to be carried out for all six joints. Once this being done, six joint torques can then be calculated using the PUMA dynamics package developed earlier by the group. The torque may be too larger (e.g. in the acceleration period) or too small (e.g. in the deceleration period). This illustrates why the dynamic scaling scheme is necessary:

$$(h_j')^2(\tau_b-g)+2h_j'Hv-[(h_j)^2(\tau-g)+2h_jHv]=0 \quad (5)$$

where τ_b is the vector of given torque bounds.

3. Solve for every one of the bounds to find all possible new time interval h_j'.

4. The new torque τ' for the new time interval can be found out by scaling the old torque τ as follows:

$$\tau' = \left(\frac{h_j}{h_j'}\right)^2(\tau-g)+g+2\frac{h_j-h_j'}{(h_j')^2}H.v \quad (6)$$

5. From the time/torque combinations that do not exceed the bounds, choose the shortest time interval for *j*th segment denoted as h_j^s.

6. Recalculate the average velocities and accelerations from equations (3) and (4) using the new time interval.

7. The permitted velocities at the end of the segment can be calculated as:

$$v=\ddot{\theta}\, h_j^s + v$$

8. Carry the above steps until *j=n (end of gird).*

4.3 The genetic operators

An initial population of trajectories is generated, as illustrated in section 4.1. During *reproduction*, the number of occurrences of the same trajectories selected for crossover is limited, which encourages higher interaction among different trajectories. To prevent any path dominating the population leading to pre-mature convergence, only a specific number of copies of the same trajectory are allowed to remain in the population after reproduction, and extra copies are replaced by new trajectories.

Single point *crossover* was adopted as an initial study. After choosing a cross site in one parent string, crossover is performed only if the crossover site of the second parent is within a certain proximity of the circle centred at the first crossing site.

Mutation has a *destroy-trajectory operator* which, when active, replaces the selected trajectory with a randomly generated one so as to give rise to new search space. Another mutation operator, *the position operator*, varies slightly the position of one or more nodes in a path. This operator helps to find trajectories which may or may not be better around a 'good' trajectory found by the crossover operator.

Reproduction was controlled to prevent pre-matured convergence, the analogous crossover directed sensible crossover operations and specially shaped mutation operators promoted new search space. The algorithm has proven to be far more efficient and is about twenty times quicker than the conventional heuristic search technique. This will be further illustrated in section 5.

4.4 The objective functions

4.4.1 Time optimisation

The fitness of a string is assigned as the value of the total time for the PUMA to travel form a start position to a final position. The total travel time is denoted as follows:

$$J=\sum_{j=1}^{n}h_j \quad (7)$$

where h_j is the time interval for the *j*th segment of motion and is calculated by the dynamic scaling scheme, as described in section 4.2. All bounds on the PUMA's joint actuators have been observed, hence ensuring the trajectories are within the arm's capability.

The end velocities should ideally be zeros if the arm is required to rest at the end of the motion. Thus, these velocity constraints are incorporated as *penalties* applied to the objective function

$$J=\sum_{j=1}^{n}h_j+\lambda_v\sum_{i=1}^{6}\left|v_n^i\right| \quad (8)$$

where λ_v is a weighting factor set to 0.1 in this application.

Considering this objective, the fitness of a chromosome is denoted by:

$$fitness = 2.0-\frac{J}{\max J}$$

where $\max J$ is the maximum objective in the same generation of populations.

4.4.2 Torque optimisation

Rather than minimising time, some applications require the minimisation of torque values applied to the arm's motors. In this case, equation (8) is re-written as

$$J = \sum_{j=1}^{n} \tau_j + \lambda_\tau \sum_{i=1}^{6} \left| v_n^i \right| \qquad (9)$$

where a new weighting factor, λ_τ, is introduced with a value of 0.06.

4.4.1 Combined optimisation

It is quite difficult to achieve the minimisation of a combined objective of both time a torques due to the their conflicting physical effects. Thus, an objective function was designed as

$$OBJ = \sum_{j=1}^{n} h_j + \lambda_\tau \sum_{j=1}^{n} \tau_j + \lambda_v \sum_{i=1}^{6} \left| v_n^i \right| \qquad (10)$$

5. Off-line Simulation

In this section, simulation results are reported for the different objective functions given in section 4.4. Considering a single example, the start and end positions for all six joints are given in Table 1.

	J 1	J 2	J 3	J 4	J 5	J 6
Start	-0.3	0.4	-0.18	0.0	-0.05	0.05
End	0.51	-0.42	0.58	0.87	0.64	0.84

Table 1 The motion start and end points (radians)

The GA parameters are chosen as follows:

- The maximum number of similar members in the new population, *Criteria*=4.
- The absolute number of grid variation allowed during an intermediate position mutation, *Variation*=2.
- The maximum number of similar members allowed to crossover during a reproduction, *Samemax*=6.
- The crossover rate, *xor*=0.80.
- The mutation rate for varying intermediate points co-ordinates, *pmutr*= 0.10
- The maximum generation na = 300.

5.1. Case Study 1: Time optimisation with velocity constraints

In this simulation, equation (8) was considered as the search objective. The best objective found was 0.431 seconds at iteration 70.

Although the discrete path points have been acquired using the genetic algorithms, a spline technique (straight line linkage is the simplest linear method) has to be used to link these points together so as to provide the joint angular information in every 28 ms or other interfacing period e.g. 1.75 ms as will be discussed in the next section.

Figure 6 shows the history of the objective value against the number of generations, while Figure 7 includes the position, torque, velocity and acceleration profiles of the minimum-time motion.

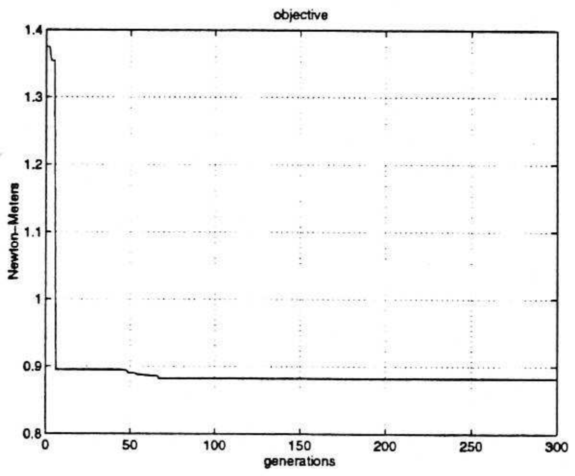


Figure 6 Objective history for case study 1

5.2. Other case studies

To provide for a study of the effect of using different an combined optimisation criteria in robot motion planning, other case studies are included, as indicated in Table 2. The results of section 5.1 is also included for comparison.

One important parameter in any algorithm using grid search is the actual size of the grid representing the searched space. The complexity of the search increases exponentially with the number of points in a chosen grid. Thus, it is always sensible to have a certain trade-off between search resolution and computation time. The results of case study 7 were obtained after running the simulation over around five days, as compared to the other cases for which simulations were accomplished in about one hour. In addition, case 7 was limited to 50,000 iterations to obtain the shown results, as compared to a limited 300 iterations for the other cases. Nonetheless, the increase in minimising the motion time is relatively small (i.e. 0.421 seconds compared to 0.431 seconds) which appears to query if an increase in the grid size is absolutely beneficial.

As expected, all cases where the optimisation is constrained by a near-zero end-point velocity exhibit a higher motion time. The precense of such constrain is important, howeve, if motion is to be planned via succesive segments, as it is the case for a point-look-ahead motion planner.

Case #	Grid size	Optimisation Criteria			Parameters		Motion Time (seconds)	Gener-ations	Simulation Results
		Time	Torque	Velocity constraints	λ_τ	λ_ν			
1	16x16	Yes	No	Yes	—	0.1	0.431	70	Fig.6-10
2	16x16	Yes	No	No	—	—	0.375	255	Fig. 11
3	16x16	No	Yes	Yes	0.06	0.1	0.502	248	Fig. 12
4	16x16	No	Yes	No	0.06	—	0.307	220	Fig. 13
5	16x16	Yes	Yes	Yes	0.06	0.1	0.873	280	Fig. 14
6	16x16	Yes	Yes	No	0.06	—	0.367	285	Fig. 15
7	25x25	Yes	No	Yes	—	0.1	0.421	40,000	Fig. 16

Table 2 Simulation results of different case studies

6. Motion Control Experimentation

Real-time motion control needs to provide the joint controller with the desired position (joint angles) and desired speed (angular velocities) in every control cycle. In the existing technology used in the PUMA, the joint positions are provided every 28 ms, and each joint controller is required to provide motion values at a less control period, usually using linear interpolation. The joint control cycle is accomplished at a very high rate of 0.875 ms.

The PUMA's linear interpolation scheme operates as follows. When a new position data is sent, the microprocessor divides the distance to be travelled into 32 equal increments. During each cycle through the servo loop, this increment is added to the encoder count and servos the joint to that position, thus accomplishing the required motion in about 28 ms.

To provide for this high sampling rate, a very fine grid is needed, which will require a large execution time for the GA planner, perhaps rendering its use as very expensive. Thus, there is a trade-off between search time and search accuracy.

A spline or joint interpolation is thus necessary to provide the joint controller with the desire inputs it need for the real-time control. Although many cubic spline scheme are reported for robot motion panning, [3,14,15], this work uses a simple technique to interpolate the discrete positions computed by the without any auxiliary information being needed, as will be described in the following. Applying the splines interpolation to the results of case 1 (Figure 7), the continuous motion is shown in Figure 17.

Experiments have been carried out using the computed GA-based motion results. The first experiment used the manufacturer defined sampling period, i.e. 28 ms, to motion control the PUMA along the trajectory planned by the GAs. As a result, large jerks were experienced as the joint servos attempted to execute the discrete points. Using the transputer system provided the potential to reduce the default sampling period of 28 ms to a large extent. Thus, a second experiment was carried out at a new sampling period of 1.75 ms. Hence, a much smoother PUMA motion was accomplished with a higher sampling period.

7. Discussion and Conclusions

The contribution in this work is mainly three fold. First, a genetic-based motion planner is formulated for a six-joint articulated arm considering all physical constraints. Second, different optimisation objectives were considered, namely minimum motion time, constrained motor torques and constrained end-point velocities. Finally, preliminary experimenations were carried out using the GA results downloaded to the PUMA via a transputer-based interface system.

In addition to reporting on many simulations incorporating multi-criterion optimisation goals, the GA algorithm allowed for the testing of the transputer-based PUMA interface system designed earlier by the investigators. With this multi-processor system, a great deal of flexibility can be achieved, which is particularly suitable for incorporating sensor-based intelligent system, e.g., vision, force and tactile., as the transputer able to provide the required computational power.

The investigators faced difficulties in attempting a further lower-level of interfacig to the PUMA's control loops. Currently, the joint level controller of the PUMA controller is still in its original design, as very little details are available on how the joint servos perform joint interpolations. In addition, the servo codes are not available with the velocity loop implemented in hardware, which makes it rather difficult to incorporate advanced neural-based algorithms for the dynamic control of the PUMA in real-time. Of course, the lower-level interface can be realised via custom built boards as well, which, in addition to extensive hardware development, defies the aims of this research project.

Further work is progressing in the direction of introducing parallelism within the GA procedure which will allow for incorporating the scheme in real-time.

References

[1]. K S Fu, R C Gonzalez and C S G Lee, "Robotics, Control, Sensing, Vision and Intelligence", McGraw-Hill International, Chapter 4, pp149-200, 1987.

[2]. M E Kahn and B Roth, "The near-minimum-time control of open-loop articulated kinematic chains", AIM 106. Stanford: Stanford Artificial Intelligence Laboratory, 1971.

[3]. C S Lin, P R Chang and J Y S Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots", IEEE Trans. Automatic Control, AC-28, No. 12, pp1066-1073, 1983.

[4]. G Sahar, J M Hollerbach (1986), "Planning of Minimum-Time Trajectory for Robot Arms", Int. J. Robotics Res., MIT Press, Vol.5 No. 3 pp.91-100.

[5] D E Goldberg, Genetic Algorithms in Search, Optimisation and Machine Learning, Addison Wesley, 1989.

[6]. A J Chipperfield and P J Fleming, "Parallel Genetic Algorithms: A Survey", Research Report, Univ. of Sheffield, May 24, 1994.

[7] Z. Micalewicz, *Genetic algorithms + data structures = evolution programs*, Springer-Verlag, 1992.

[8] Y. Davidor, Geneticalgorithms and robotics: a heurestic strategy for optimisation, World Scientific, Singapore 1991.

[9] C.H. Leung and A.M.S. Zalzala, "A Genetic Solution for the Motion Planning of Wheeled Robotic Systems in Dynamic Environments", In *Proc. IEE Int. Conf. on Control*, Vol.1, pp. 760-4 , Warwick, 1994. 493

[10]. A M S Zalzala and K K Chan, "An Evolutionary Solution for the Control of Mechanical Arms", ICARCV'94 3rd Int. Conf. Automation, Robotics & Comp. Vision, 8-11 Nov. 1994, Singapore. 476

[11] M. Chen and A.M.S. Zalzala, "Genetic algorithms for the motion planning of redundant arms",In *Proc. Int Workshop on Advanced robotics and intelligent machines*, Salford 1995. 600

[12]. A A Goldenberg and L Chan, "An Approach to Real-Time Control of Robots in Task Space. Application to Control of PUMA 560 Without VAL-II", IEEE T. Industrial Electronics, Vol. 35, No.2, pp.231-238, May 1988.

[13]. Q Wang and A M S Zalzala, "Real-time Transputer Interface System for the PUMA560 Industrial Robotic Manipulators", Research Report #530, Dept. of AC&SE, University of Sheffield, 1994.

[14]. A De Luca, L Lanari and G Oriolo, "A Sensitivity Approach to Optimal Spline Robot Trajectories", Automatica, Vol.27No.3, pp535-539, 1991.

[15]. S-J Yi and K Kim, "Effect of Tension Parameters and Intervals on Spline-Under-Tension Based Robot Trajectory Planning", J. of Robotic Systems, 11(2), pp91-102, 1994

[16]. J M Hollerbach, "Dynamic Scaling of Manipulator Trajectories", J. of Dynamic Systems, Measurement, & Control, T. of ASME, vol106, March 1984, pp102-106.

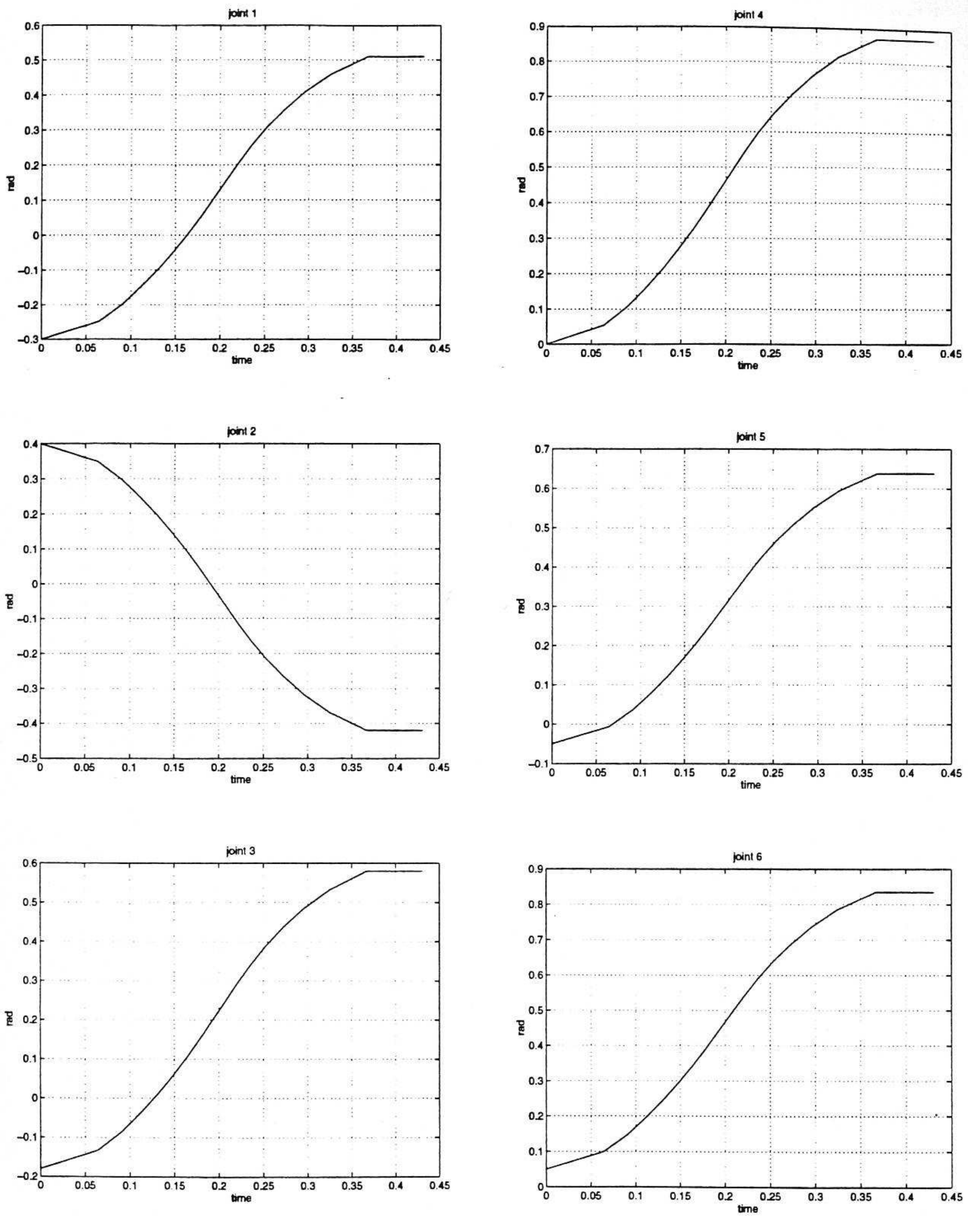


Figure 7
Case Study 1: Minimum-time path for all 6 joints of the PUMA

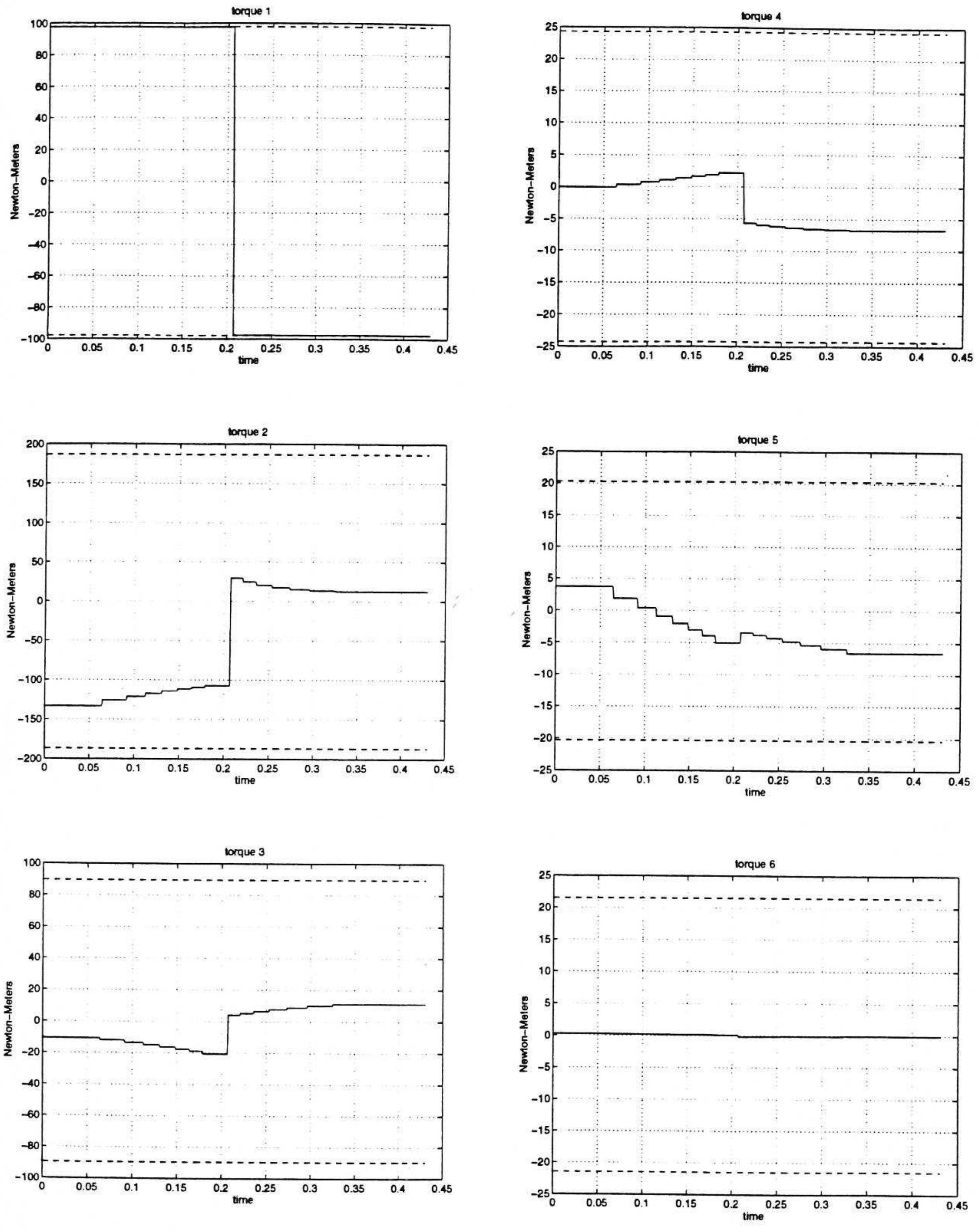


Figure 8
Case Study 1: Joint torque values. (The two dashed lines in each plot represent the bounds. Thus, only joint 1 enjoys a bang-bang motion).

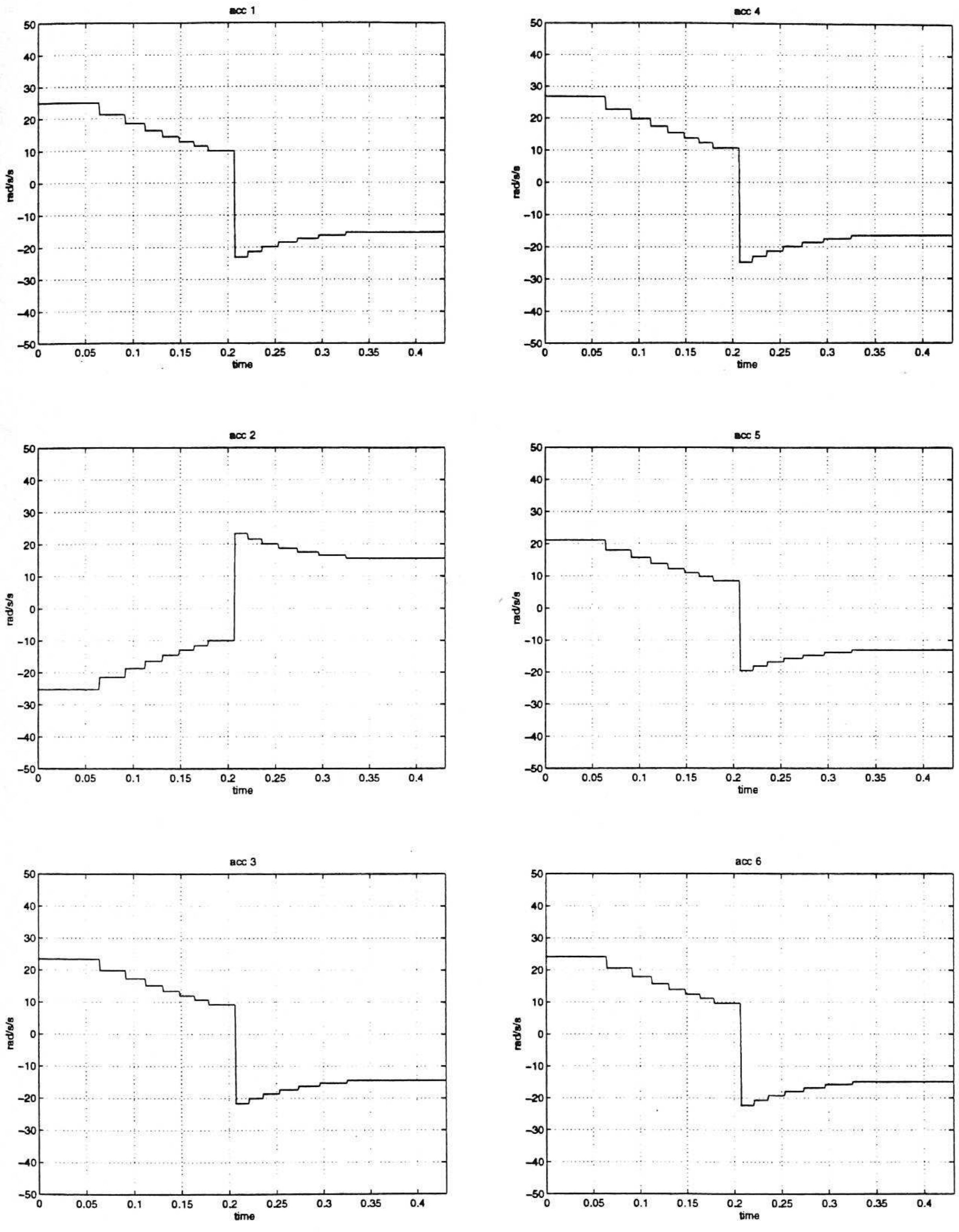


Figure 9
Case Study 1: Joint acceleration values

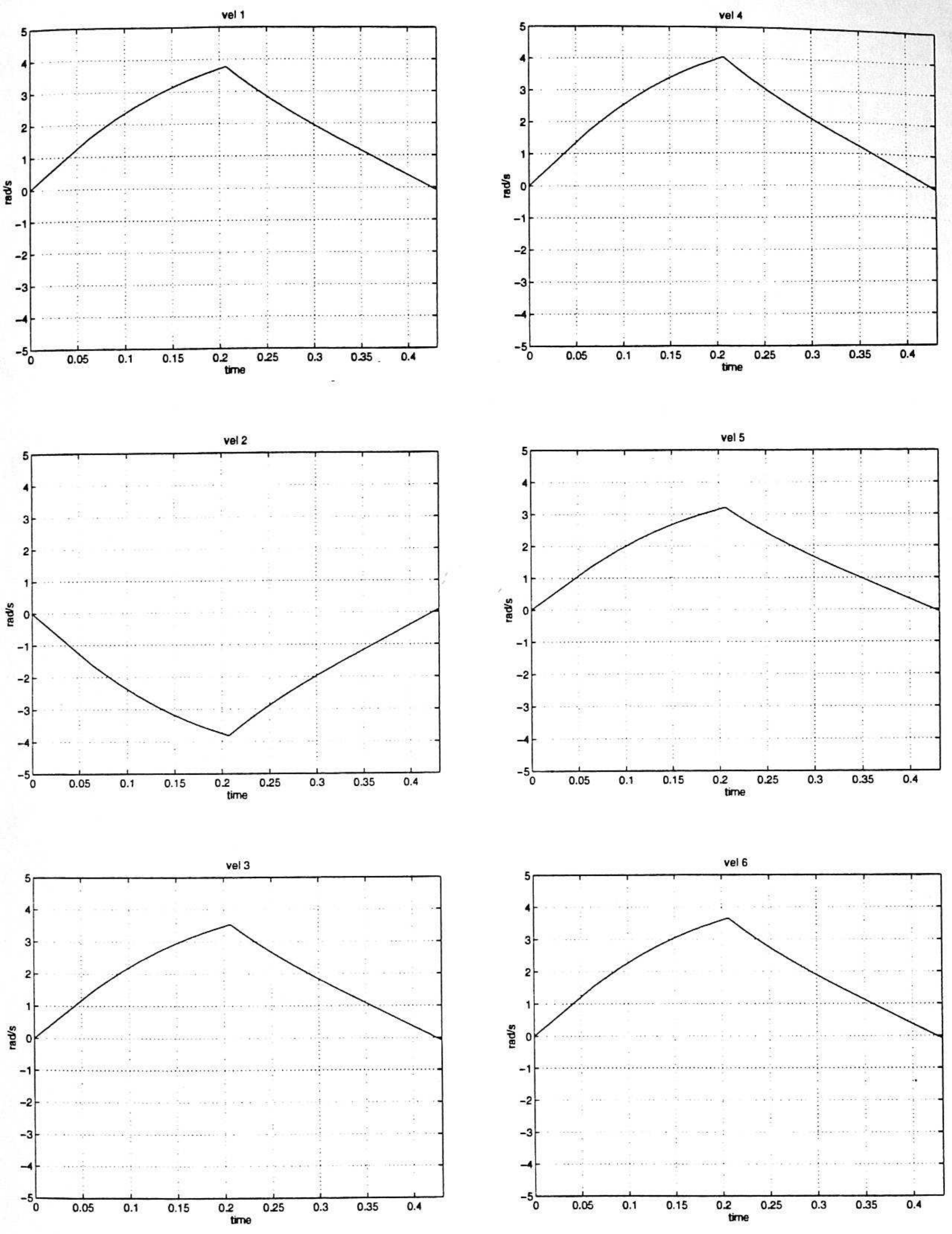


Figure 10
Case Study 1: Joint velocity values

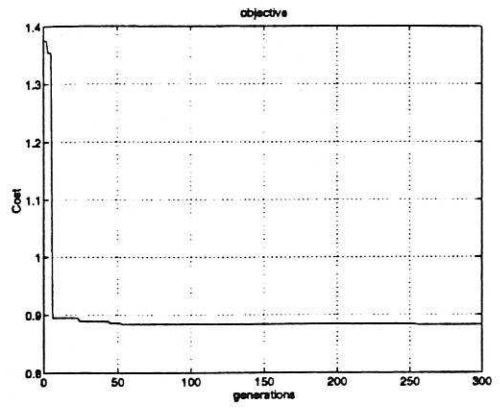


Figure 11(a) Cost function history

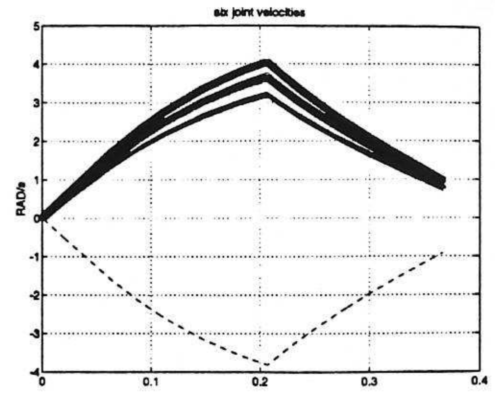


Figure 11(d) Six joint velocities

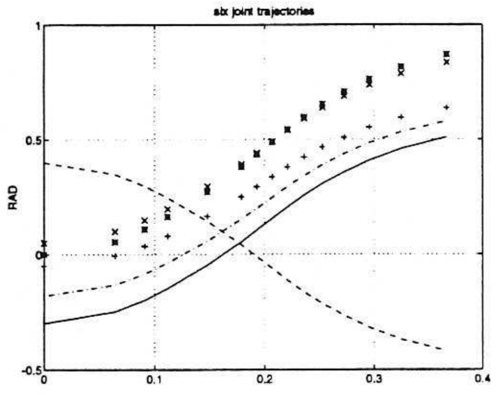


Figure 11(b) Six joint trajectories

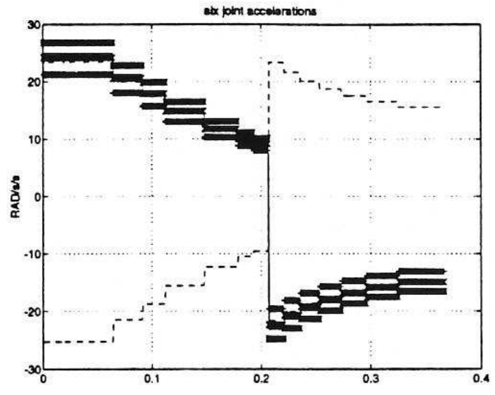


Figure 11(e) Acceleration of six joint

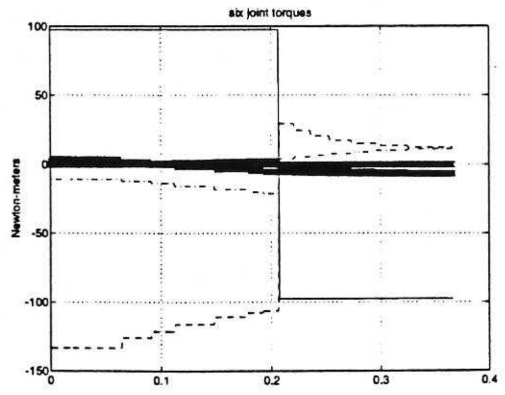


Figure 11(c) Six corresponding torques

Figure 11
Case Study 2: Time without Velocity constraints

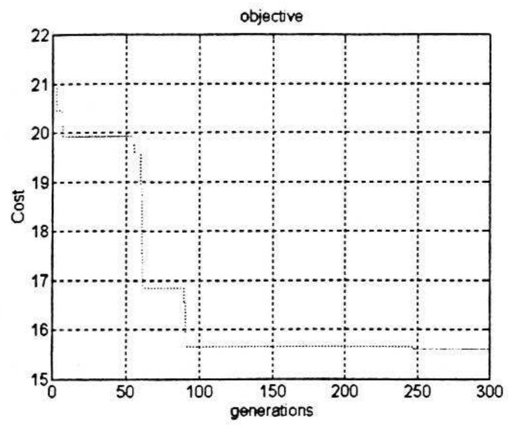


Figure 12(a)

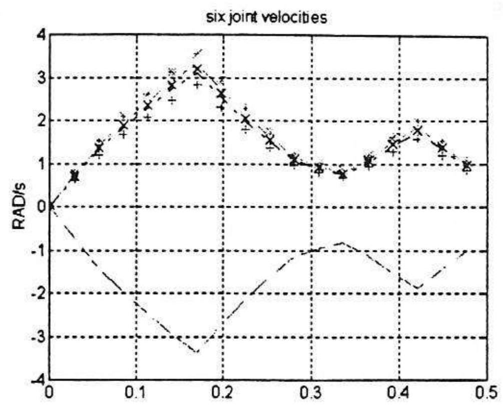


Figure 12(d)

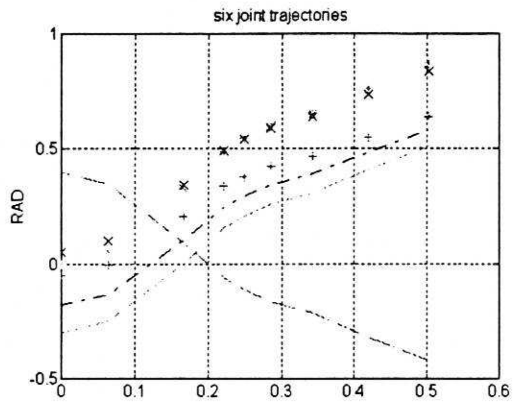


Figure 12(b)

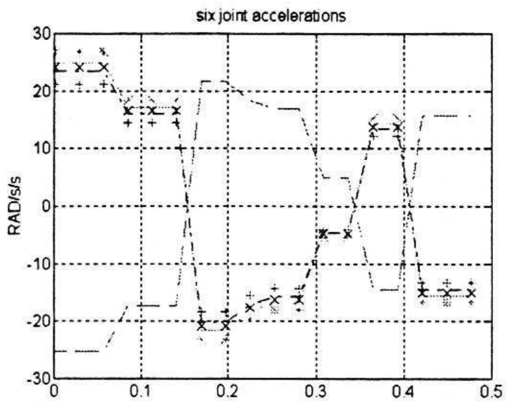


Figure 12(e)

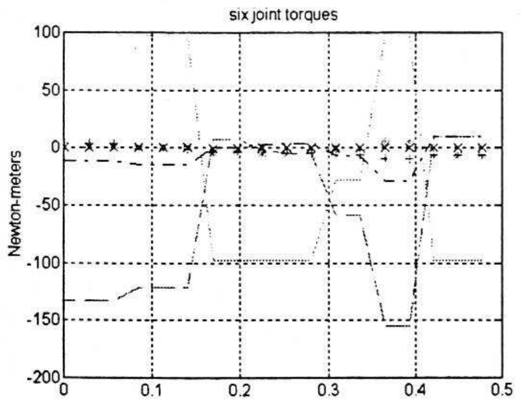


Figure 12(c)

Figure 12
Case Study 3: Torque with Velocity constraints

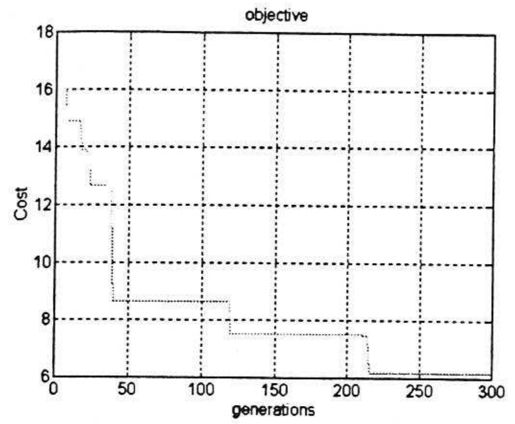


Figure 13(a)

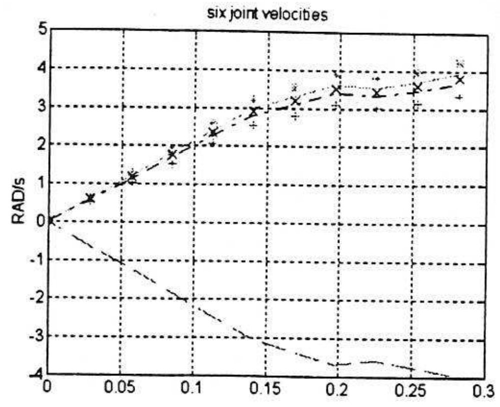


Figure 13(d)

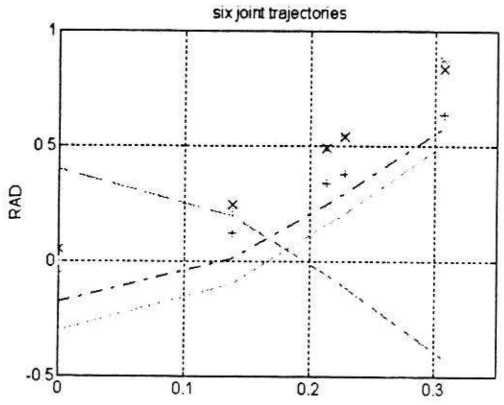


Figure 13(b)

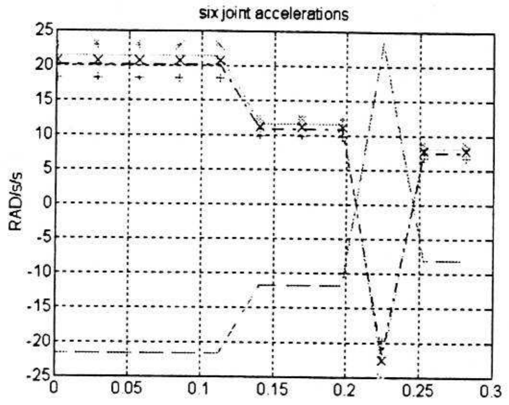


Figure 13(e)

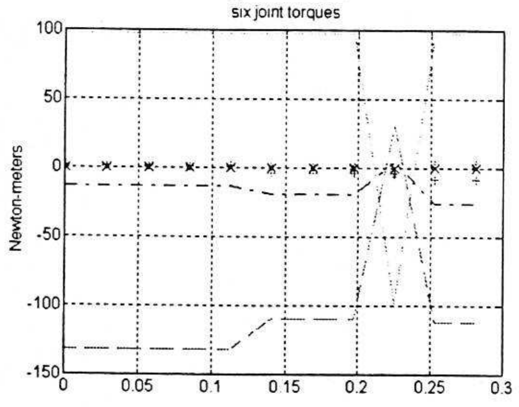


Figure 13(c)

Figure 13
Case Study 4: Torque without Velocity constraints

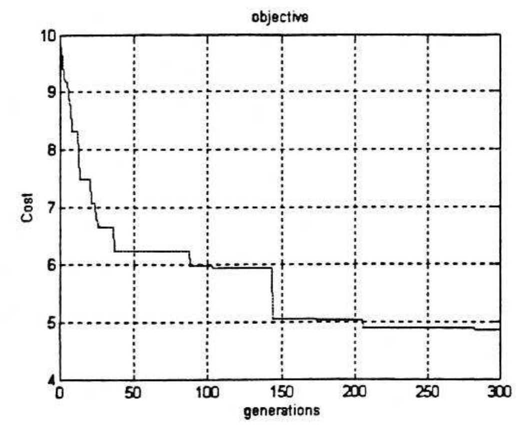


Figure 14(a)

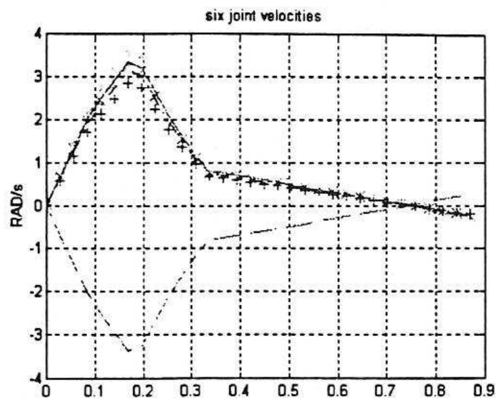


Figure 14(d)

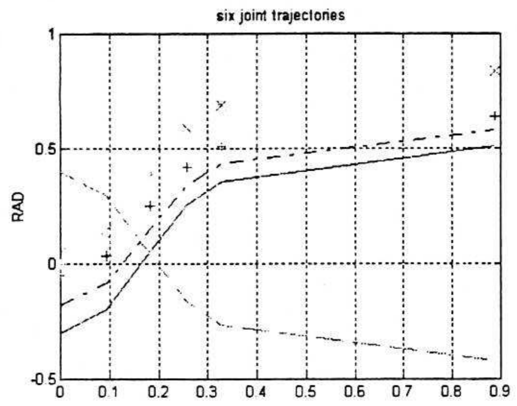


Figure 14(b)

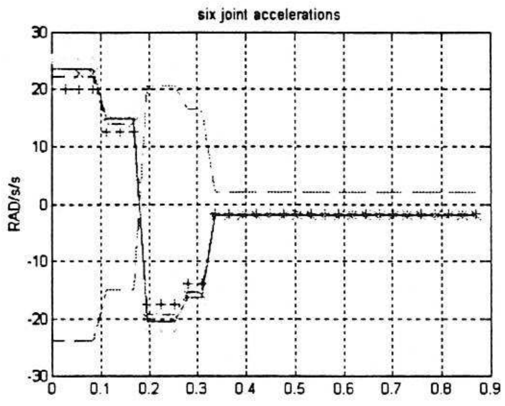


Figure 14(e)

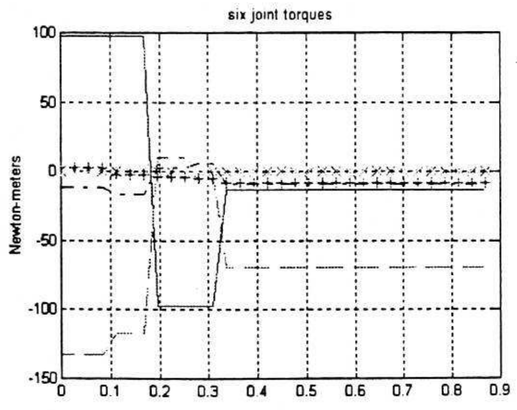


Figure 14(c)

Figure 14
Case Study 5: Time and Torque with Velocity constraints

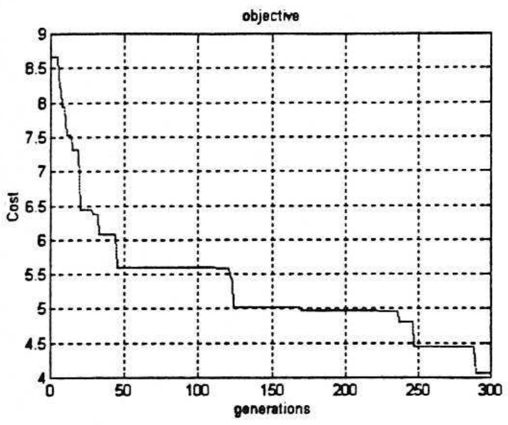


Figure 15(a)

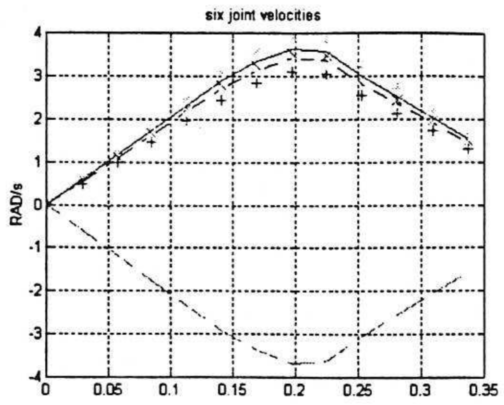


Figure 15(d)

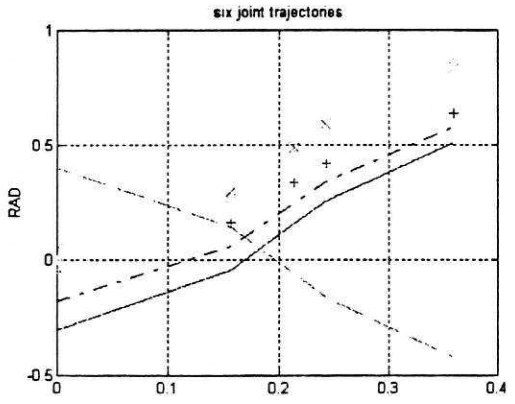


Figure 15(b)

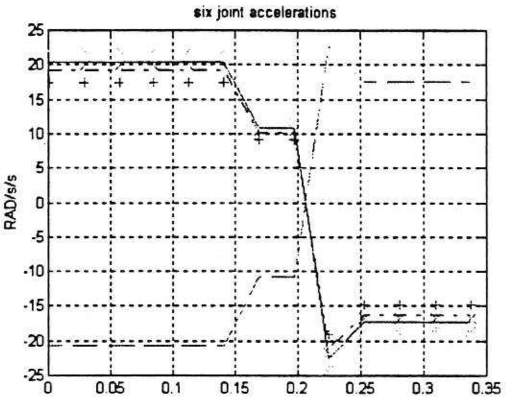


Figure 15(e)

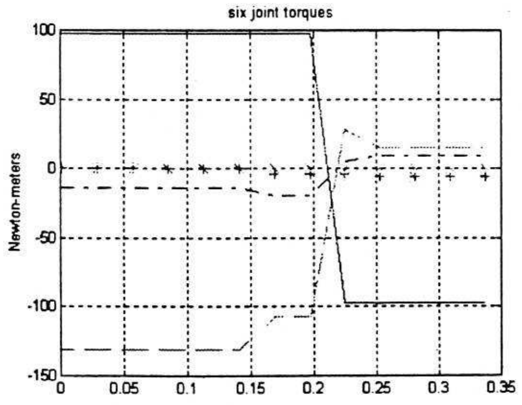
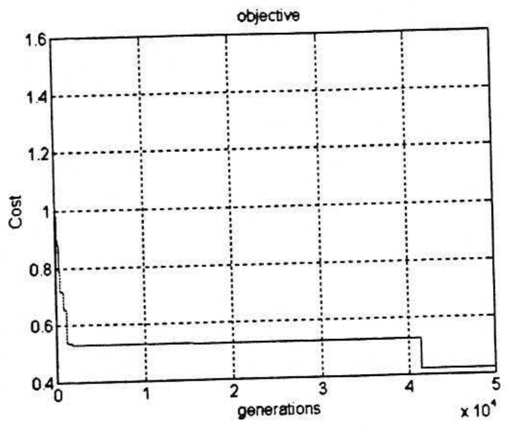
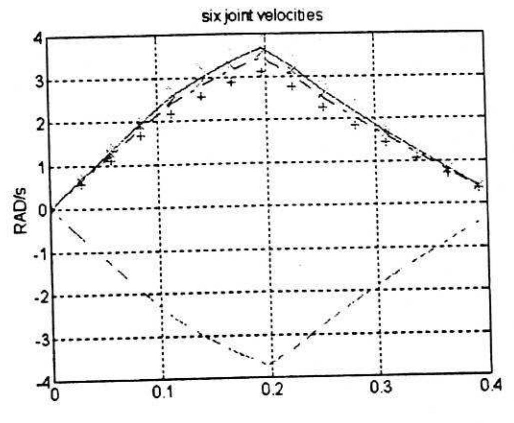


Figure 15(c)

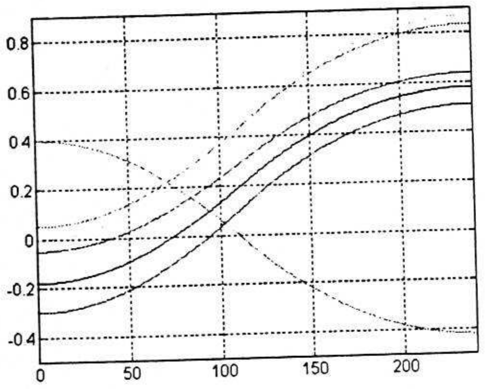
Figure 15
Case Study 6: Time and Torque without Velocity constraints



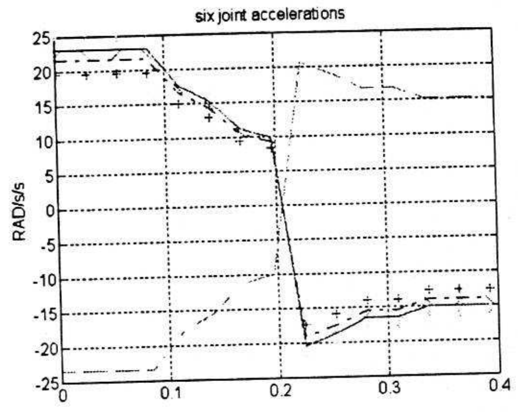
(a)



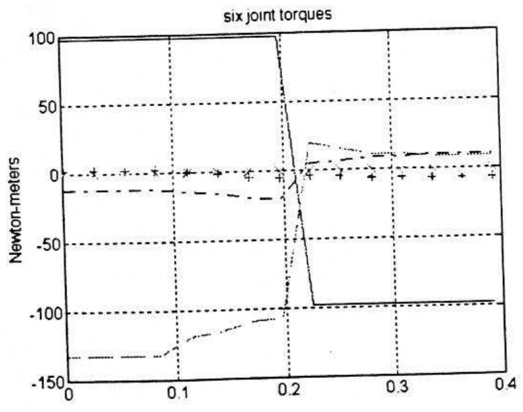
(d)



(b)



(e)



(c)

Figure 16
Case Study 7: Results for a 25x25 grid space

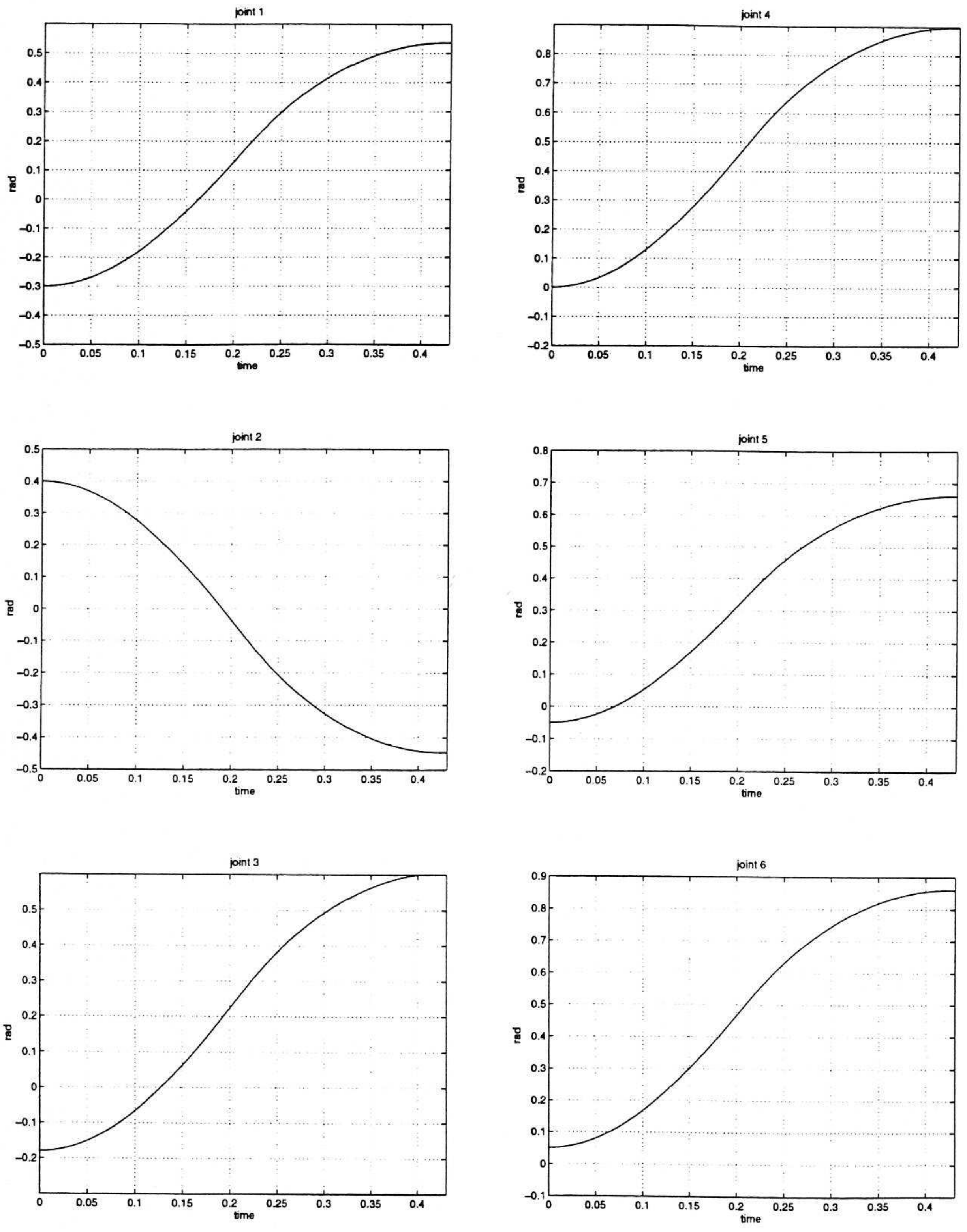


Figure 17
Splined trajectories for Case Study 1 (position profiles)