ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

On-line Learning for Robotic Assembly Using Artificial Neural Networks and Contact Force Sensing

Ismael Lopez-Juarez

A thesis submitted in partial fulfilment of the requirements of
The Nottingham Trent University for the degree of
Doctor of Philosophy

April, 2000

Abstract

Traditionally, robotic assembly techniques have depended on simple sensing systems and the robot manufacturer's programming language, which has severely restricted the extensive use of robots in complex manufacturing operations. The research reported in this thesis is related to the creation of self-adapting robots capable of learning manipulative skills on-line. This work involves the use of Artificial Neural Networks (ANNs) and contact force sensing to "teach" the robot how to behave in poorly structured environments.

An industrial PUMA 761 robot arm was provided for this research by Rolls Royce & Associates, who are interested in autonomous robot operations. The investigation includes the design of a novel Neural Network Controller (NNC), which is based on the Adaptive Resonance Theory (ART) and a knowledge base, whose knowledge is generated by specific assembly operations.

The research used a force/torque sensor attached to the robot's wrist. This was the only sensory information available to the NNC during assembly operations since the precise location of the components was unknown. The communication with the robot controller was made through a PC master-slave architecture, which provided data acquisition and control in real-time.

The design of the NNC was founded on ART's strength to learn incrementally in combination with a dynamic knowledge base. Initially, the robot was provided with a Primitive Knowledge Base (PKB), which contained a minimum set of *primitive* contact force conditions and the corresponding motions to reduce these forces. The knowledge is enhanced on-line, based on the success in predicting the motion that reduces the constraint forces. New knowledge information is only accepted in the PKB when it has contributed strongly towards the success of the assembly. The robot actually enhances its overall assembly performance which is measured by a reduction in assembly time. Additionally, mistakes made earlier do not recur, which demonstrates the new expertise acquired by the robot.

The results also demonstrate the generalisation capability of the NNC by learning the assembly of different part geometries using the same PKB. The overall results show the effectiveness of the methodology and clearly define the requirements for implementing the skill acquisition onto other industrial manipulators, hence, providing an important contribution to the creation of new self-adapting robots with on-line incremental learning capability.

To My Wife, Rosario – Thanks for your patience;

To My Daughter, Celine – Thank you for having learnt to call me Papa by phone, enjoy the wonders of life!.

Acknowledgements

This research has occupied a very important part in my life to myself and my family. I have to definitely acknowledge the support and patience of my wife, who was there always to encourage me, (although there were complaints too!), but more love and tenderness than anything else. Thanks.

I also owe a big thank you to my parents, brother and sisters who always in some manner suffered the ups and downs during my stay in the UK.

There have also been many people involved in this research that helped me in different aspects during this investigation and whom I wish to acknowledge:

- My supervisor, Martin Howarth for giving me the opportunity to carry out this research and his support during these 4+ years.
- The Nottingham Trent University, the Consejo Nacional de Ciencia y Tecnologia (CONACyT) for their financial support and Rolls Royce & Associates for providing the manipulator.
- Past and present members of the MARG group. Thanks are due to Dr Balendran and Dr Sivayoganathan for their support. Special thanks go to Nathan Chandler, for his valuable disscussions about ART. Also thanks are due to Collin D'Souza for the fruitful discussions.
- The support of Mr. & Mrs. Corlett is also acknowledged for the access to research materials. The support of the technical staff within the Department is also acknowledged.
- Thanks to David Glennie for his assistance in communicating the robot. Finally, to Steve, Askary, Warren, James for partially reading this manuscript.

Contents

List of Figures

List of Tables

List of Abbreviations

The abbreviations used in this thesis are explained at the point of first reference. However, they are reproduced here for the benefit of the reader.

ADC	Analogue to Digital Converter
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
CMOS	Complementary Metal Oxide Semiconductor
DAC	Digital to Analogue Converter
DDCMP	Digital Data Communication Message Protocol
DOF	Degrees Of Freedom
DSP	Digital Signal Processor
EKB	Enhanced Knowledge Base
F/T	Force/Torque
FA	Fuzzy ART
FAM	Fuzzy ARTMAP
LTM	Long Term Memory
NNC	Neural Network Controller
OAE	Orthogonal Angular Error
ODE	Orthogonal Displacement Error
PD	Proportional Derivative
PID	Proportional Integral Derivative
PKB	Primitive Knowledge Base
PM	Primitive Motion
PWM	Pulse Width Modulation
RAC	Robot Assembly Controller

RCC	Remote Centre Compliance
RL	Reinforcement Learning
SM	Sensory Memory
SOM	Self Organising Maps
SRV	Stochastic Reinforcement Value
STM	Short Term Memory
VDT	Video Display Terminal
VDU	Video Display Unit
VLSI	Very Large Scale Integration

Chapter 1

Introduction

The majority of industries have a requirement for assembly at some stage of their operations. Martin-Vega et al. [1] have identified in a survey of industrial companies in the USA that 20% of unit production cost is attributed to assembly. One of the objectives of this research is to improve the fundamental understanding and capability of robotic assembly. The research is intended to apply new knowledge in an effective way to enable robots to work and assemble components in poorly structured environments by using sensor data and pre-learnt manipulative skills. The work will contribute to manufacturing industry by improving the capabilities of robot assembly systems through the application of novel learning and manipulative skills.

Traditionally, robotic assembly techniques have depended on simple sensing systems and the robot manufacturers programming language, which has severely restricted the application of robots to complex manufacturing operations. In this project Artificial Neural Network (ANN) techniques and force sensing are used to generate self-adapting robots.

The research is concerned with methods by which components and their interaction with other components during the assembly operation can be described. These descriptors can then be implemented in the learning environment which, combined with sensory information, enables the robot to recognise the interactions between components. This provides the robot with the ability to automatically compensate for typical assembly errors. In other words, the robot is

"taught" to recognise the force patterns that occur during assembly employing Force/Torque sensors. In this manner, the robot is able to perform mechanical assembly by using force sensing information and compensate for misalignments between components which ocurr at the start of the operation.

Early work by V. Balendran [2] used ANN's to assess motor vehicle components. This was adapted and extended by M. Howarth [3] to enable robot manipulators to recognise 3D sensor data collected during mechanical assembly sequences. These techniques developed for inspection and machining by the Manufacturing Automation Research Group within the Department of Mechanical and Manufacturing Engineering have therefore been widened into the field of assembly using sensor data to embed robots with self-adapting behaviour.

The research presented in this thesis compares favourably with work by V. Gullapalli et al. [4] at the University of Massachusetts, who used an ANN to teach a robot *peg-in-hole* tasks, but where the target point had to be known in advance to enable the training phase to be completed. Similarly, the work published by H. Asada [5] at Massachusetts Institute of Technology has shown the possibility of teaching a robot manipulative skills by the recording of manual operators' actions. The investigation for this research differs from Asada's approach by the use of contact conditions and on-line incremental learning to enable manipulative skills to be developed by the manipulator. In addition, the precise location of the assembly parts is unknown at all times.

In the following sections, an overview of force control in robotics is provided, highlighting the issues involved during assembly. Attributes of connectionist models that allow their implementation in the robotic system to perform the assembly are introduced. The scope of the research is established followed by the review of the aim and the author's original contribution. Finally, the organisation of this thesis is described.

1.1 An Introduction to Force Control in Robotics

Most of the current research in robotic assembly is devoted to force and position control. There are many approaches, some of them based on classic control, adaptive control and more recently, neural control. The type of controller to be used depends greatly on the specific application. In certain circumstances only one parameter is important, for instance, position (e.g. motion in *free space* to reach the target point). In other circumstances force may be important (e.g. handling soft materials), but in most cases a combination of both, position and force control, is needed.

Traditional control methods based on classic control architectures have proved to work well in a range of processes. However, they are robot oriented techniques rather than task oriented, and programs written for a specific operation have to be modified if applied to a different robot. Algorithms also rely on *a priori* knowledge of parameters such as part geometry, robot arm stiffness, environment[1] stiffness, etc. However, in real-world systems these parameters change and other tools are needed to analyse and control non-linear behaviour.

In some industrial processes such as assembly and metal removal, an accurate force tracking control is essential to achieve a successful operation. In assembly, force tracking is important since it can prevent mating pairs from being jammed or broken and it can also be used for component alignment. Force control may also avoid any damage to the end-effector of the robot, i.e. gripper[2]. In other operations such as grinding, deburring and polishing, the normal contact force is related to the metal removal rate and wheel wear, hence a precise force must also be applied[6, 7].

There are different approaches to control the above processes and it can be said that force/position control falls into two categories, hybrid control and impedance control[8]. In hybrid control, a switched control structure between constrained and unconstrained motion should be considered. Gain parameters

[1] In the robotics jargon, the term "environment" is widely used and refers to the objects the robot make contact with, namely, assembly parts, tool changers, obstacles, etc.

[2] The terms end-effector and gripper will be employed indistinctly throughout this thesis.

must be adjusted according to the type of movement and depending on whether the end-effector is acting on constrained motion (i.e., contacting the surface), or free-motion. Extensive work has been carried out on this area [9, 10]. On the other hand, impedance control does not treat force and position separately, but it takes the relationship between both, based on the sensed forces. In either control method, the strategy to correct a planned motion namely, *compliant motion*, has to be able to suppress any bouncing at the contact surface. This transient stage is difficult to control due to a lack of knowledge about the environmental stiffness and robot stiffness which varies according to each robot configuration and position in its working envelope. This unavoidable stage introduces noisy information to the system controller, therefore a careful design and study of this input information is a preliminary task. In both hybrid and impedance control, motion compliance is achieved by mapping force input values to motion/torque commands, which eliminates the non-linearity involved.

In the above techniques there is a trade-off between error in motion, error in contact and switching controllers. This non-linear mapping can be solved by discriminating and classifying contact points (i.e. contact force states) during assembly. It is precisely in this stage where an ANN can be used due to its capability to *associate* and *generalise* all contact force states during the assembly process. These contact states can be grouped according to certain common characteristics to facilitate the decision stage, which ultimately selects the appropriate motion towards the end condition.

1.1.1 Force feedback control

The general case of robot feedback control is shown in Figure 1.1. Motion commands are applied to the robot arm via the coordinates conversion stage and joint servos.

This conversion calculates the *Inverse* and *Forward* kinematics of the arm. The tool position (also called *end-effector* position), moves as directed by these commands towards the desired position and modifies the contact forces with the environment. This change and any disturbance is then measured in the feedback

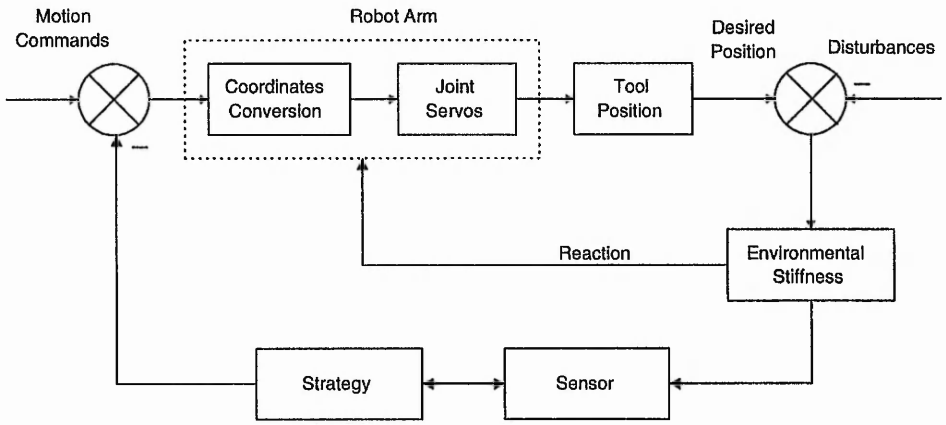


Figure 1.1: Force feedback structure

loop by the force sensor. The new contact forces and arm position are then used in the strategy algorithm to move the arm towards a desired position, so the previous motion command has to be altered accordingly. It is important to note that due to contact forces with the environment, a reaction force will appear on the arm joints. However, the force feedback algorithm will not consider these forces directly because the robot controller is limited to a built-in positional PID controller for each servo motor. Therefore, it is easier to control contact forces at the end-effector and to consider the arm working in a linear region rather than controlling the arm in joint-space. Controlling the arm in the joint-space would imply the implementation of a sensing system on each joint of the arm increasing the complexity of the control law.

More research has been conducted in the area of impedance controllers because this type of control is smoother than the hybrid method. S. Lee developed a model of a Generalised Impedance Controller (GIC) [8]. Its performance is based on a trade-off between motion and contact force error. S. Payandeh [11] considers the environment to be a spring-damper system and to support his theory of compliance on environmental compliance, he used a 2 DOF arm robot to demonstrate that the system reduced the error under disturbances whilst maintaining a constant contact force. The disadvantage with this model is its need for *a priori* knowledge of the environmental stiffness. Seul Jung [12] used contact force information at the end-effector and formulated a relationship that also took into

account the environmental stiffness.

Hybrid control considers a switching strategy between force and position control i.e., the arm is position controlled during motions in free-space or any other direction different from the main direction. During contact with the environment or during operations the arm is force controlled, therefore switching between both strategies is required.

1.2 Force Tracking and Compliance

Force tracking on the end-effector is important in applications where motions in constrained space have to be maintained within a certain range during compliance. The term *compliance* refers to manipulative tasks which involve continuous contact between the manipulator and the environment. There are two classifications within compliant motion. *Passive compliance* (sometimes referred as natural compliance), in which contact configurations depends on mating pair geometries, and *active compliance* applied to aid mating, which is based on sensor readings and active control of the manipulator.

1.2.1 Peg-in-hole operation

The most studied operation for analysing compliant motion in insertion tasks is the peg-in-hole operation. In fact, this is the most common operation in assembly and has also been set by many roboticists as a canonical test operation. There have been many researchers who have studied contact forces acting during this operation [13, 14, 15, 3, 4, 16, 17]. The four stages during part mating in the peg-in-hole operation as identified by Daniel E. Whitney [18] are shown in Figure 1.2. The figure shows the peg approaching the hole, contacting the chamfer and when the peg contacts inside the hole in one and two points. Whitney also analyses these forces identifying failures during operation such as *jamming* and *wedging* and diverse relationships between contact states. Jamming is a condition in which the peg will not move because the forces and moments applied to it through the support are in the wrong proportions. It can occur during one or

two contact points and is due to the resultant force being within the *friction cone* and increasing the applied force will not cause the peg to slide. To exit from this condition, a force in the opposite direction is needed. On the other hand, wedging is worse than jamming because it may involve deformation in one or both parts therefore, removal of the applied force may not remove the reaction forces.

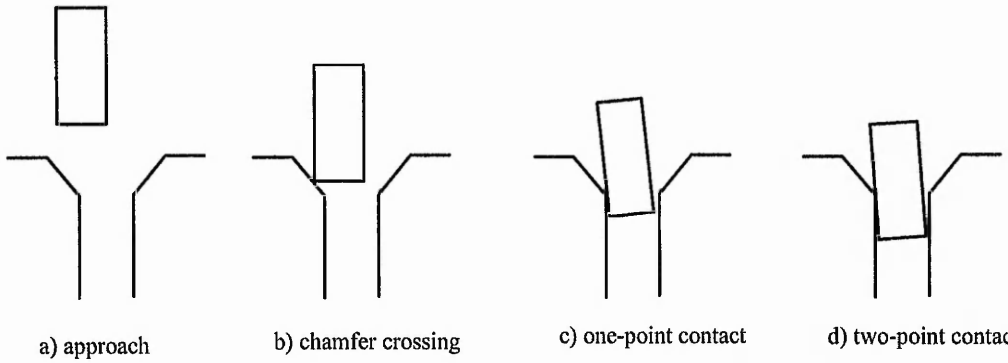


Figure 1.2: Four stages during peg-in-hole operation

1.2.2 Use of a passive compliance device

The *Centre of Compliance* is defined as a point located at the tip of the peg that will not be affected by rotation if a force that produces a pure translational motion is applied and vice versa. Hence, to facilitate compliant motion, a strategy such as tilting the peg can be used. From the analysis of the jamming, wedging and the definition of Centre of Compliance, a mechanical device named *Remote Centre Compliance (RCC)* was designed to overcome misalignments between mating pairs [18]. The RCC can be thought of as a mechanical spring that permits lateral motions in response to laterally directed contact forces without angular motion. Similarly, angular motions are permitted in response to applied torques but no lateral motion is developed. This device is depicted in Figure 1.3.

With the use of the RCC, positional errors can be eliminated, within a certain misalignment range and for specific part sizes. However, out of this range or working with longer mating pairs, either the RCC has to be changed or an alternative procedure sought. Research in RCC design has looked at these limitations and produced a variable RCC [19] that improves the performance of the fixed

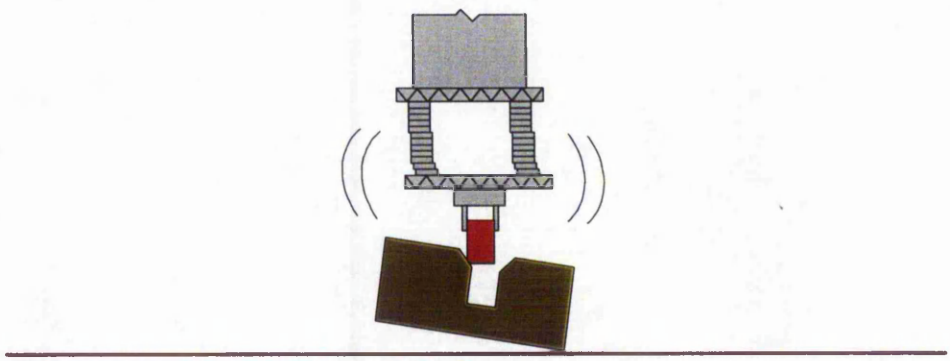


Figure 1.3: Remote centre compliance device

type RCC. In this device, the compliance centre can be modulated via one or two actuators, or it can be modified manually. The main advantage of this device is that different length of parts can be assembled without changing the RCC.

The RCC is a passive device hence, force sensing and monitoring capability are not present and therefore it cannot provide sensory information needed in a learning assembly system. However, it is mentioned here due to its practical importance for correcting passively the misalignment between parts.

1.3 Task-Level Programming and Fine Motion

Robot programming languages have to deal with external objects, which generate significant problems due to their inconsistency. In non-robot oriented languages, variables are manipulated within the same computer therefore they are not affected directly by external events. For these languages, control of the variables is easier than in robot oriented languages because they do not have to deal with uncertainties. Robot systems must deal with workspace uncertainties which come from a variety of sources such as accumulated errors due to part tolerances, dimensions of parts, gear backlash, ageing of arm mechanisms, etc. Unpredicted errors such as the sliding of mating pairs in the gripper and misalignments have also to be considered. Furthermore, robot programs are for a specific robot and cannot easily be transferred to other robots or workspace.

Motion planning involves *gross motion* and *fine motion*. The former deals with

motions of the end-effector in free-space avoiding any collision with the environment whereas the latter considers a *quasi-static* motion in contact with the environment. Fine motions are related to the aims of this research since they are necessary for insertion. Furthermore, fine motion makes manipulation safe in the presence of uncertainties.

There has been research in this area, especially to create strategies in compliant fine-motion such as those developed by Tomas Lozano-Perez, Mattew T. Mason and Russell H. Taylor [20]. The strategy LMT, named after the authors, synthesised fine motion strategies which are based on the view that the structure of a fine motion strategy for a task is determined by the set of geometric interactions that can occur during the execution of the task. Their strategy considers *configuration space* frames. These frames specify the motion at a point, for example, in the *compliance frame* that can move with freedom during an insertion operation. Work done by J. De Schutter and H. Van Brussel [21] [22] extends the LMT work by showing other formalisms in specifying compliant motion that require hybrid approaches. It achieves a strict separation between control and programming, which is important for the integration of compliant motion to a robot programming language.

Control architectures have been used widely and with success in different applications. L. Liu et al. implemented a PID controller on a PUMA robot in grinding operations, demonstrating improved profiles of finished workpieces [7]. Jeoun and Tomizuka applied a feedforward compensator on a SCARA robot using a deburring tool [6]. S. N. Gottschlich and A. C. Kak, [17] applied compliant motion to insert a peg into a hole. For these operations, authors also have developed geometric models using passive compliance [13, 14].

The methods mentioned above are either robot or geometry based (henceforth, model-based solutions). In these models, there has not been a unified criteria to specify task-level programming independent of the robot or environment and parameters have to be measured or determined for each individual element in the model. Geometry of the mating pairs is known, and parameters such as environment stiffness are considered as known *a priori*. However, there is still a lack of

such criteria to generalise task-level programming, to improve operations and to speed up programming. Research continues in this area looking for alternative approaches, such as those based on AI techniques and Artificial Neural Networks.

1.4 Connectionist Models and Robotics

Model-based methods do not offer a complete solution to the problem due to the complexity associated with mechanical assembly operations. In the particular situation of mating pairs for assembly, the number of contact states cannot easily be defined and therefore an alternative method is preferable. This alternative method should provide the right *mapping* function from contact states to arm motions. This non-linear function should also be able to *generalise* and *associate*. Generalise means that if a certain state is encountered within specified limits, it has to be clustered within the same group according to pre-defined features. The mapping must also be able to associate these clustered groups with a particular output, which will be a desired force or position. The above requirements for the mapping function compare favourably with the characteristics of connectionist methods.

The use of connectionist models in robot control to solve the non-linear problem has been demonstrated in a number of publications, either in simulations [23, 24, 25] or, working with real robots [26, 27, 3].

It should be noted that the terminology *connectionism* owes its origin to brain modelling. These learning algorithms are only "approximations" to the model itself due to the complex parallel processing involved in a complete model. Some authors refer to these algorithms as Neural Networks, however, to avoid ambiguity with biological principles in this thesis, they will be referred to as Artificial Neural Networks (ANN's), unless stated otherwise.

From reported work, ANN's have shown positive attributes which can be exploited in robot control such as:

- Ability to learn on-line and improve system actions from experience.
- Ability to recognise, classify and associate patterns with pre-defined actions.

- Stability and Adaptation. i.e., in the case of competitive learning, if new or novel patterns are encountered, the system self-adapt by classifying them without forgetting previous patterns, hence the system is also stable.

The above attributes can be used in robotic systems to reduce complexity due to uncertainty and to deal with the inherent non-linearity. The uncertainty involved during operations is mostly due to pair mating misalignment. In this case, the controller has to develop a static global model by associating behaviour with a particular situation, which is only achieved through learning. A system that treats every distinct operating situation as novel is limited to adaptive operations, whereas a system that correlates past experiences with past situations, and that can recall and exploit those experiences, is capable of learning. Since a learning system must be capable of adjusting its memory to accommodate new experiences, a learning system must, in some sense, incorporate an adaptive capability. ANN techniques have been used in robot control in the following areas:

- To solve the Inverse Kinematic Problem.
- Force/Torque control in joint space.
- Fine motion manipulation.

The first two areas are out of the scope of this investigation since the problem is not to learn a transformation from workspace co-ordinates to joint angles, or to control the robot to move along a given free-space trajectory. These abilities are already built-into real robots (within a certain dynamic range). This research is focused on how to react to real sensations of position and force to insert a peg into a hole, hence the aim is concentrated on learning fine motion manipulation.

1.5 Research Scope

The aim of this work is to develop autonomous robotic assembly operations. However, due to the complexity of the operations, the scope of the research must be clearly defined. The system to be designed forms part of an intelligent

cell with different sensory capabilities currently being investigated within the Manufacturing Automation Research Group. These areas also include language understanding [28, 29] and object recognition [30, 31]. A robotic autonomous assembly operation can be divided into four stages as follows:

Stage 1 *(Recognition):* During this stage the aim is to identify the part to be grasped. This is achieved by first understanding human commands such as *"pick the ball"* or *"move right"*. By employing visual feedback the robot is able to move the end-effector close to the object in readiness for grasping.

Stage 2 *(Grasping and trajectory planning):* The part is grasped and the arm is moved in free-space towards the mating component, e.g. peg in close proximity to the hole. This stage is completed when the end-effector makes contact with the female component.

Stage 3 *(Exploration):* At this stage the arm moves in constrained motion searching for the hole location by scanning the surface. After locating the hole, the arm aligns the peg to the insertion starting position.

Stage 4 *(Insertion):* During this stage the peg is inserted properly into the hole and the robot must be able to compensate for any misalignment occurring during the operation. The assembly terminates when a stop condition is reached, such as depth of insertion or a pre-defined force threshold.

> The scope for the research described in this thesis is centred on the last stage. It is assumed that the manipulated part (peg) has already been recognised and is held by the gripper ready to start the insertion.

1.5.1 Aim

Model-based techniques have been employed by many authors as an alternative to solve the assembly problem. Representative work in this area has been reviewed in this Chapter and Chapter 2. The success of the model-based strategy is built upon the precise knowledge of the parameters of the system such as geometry, friction, stiffness, etc. In practice, it is difficult to quantify all these parameters,

or at least with the required accuracy. Hence, errors are likely to occur during operations.

In addition, programs involving robot interaction, in comparison with stand alone computer programs, are very likely to experience errors because of its interaction with external variables in real world operations. Unfortunately, the uncertainties involved during assembly processes are difficult to quantify.

Some of these uncertainties come from different sources such as:

- *Backlash.* Present in the gears of the joints, due to ageing or poor manufacturing of the overall mechanisms.
- *Arm deflection.* Due to individual deflections in the robot links, which are proportional to the robot payload. Arm links can be made stiffer and therefore reduce its deflection, however there is a trade-off between the stiffness of the links, weight and payload.
- *Noise.* Force sensors used to feedback information into the control system exhibits noise due to thermal drift or cross-coupling errors. Noise can also be produced by a quantisation error that occurs during analogue to digital conversion.
- *Implicit errors.* These errors include the effect of random part sliding into the gripper mechanism during part mating.
- *Explicit errors.* Any external force disturbance acting on the end-effector.

Connectionist-based solutions in comparison with model-based techniques are capable of learning and minimise the need for explicit knowledge of the environment. Connectionist models have a strong ability to generalise from examples and acquire new knowledge using on-line learning. The NNC does not need to know about particular parameters of the system, it only requires to be trained on particular force conditions and be provided with the corresponding output motions. With this information the NNC should be able to start the insertion and be directed solely by the contact forces as they develop during operations and continue until the end-condition is encountered.

Considering the issues discussed above, the overall aim of this research is to develop a NNC capable of performing assembly operations and learning the insertion on-line. This means, the design of a mechanisms into the NNC to learn all contact force states as they occur and if useful, absorb them in its current Knowledge Base. By doing so, the NNC will become more skillful as the learning progresses. The assembly speed is expected to be reduced and the alignment motions to be more efficient.

1.5.2 Original contribution

The working conditions under which the NNC performance will be evaluated are as follows:

- The locations of the mating components are unknown.
- Limited spatial orientation. This means, similar to a blindfold human operator, the robot is provided with only a minimum generic Knowledge Base enough to start "recognising" and learning the actual contact force patterns.
- It is assumed the insertion direction to be in the vertical direction.

The assembly learning process for the NNC follows a strategy for insertion similar to that which a blindfold person would develop. The very first attempt will be erroneous due to the limited information that is given (i.e. insertion direction) and the uncertainty involved, since no information is given regarding the spatial location of the mating pair. At the second and subsequent insertions, the operation will become faster and more accurate due to the increased information the human operator holds in their brain and is able to retrieve to accomplish the task. In a similar manner, the NNC is expected to operate and make effective use of past events stored in its knowledge base and to improve its performance over time. Continuing with the analogy with the human operator and considering him as a base line for an *ideal operator*, this human operator possesses two important features which are to be implemented in the NNC: *error recovery* and *incremental learning.*

The NNC should be able to recover from motions which produce high forces and to stop learning when the insertion has been mastered. Similarly, it should be capable of further learning and be able to learn new skills and speed up this learning by using previous experience (learned tasks).

From the information described above and the requirements of the NNC, it makes sense to consider for its design, a connectionist network capable of incremental learning. Connectionist model algorithms that have shown favourable attributes to be implemented in the robotic system were considered. For reasons that will be explained, the connectionist model that completely fulfilled the requirements for the NNC was the Adaptive Resonance Theory (ART) created by Steve Grossberg and Gail Carpenter at Boston University. The basic ART-1 model [32] is a self-organising network with unsupervised category learning. The evolution of the theory has led to join two basic ART modules and produce networks with supervised learning capability, hence producing the so-called Predictive ART or ARTMAP network [33]. Fuzzy logic has also been combined with ART learning forming the Fuzzy ARTMAP network [34].

The NNC design is based on an enhanced version of the Fuzzy ARTMAP algorithm that includes a dynamic Knowledge Base, whose knowledge is regulated by the assembly process. Previous to the start of the operation, a Primitive Knowledge Base (PKB) is formed by teaching the robot generic examples of force contact in its six degrees of freedom. Basically, this information serves to bias and initialise the learning at the first insertion attempt. Later on, the Knowledge-Base may be modified with new information based on whether the force pattern-action pair has been good enough in reducing the constraint forces and if the motion has been made towards the end-condition.

In summary, the original contribution of this research to the field of robotic assembly is the creation of self-adapting robots that overcome the limitations of current techniques and enhance the capability of robot programming languages. The ART model is a truly adaptable network that meets many of the requirements for the NNC for an improved performance over the other algorithms. Furthermore, to the best knowledge of the author, the use of the Fuzzy ARTMAP network as

a core for the design of the NNC is the first time this technique has been applied to robot manipulators, opening new research in the area of incremental learning for robotic assembly.

1.6 Thesis Structure

In this introduction, issues regarding force control techniques for assembly have been presented. Non-linear problems occurring during operations highlighted the requirement for alternative methods to facilitate assembly operations. These methods are currently being investigated in the Manufacturing Automation Research Group (MARG) within the Department of Mechanical and Manufacturing Engineering. Previous work by M. Howarth [35, 36, 3] has demonstrated the applicability of ANNs and force sensing for skill acquisition in a robot with 4 DOF. Based on this work, it was proposed to implement a testbed site to further investigate the learning of manipulative skills with a six Degrees Of Freedom (DOF) industrial robot. This investigation is basically centred in two general aspects:

- Design and implementation of the testbed site, which includes the use of a PUMA 761 robot provided by the industrial collaborator.
- Design and implementation of a Neural Network Controller (NNC) and testing its performance in terms of on-line learning and assembly speed.

Both aspect are reported in this thesis following the structure outlined in the following paragraphs.

Chapter 1. Provides an insight into the area of robotics, force control and self-adapting robots for assembly. The scope of the research is established, the aim reviewed and the author's contribution to field of robotics assembly highlighted.

Chapter 2. Analyses in detail mechanical insertions underlying the importance of the peg-in-hole insertion and its characteristics. Current work in connectionist-based solutions is reviewed as well as its appropriateness to solve the assembly problem.

Chapter 3. Issues regarding the design and implementation of the robotic learning environment, including its control and sensory systems are explained. These issues include: security inter-lock system, force sensing software, custom-based control unit using a host-slave computer system, hardware interfacing and communication software.

Chapter 4. The accuracy and uncertainty of the robotic assembly system are evaluated in this Chapter. Two central aspects are studied: positional errors during incremental motions and errors produced by the force sensing system.

Chapter 5. In this Chapter a number of experiments are presented that describe the nature of the forces involved during assembly. Based on this analysis two important stages in the NNC are identified: Adaptation and Decision. An initial structure of the NNC is proposed based on the results obtained during experiments.

Chapter 6. In this Chapter, the Adaptive Resonance Theory (ART) is formally introduced. The mechanics of the ART-1 and FuzzyARTMAP learning systems are explained.

Chapter 7. Initial investigations into the unsupervised ART-1 algorithm were carried out using information from assemblies and a binary encoding according to ART-1 requirements. Assessments were carried out in terms of learning speed and recognition.

Chapter 8. In this Chapter, it is described how a predictive ART network in conjunction with a dynamic Knowledge Base can provide on-line learning and predictive capability to the NNC.

Chapter 9. In this Chapter the assessment result of the NNC performance is presented. Several insertions were made using different working conditions, i.e. different part geometry and positional offset. Also the knowledge discovery capability of the NNC was verified by stopping/inhibiting its learning during operations.

Chapter 10. Provides the conclusion of the thesis. A review of the main contribution of this research to the development of self-adaptive intelligent robots for mechanical assembly is given. Further developments of the system to enhance

the autonomy of the assembly system are identified and indicated as future work. Finally, additional results obtained during the assessment of the NNC are included in AppendixA. The ART algorithms upon which the NNC was designed are provided in Appendix B. The work published by the author during this research is listed in Appendix C.

Chapter 2

Force Control and Connectionist Models

This Chapter presents a review of current research in force control for robotic assembly and the main issues involved in non-linear control. The neural approaches reviewed include supervised and unsupervised connectionist controllers. The backpropagation network was used by H. Asada, reinforcement learning in conjunction with backpropagation was used by V. Gullapalli and M. Howarth, and Self Organising Maps (SOM) was used by E. Cervera. The analysis provided useful guidelines for the design of the neural controller which concludes the Chapter.

2.1 Non-linear Mapping — H. Asada

Asada analysed the peg-in-hole operation using a supervised ANN [23]. He analysed the insertion of a peg in a chamferless hole taking into account the following considerations: a) point contact, b) negligible friction and c) small displacements. A representation of the contact point is illustrated in Figure 2.1.

The part shown is in contact with the environment at point C_i, where r_i and n_i are the position and normal vector of the i_{th} contact point C_i respectively. The force and moment acting on the rigid body are collectively represented by a six-dimensional vector $b_i s_i$, where:

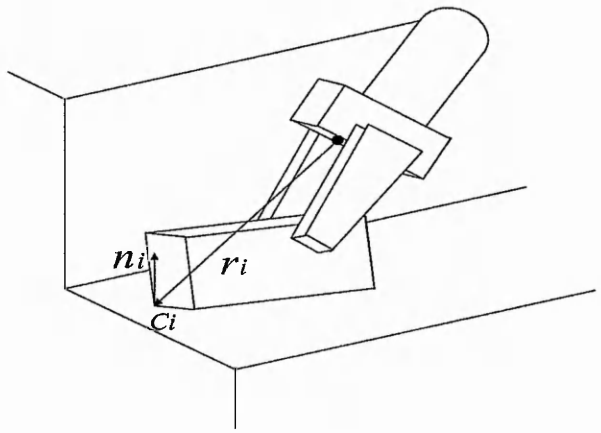


Figure 2.1: Contact point

$$s_i = \begin{bmatrix} n_i \\ r_i \times n_i \end{bmatrix} \tag{2.1}$$

and the resultant force and moment, collectively represented by $f \in \Re^6$ is then given by

$$f = b_1 s_1 + \cdots + b_m s_m \tag{2.2}$$

where $b_i \geq 0$. It is assumed that the m vectors are linearly independent. The contact at point C_i is detected by examining whether $b_i \geq 0$ or not. He demonstrated that a necessary and sufficient condition for b_i to be positive is given by $a_i\ f \geq 0$ where $a_i = (I - P_i)s_i$ and $P_i \in \Re^{6\times 6}$ is the projection matrix of vector s_i to the hyperplane spanned by vectors $s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_m$. Each contact point can then be detected separately by using the above linear inequality. He used a discriminatory function in a neural unit as shown in Figure 2.2.

The inputs to the unit are the six components involved in the force and moment vector f, and the weights are components of the coefficient vector a_i. The activation function is a threshold function. The output indicates whether or not the contact at point C_i has been made. Therefore, this unit can also be considered as a discriminatory or clustering function in terms of contact points. Starting from this unit, he constructed a three layer unit as shown in Figure 2.3.

Physical interpretation of the network is as follows. The first layer comprises m neural units that detect contacts at the m individual points and provide binary

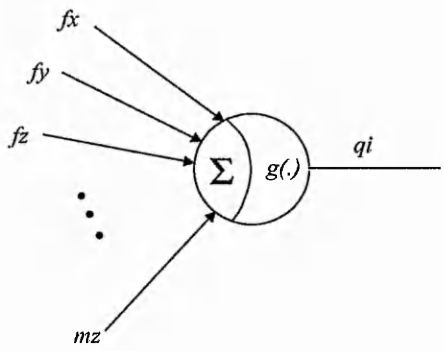


Figure 2.2: Neural unit for detecting contact point

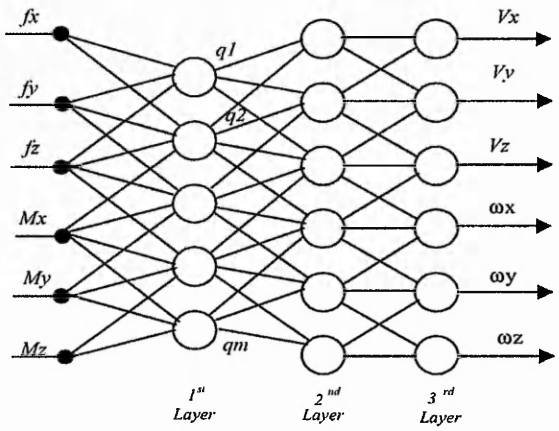


Figure 2.3: Three layer unit

outputs $q_1, \cdots, q_m$. These outputs described by a m-dimensional vector, determine the current contact state from which described velocities are directly derived. The second layer actually maps the m-dimensional vector to discrete contact states, $r_1, \cdots, r_n$. The third layer generates desired velocities in accordance with the individual contact states detected at the second layer.

A chamferless insertion of the structure shown above was simulated using a supervised backpropagation algorithm, where the weight of the mapping input-output was determined by the Widrow-Hoff procedure [37]. Results from the simulation showed that the network was able to determine three different contact states and that the output velocity vector was very close to the desired velocity.

2.1.1 Stochastic reinforcement value (SRV) — V. Gullapalli

ANN algorithms can be roughly divided between supervised and unsupervised methods. However, the Reinforcement Learning (RL) category has a fuzzy definition since it depends on the authors' view point on how they define *supervision.* RL is frequently associated with unsupervised methods since it provides a good evaluation about how well or how badly the network is performing [26, 25, 4].

RL refers to improving performance through trial-and-error. Supervised methods use a *teacher* that tells the system how to map input information with the target output. The weights of the network are adjusted based on a list of errors derived by comparing each output units' actual activity with its target activity. In contrast, the training information used in RL is evaluative feedback, it tells the learner whether or not, and possibly by how much, its behaviour has improved or has got worse. The teacher is then no longer needed and it becomes a *critic.* One of the main advantages of RL is the fact that it does not know what the system's output will be, as it only evaluates consequences and then it requires less information than supervised methods. However, since RL works with feedback, this information can have either delays due to the nature of the action or immediate consequences. Hence, it has to solve what is termed the *credit assignment problem* [38, 39]. Under uncertainty other parameters have to help to decide which action to take, normally those assumptions are based on probability theory. The Stochastic Reinforcement Value (SRV) algorithm designed by Gullapalli deals with uncertainty and is based on this sort of statement.

The network controller developed by Gullapalli is based on backpropagation units, whose outputs are SRV reinforcement learning units. The activation at any time of the units depends on two parameters, the mean (μ), and the standard deviation (σ) used in the normal distribution, which in turn depend on the current input to the unit. Learning takes place by adjusting these two parameters, therefore increasing the probability of producing the optimal real value for each input pattern. The algorithm does this by maintaining the mean of its activation as an estimate of the optimal activation (the activation that has the maximum expec-

tation of reinforcement from the environment), and using the standard deviation to control the amount of search around the current mean value of the activation. The basic idea for the algorithm is as follows [25]: the mean (μ) of the distribution is made equal to a weighted sum of the inputs of the unit at time t:

$$\mu(t) = \sum_{i=1}^{n} w_i(t)x_i(t) + w_{thres}(t) \tag{2.3}$$

The determination of σ is more involved. The conditions stated above indicate that for a given input, the standard deviation should depend on how close the current expected output (i.e. the current value of the mean), is to the optimal output for that input. Since the reinforcement signal returned by the environment is a measure of this, the standard deviation used in computing the output should depend on the expected reinforcement. If the expected reinforcement is high, the unit is performing well for that input and σ should be small. Conversely, if the expected reinforcement is low, σ should be large so that the unit explores a wider interval in its output range. The *expected reinforcement* $\hat{r}$ is calculated by

$$\hat{r}(t) = \sum_{i=1}^{n} v_i(t)x_i(t) + v_{thres}(t) \tag{2.4}$$

This expected reinforcement is used to compute the standard deviation as

$$\sigma(t) = s(\hat{r}(t)) \tag{2.5}$$

where $s(\cdot)$ is a monotonic decreasing, non-negative function of $\hat{r}(t)$. Moreover, $s(1.0) = 0.0$, so that when the maximum reinforcement is expected, the standard deviation is zero. $s(\cdot)$ is defined as

$$s(\hat{r}(t)) = max\left(\left(\frac{1.0 - \hat{r}(t)}{5.0}, 0.0\right)\right) \tag{2.6}$$

Based on $\mu(t)$ and $\sigma(t)$, the unit computes its *activation* $a(t)$, which is a normally distributed random variable:

$$a(t) \sim \Psi(\mu(t), \sigma(t)) \tag{2.7}$$

Finally, the activation $a(t)$ is transformed into the output of the unit $y(t)$ using the output function $f(\cdot)$, so that

$$y(t) = f(a(t)) \tag{2.8}$$

Equations 2.3 to 2.7 describe how the unit uses its inputs to compute its output at a given time step. Learning is achieved through modifying the unit's weights as follows:

$$w_i(t+1) = w_i(t) + \alpha\Delta_w(t)x_i(t) \tag{2.9}$$

$$w_{thres}(t+1) = w_{thres}(t) + \alpha\Delta_w(t) \tag{2.10}$$

where α is the learning rate parameter and

$$\Delta_w(t) = (r(t) - \hat{r}(t))\left(\frac{a(t) - \mu(t)}{\sigma(t)}\right) \tag{2.11}$$

The fraction in equation 2.11 is seen as a *normalised noise* (or jitter) that has been added to the mean activation unit. If this noise has caused the unit to receive a reinforcement signal that is more than the expected reinforcement, then it is desirable for the unit to have an activation closer to the current activation $a(t)$. It should therefore update its mean output value in the direction of the noise. i.e., if the noise is positive, the unit should upgrade its weights so that the mean value increases. Conversely, if the noise is negative, the weights should be updated so that the mean value decreases. On the other hand, if the reinforcement received is *less* than the expected reinforcement, then the unit should adjust its mean in the direction *opposite* to that of the noise. In terms of training, a supervised algorithm is used since each input vector has to be associated to a corresponding reinforcement value. This is done by using the LMS rule of Widrow and Hoff as shown in the following equations:

$$v_i(t+1) = v_i(t) + \beta\Delta_v(t)x_i(t) \tag{2.12}$$

where β is the learning rate parameter and

$$\Delta_v(t) = r(t) - \hat{r}(t) \tag{2.13}$$

To demonstrate the feasibility of the SRV algorithm, Gullapalli used a Zebra robot arm to accomplish a peg-in-hole operation. A F/T sensor is placed between the gripper and the flange of the robot. The robot is commanded via desired motions which are the output of the neural controller placed in the feedback loop of the system. For the task, the peg is circular and 30 mm long and 6 mm in diameter,

while the hole is chamferless and 6.35 mm in diameter, giving a clearance of 0.175 mm.

The controller is a backpropagation network with 11 inputs. These are the sensed positions and forces, $(X, Y, Z, \theta_1, \theta_2)$ and $(F_x, F_y, F_z, M_x, M_y, M_z)$. The output of the network are the position commands $(x, y, z, \Theta_1, \Theta_2)$. In terms of hardware, the position of the end-effector is computed from the sensed joint position of the arm by using the Forward Kinematics of the arm. For the output command, the arm uses a PD control law. The performance of the operation is evaluated by a parameter r, which measures the performance of the controller. r varies between 1 to 0 and is a function of the sensed peg position and the nominal hole location. Results show that the algorithm performs well despite uncertainty in the end-effector location (e.g. *backlash*).

2.1.2 Reinforcement learning using a SCARA-type robot — M. Howarth

M. Howarth implemented a backpropagation algorithm in conjunction with reinforcement learning using a 4 DOF SCARA robot to carry out peg-in-hole operations [3]. In comparison with Gullapalli's work, where the reinforcement learning values were stochastic, Howarth's reinforcement value was based on two principles: minimisation of force and torque values and continuation of movement in the assembly direction. This implies that whenever a force or torque value is above an acceptable threshold, an action (i.e., reorientation), should occur to minimise the force. Additionally, movement in the target assembly direction is favoured.

After completing the incremental move generated by the network, the inputs are read (sensor values), and these are propagated through the network and the output values calculated and scaled to provide the next incremental motion of the manipulator (δ). The incremental motion is completed and new sensor values are taken at this new location. The effectiveness of the last action is evaluated and the error signal generated. This value is then propagated back to the network adapting the weights at each layer of the network. The reinforcement value (k)

is associated with the backpropagation algorithm as follows:

$$\varepsilon_i \;=\; k\; f'\;(S_i)$$

where ε_i is the error at the i_{th} output neuron resulting from the forward pass of the network. $f'(S_i)$ is the derivative of the activation function $(f(tanh(x)))$, which is given by

$$f'(S_i) \;=\; \frac{4}{e^{2S_i} + e^{-2Si} + 2}$$

The value of k varies according to the stage. If the peg is moving in free space, then k is governed by:

$$k_i \;=\; -\,\delta_i$$

for all axes except Z and $i = x, y, or\ \Theta$.

For the Z axis:

$$k_z \;=\; -\,1(\delta_z + 1) \;\; for - 1 < \delta_z < 0$$

$$k_z \;=\; 0 \;\; for\; \delta_z \leq \; -\,1.0$$

$$k_z \;=\; -\,1.0 \;\; for\; \delta_z \geq \; 0.0$$

When the peg is in contact with the environment (female component):

$$k_i \;=\; F_i \;-\; \delta_i$$

2.1.3 Identifying contact states by SOM — E. Cervera

Cervera et al. [16] have also used ANN techniques to classify contact states in the peg-in-hole operation. In comparison with some other approaches, they use an unsupervised network to cluster the different states during the assembly. The contact states are classified by using Self Organising Maps (SOM). The network is able to classify the three contact states as made before by Asada [23] and three additional contacts as shown in Figure 2.4. After contact states have been identified, they are labelled to the corresponding state in order to differentiate them.

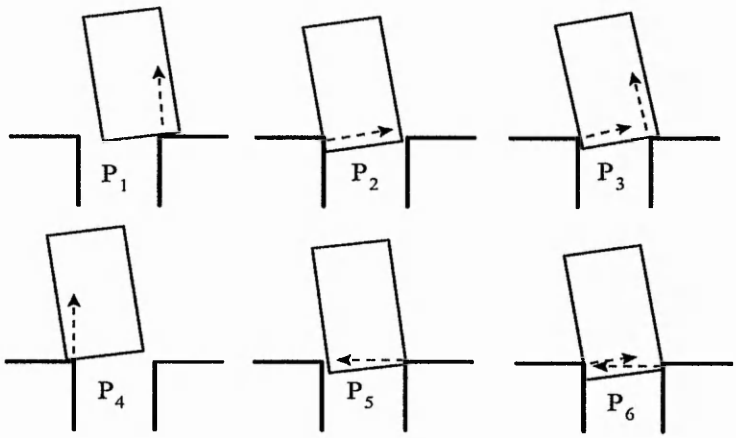


Figure 2.4: Contact states

A SOM is a 2D representation of multidimensional input patterns. The cells are fully connected to the inputs and become tuned to different signal patterns. Those units which are best tuned to a given signal pattern become active and a response is concentrated in the area with the most active units in the network. The difference with other connectionist networks is the fact that after learning, SOM's responses are topologically arranged in the map i.e., two similar input patterns which are near each other in the signal space, are correspondingly located near each other in the map sheet. The ordering takes place automatically without external supervision, based only on the internal relation in the structure of the input signals themselves and, on the coordination of the unit activities through the lateral connections between the units. During training, units adapt their weights following the input values but each adaptation step involves not only a unit but a neighbourhood of units, providing the topology preservation properties. Through the self-organisation of the responsive areas in the map, the SOM algorithm creates an internal representation of the incoming signal structure.

The output of the network and the corresponding contact states and symmetry are classified as shown schematically in Figure 2.5.

Here, $p'_1 - p'_6$ are the symmetric contact states of $p_1 - p_6$. When more contact states were presented to the network, it could not resolve the difference, so additional information (orientation of the peg) had to be included in the training set. Insertions with different tolerances, (offset between the nominal insertion point

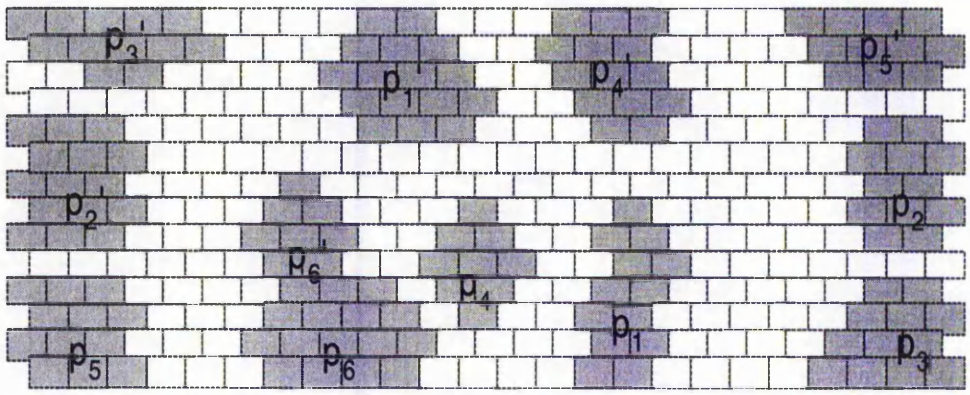


Figure 2.5: Activation patterns

and the peg location) were made, so the output patterns of the network were as it is schematically shown in Figure 2.6.

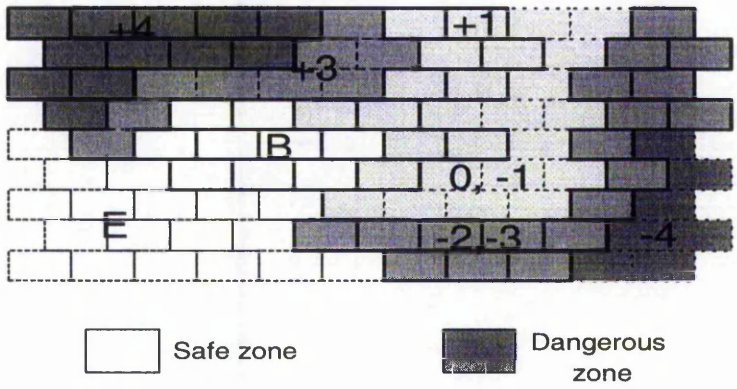


Figure 2.6: Regions for the insertion task

The degree of *safety* is shown in grey scale according to the level of tolerance. The higher the tolerance, the more dangerous the operation is. However, due to the nature of the mapping, some of the areas overlapped as shown in the areas marked +1, 0, -1 and E, B. When an unlabeled group was identified, the output of the network was defined as *"unknown"*. To overcome this, further enhancements were developed [24], in which additional parameters were taken into account such as hole clearance and friction coefficient. An additional parameter that helps the algorithm to discriminate between states is the introduction of a probability function that gives a measure of confidence over the network output.

In a more recent work, E. Cervera and A. P. del Pobil used Reinforcement Learn-

ing to carry out the peg in hole insertion using cylindrical and cubic pegs [26]. Here the relationship between contact states and action is further improved by using a Q-learning algorithm [39], where a discrete number of actions are chosen, i.e., different direction of motion. Each learning algorithm updates these values according to the reinforcement signal obtained when each action is executed in a given state. A simple action-penalty representation is used. The system is penalised with the same value for every action that it executes. The method converges to the value which minimises the accumulated penalisation (which corresponds to the minimum insertion time). Initially the system chooses actions randomly. After learning, the action with higher value for the current state is chosen.

2.1.4 Conclusions

H. Asada provided an important contribution towards non-linear mapping between force data and position(velocity) data. During simulations using back-propagation he showed that this network topology could be used for clustering purposes. However, the network was able to discriminate only three contact states. Also, there was no indication that the network could easily be re-trained to cope with a larger number of contact states.

In the SRV algorithm, the learning depends on the peg geometry on which the network is trained and is specific to the location of the peg in the workspace because the absolute peg position is produced as an output. If the peg location were moved in the workspace, the learned operation would probably fail because it would be very difficult to accurately specify the relative transformation between the new location and the training location.

Another approach given by M. Howarth also used RL, but in comparison with Gullapalli's work, the final location of the hole is unknown. He used a reinforcement value that rewarded motion in the main insertion axis ($-Z$). The reinforcement value was distinct depending on the relative motion in Z axis and if the motion is made in free or constrained space. The implementation on the 4 DOF SCARA-type robot limited the robustness of the algorithm since it con-

sidered 4 axes of reorientation only and therefore reinforcement values were not related to the full 6 DOF.

The use of an unsupervised SOM approach demonstrated its capabilities as classifier or clustering algorithm for the different contact states. When the dimension of the net increased (with more contact states), further parameters had to be used to discriminate between states. In this respect, reinforcement learning used in conjunction with a probability function were used to give a confidence to the network's output. In comparison with the work by Gullapalli and Howarth, the approach used by Cervera considered only a finite number of contact states.

The revision of the work presented here has provided essential information regarding two important aspects for the design of the neural network controller:

- It has to be adaptable by allowing more contact states to be classified in order to work better under uncertainty with different shapes and assembly orientations.
- It has also to be able to recognise autonomously those clustered contact states and associate them with the corresponding motion output.

Chapter 3

Design of the Robotic Assembly System

This Chapter is concerned with the design and implementation of the robotic learning environment, including its control and sensory systems.

The author was responsible for the specification, design and implementation of all issues related to the system. These included: security inter-lock system, force sensing software, custom-based control unit using a host-slave computer system, hardware interfacing and communication software.

3.1 Robot Control System

The testbed was designed for the PUMA 761 robot arm provided by the industrial collaborator Roll Royce & Associates. The hardware design is centred on this arm, which is driven in real-time through continuous path modification using the ALTER mode described in section 3.4.2.

The control architecture is depicted in Figure 3.1. The plant is integrated by the robot arm and the insertion process itself in the open loop path (highlighted area 1). A force f is produced during manipulation. This force is measured at the wrist of the arm, which may include disturbance forces f_{dist}. The F/T sensor is located in the feedback loop to provide moment and force signals to the Neural Network Controller (NNC). The NNC then associates and generalises this information from

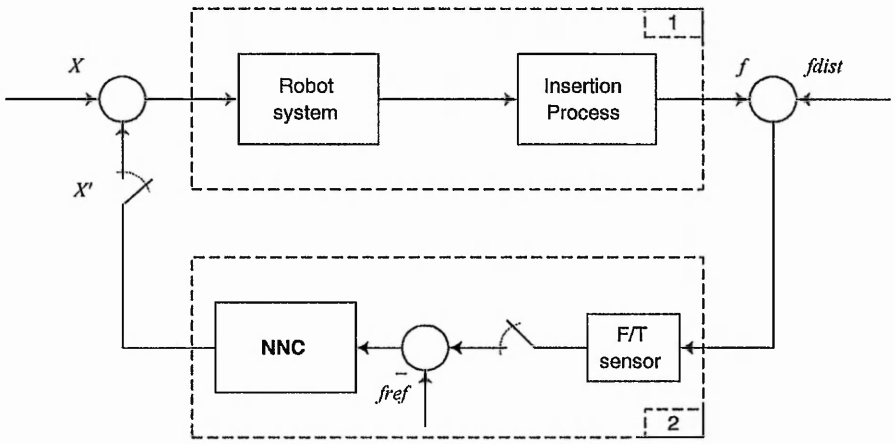


Figure 3.1: Control system

the sensor and from the force reference signal, f_{ref}. This f_{ref} varies according to the assembly stage, depending on the orientation of the end-effector during assembly. In practice, this force represented the gravitational force component due to the weight of the end-effector. The output of the NNC are actually incremental motions X' that are added to the end-effector's current location. This incremental motion is sent to the robot's controller during continuous path modification.

Looking at the feedback loop path (highlighted area 2), it can be observed that this section is critical from the design point of view since it provides the input information to the NNC. This information must be as reliable as possible to provide fast training to the NNC. Inaccurate input patterns at this stage may slow the learning algorithm or produce an unstable system.

The reliability of the control system and consequently the assembly relies then on these two subsystems, the robot arm and the sensory system, which are described in the following sections.

3.2 The PUMA 761 System

The **P**rogrammable **U**niversal **M**achine for **A**ssembly (PUMA) is a member of the Unimation 700, 500 and 200 robot series designed by Vic Schienman at MIT in the mid-70's. These 6 DOF robot arms are still widely used in industry and for research and teaching purposes in academia. The robot employed for this

research is one of the biggest of the PUMA family, namely PUMA 761. The PUMA system is shown in Figure 3.2.

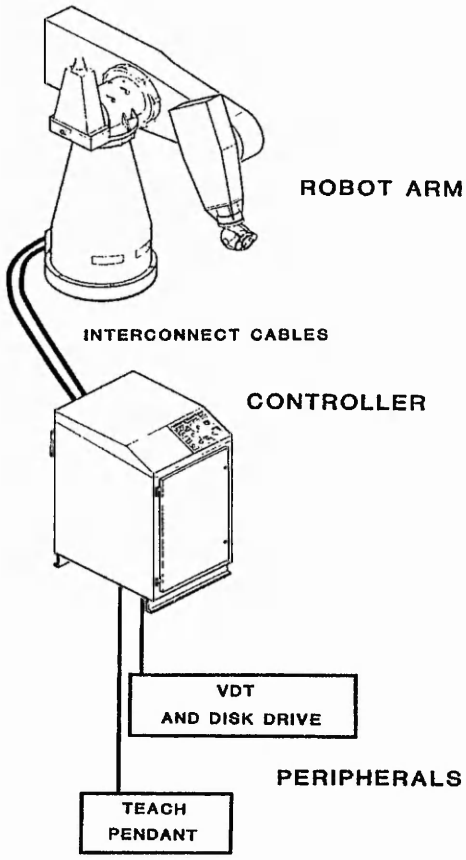


Figure 3.2: PUMA system

The robot arm or manipulator is the main mechanical part of the system. Basically, it consists of servomechanisms for the motion of the six joints and optical encoders to pass the incremental motion information of each joint to the controller unit.

The other significant part of the system, the controller, houses the components that control and power the robot arm. The major components of the controller are the control module (computer and software), power component chamber, I/O boards, ventilation and cooling system, and peripherals. The peripheral include the Video Display Terminal (VDT)/disk drive unit, external printer, and a teach pendant also called hand-held control.

The operating system that controls the robot arm is termed VAL II or VAL

for short. VAL II is stored in the computer memory of the control module. The controller also houses the operating controls for the robot system such as emergency stop, on/off power arm, teach mode, etc.

The controller's master processor of the PUMA robot is the LSI-11/73. This processor interfaces with six 6503 microprocessor based boards as shown in Figure 3.3.

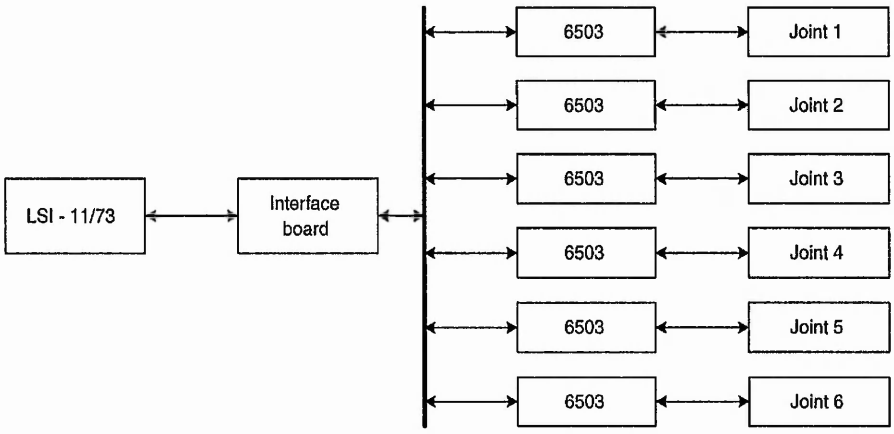


Figure 3.3: Architecture of the PUMA system

Each board has a PID control law which computes the desired motion for each joint. The master processor also takes care of the high level interface with the operating system and calculates the inverse kinematic parameters. The PID controller can be switched off to diminish vibration effects at the end-effector. However, this would limit the ability of the controller to cope with positional errors in steady state, affecting the overall performance.

3.2.1 Safety regulations and Remote Teach Mode

Following safety regulations within the Department of Mechanical and Manufacturing Engineering, it was recommended to surround the robot by a metallic guard and to design a security interlock system. The main requirement for the security system was as follows:

> *It must prevent people entering the working envelope while the robot is in operation. This has to be achieved by using a key-lock*

> *system at the door's guard, which must activate the emergency stop in the robot controller if the door is opened when the arm power is on.*

However, during experiments and especially when teaching precise points to the robot, it is preferable to teach some of them inside the guard and work closely alongside the manipulator. The only way to achieve this is by using the robot in *remote control mode*, which disables all controls on the front panel of the robot controller (except power key and emergency stop), and enables remote control (including the teach pendant).

By combining both requirements the author designed a system that can be used in either *local* or *remote mode* safely. The user is able to teach points outside or inside the guard with full control on the manipulator. In remote mode the only person with control of the robot's motion is the user via the teach pendant.

3.3 Controlling Contact Force via Directed Motion

Ordinary methods to program robots include the use of predefined workspace points stored in memory by the teach pendant. This method is suitable when environment conditions prevail during operations e.g. motion in free-space. However, during mechanical insertion and part manipulation the end-effector necessarily contacts the environment making the operation difficult. To overcome these uncertainties either a model-based or a connectionist-based solution may be employed as pointed out in Chapter 1 and 2. The force control will be ultimately achieved by position control as it is explained in the following paragraphs, where related work with PUMA robots is reviewed.

Perhaps, one of the most comprehensive options in implementing the position/force control as far as the hardware is concerned, is the utilization of the TRC004 PUMA interface card manufactured by Trident Robotics and Research, Inc. [40]. The TRC004 was designed to replace the LSI-11 VAL computer and the servo cards in the Unimate PUMA controller to allow high-speed, direct access to joint motor torques and positions. This card is suited to interface the 500 series robot

and some of the 200 and 700 series with Mark II type controllers. This card physically replaces the original CPU, RAM, EPROM, serial controller, interface cards and joint servos from the controller backplane. The TRC004 can handle joint encoder information from the arm and provide an analogue output to be connected to the power amplifiers for commanding motor torque.

This option is attractive considering that DEC announced in 1996 the "Retirement of the PDP-11 Product Family" [41] from late 1997.[1] However, Staubli Unimation has guaranteed the supply of parts for these controllers for some years more. The drawback of choosing this option is undoubtedly the fact that the user has to create his own power-on calibration, joint potentiometer calibration and joint position control. In other words, programming is needed to restore many of the functions of the original operating system, VAL II. Such changes were out of the scope, time and budget allocated for this research. However, this option is mentioned here since it offers a good alternative for creating robust controllers using PUMA architectures. Some other researchers have also modified the original architecture as can be seen in [42], however, they do not use a ready to implement commercial board, but a purpose specific design.

The PUMA architecture's built-in PID control law for each joint and each control board is based on a microprocessor that modulates the power applied to each of the PWM[2] amplifiers. The modification of the signal power to the servos could be made by using *preshaping* input signal as pointed out by J. M. Hyde and M. R. Cutkosky [9]. Nevertheless, altering this signal would modify the total force at the end-effector and also may change the arms behaviour when the end-effector and the environment make contact. But, such a modification would necessarily affect the control boards and as a result, the overall robot performance.

Bukowski, et al. [43], used a novel approach to alter the arm position by using a Digital to Analogue Converter (DAC) in the host computer and an Analogue to Digital Converter (ADC) in the robot controller. This has the advantage of reducing the communication time in comparison to the required time during the continuous path modification with the robot. Obviously, a trade-off must

[1] It is important to remember that the PUMA's controller design was centred on this family.
[2] Pulse Width Modulation.

be considered between accuracy in position and the communication overhead experienced by using the continuous path modification feature of the PUMA architecture. The reported positional uncertainty was equivalent to only three counts of the ADC resolution.

Having considered the options to control the force and the objectives of the research, it was decided to control the force indirectly by controlling the end-effector position. In this manner, the built-in PID controller did not need to be modified. Furthermore, following this approach the implementation of the techniques developed here onto other industrial robots should be straightforward.

3.3.1 Servo sampling rate

There are two possibilities for moving the arm by using low-level control. It can be achieved either directly by writing motion programs using the *VAL II Z-Instruction Set*, or by using the ALTER mode [44]. In the former, the instruction set allows the user to write LSI-11 machine code routines to run in VAL II controllers improving the communication abilities of the robot. In the latter, ALTER mode is restricted to a fixed communication timing which should be considered as the bandwidth available in real-time for signal processing and control.

Using machine code and writing directly to the LSI-11/73 processor certainly gives great flexibility in terms of hardware. However, it also imposes a serious consideration on the *natural frequency* of the arm and its stability. Modifying updating times to the servo motors may excite some resonances on the structure of the arm. An analysis of this stability is very complex in a serial link manipulator since the mechanical structure and its description is dependent on configuration, which is infinitely variable.

With regard to these considerations, it was decided to use ALTER mode to communicate to the controller and to consider a bandwidth of 28 ms available for processing. This servo sampling rate (36 Hz) is considered "safe" due to the low resonance frequency of this type of robot manipulators, which is between 5 Hz and 25 Hz [45, 46]. Furthermore, since the arm is moved in *quasi-static* motion (which is the case in fine manipulation), then the communication overhead does

not affect the system. From experimental data it was observed that a settling time longer than 28 ms was needed when the F/T data was acquired. This situation is discussed further in sections 7.1 and 7.2.

3.3.2 Force sensing and system architecture

Force sensing is not a standard function built into industrial robots, but is available as an option from a few manufacturers. Adept Technologies pioneered the implementation of force sensing capability in their controllers in 1989 and some others have also announced some form of integrated force sensing [47]. Following the Adept-Staubli partnership, the new RX family robots from Staubli uses the Adept MV type of controllers [48], and consequently, force sensing is also offered as an option. The RX family and its programming language V^+ are the successors of the PUMA family and the VAL language respectively. Other manufactures such as Integrated Motions, Inc. offers force sensing capability in the Zebra model, however this capability is limited due to the 1 kg maximum payload of the robot.

In the present design, centred on the PUMA system, a custom solution was implemented that included this sensing capability as well as an open architecture to control the arm motion. This architecture is illustrated in Figure 3.4.

The control and learning algorithms are implemented in the supervisory host computer. This supervisory host computer is based on a Pentium processor working at 100MHz. The Host communicates to the robot controller modifying the previous arm motion according to the force feedback signal. The original robot controller and the VAL language were not used to implement the algorithms since the language does not allow the creation of complex programs nor to implement easily the force sensor interface. The robot controller is only used as a *motion slave*, which alters the current motion of the arm according to supervisory host computer commands. It can also be seen in Figure 3.4 that the supervisory computer communicates with the robot controller via two serial lines. The supervisory line carries all commands to the controller and replaces the original VDT. The computing capabilities of the supervisory computer are better than

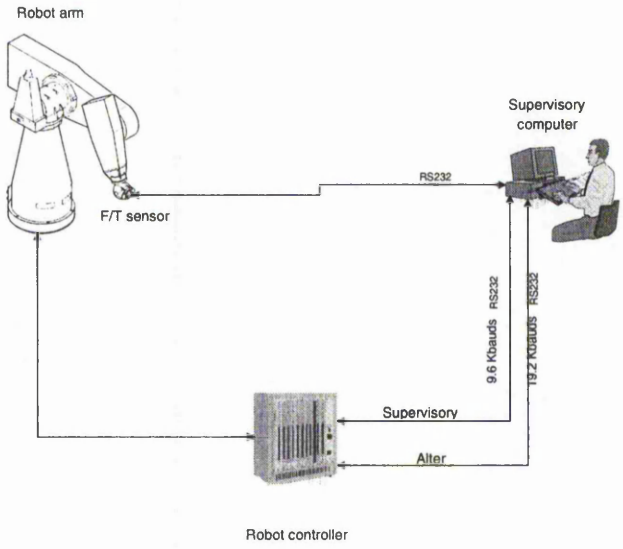


Figure 3.4: System architecture

the original terminal. The second line is a dedicated serial line (ALTER), which sends the incremental motions to the controller and also carries information back to the supervisory computer on the current position of the end-effector. The use and design of both lines are explained in the next section.

3.4 Communicating with the Robot Controller

The PUMA system has the capability to communicate with an external system (stand alone or networked computer) using a protocol that ensures the integrity of information transferred between the VAL II system and the external system. There are two levels of communication as mentioned in section 3.3, where the level dealing with commands to the controller is termed Supervisor mode, whereas the one carrying information about how much to modify the current arm position is referred to as ALTER.

3.4.1 Supervisor mode

This is how VAL II can be interfaced to a supervisory computer via a EIA RS-423 serial line operated at a maximum speed of 9600 bauds using a software communi-

cation protocol DDCMP [49]. DDCMP stands for Digital Data Communication Message Protocol and it was created by DEC to interface all the PDP family processors including the LSI-11/73, the master processor in the controller unit. DDCMP resides at a level above the communication medium (i.e., the physical transmission of bits over the communication channel). It is concerned with the transmission of data organised into physical groups as data messages and it uses a coding scheme within every message to identify the type of communication. This scheme is implemented by considering the system as being subdivided into separate subunits, referred to as "logical units". Each type of communication is then associated with a particular unit [50].

Despite the advantages of using a well structured synchronous protocol such as the DDCMP, it was decided not to use this protocol since it would increase the time overhead in the serial communication. Other alternatives reduce time, implementation effort and cost, so it was decided to communicate directly through the serial line via a dummy terminal program to interface the supervisory computer to the serial VDT port at the robot controller. A 'C' program was written and tested by directly connecting the RS-232 of the supervisory computer to the J10 terminal at the robot's controller, so that the original VDT and keyboard are no longer used and all the commands to the robot were issued from the supervisory host computer.

3.4.2 Continuous path modification — ALTER mode

A reliable communication was established to the robot controller to issue commands from the supervisory computer, which were originally available only from the controller's terminal. However, if the robot is to be interfaced with sensors as this is the case, then a supervisory communication is not enough, since another channel is needed to convey such sensory information to the VAL II system. With a force sensor attached to the wrist of the robot, the sensor information will serve to modify the current arm motion in real-time by using the ALTER mode.

The ALTER mode can be achieved in two ways. Internally, by using an auxiliary program that executes in parallel with the robot-control program that contin-

uously monitors the motion of the arm and acts accordingly. Externally, using the information sent by the supervisory computer. The information contained in the data stream includes, among other parameters, the type of coordinates to be used (world or tool coordinates), the type of motion e.g. incremental or absolute, and the incremental motion for each axis. Software implementation of the ALTER mode was not supported by Staubli, hence the author developed the serial communication software for continuous path modification.

When in ALTER mode, the operating system sends messages to the supervisory computer about 36 times per second (every 28 ms) requesting path control information. The computer must respond by sending data that determines how the nominal robot tool trajectory is to be modified. This sequence continues until VAL terminates ALTER mode for some reason, or the supervisory computer signals an exception condition.

The details of ALTER mode are specified in a user program with an ALTER program instruction. When this instruction is executed, VAL immediately sends an "ALTER starting" message to the supervisory computer and waits for the start up acknowledgement. If the computer does not respond, the VAL program terminates with an error. If the computer does respond, VAL enters the ALTER mode at the start of the next robot motion.

VAL remains in ALTER mode until NOALTER instruction is executed, or the robot program stops executing for any reason. The latter includes unexpected termination of program execution due to an error, such as attempting a joint-interpolated motion (since in this mode only straight line motions are allowed).

Details about the structure of ALTER messages are found in [44]. However, the general structure of the messages will be given here to understand how the protocol handles information between the controller and the supervisory computer.

The communication is bi-directional between the controller and the computer. It starts with an ALTER instruction in a user program as mentioned previously, then after executing this instruction, the controller sends a starting message to the computer indicating its readiness to start communicating. The computer has to acknowledge this message and start sending information. VAL imposes

strict timing between messages and there is no error checking on messages after starting the ALTER mode. If the supervisory computer message is delayed by more than 28 ms, the controller request the information again and if the delay persist, then the controller interrupts the communication. VAL identifies which error has occurred and indicates the type of error. However, it cannot recover from it and the communication terminates. During software development many attempts were made before a stable communication could actually be achieved.

Message packet

The complete format of every transmission to and from the serial communication port of the controller and the supervisory computer is as follows:

$$DEL \;\; DLE \;\; STX \;\; < data > \;\; DLE \;\; ETX \;\; < check >$$

where,

DEL: Byte containing 255 (FFh)

DLE: Byte containing 144 (90h)

STX: Byte containing 130 (82h)

< *Data field* > Sequence of any number of 8-bit bytes

DLE: Byte containing 144 (90h)

ETX: Byte containing 131 (83h)

< *check* > Single-byte checksum containing the two's complement of the sum of all the bytes in the *Data field.*

All messages either from the supervisory computer or VAL system follow this format (except when starting communication). All information and control bytes are transmitted in the Data field. The reader is referred to [44] for a more detailed description of all data within this field.

A 'C' program was written to establish this data packet and to communicate with the controller, tests were carried out with first, motion in one axis at a time and then all axes in full motion.

Physical link

The serial interface between the robot controller and the supervisory computer was agreed to be made using the RS-232 standard. Therefore, the RS-422 original standard in the DVL11-J asynchronous serial line board had to be modified. The change was simple since only a number of resistors were removed from the board to allow single-ended voltage input. There were no changes at the supervisory computer end since its port were already using the RS-232 standard.

3.5 Improving Hardware Design

Initial experiments with the proposed architecture were carried out. A program was written in Borland 'C' to control the incremental motion of the robot. The scheme proved to work well when in ALTER mode, however under different conditions the ALTER mode was aborted since the communication time exceeded the 28 ms limit. The ALTER mode is very strict and once it has started, the interchange of data between the robot controller and supervisory computer must continue. If the data communication time is exceeded, the controller requests the information again and if the supervisory computer does not respond then the communication link is broken.

The above can be better understood using Figure 3.5 which shows the host and robot controller.

The main program in the host computer is in charge of:

- F/T data acquisition and processing.
- Communicating with the controller using the ALTER mode.
- Neural network training.
- Terminal emulator.

At the start of the communication, the host computer request the initiation of the ALTER mode by the terminal emulator via the serial port COM2. The robot controller starts communicating at a 28 ms rate via the COM1 port and expects

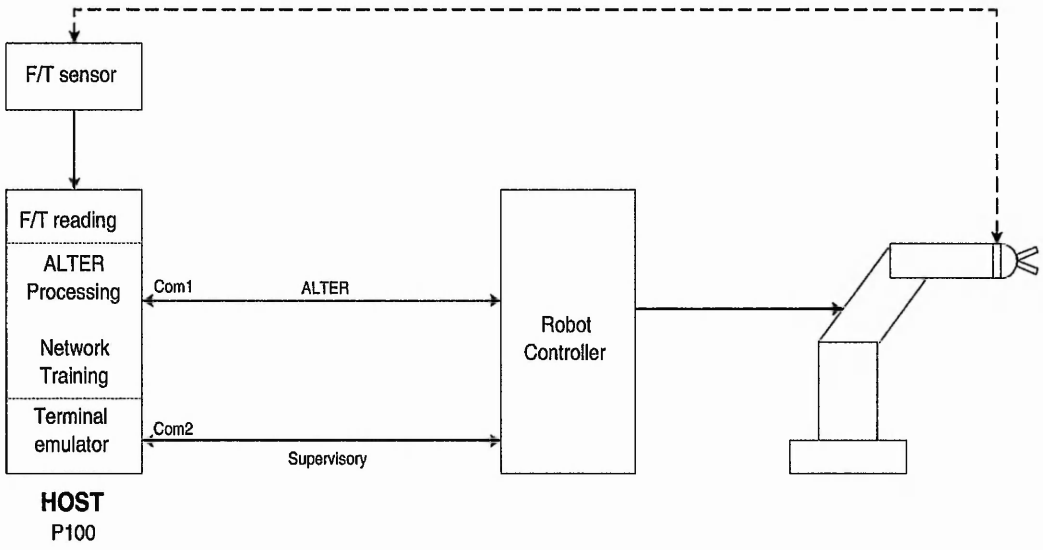


Figure 3.5: Host-Controller architecture

the host computer to respond giving an incremental motion request within that time. However, the host computer should also allocate resources to read from the F/T sensor in order to determine the next incremental motion. At this point the ALTER mode sometimes failed, since the communication overhead exceeded the allowed time.

To solve the problem, two possible solutions were considered. One solution would be creating an interrupt-driven program and allocate time for each of the processes mentioned above. In this manner, the host computer is only interrupted when the motion information is requested by the controller in ALTER mode otherwise it can continue with the current process. The disadvantages of this approach is that the F/T reading or neural network training may be interrupted asynchronously. Hence, the process had to be reinitiated after servicing the interrupt. As a result, the remaining time slice to continue both processes may not be enough and most probably another ALTER interrupt could occur before actually reading the F/T pattern or training the network. If that happens, the system can be trapped and become unstable. Additionally, with this approach the F/T data cannot be read in real-time as the contact forces develop during operations. In the ideal case, it is always desirable to monitor the forces as they occur and to react accordingly. From the discussion above, it is clear that parallel processing is an ideal solution

to service all processes, but mainly to release the host computer from the ALTER mode data transfer giving it more processing time capability.

Adding parallel processing capability to the supervisory computer can be achieved in different ways. However, a very simple solution was found: a slave computer was placed between the supervisory computer and the robot controller. This new slave computer would be solely in charge of the incremental motion during the ALTER mode.

3.5.1 Host-slave architecture

A 486-based computer was used as the slave in the architecture shown in Figure 3.6.

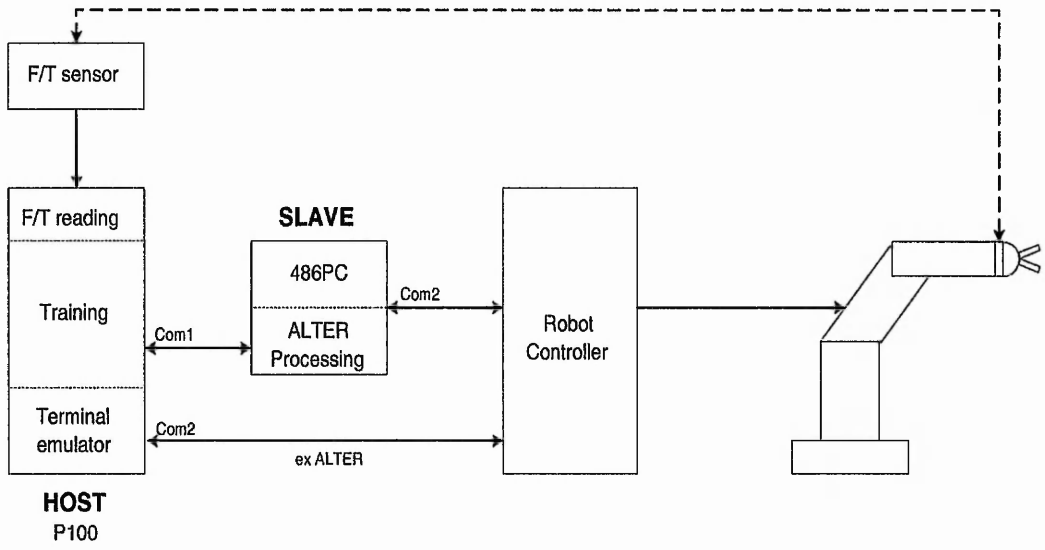


Figure 3.6: Host-slave architecture

In the host computer three algorithms are implemented:

1. F/T processing. The processing includes the following features: F/T data acquisition, force data scaling, offset removal and rotation/translation of the sensor coordinate frame.

2. Neural Network training/testing. Includes all the necessary settings for the network training and testing, and the control of the continuous path modification in ALTER mode.

3. Terminal emulator. The emulator resembles the use of the former controller's terminal. Commands and instruction to VAL can be issued from the host computer as if it were the terminal. Full functionality is provided so that the terminal is not longer used, which provides more flexibility since the overall control is in the host.

With this architecture a typical interchange of information between slave-host is as follows:

> The host computer reads the F/T data and the neural network is tested with this new information. As a result, the neural network's output produces the desired incremental motion for the end-effector. At this time the host computer transmits the incremental motion value to the slave and immediately after, the slave computer signals the robot controller to initiate the ALTER mode. The host computer is released at this point and the slave computer is in charge of encoding the information as required by the robot controller in ALTER mode. The host can then monitor the F/T force traces as they occur in real-time or perform any other task.

3.5.2 Software design

The software for handling data during ALTER mode was implemented in Borland 'C' in the slave computer working under the DOS operating system. This program basically gets the direction, speed and mode of the motion (world or tool coordinates) from the host and sends it, in compatible format, to the robot controller in ALTER mode as explained in section 3.4.2.

The NNC is implemented in the host computer, this main program is in charge of handling required information for the processes mentioned above by the slave computer. The programming in the host was developed in Visual C++ version 5.0.

Manual motion

The Windows-based program in the host computer facilitated the robot operation. For instance, the arm can be positioned manually using the dialogue control shown in Figure 3.7.

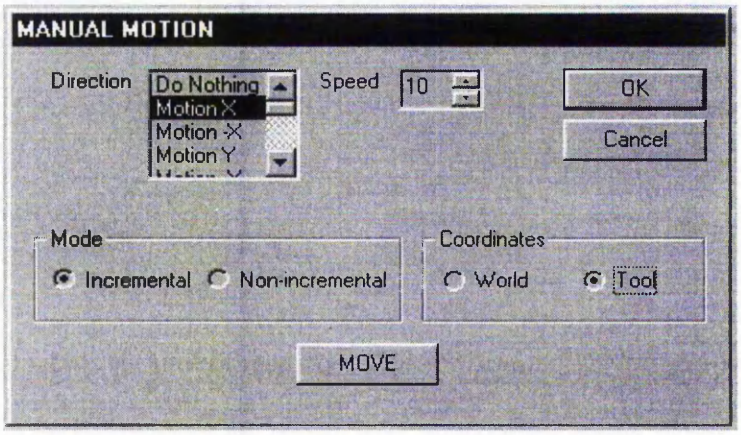


Figure 3.7: Manual motion dialogue

The direction can be selected as translation or rotation and it can be either positive or negative. The speed can be selected between 0 and 50, which represents a percentage of the nominal speed value. For example, a selected value of 10 will imply a 10% speed of the nominal arm speed given by the SPEED command in VAL language. This speed was limited to 50 due to the high acceleration that appeared at the end-effector above this value.

The type of motion can be selected in the Mode option. It can be incremental, which means that the value will be added to current arm's position. In the non-incremental option, the new value will be added to the arm's starting location, so that incremental motions are relative to this initial position. The arm can be positioned either in World or Tool coordinates, with origin at the robot's base or flange edge of the robot arm, respectively.

3.6 Force Sensing Implementation

Figure 3.6, showed a feedback line connected from the sensor to the supervisory computer carrying information about the current state of the end-effector. This

data contains information about force and torque related to the sensor frame in the workspace. In this section, a brief description of the F/T sensor is given and parameters for the selection, calibration and software implementation are explained.

3.6.1 Force/Torque sensing and selection

The selected sensing element is a wrist Force/Torque sensor which is placed between the flange edge of the arm and the adaptor plate of the robot. The force and torque are resolved into a six degrees of freedom force-torque vector with respect to the sensor coordinate frame.

The sensing principle is based on strain gauges, which are mounted on beams connected to the shaft of the sensor case. The strain gauges can be made from metal foil or silicon. Generally speaking, silicon-based sensors are temperature dependent but when compensated they offer good accuracy. Metal foil strain gauges are less accurate when used at low pressures, however they are more robust when used at higher forces. The mechanical placement of the gauges, coupled with the mechanical location of the beam, determines the combination of forces in a bridge output signal. The number of beams is three or four depending on the manufacturer. According to Assurance Technology Inc. [51], using three beams instead of four performs better since with four beams, an eight-component strain vector requires an eight-by-six calibration matrix compared to the six-by-six required with three beams. This reduces sensor costs and increases reliability and speed because the available computational power can be dedicated to fewer control algorithms.

Information was gathered from different sources and suppliers to compare different sensors readily available in the market. Table 3.1 shows the features for each of the sensor reviewed.

It can be observed from the table that the JR3 sensor specifications are superior to the others, this sensor was chosen. Its sampling rate is the highest available, the built-in software capabilities are varied and provide more flexibility for software design. Additionally, another advantage is that this sensor only needs a receiver

FEATURES	SENSORS		
	MINIFORCE	JR3	ATI
Models	SCT-05-1	100M40	9105Gamma2(130N/10Nm)
Measuring range	100 N (fy)	800N (fz)	260 N (fz)
Technology	metal foil	metal foil	Silicon
Sampling rate	5 kHz	8 kHz	572Hz(serial)1.2kHz(parallel)
Communication	RS-485	RS-485	RS-232/ LSTTL
Processor (clock freq.)	DSP (16 MHz)	DSP ADSP-2105	D S P
Resolution	0.2 N	0.01 lb (x,y); 0.02 lb (z)	0.10 N (10 C/N)
Sensitivity	1 mV/V		
Nonlinearity	0.10%		
Hysteresis	0.10%		
Repeatability	0.10%		
Cross sensitivity	1 % (corrected)		
Excitation	5 V DC/AC	Provided by the Bus	
Weight (kg)		0.65	0.181
F/T reference point	Yes	Yes	
Digital filters (Hz)	20 to 1000	0.5 to 500 @ 8 kHz	235
Processing	Sync/Async		
Data format	N, Nm, Binary	lb, N	
Tool transformation	Yes	Yes	
Biasing	Yes	Yes	Yes
Decoupling		Yes	
Max/Min peak capture		Yes	
Vector magnitude calc.		Yes	
Threshold calculation		Yes	Yes
Rate calculation		Yes	
Price (£)	3700	6000	5626

Table 3.1: Comparison of F/T sensors

card plugged into the PC slot to process the serial data from the sensor unit. The sensor unit houses all the conditioning electronics, amplifiers and converters so no additional cards are needed, as would have been the case with the other options. The sensor unit is shown in Figure 3.8. For additional comparisons, the reader may refer to the report written by R. Voyles at Carnegie Mellon University [52].

Figure 3.8: JR3 sensor unit

Sensor adaptor plate and gripper

A sensor adaptor plate was designed to attach the gripper to the JR3 F/T sensor. The sensor is located between the flange edge of the arm and the sensor adaptor plate as illustrated in Figure 3.9.

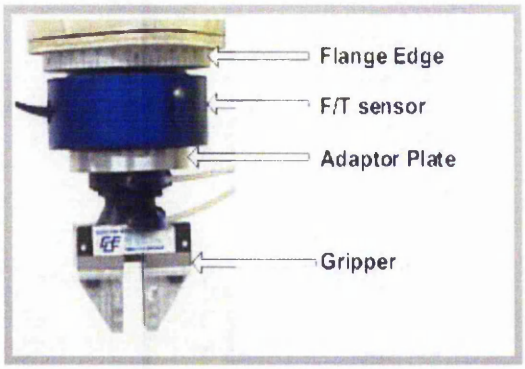


Figure 3.9: Sensor-Plate-Gripper

Additionally, dowels and grooves were included in the design of the sensor plate to provide a perfect fit of the sensor and gripper. The gripper is pneumatically powered and controlled from the supervisor computer or via a VAL program.

JR3 memory mapping

The JR3 receiver board is based on a Digital Signal Processor (DSP), that contains a shared dual-ported RAM that both the user and the DSP can read and write. This memory area consists of 16k 2-byte words. The first 8kb of this space is RAM, while the remaining 8kb consists of status registers and other features. The majority of the user activity takes place in the first 256 words of the shared space. In these locations, many operations are available to the user such as offset removal, data decoupling, saturation detection, digital low-pass filtering, force and moment vector magnitude calculation, peak detection, rate calculation, threshold monitoring, coordinate translation and coordinate rotation.

3.6.2 Force representation

The sensor was mounted according to manufacturer's specifications, who also provided a core code example to access the DSP memory locations. It was thought that a graphical representation of force and moment on the computer screen would facilitate the analysis of contact forces, so a Graphical Interface Program was written for this purpose.

The sampling rate for data acquisition is 8 kHz although the refresh on the computer's screen is slower due to the time used for reading and writing to the DSP and displaying. A typical F/T trace on the screen computer is shown in Figure 3.10. There are two graphs, the upper graph represents Fx, Fy and Fz signals while the values of Mx, My and Mz are represented in the lower graph. Some other features include selection of signal gain (Y axis), time scale (X axis), time delay interval (for updating screen values) and storing and loading signals from and to a file.

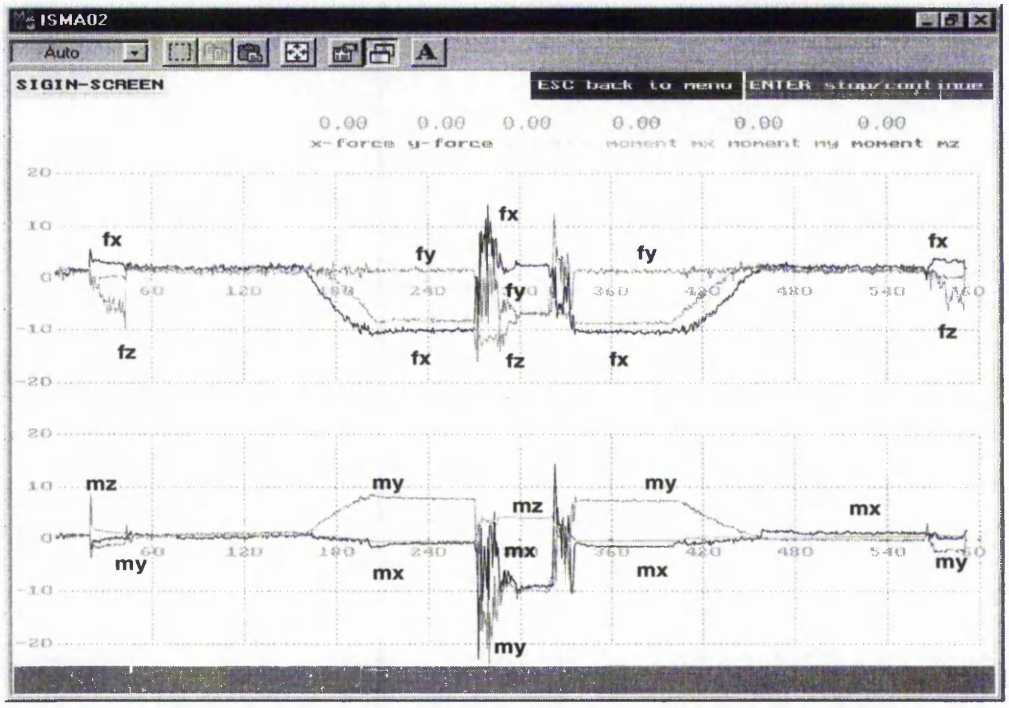


Figure 3.10: Force and Moment signals

3.6.3 F/T sensor features

Rotation and translation

The JR3 sensor has many useful features that facilitate data manipulation and analysis, some of these were included within the NNC. Coordinate transformation is possible via the dialogue shown in Figure 3.11.

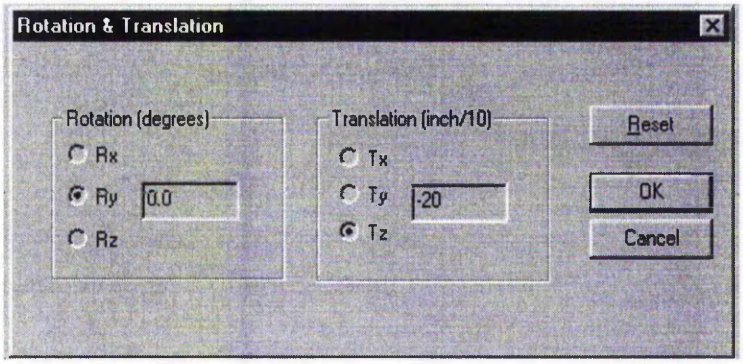


Figure 3.11: Rotation/Translation dialogue

The sensor coordinate frame can be rotated and/or translated. During experiments these transformations were very useful since the Tool, World and Sensor frames were superimposed, so that only one common reference frame was used. This facilitated easier motion interpretation and data analysis.

Force/Torque reading and pattern acquisition

The force/torque patterns needed during network training can be acquired through the dialogue box shown in Figure 3.12.

Force and moment values are displayed in the appropriate boxes. The *tare* or offset in the signal can be eliminated by clicking the "Reset offsets" button. There is also a boundary limit for the force. In the figure, it is shown as a value of 10 lb and -10 lb for the upper and lower limit respectively. By setting these limits the NNC will stop the operation whenever any of these values are reached and the user prompted to take an action. This prevents potential damage to the parts, end-effector or sensor. The button labelled "Acquire" is used only when training the network the first time. The current F/T pattern can be acquired

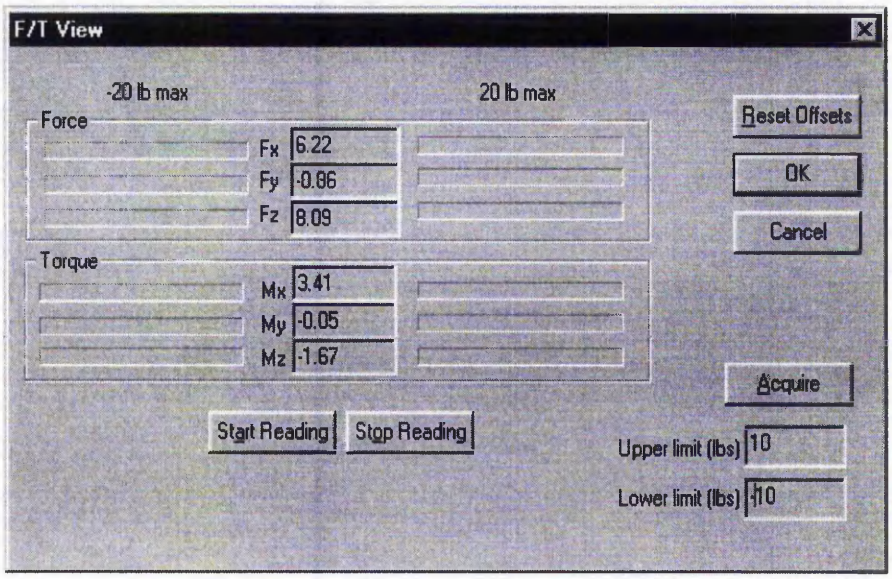


Figure 3.12: Settings and pattern acquisition

using this button and in a subsequent Window the user will be prompted to selected the motion corresponding to this pattern. This is to create the initial learning or knowledge base for the network, which will be described in Chapter 8.

Sampling rate, signal filtering and F/T data acquisition

The JR3 sensor can provide either unfiltered decoupled data or data passed through cascade low-pass filters. This is particularly useful to diminish unwanted noise signals. The cut-off filter frequency is 1/16 of the sample rate of the sensor. Each succeeding filter has a cut-off frequency of 1/4 of the preceding filter. During experiments, the sampling rate was maintained at the nominal value of 8 kHz. Hence, the cut-off frequencies for the digital filters were 500 Hz, 125 Hz, 31.25 Hz, 7.81 Hz, 1.95 Hz and 0.48 Hz. The transient stage and the processing time for the NNC have to be considered during the reading of the F/T data. The initial F/T reading was carried out as shown in Figure 3.13.

The start and end positions are marked as **A** and **B** respectively. Between these points there is a transient stage due to constrained motion or arm impact. After an appropriate delay, the F/T vector is sampled and the NNC processes the in-

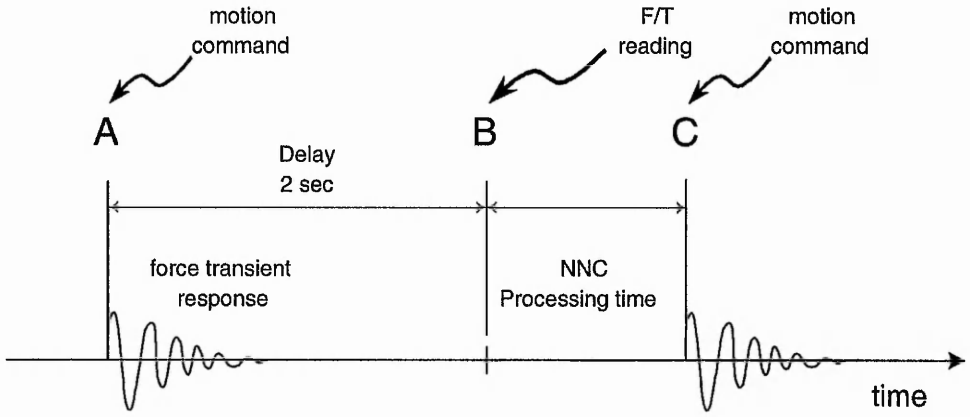


Figure 3.13: Delay for force reading

formation producing a motion command at time **C**. After this point a new cycle may be initiated. The delay is necessary to ensure that the F/T reading has reached a steady state. Theoretically, this delay should be at least the magnitude of the servoing time (28 ms). However, during experiments a longer delay was found to be necessary as will be seen in Chapter 7 where additional time considerations are explained. A delay of two seconds was found to be appropriate during the implementation of the F/T sensing and during early experiments with the manipulator.

3.7 Summary

The testbed design included relevant issues ranging from safety regulations, robot controller modification and force sensing implementation.

The implementation of a reliable communication with the controller in ALTER mode was essential. ALTER mode was created by the manufacturer to provide control in real-time. The exchange of data between the supervisory computer and the robot controller is very strict and there are no ways to recover from data errors or to stop the communication. The robot controller simply sends a timeout message and terminates the exchange of data if no reply is received from the supervisory computer during the allocated time. With the other serial line in supervisor mode the original terminal is no longer needed since its functions

were transferred to the supervisory computer.

With the improved master-slave architecture, the communication tasks in ALTER mode were delegated to the slave, hence leaving more processing power and time to the supervisory computer. With this architecture, it is also possible to monitor the force continuously, although this will not be made due to the quasi-static incremental motions required during assembly operations. However, this capability can be used if required.

Chapter 4

Quantifying System Accuracy/Uncertainty

Success in robotic assembly is directly linked to the accuracy of the overall system. However, robots and associated systems are not accurate by definition and an estimate of the uncertainty is desired to compensate for these errors. In order to identify the absolute accuracy of the manipulator several factor should be considered first such as arm configuration, payload, arm deflection, etc. The absolute accuracy of the manipulator varies according to these factors and this is the reason why manufacturers do not normally give this parameters in their data sheets, but only provide the repeatability. In this Chapter, the associated errors in positioning the end-effector are determined. The manipulator was driven with the same type of data that the NNC uses during automated assembly operations. As these tests only relate to local incremental motions, it is necessary to clarify that the assessment of the accuracy is only related to *local uncertainty* within the work space. Therefore, it should not be considered as an exhaustive analysis of absolute accuracy for the PUMA 761. The following sections briefly review:*resolution, accuracy and repeatability.*

4.0.1 Resolution

The spatial resolution is the minimum distance that the tool centre can be guaranteed to move. In other words, the smallest increment of movement into which

the robot can divide its work volume.

Factors that influence the spatial resolution are basically the resolution in the feedback measuring system to move the arm and the mechanical inaccuracies in the robot's links and joints. Errors in the feedback system are associated to the ability of the encoders to resolve for the minimum increment and the computer to process this information. The mechanical inaccuracies basically come from the deflection of the links and gear backlash. The contribution of these errors will determine the actual spatial resolution.

4.0.2 Accuracy

The accuracy can be thought of as the difference between the actual and the commanded location. The accuracy of a robot can be defined in terms of spatial resolution because the ability to achieve a given target point depends on how closely the robot can define the control increments for each of its joint motions. In the worst case, the desired point would lie in the middle between two adjacent control increments[53].

In fact, the mechanical inaccuracies would affect the ability to reach the target position as shown in the statistical distribution in Figure 4.1.

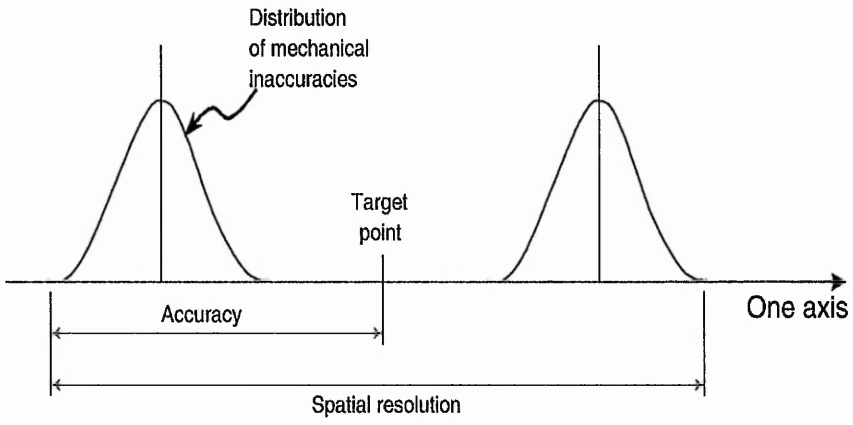


Figure 4.1: Statistical distribution of accuracy and spatial resolution.

As it is shown, the accuracy will depend on the mechanical inaccuracies, which on their own are affected by the robot configuration, work volume, payload, stretch of the links, etc.

4.0.3 Repeatability

The repeatability refers to the variation of the actual location when the manipulator repeatedly moves the tool centre point to the same commanded location. Most robot manufacturers provide a numerical value for the repeatability rather than the accuracy of their robots. The reason is that the accuracy depends upon the particular load that the gripper carries. A heavier weight causes larger deflections of the robot links and larger load on the joints, which degrades the accuracy. The repeatability value, however, is almost independent of the gripper load and therefore can be specified for any robot arm [54].

4.1 Uncertainties in the robot system

The main source of uncertainty is the robot interaction with the environment. In real-world applications the arm may contact the environment in multiple robot configurations, hence the number of possible contacts are innumerable. Added to this complexity is the uncertainty in sensory systems and motion/control mechanisms. From this reasoning, three types of uncertainties can clearly be identified:

- Sensing uncertainty
- Control uncertainty
- Model uncertainty

Sensing uncertainty. Typically, in force sensors, noise is added to the force signal. Temperature plays an important role in silicon-based force sensors. These types of sensors are temperature dependent and their output is directly affected unless a scheme compensation is added to the signal conditioning circuitry. Generally speaking, the transfer function of silicon-based sensors suffer a change in both, gain voltage and DC offset. There are analogue and digital techniques to compensate for these errors, however the latter methods have proved to outperform analogue techniques[55].

Control uncertainty. This is due to the limitations of manipulators to accurately sense the feedback information from the position encoders. Quantization

errors are likely to occur due to the analogue to digital conversion. On the other hand, uncertainty is also associated with the accurate positioning of the end-effector since errors are present in the mechanisms as discussed before.

Model uncertainty. It would be impossible to accomplish an exact model of the robotic system, since it depends on robot configuration and relative position of the sensor, adaptor plate, gripper, assembly and fixed part. Static and sliding friction is also present in the model and it is difficult to quantify accurately.

Robotic assembly operations are accomplished under extreme uncertainty for all the reasons explained above. Some researchers have included system uncertainties in their models to overcome this situation. Lozano-Perez et al. [20], included the uncertainty in the geometric model task in configuration space, this method was enhanced by Michael Erdmann [56], where the task is modelled as geometrical goal. For the method to succeed, the velocity must lie within a pre-determined uncertainty cone, this availability is a strong assumption in real world-systems. Only a few researchers actually quantify the errors in their systems as a baseline to objectively assess their control scheme performance. V. Gullapalli compared favourably his stochastic reinforcement learning method based on the actual uncertainty involved in the system [57, 27].

Undoubtedly, because of the complexity of the robot system it would be impossible to determine all parameters. However, an analysis of the accuracy of the control and sensory system will be important to assess the performance of the Neural Network Controller. In the following sections, an analysis of the accuracy of the PUMA robot and the sensor system is therefore presented.

4.1.1 Positional uncertainty

Positional uncertainty is mainly due to gear backlash and friction in the manipulator mechanisms. Backlash is a well known problem in gear design. Friction is a very complex problem when considering *sliding* and *static* friction. Popovic and Goldenberg [58] demonstrated that friction is position and velocity dependent using the joint 1 mechanism of a PUMA 560 robot. In their model using spectral analysis, they developed a method to determine the contribution of individual

parts to the friction model, which included static and sliding friction.

For practical purposes in this research it is sufficient to determine the positional error and to have an idea of the involved uncertainty. The positional accuracy of the arm was therefore measured and compared with the manufacture's specifications.

The rig used for measuring linear and angular displacement is shown in Figure 4.2.

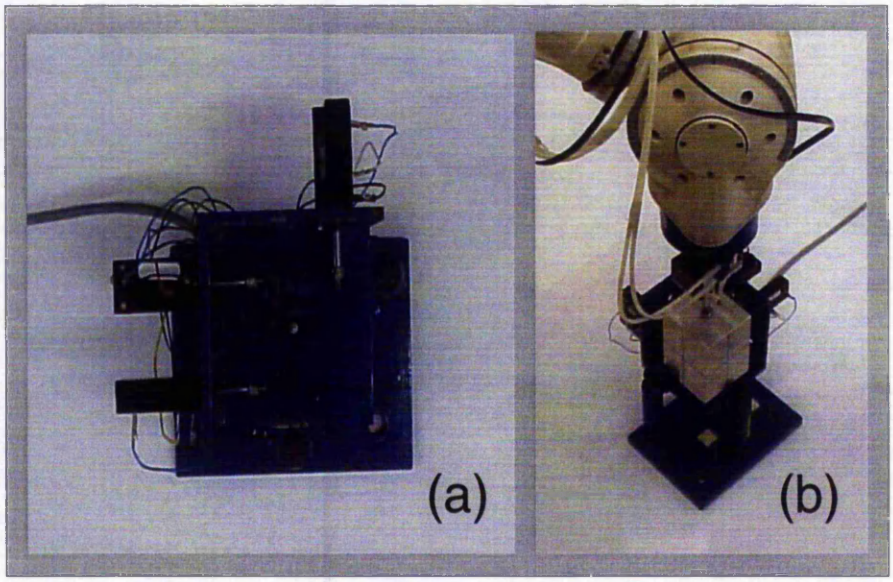


Figure 4.2: Test rig to measure linear and angular displacements

The sensing elements are six $10\,k\Omega$ "Sakae" linear potentiometers with a specified linearity of 0.5%. The potentiometers were placed as illustrated in the Figure. A voltage supply was connected to each potentiometer and the voltage measured at each output.

Before using the potentiometers, they were calibrated separately. This was accomplished by applying incremental displacements of 0.5 mm and measuring the corresponding voltage. The resultant calibration curve is shown in Figure 4.3.

The graph shows a very similar characteristic for all potentiometers, except for the fifth potentiometer, which resulted in different calibration. The area enclosed by the broken line is the region where the potentiometers were used during the actual displacement measurements. The least-squares regression method [59] was used

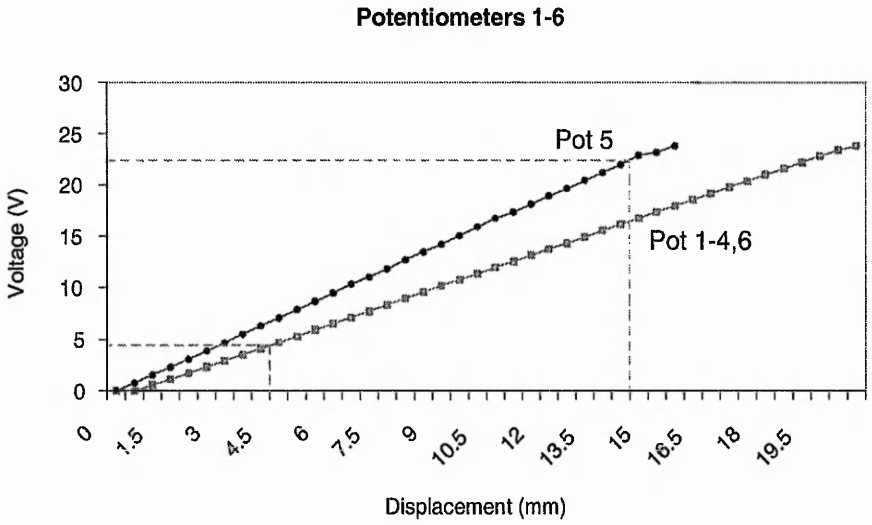


Figure 4.3: Calibration Curve of Potentiometers

Potentiometer	Voltage/Displacement (V/mm)
1	1.20
2	1.19
3	1.20
4	1.21
5	1.55
6	1.19

Table 4.1: Voltage-displacement characteristics

to fit the best equation for each curve. The *Voltage/displacement* characteristic found for each potentiometer is indicated in Table 4.1:

During measurements a 3 1/2 digit voltmeter and a voltage range from 10 V to 25 V were used. It implied that the voltage resolution within the measuring region was 0.01 V. Considering this voltage and the values given in the Table 4.1, it is simple to determine the resolution of the measuring system. It was approximately 8 μm for pots 1-4,6 and 6 μm for pot5. This value is clearly lower than the robot's repeatability ($\pm$ 200 μm), hence the measuring system using the rig and the potentiometers was considered satisfactory.

The total linear displacement can be measured with any set of three orthogonally

placed potentiometers. The displacement can be measured in any axis as well as the corresponding error termed *Orthogonal Displacement Error (ODE)*, which is the displacement that occurs in the orthogonal axes where the linear motion is not applied. This error is mostly due to gear backslash.

For measuring angular displacement three potentiometers were needed, two for the actual rotation and potentiometer five for measuring the displacement error in the vertical axis. Despite of the need of three potentiometers only, the readings were taken from all the potentiometers to quantify the *Orthogonal Angular Error (OAE)*. Similarly to the ODE, the OAE is an undesired error appearing in the complementary rotational directions.

The graphs for linear displacement are shown in Figures 4.4 to 4.6, whereas the graphs for rotational displacement are shown in Figure 4.7. In these graphs, the associated error can be seen near to the horizontal. In the ideal case, the linear and angular displacement should occur only about the corresponding axis where the displacement is desired. In any of the other axes the displacement should be zero.

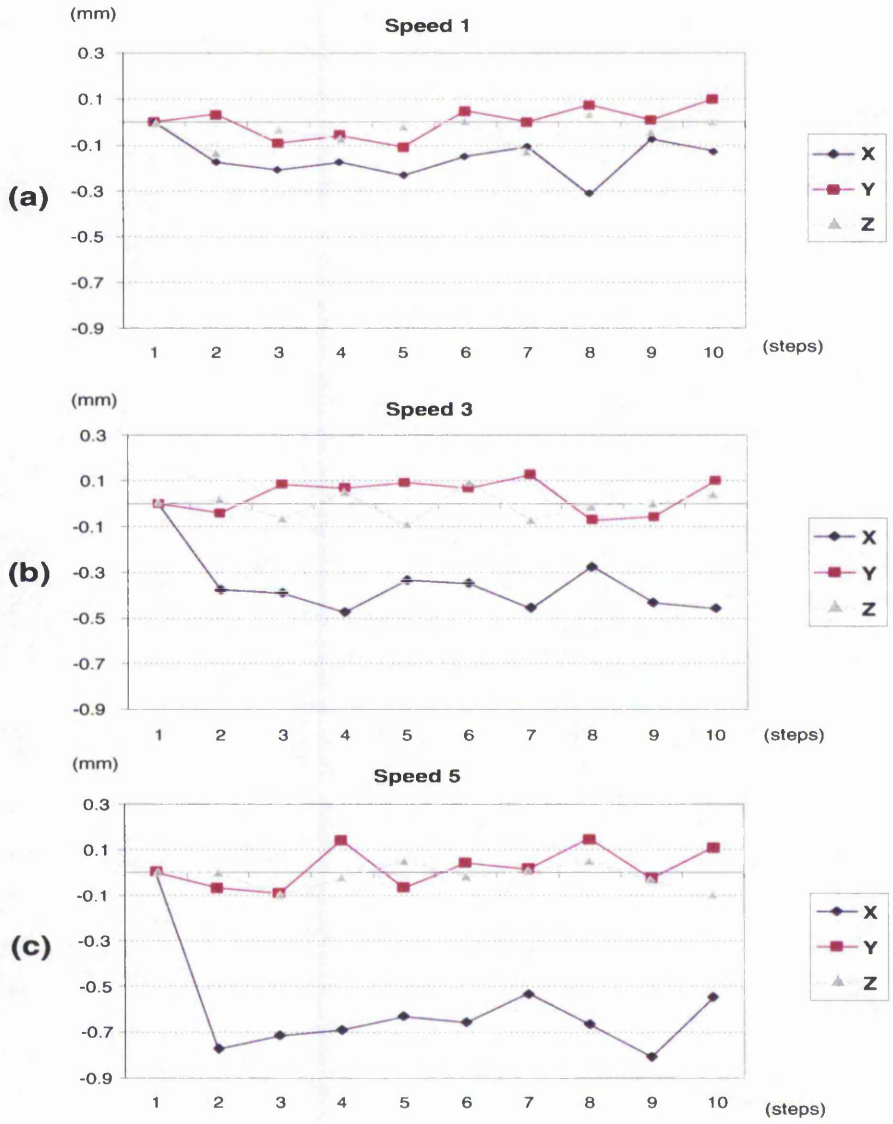


Figure 4.4: Accuracy in X direction

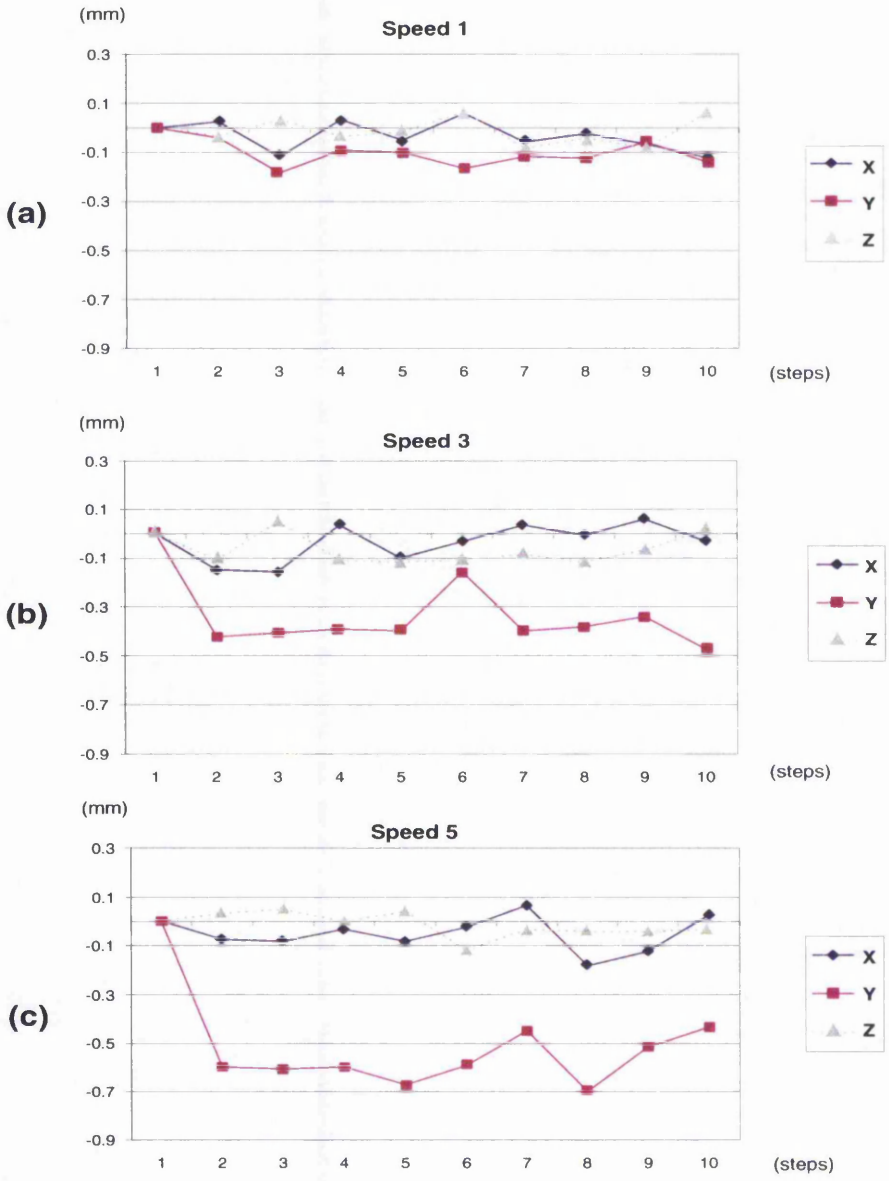


Figure 4.5: Accuracy in Y direction

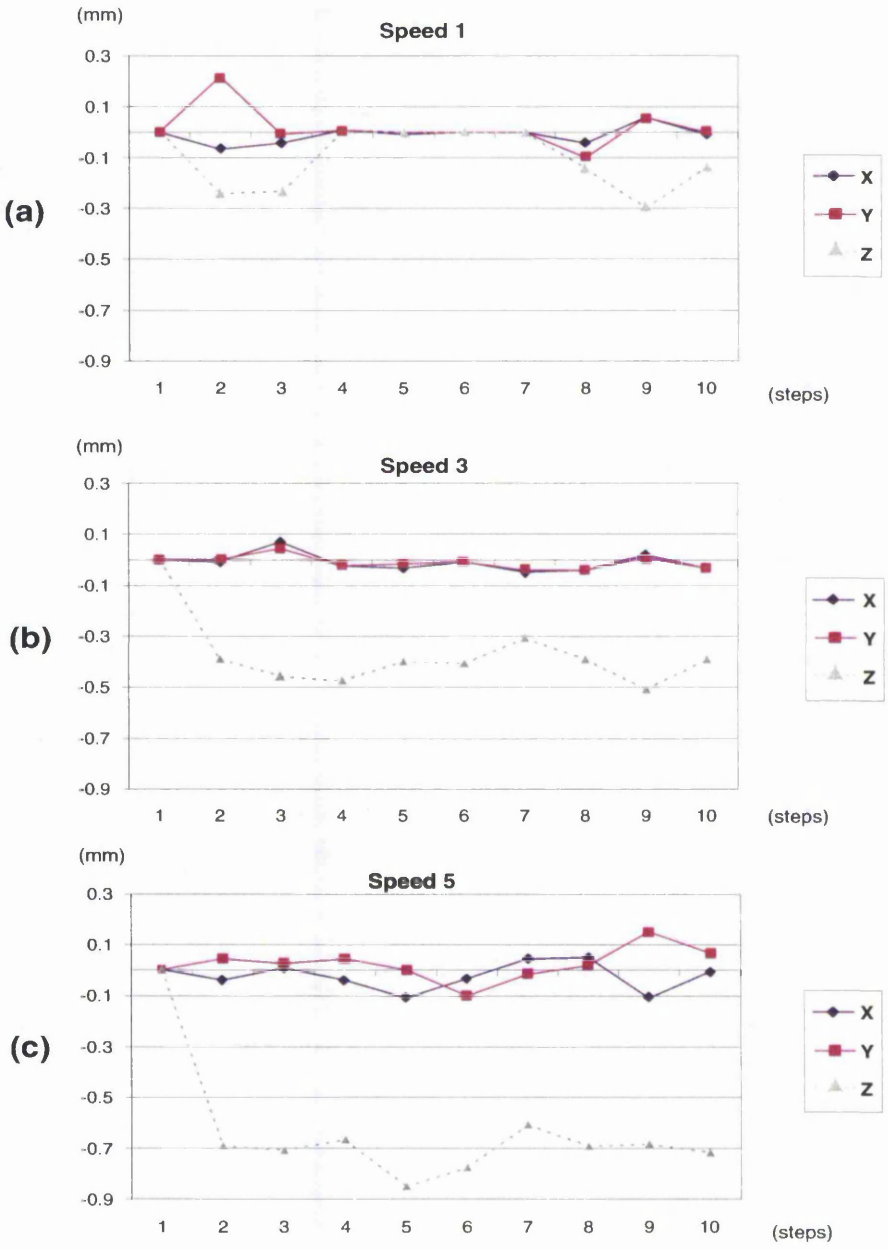


Figure 4.6: Accuracy in Z direction

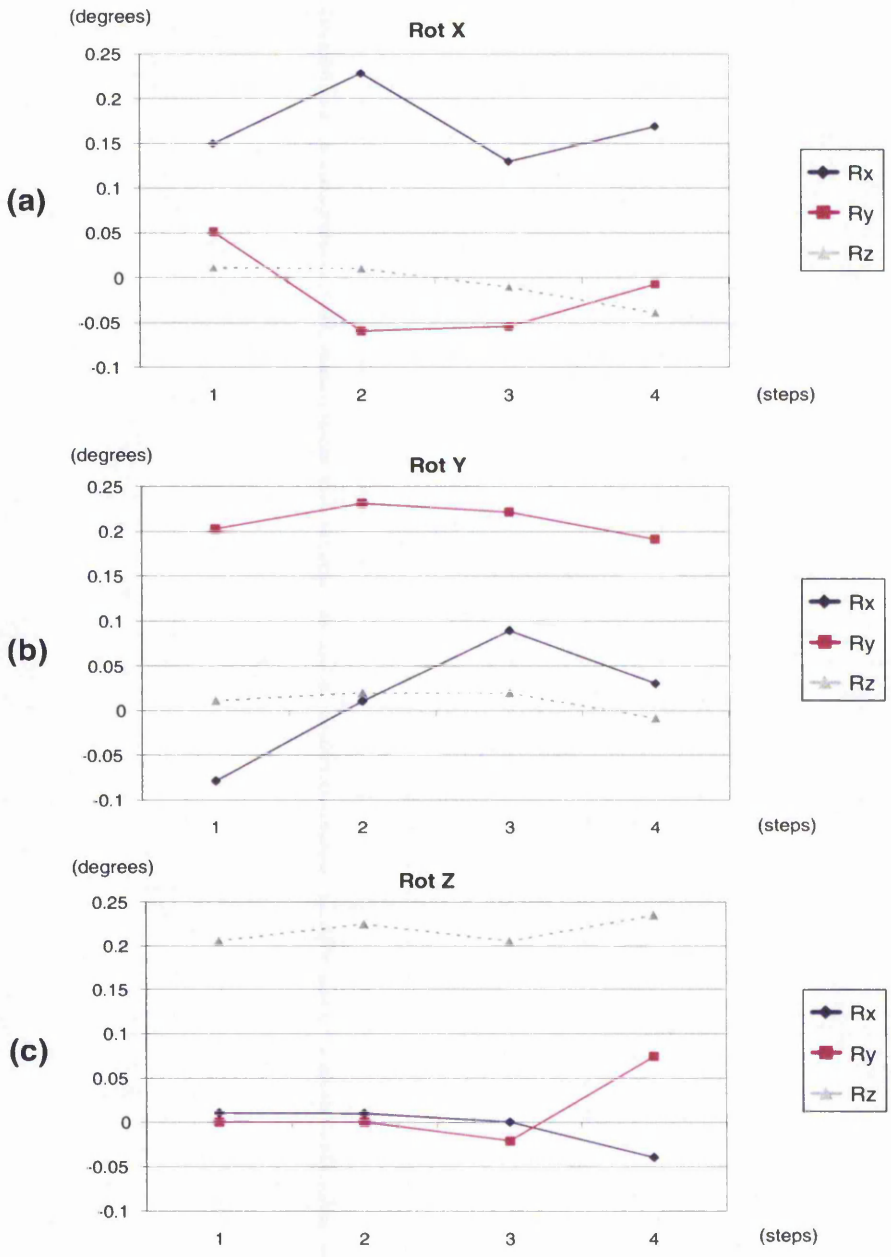


Figure 4.7: Rotational accuracy

The graphs in Figure 4.4 to 4.6 show the displacement value of the end-effector in the X, Y and Z directions at speeds 1, 3 and 5. These values were recorded during 10 motion steps. The actual size of the step was the equivalent to one communication cycle in ALTER mode in the same manner as the NNC would command the robot moves. It can be observed that at speed 1, the system presented a high ODE, sometimes more than 100% compared to the motion in the complementary orthogonal axes. This means that the arm moved in the wrong direction more than it was expected to, in the desired direction. At speeds 3 and 5 it can be seen that this error was significantly reduced compared with the actual move in which the linear displacement was commanded. Higher speeds were tested, however the end-effector presented abrupt jerky motions since the robot controller accelerated and stopped in a very short time. From the graphs, it can be concluded that speeds 3 and 5 are appropriate for moving the arm. However, later during an independent test, it was found that moving the arm during direct impact at speed 5 resulted in a force increment in the range of 10 lb to 18 lb. Bearing in mind that forces during manipulations are about 10 lb[1], the total forces then may well surpass the 25 lb limit of the sensor and therefore damage was likely to occur. Although the ODE was higher at speed 3 than at speed 5, speed 3 was selected to move the arm during constrained motion so as to prevent any damage to the sensor if direct impact occurred.

To determine the optimal rotational movement at the end-effector, different increments were tested at the corresponding speed in free and constraint motion. It was found that a value of speed 30 was appropriate. The corresponding graphs are shown in Figure 4.7. The OAE can be observed at the lower part of each graph.

The results of the experiments above are summarised in Table 4.2. This Table shows the positional accuracy of the PUMA robot relative to the work space in which the arm will be operated during NNC performance tests.

Table 4.2 is divided in two major parts. The first section is related to the resolution of the arm and the second refers to the cross-coupling maximum error (ODE

[1]This value was observed experimentally.

Resolution		Cross-coupled maximum error	
Motion Direction	Value	(%)	Error Direction
X	0.3940 ± 0.064mm	28	Z
Y	0.3760 ± 0.084mm	64	Z
Z	0.415 ± 0.055mm	16	X
Rx	0.168 ± 0.037 °	116	Z
Ry	0.211 ± 0.016 °	37	Z
Rz	0.218 ± 0.012 °	31	Rx

Table 4.2: PUMA 761 positional accuracy

and OAE).

In the first and second column, the applied motion direction and their corresponding value are given. In the third column, the percentage of the maximum error is given whereas the axis in which this occurred is indicated in the fourth column. These resolution values were determined averaging the result values from ten incremental motions at every particular direction.

4.1.2 Sensor uncertainty

The sensing system is of utmost importance since control actions are taken according to the interaction of the arm with the environment. Hence, a reliable sensing system will improve the control accuracy. The more accurate the sensing the better the decision making within the force-action mapping in the NNC will be.

Due to its complexity, an exhaustive analysis of the sensor error is out of the scope in the present study. However, main errors such as cross-coupling, thermal drift and linearity that most affect the readings are analysed.

Cross-Coupling error

This error refers to the erroneous measurement at a particular axis as a result of an applied force in an orthogonal axis. For instance, a pure moment applied

around X-axis should not produce any moment around Y or Z axis.

Similar to Voyles' approach [52], a constant force was applied to produce different moment values around a particular axis. This was achieved by attaching a metal rod to the end-effector and producing different torques by using a constant force. This is illustrated in Figure 4.8.

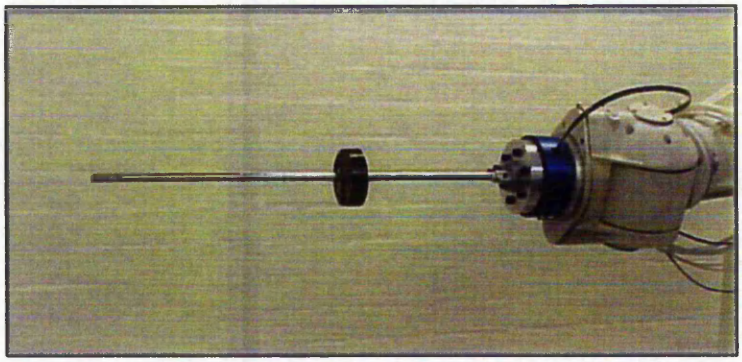


Figure 4.8: Cross-Coupling measurement

The rod was positioned in such a manner to produce only the required moment component. A mass of 1 lb was hung from the rod and moved along it at increments of 1 in. The output from the sensor was recorded for both Mx and My, the actual plot is shown in Figure 4.9. It is important to mention that the F/T sensor factory units are imperial and its output could not be changed to the SI equivalent by software but by hardware. It was decided then to continue using the imperial system.

In both cases the response was satisfactory and the cross-coupling error minimal for practical purposes.

Signal drift

An experiment was set to determine the stability of the force signal due to variation in temperature. The temperature increase was primarily due to the electronic circuitry in the sensor. The results are shown in Figure 4.10.

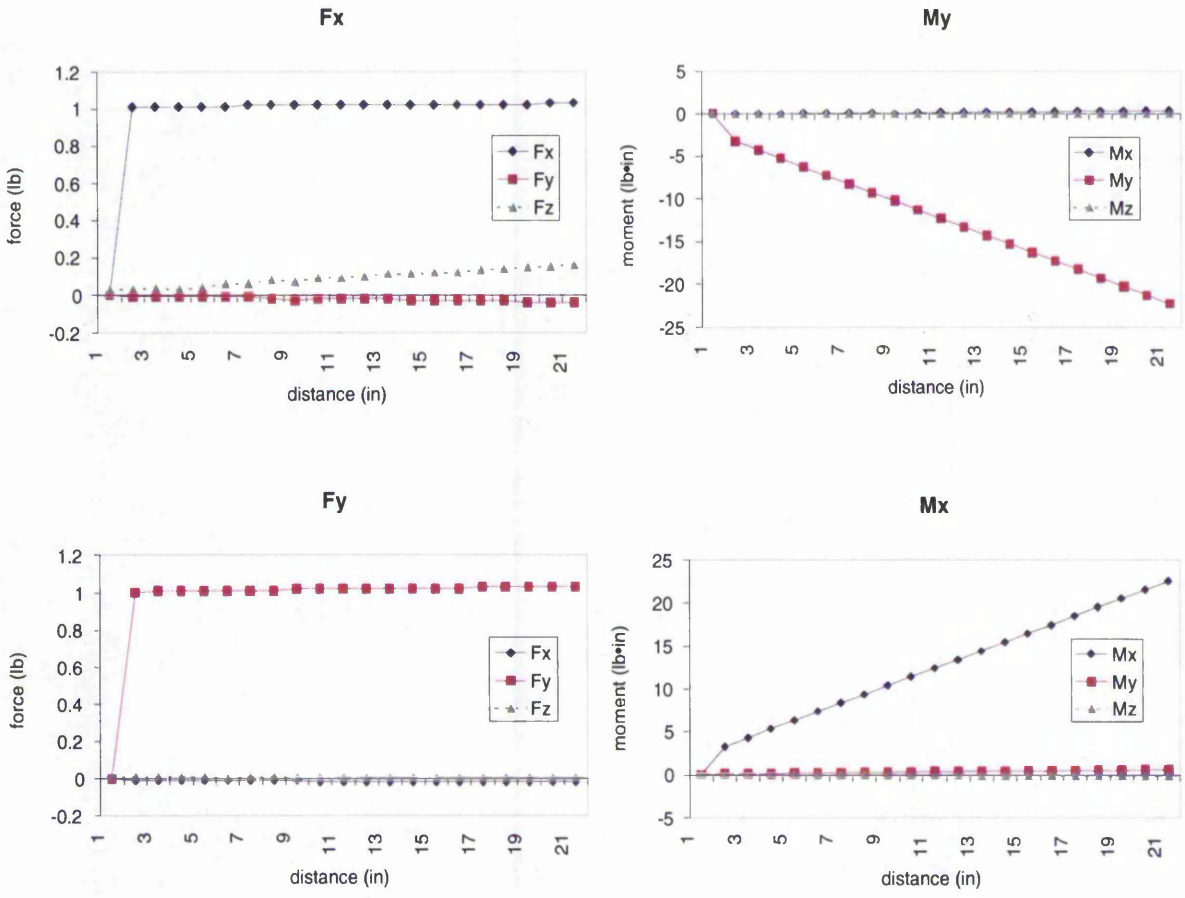


Figure 4.9: Measuring moment applying constant force

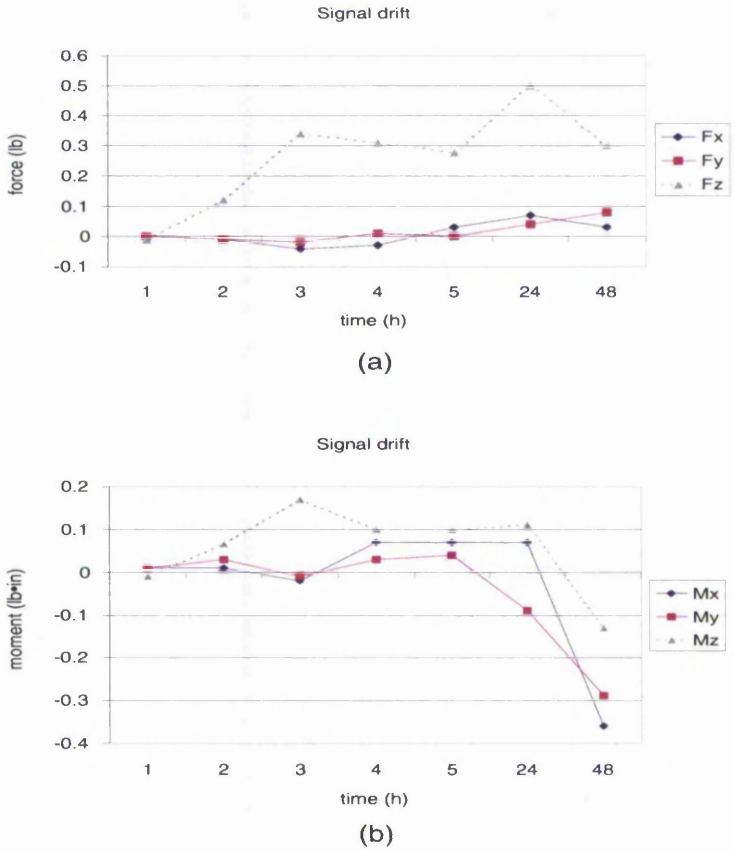


Figure 4.10: Force and moment drift due to temperature increase

The force and moment signal were measured at intervals of 1 hour for five hours and then at 24 h and 48 h. The maximum change was detected at 24 h where the force signal increased to 0.5 lb, after this time this value started to decrease. With regard to the drift in moment signals, their absolute value increased. From both plots, it is concluded that this signal drift is important and a dynamic compensation should be considered for the NNC design.

Linearity

This test was carried out by positioning the arm at a certain angle to produce a force component in all axes. The resultant force was plotted against the applied force as shown in Figure 4.11.

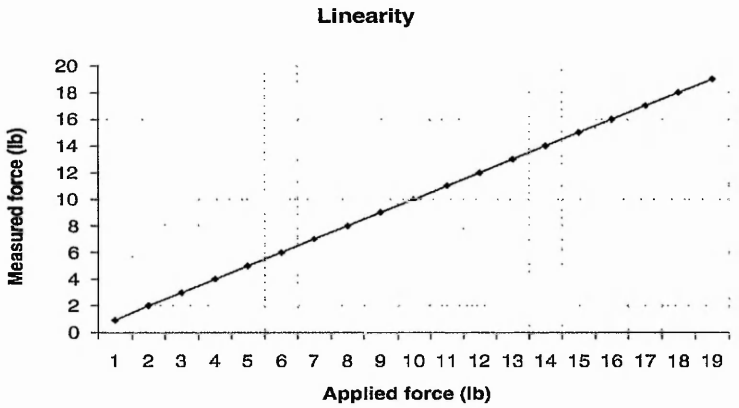


Figure 4.11: F/T sensor linearity

The sensor response to the applied force was found to be virtually linear since the discrepancies with the measured value were very small.

4.2 Conclusions

The *local accuracy* of the robot system was analysed and the main errors identified. In terms of positional accuracy, the most important parameters were the cross-coupling errors identified during linear and rotational displacements. Regarding force sensing, errors such as cross-coupling and thermal drift were also quantified.

The accuracy analysis of the manipulator was very important to specify appropriate incremental motions for the arm under ALTER mode. Results revealed the extreme uncertainty under which the incremental motions develop. The overall uncertainty was mostly due to the contribution of the ODE, OAE and thermal drift in the F/T signal. The NNC design should address this uncertainty by providing a self-adaptive behaviour to the robot and compensating for the above errors.

Chapter 5

Forces During Assembly and NNC Requirements

A number of experiments are carried out to analyse the nature of the forces involved during assembly. Based on this analysis two important stages in the NNC are identified: Adaptation and Decision. An initial structure of the NNC is proposed based on the results obtained during experiments.

5.1 Investigating the Forces Acting During Mechanical Assembly

Having set up the robotic system, the next stage was to design the NNC. Regarding its design, the logical step in a bottom-up approach is to study the nature of the force traces during assembly. From the analysis of these forces, the requirements for the NNC were identified. This analysis included several peg-in-hole operations carried out using a VAL program and different pegs at different orientations.

The Figure 5.1 illustrates the implementation of the testbed site and the assembly fixtures used for the assembly.

All assembly components were made from aluminium. To measure insertion forces at different geometry and orientation, a *master block* was made in which inter-

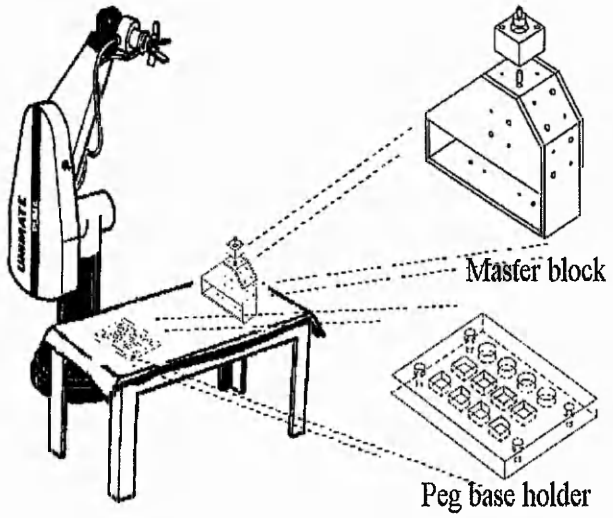


Figure 5.1: Testbed site and assembly components

changeable *female blocks* were placed at a variety of orientations (180°, 90° and 45°). The *peg base holder* was used to place the pegs while not in use. The clearance between female and male components was set from 0.075 mm to 0.1 mm and the chamfer set to 45° in the female components. Sets of three different pegs were used with shapes matching the female blocks (square, radiused-square, and circular). These components are illustrated in Figure 5.2.

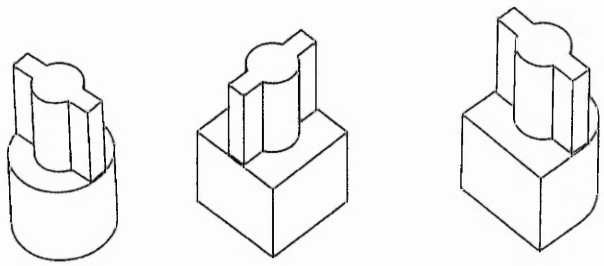


Figure 5.2: Male components (pegs) used for assembly

As it can be seen, the pegs were manufactured with *"wings"* to avoid rotation within the gripper. The fingers were also especially manufactured to grasp these pegs and to guarantee the best grasp. This was particularly important for the assembly of the square and radiused-square pegs.

Forces acting during the assembly of the square peg are shown in Figure 5.3 and Figure 5.4. The plots show forces and moments occurring during the operation. For this experiment the female block was orientated at 45° on the master block. The sensor was set to its maximum sampling rate (8 kHz) and its reference frame was aligned with the robot's tool coordinate frame. The complete operation was carried out in approximately one minute including grasping the part, gripper alignment, insertion, withdrawal and returning the component to the base holder. Constrained motion, as can be seen in the plots, is divided into *gripper alignment, insertion* and *rest* stages. The stages during the operation can be described as follows:

Stage 1 *(pick)*, is the period when the peg is picked up and stage 7 *(leave)* when the peg is returned to the base holder. It can be seen that there is no symmetry between these stages however, the magnitude of the force signals does follow a similar pattern, which was not the case for the corresponding moment forces.

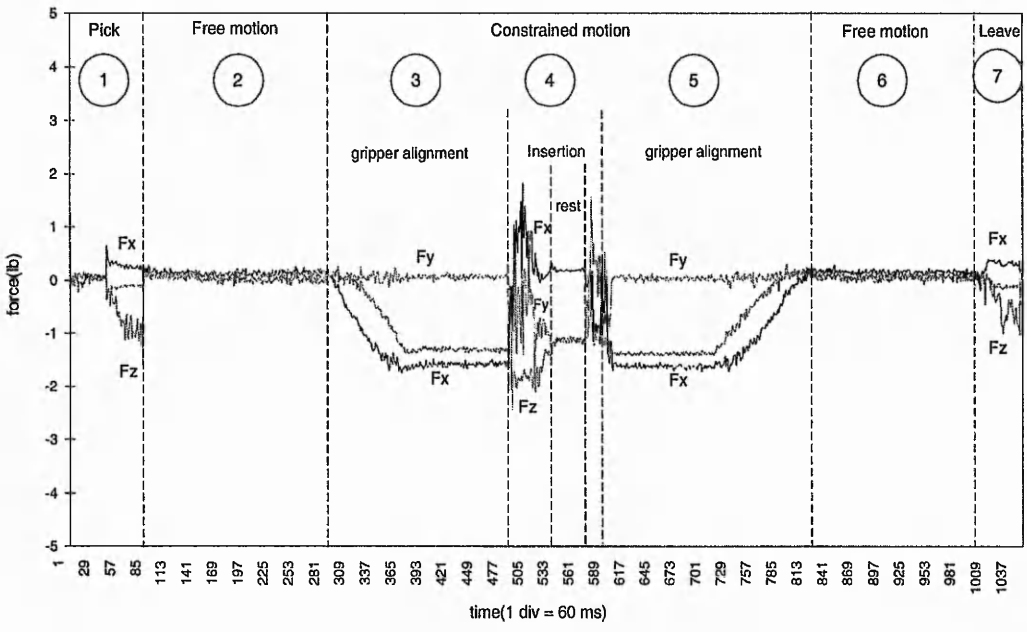


Figure 5.3: Forces acting in the square peg assembly

Stage 2 is a *free-motion* movement parallel to the X-Y plane, which means that the arm end-effector is moving towards the insertion block with no rotation in X

or Y axis, i.e. only translation is occurring. Within this period it can be seen that vibration appears during the motion. A free-constrained transition starts (stage 3), when the gripper is aligned perpendicular to the contact surface (45°) in readiness to insert the peg into the hole. Even though this stage is strictly not constrained since no contact is being made onto the surface, rotation occurs and gravity force component g_x increases, which contributes to an increase in M_y. The orthogonal relationship between force and moment in the X and Y axis can be observed clearly.

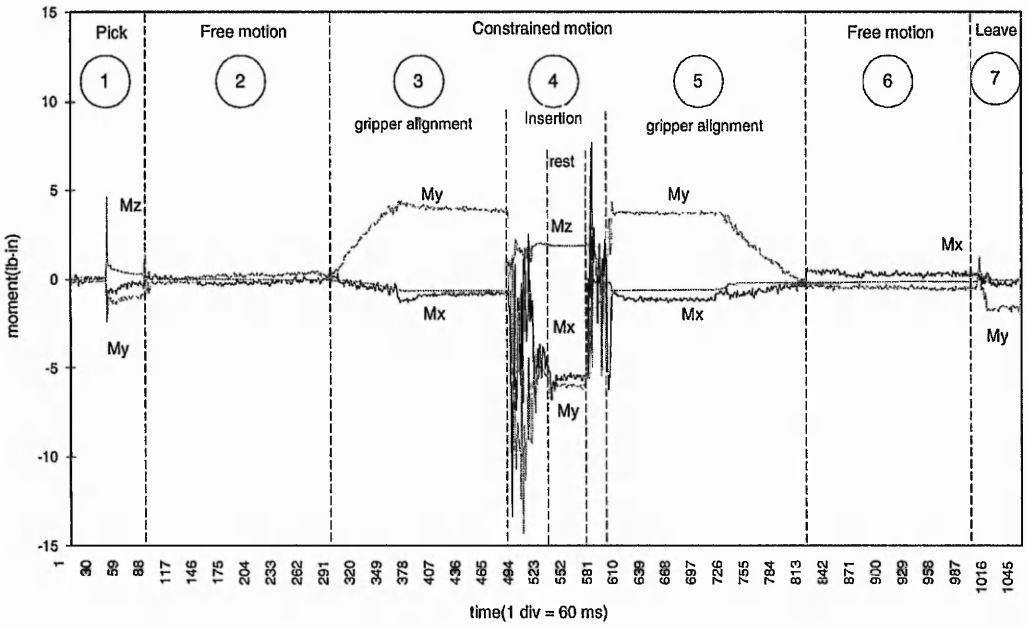


Figure 5.4: Moment signals in the square peg assembly

Since neither the centres of mass nor velocity vectors were aligned to the line defined by the common normal of the contacting surfaces in stage 4, an *oblique eccentric* [17] impact force was observed. This was useful in terms of looking at sliding motions and small misalignments which were corrected by the hole's chamfer while inserting the peg. This fourth stage is the most critical, which started with a high spike due to the arm inertia. During this stage, oscillations appeared due to either a high electrical noise in the current supplied to the motor or an underdamped response on the strain gauges in the sensor. The

end-condition was set when the peg was fully inserted without contacting the bottom of the hole, thereafter a time delay of about 2 seconds was used to observe the response. During this *rest* period, moment signals were more stable and lasted longer especially in M_z. Once again, while extracting the peg, a high spike and oscillations appeared until the peg was withdrawn from the hole. From there and up to leaving the peg in its base holder, the patterns were very similar to those before insertion.

5.2 Analysis of Contact Force Information

Similar experiments to those given above were carried out by placing the female chamfered blocks at different angles on the master block. The block for the circular peg was placed at 90° and for the radiused-square peg at 180°. The plots for these insertions are shown in Figure 5.5 and 5.6.

For the radiused-square peg insertion (Figure 5.6) only the withdrawal of the peg from the base holder to the assembly is shown. It can be observed that since the arm was moved only vertically and parallel to the X-Y plane there was no change in force during gripper alignment. On the other hand, with the cylindrical peg inserted at 90°, forces during gripper alignment were higher (Figure 5.5), due to the increment of the gravity component vector in Z axis, g_y.

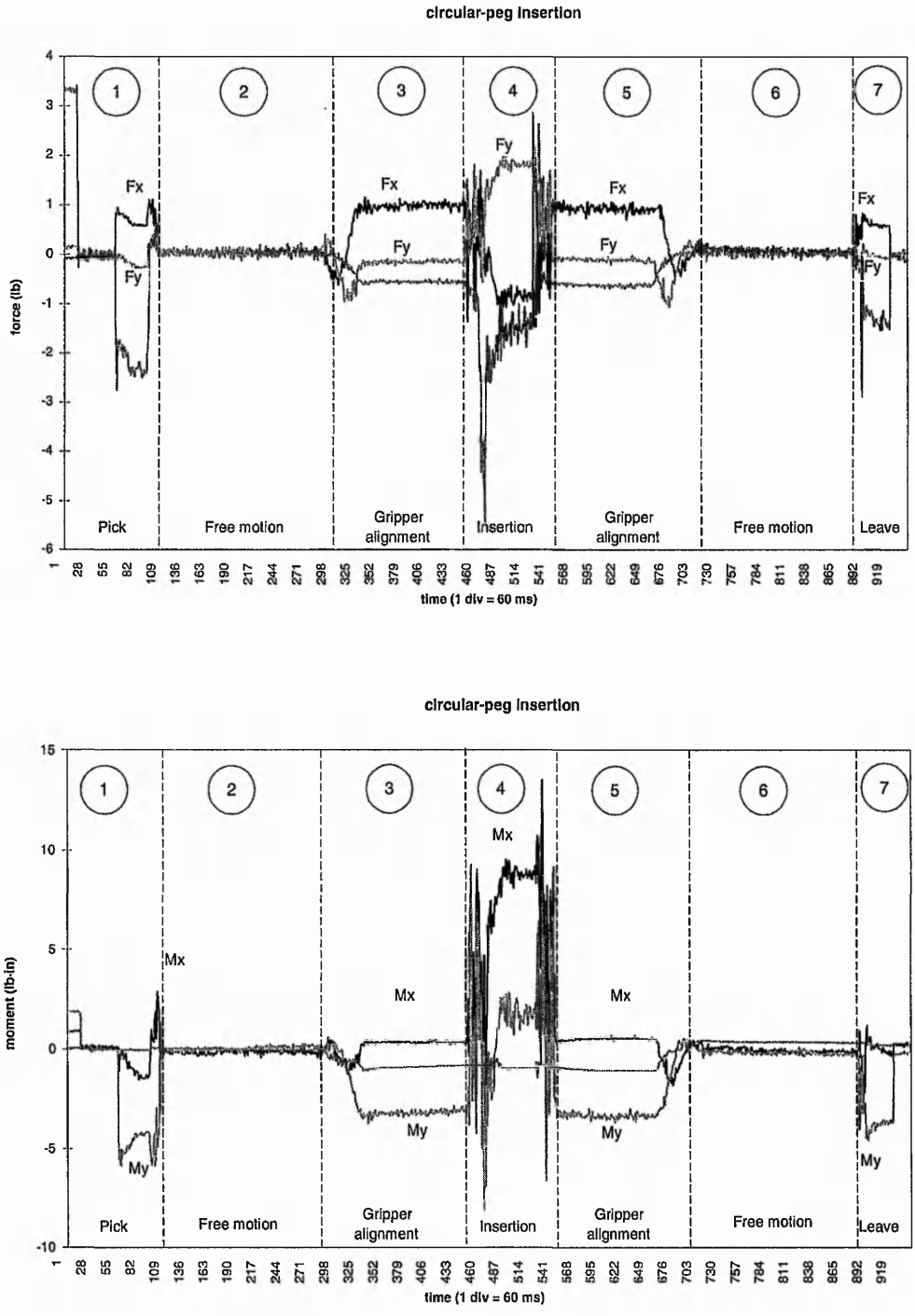


Figure 5.5: Circular-peg assembly

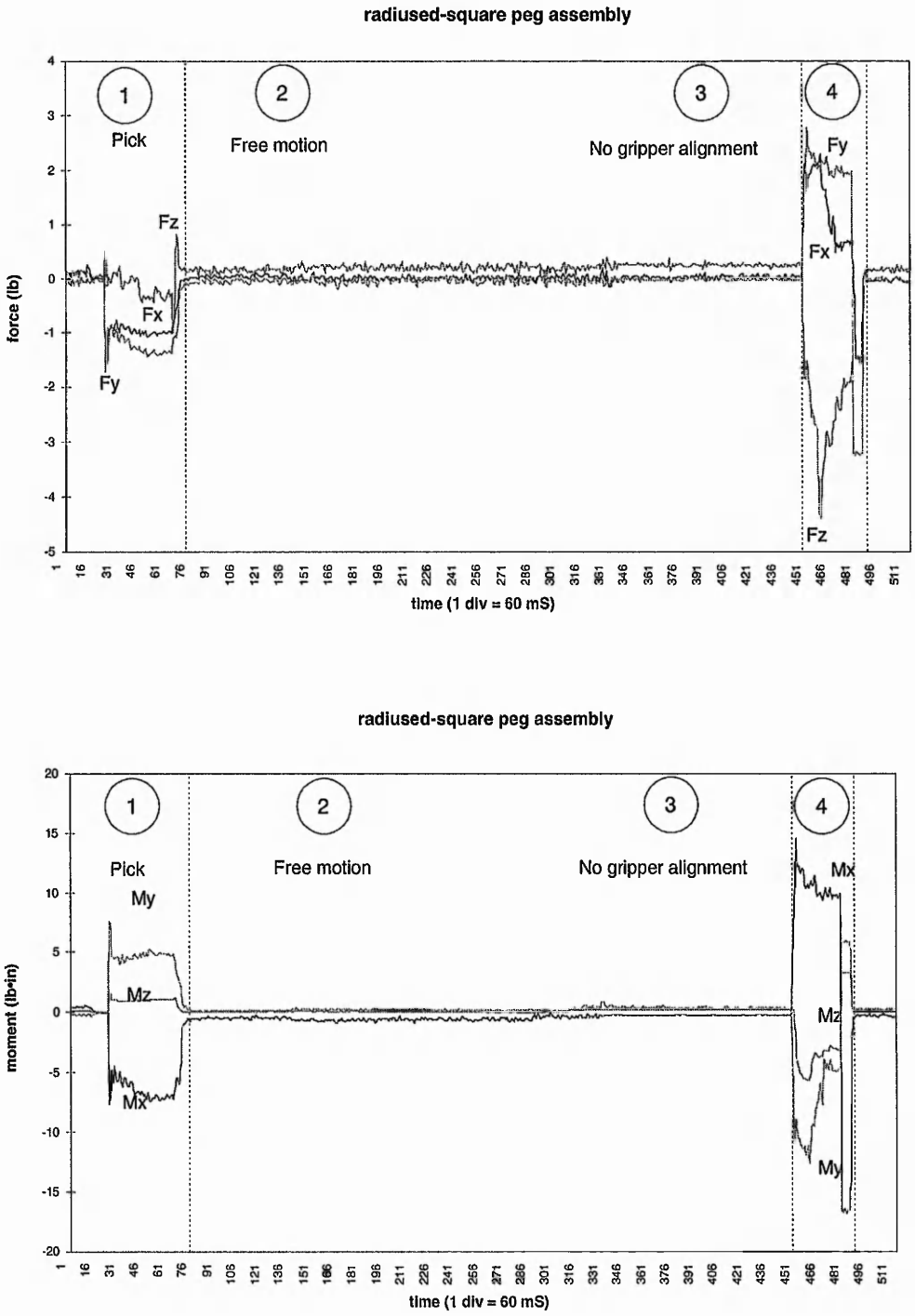


Figure 5.6: Radiused-square peg assembly

In terms of arm motions during gripper alignment the orientation of the female block on the master block is of minor importance. Even if the movement is not parallel to the X-Y plane, it is always possible to adjust the sensor output to a zero value by software and discard any force component not associated to the assembly. In other words, deducting the weight of the peg, gripper and sensor itself from the output reading. However, the orientation information would be lost, which can help to recognise the assembly stage, if needed.

Looking at finer details during assembly (insertion and extraction), the orthogonal relationship that exist between forces acting in X and Y axes in tool coordinates (frame origin at the tool plate) can be observed. Three representative graphs are shown in Figure 5.7.

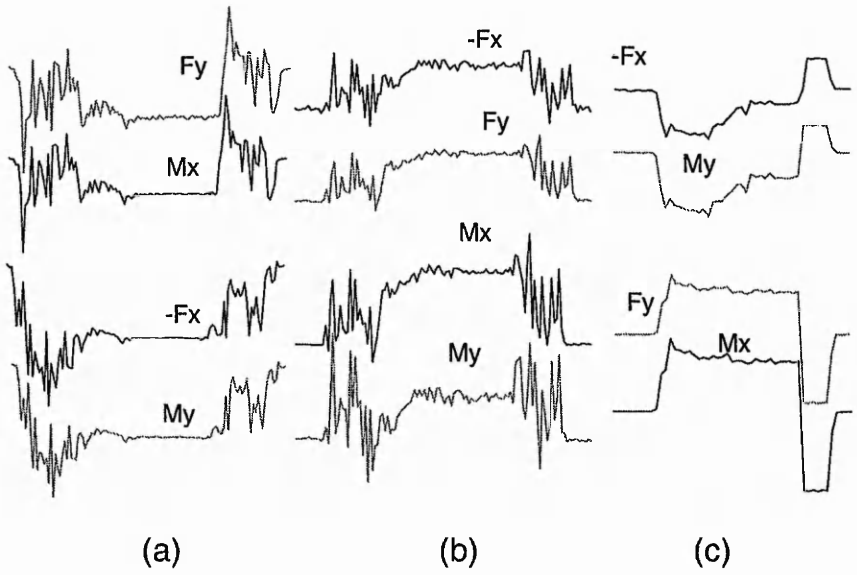


Figure 5.7: (a)square, (b)circular, (c)radiused-square

It should be noted that magnitudes in the force and moment signals were scaled to interpret properly their similarities. Signals corresponding to the Z axis (insertion direction) are not shown since they varied completely between each sample. The relationship between the signals in Figure 5.7 can be mathematically represented by the following expressions:

Square peg assembly,

$$(M_x \propto F_y) \sim (M_y \propto -F_x) \tag{5.1}$$

Circular peg assembly,

$$(M_x \propto M_y) \sim (F_y \propto -F_x) \tag{5.2}$$

and Radiused-square peg assembly,

$$(M_x \propto F_y) \neq (M_y \propto -F_x) \tag{5.3}$$

It can also be observed that signals in Figure 5.7a and 5.7b corresponding to pegs with symmetric cross-section (square and circular) followed a similar pattern, while for the non-symmetric radiused-square peg insertion (Figure 5.7c), the patterns were totally different. This occurred because the distribution of the contact forces on the pegs were also dissimilar due to the non-symetric shape of the peg. Therefore, it can be said that the type of pattern depends on the *magnitude* of the force applied during assembly and the *shape* of the peg. However, it is important to mention that this assumption is only valid when both mating pairs are aligned, i.e. the peg has been aligned perpendicular to the female component. The magnitude and type of proportionality given by expressions 5.1 to 5.3 changes according to the *offset* location of the peg within the X-Y plane quadrants. This is exemplified by the cylindrical peg insertion in Figure 5.8.

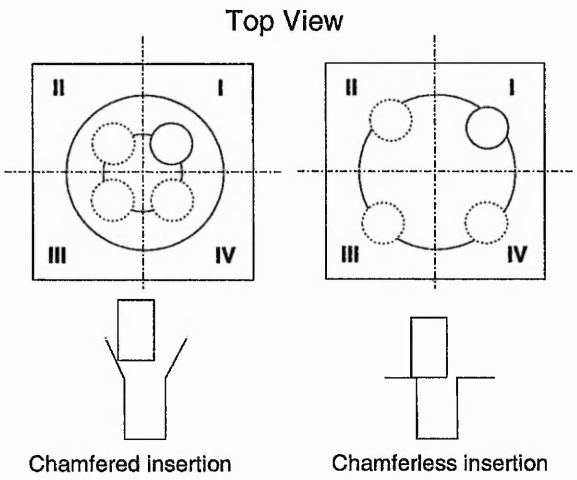


Figure 5.8: Symmetry and contact forces

The peg and the female block are shown in plan view. The chamfer and female component have been enlarged to clarify the placement of the peg within the quadrants. The level of cross-correlation between force and moment patterns

depends on the placement of the peg within these quadrants. An important observation is that the correlation was related to the cross-section symmetry of the mating pairs. Circular and square pegs showed higher correlation (infinite symmetry) compared to the radiused-square peg.

Another important consideration from the analysis above is the fact that the NNC must be robust and also be able to cope with the inertia spikes produced during contacts between the peg and the female component (see Figures 5.3, 5.4, 5.6 and 5.5). The inertia spikes may either cause a false triggering due to the force threshold limits or produce a malfunction on the F/T sensor due to its transient response.

5.3 Requirements of the NNC

From section 5.2, the complexity of the system becomes evident especially if the infinite number of patterns that can be generated through the assembly process are considered. To deal with this complexity the NNC has to classify and recognise these patterns first and then allocate appropriate actions for every contact force pattern.

An approach to solve this situation is to take into account all parameters involved and associate them with the *"experience"* of the system. Such experience may be acquired, for example by perturbing the system with random initial movements at the end-effector and determining the appropriate action in each of the major axes. This provides certain expectancy of success which may be combined with part geometry information boosting the decision. However, a method based on measurable parameters rather than a "trial and error" approach is preferable.

The author analysed different contact force patterns during experiments described in section 5.1 which led to specify the requirements for the NNC. Basically, it should consider two stages:

1. Adaptation: The NNC should be capable of recognition and classification of the different contact states as they occur during assembly. In other words, the controller should have an autonomous operation with the capability to recognise

new and novel contact states from those stored previously.

2. Decision: In the same manner, it also has to allocate those "recognised" contact states with the corresponding actions 'on the fly'. That is, using on-line learning. The NNC should be able, based on past experiences, to predict or decide which action will correspond to a completely new contact state.

To improve the performance of the NNC and overcome some of the limitations of current ANN approaches, it is desirable to use an ANN able to learn in real-time and to adapt to changing environments (as occurring during insertion operations). In other words, a network capable of fast learning and adaptation. Prominent research in this area has been reviewed in Chapter 2.

Different connectionist networks have been tested in real robotic operations. The reinforcement algorithm implemented by V. Gullapalli demonstrated to be able to learn circular and square peg insertions. The network showed a good performance after 150 trials with insertion time lower than 100 time steps [57]. Although the learning capability showed during experiments improved over time the network was unable to generalise over different geometries. Insertion was reported with both, circular and square geometries, however, when inserting the square peg, its rotation around the vertical axis was restricted otherwise the insertion would not have been possible. M. Howarth [3], followed a similar approach using back-propagation in combination with reinforcement learning. During simulation it was demonstrated that 300 learning cycles were needed to achieve a minimum error level with his best network topology during circular peg insertions. A cycle meaning an actual motion that diminished the forces acting on the peg. For the square peg, the number of cycles increased dramatically to 3750 cycles. These figures are high and this is important, especially when fast learning is desired during assembly. On the other hand, E. Cervera using SOM networks and a Zebra robot (same type used by Gullapalli), developed similar insertions. Cervera in comparison with Gullapalli, improved the autonomy of the system by eliminating the need to use the knowledge of the part location and using only relative motions. However, the trade-off with this approach was the increased number of trials to achieve the insertion [26], the best insertions were achieved after 1000

trials. During Cervera's experiments the network considered 75 contact states and only 8 motion directions were allowed. For square peg insertions, 4000 trials were needed to reach 66% success of insertion and that did not improved any further. According to Cervera's statement: "We suspect that the architecture is suitable, but the system lacks the necessary information for solving the task". The situation clearly recognises the necessity to embed new information in the control system as it is needed, which is likely to be obtained with an architecture based on ART. Additionally, independent studies report ART, to be to 533% faster than SOM when compared with the same database [60].

5.3.1 Structure of the NNC — adaptation and decision

The proposed structure for the NNC is illustrated in Figure 5.9.

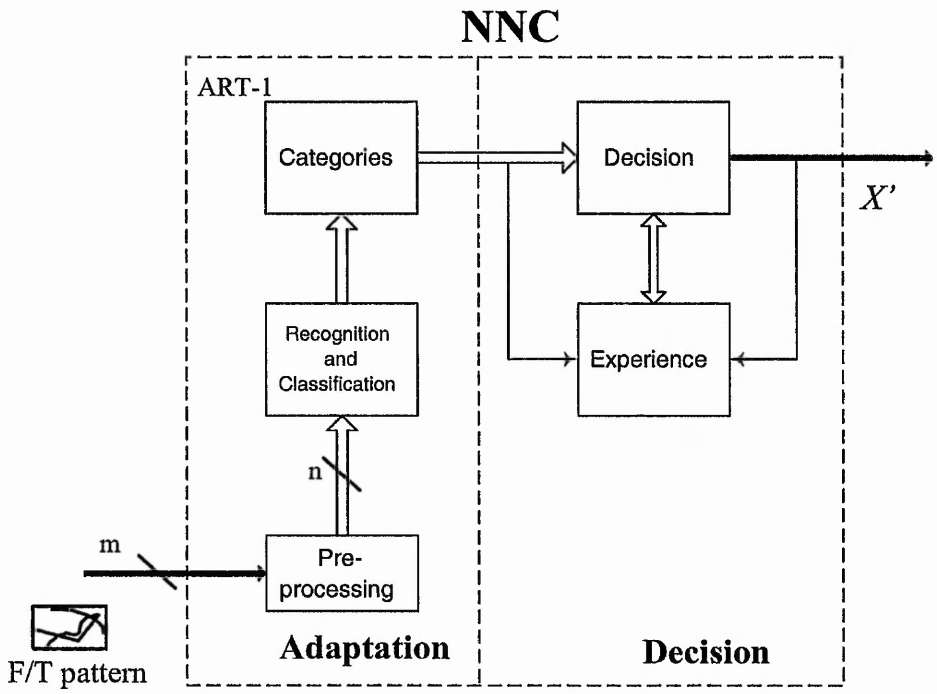


Figure 5.9: Structure of the NNC

The structure is divided into two stages. The first stage (*Adaptation*), has an initial section where the m-dimensional F/T value vector is pre-processed to provide an adequate representation of the contact forces. This pre-processing section may include some function in order to normalise the vector components

or to eliminate redundant contact forces, hence producing a n-dimension input vector.

The *recognition/classification* and *category* sections are in charge of recognising contact states and classifying them. In case new force patterns arrive, the adaptation stage will allocate space for these new patterns into the current categories. The system still preserves the previous categories, hence providing the required flexibility for the Knowledge Base.

During the *decision* stage, the corresponding category contains a group of contact state prototypes with similar pattern features. However, due to the geometry of the mating pairs there might not be an unique motion direction, hence this information has to be associated through the *experience* section in order to make a decision about the optimum motion.

Finally, the output X' is an incremental motion relative to the previous position of the arm, that produces a reduction in contact forces as well as a motion towards the end condition.

5.4 Summary and Conclusions

Force and moment patterns for peg-in-hole operation were analysed using different components and assembly orientations. A direct relationship between the applied force and the shape of the peg was found. Also, the cross-sectional symmetry of the peg and its offset location at the start of the insertion with respect to the hole's centre contributed to the resulting type of pattern.

The information extracted from these patterns was important to determine the design parameters for the NNC. The NNC has to recognise a wide range of patterns and to accommodate this information into an 'experience' stage, where information can be retrieved. The NNC must be adaptable and able to use effectively its experience in future insertions.

Other important parameter is the learning time since the operations and learning should be made on-line. One of the requirements is also that the NNC should be able to insert non-symmetrical pegs. A reduced learning time can be achieved by

selecting a neural network such as ART with fast learning. Some of the aspects for choosing this ANN are described in Chapter 6.

Chapter 6

Adaptive Resonance Theory (ART)

In this Chapter, the Adaptive Resonance Theory (ART) is formally introduced. The mechanics of the ART models upon which the NNC was built namely, ART-1, FuzzyART and Fuzzy ARTMAP are included.

6.1 Introduction

The Adaptive Resonance Theory (ART) is a well established neural network theory developed by Gail Carpenter and Stephen Grossberg at the Center for Adaptive Systems at Boston University. The theory was first introduced by Grossberg as a theory of human cognitive processing [61], although the core principles regarding how Short Term Memory (STM) and Long Term Memory (LTM) interact during network processes of activation, associative learning and recall were earlier published in the scientific literature back in the 60's [62, 63, 64]. The theory has evolved in a series of real-time architectures for unsupervised learning, the ART-1 algorithm for binary input patterns [32], ART 2-A for analogue and binary input patterns [65], and ART 3 based on chemical transmitters. Supervised learning is also possible through ARTMAP [33] that uses two ART-1 modules that can be trained to learn the correspondence between input patterns and desired output classes.

The ART-1 unsupervised network was later enhanced by incorporating Fuzzy set theory concepts into the learning and recognition mechanisms. This resulted in a generalised unsupervised network called Fuzzy ART. Similarly to the ARTMAP case, is that the connectivity of two Fuzzy ART networks resulted in the Fuzzy ARTMAP network [34]. Continual development of ART systems have produced other models for supervised learning such as Gaussian ARTMAP [66], ART-EMAP [67], and many other variants adapted for specific applications [68]. In terms of hardware implementation, perhaps the most prominent work in this area has been made by T. Serrano and B. Linares at the National Microelectronics Centre, Spain. They have developed a VLSI design of an ART-1 chip. The chip was fabricated using a 1.6μ CMOS process and contains an ART-1 system. Results have encouraged further investigations at the Johns Hopkins University, where a chip design that can be configured as ART1, Fuzzy ART, ARTMAP and Fuzzy ARTMAP is being investigated [69].

6.1.1 Learning and forgetting

— The stability-plasticity dilemma

Learning in natural cognitive systems, including our own, follows a sequential process as it is demonstrated in our daily life. Events are learnt incrementally, for instance, during childhood when we start making new friends, we also learn more faces and this process continues through life. This learning is also stable because the learning of new faces does not disrupt our previous knowledge. – We do not come home after having learn new faces at school and forget our parent's face. These premises are the core for the development of Connectionist Models (CMs) of the human brain and are supported by Psychology, Biology and Computer Sciences. Psychology studies suggest the sequential learning of events at different stages or "storage levels" termed as Sensory Memory (SM), Short Term Memory (STM) and Long Term Memory (LTM). [70][1] The SM refers to the initial, momentary storage of information, lasting only an instant and it is recorded

[1] Although there are some other memory divisions that have been identified during studies such as Phonological and Visuospatial Short-Term Memory [71].

by the person's sensory system as a raw nonmeaninful stimulus. Examples of these stimuli are the momentary flash of lightning or the sting of a pinprick. This type of information is like a snapshot that is replaced by a new one. Unless this information is transferred to other storage levels of memory, it is lost. The STM —also referred to as the working memory— is the storage where psychologists believe the information is transferred immediately after a person makes sense of the information. Returning to our example of memorising familiar faces, an example of STM information can be placed in our daily journey to work. We see many faces while travelling on the morning bus and that information is retained may be seconds or minutes, but, as soon as we get off the bus this information may have already been erased. But, how short is that STM?. Psychology studies suggest that information is retained in the STM for about 15 to 25 seconds before passed to a permanent storage, known as the LTM [70]. The time that information can be retained in the LTM can be of a life-long duration, e.g. our parents' faces.

The connectionist model of memory within ART networks has been inspired by the above thoughts. Memory is sequential and allows variability —*plastic*, according to ART vocabulary—. That is, knowledge can always be accommodated within the LTM storage. For instance, as we grow older we can recall best friends' faces during primary school, high school, university, work and so on. Forgetting if present is gradual rather than a plummeted process, therefore *stable* —We may gradually forget classmates' faces after a long time, but not immediately after the school term has ended—. This forgetting problem is readily recognised within the scientific community and referred to as "catastrophic forgetting." Connectionist models such as the standard Backpropagation can suffer from this catastrophic forgetting, as studied by McCloskey and Cohen [72]. They showed in a set of experiments the learning of "one's addition facts" (i.e. the 17 sums 1+1 through 9+1 and 1+2 through 1+9). Then the network learned the 17 "two's addition facts" (i.e. 2+1 through 2+9 and 1+2 through 9+2). Recall fall abruptly as soon the network began the learning of the two's addition facts. Within 1-5 two's learning trials, the number of the correct responses on the one's fact had dropped from 100% to 20%. In five more learning trials, the one's learning was only 1%

and by trial 15, no correct answers from the previous one's learning could be produced. Architectures that rely on the separability of the previously learn representation from those that are currently being learned have been suggested to solve this paradigm [73, 74].

Grossberg resumed the above situations in what he called the *stability-plasticity dilemma* suggesting that connectionist models should be able to adaptively switch between its plastic and stable modes. That is, a system should exhibit plasticity to accommodate new information regarding unfamiliar events. But also, it should remain in a stable condition if familiar or irrelevant information is being presented. He identified the problem as due to basic properties of associative learning and lateral inhibition. An analysis of this instability, together with data of categorisation, conditioning, and attention led to the introduction of the ART model that stabilises the memory of self-organising feature maps in response to an arbitrary stream of input patterns [61].

6.2 ART Mechanism

The ART-1 architecture consists of two parts: attentional subsystem and orienting subsystem as illustrated in Figure 6.1:

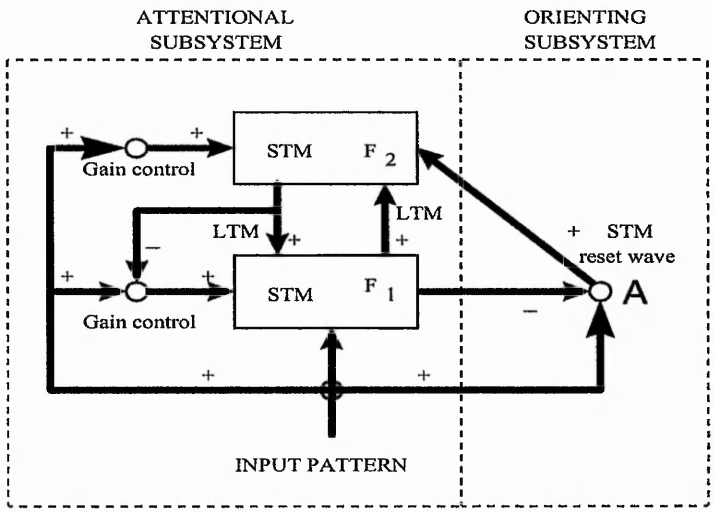


Figure 6.1: ART architecture

The attentional subsystem is made up of two STM layers of nodes (F_1 and F_2).

In an ART network, information reverberates back and forth between both STM layers. If a stable resonance takes place, then learning or adaptation can occur. On the other hand the orienting subsystem is in charge of resetting the attentional subsystem when an unfamiliar event occurs so that a new node can be tested.

A *resonant state* can be attained in one of two ways. If the network has learned previously to recognise an input vector, then a resonant state will be achieved quickly when that input vector is presented. During resonance, the adaptation process will reinforce the memory of the stored pattern. If the input vector is not immediately recognised, the network will rapidly search through its stored patterns looking for a match. If no match is found, the network will enter a resonant state whereupon the new pattern will be stored for the first time. Thus, the network responds quickly to previously learned data, yet remains able to learn when novel data is presented, hence solving the *stability-plasticity* dilemma. The activity of a node in the F_1 or F_2 layer is the activity of the STM whereas the adaptive weights are LTM traces or simply *weights* according to the ANN's jargon. Gain controls handle the discrete presentation of the input signals.

The STM layers and the LTM adaptive weights were first introduced in the literature as a set of nonlinear equations. The STM equations described the instantaneous activation of the neurons as a function of the externally applied inputs and the connecting weights. The LTM equation described the evolution of the adaptive connecting weights. If the input patterns are held long enough to allow the system to reach its steady state then both equations resumes to their algebraic form[2]. This form is also employed when ART systems are used in their *fast learning mode.* When ARTMAP, Fuzzy ART, Fuzzy ARTMAP and Gaussian ARTMAP were reported, those algorithms were presented in their fast learning mode[33, 76, 34, 66]. The term of fast learning is given due to the fact that learning can be done in one shot that is in one input pattern presentation.

[2]The reader is referred to [32, 75] for the mathematical foundation of these equations.

6.3 ART Processing

The dynamics of the ART processing are represented in Figure 6.2 as follows:

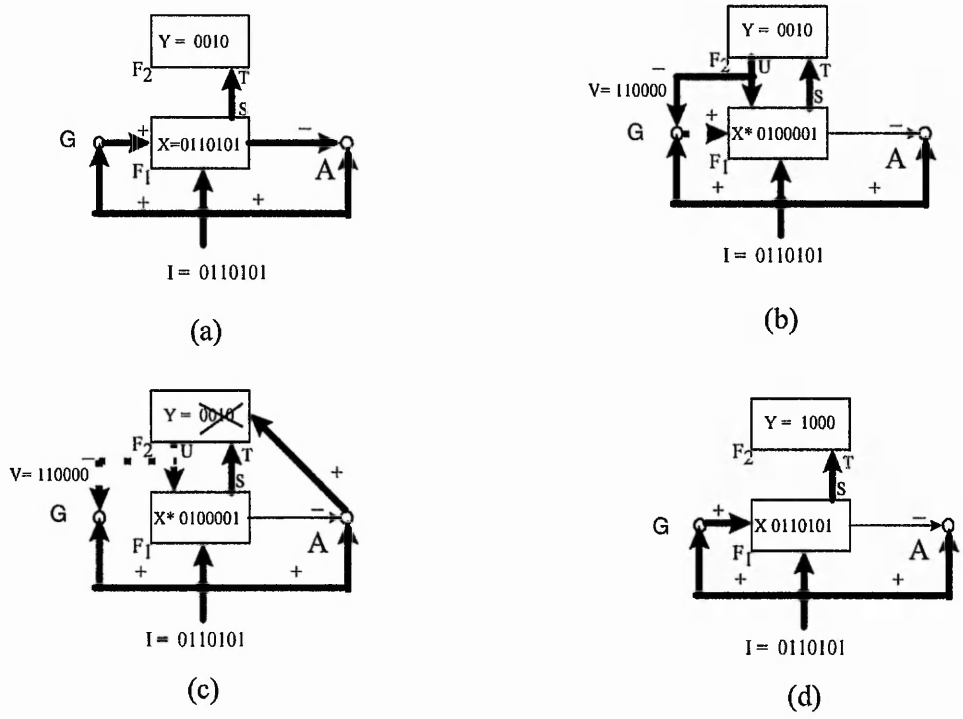


Figure 6.2: ART mechanics

6.3.1 F2 choice — hypothesis

An input vector I registers itself as a pattern X of activity across level F_1 (Figure 6.2(a)). The same input pattern excites the orienting subsystem A, and the gain control G. The pattern X of STM activated across F_1 therefore elicits a pattern S of output signals from F_1. This transmission event multiplies the vector S by a matrix of adaptive weights or LTM traces to generate a net input vector T to level F_2. The transformation from S to T is an *adaptive filter.*

The output pattern S, results in an inhibitory signal that is also sent to A cancelling the excitatory effect of I, so that A is inactive. Note that G also supplies an excitatory signal to each node on F_1(non-specific signal).

The internal competitive dynamics of F_2 vector T results in the F_2 activity vector Y. The competition is sharply tuned to select only the F_2 nodes that receive the

maximum $F_1 \rightarrow F_2$ input, leaving just one positive component of Y. Such a category represents all the F_1 inputs I that send maximum input to the corresponding F_2 node. In other words the selected F_2 node or 'winning' is deemed as making a hypothesis about which category the input I belongs to.

6.3.2 Hypothesis test, resonance or category reset

The pattern of activity Y results in an output pattern U from F_2 (see Figure 6.2(b)). This pattern is sent as an inhibitory signal to the gain control which ceases activity as designed. After output vector U undergoes multiplication by the adaptive weight matrix of the top-down filter, net vector V becomes the input to F_1. Vector V plays the role of a learned top-down expectation.

The network matches the "expected prototype" V of the category against the active input pattern I. Nodes in F_1 that were activated by I are now suppressed if they do not correspond to large LTM traces in the prototype pattern V. Thus F_1 features that are not "expected" by V are suppressed. The resultant matched pattern is X^* which develops on F_1.

Since the new output pattern S^* is different from the original pattern (S), if the mismatch is severe, A can no longer be prevented from releasing a non-specific arousal wave to F_2, which resets the active node at F_2 (Figure 6.2(c)). There is a *vigilance parameter* (ρ) that determines how much mismatch is tolerated.

After the F_2 node is inhibited, its top-down expectation is eliminated and X can be reinstated at F_1 (Figure 6.2(d)), and then the cycle begins again. X then again generates input pattern T to F_2, but a different node is activated. The previously chosen F_2 node remains inhibited until the gain control of the F_2 layer is disengaged by removal of the input pattern.

The attentive matching process combines three different types of inputs at level F_1. Bottom-up inputs, top-down expectations and attentional gain control signals. Attentive matching obeys a 2/3 rule that permits an F_1 node to reach its output threshold only if 2 of 3 input sources that converge on it are high. At this point, the system is said to be in *resonance.*

6.4 Prediction in ART systems

The supervised version of ART is made by the connection of two ART modules and referred to as Predictive ART or more commonly, ARTMAP. This type of architectures can learn to predict a prescribed m-dimensional output vector **b** given a prescribed n-dimensional input vector **a**. The term ARTMAP is used because the transformation from vectors in $\Re^n$ to vectors in $\Re^m$ defines a map. The main elements of an ARTMAP system are shown in Figure 6.3

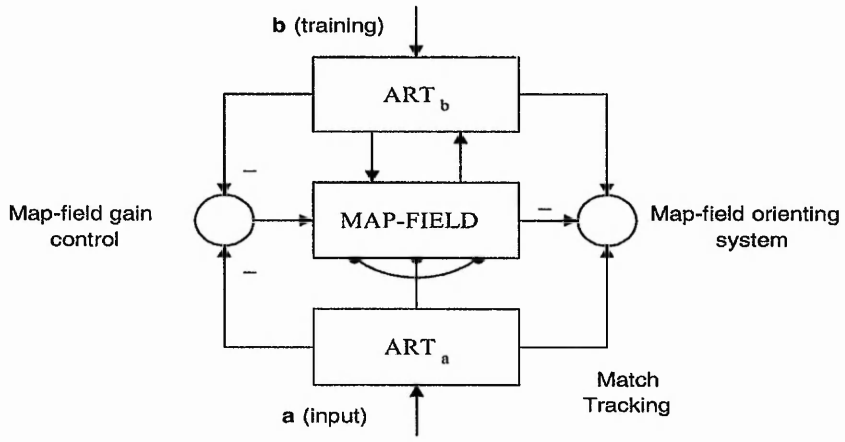


Figure 6.3: ARTMAP system

There are two modules ART_a and ART_b and an inter-ART module "map field" that controls the learning of an associative map from ART_a recognition categories to ART_b recognition categories. The map field module also controls the match tracking of ART_a vigilance parameter. A mismatch between Map field and ART_a category activated by input **a** and ART_b category activated by input **b** increases ART_a vigilance by the minimum amount needed for the system to search for, and if necessary, learn a new ART_a whose prediction matches the ART_b category. The search initiated by the inter-ART reset can shift attention to a novel cluster of features that can be incorporated through learning into a new ART_a recognition category, which can then be linked to a new ART prediction via associative learning at the Map field. The Gain control is in charge of the discrete inputs **a** and **b**. Inhibitory paths are denoted by a minus sign, other paths are excitatory. In the case of Fuzzy ART and Fuzzy ARTMAP, both can handle analogue or binary data. The algorithm uses a preprocessing step, called complement coding

which is designed to avoid category proliferation. The internal operations are similar to those in ARTMAP for the category choice, matching criterion and the update of weights. The difference between them is the interchange of the logical AND intersection ($\cap$) by the AND operator ($\vee$) of fuzzy logic (see Appendix B for details).

6.5 Implementation in the Robotic System

The implementation of the ART systems into the robotic assembly was accomplished in two stages. The ART-1 was implemented first followed by the Fuzzy ARTMAP. Both algorithms were implemented using Visual C++ 5.0. ART-1 was implemented and tested using data retrieved from experimental insertions. The objective was to test off-line the learning speed and clustering capability of the network. The procedure is detailed in Chapter 7. Having verified the suitability of the algorithm for real-time operation, then the Fuzzy ARTMAP algorithm was implemented into the NNC in combination with a dynamic knowledge base, whose knowledge is regulated by the assembly task. Simulations to verify the prediction capability of the algorithm were carried out first using data retrieved from different contact states. After simulations, the learning was tested on-line using the manipulator. Results on these tests as well as the network parameters used during experiments are given in Chapter 9, where the NNC performance is assessed. The corresponding algorithms are given in Appendix B for further reference and completeness of this Chapter.

Chapter 7

Towards the Implementation of the NNC: Investigations into ART-1

Broadly speaking, the capability of developing insertions in real-time is bounded by the mechanical response of the manipulator and the NNC processing time. Issues regarding the robotic system design were addressed in Chapter 3 and 4. In this Chapter, factors that affect the data availability to the NNC are described followed by result simulations using the ART-1 algorithm. These initial results investigate the classification and learning speed of the network using data retrieved from peg-in-hole insertions.

7.1 Timing Considerations

How quickly the robot can respond to changing patterns will depend on many factors. However, with a good approximation it can be said that it primarily depends upon the neural network processing time and the mechanical response of the arm during fine motions. The ability of the NNC to effectively respond to the changing force conditions during manipulation is bounded by the following factors:

Time for network processing (learning and testing). This is a paramount requirement for the NNC. The learning should be achieved as fast as possible and with a small number of learning epochs.

Servoing the arm motors (ALTER mode). As explained in Chapter 3 during normal ALTER mode, the robot controller expects information about incremental motions every 28 ms otherwise a time-out occurs[1]. By using the Host-slave architecture (also described in Chapter 3) this no longer applies since the ALTER mode can be switched ON/OFF at any time, which gives more time for processing if required. However, the 28 ms still needs to be considered as the minimum allowed time between arm motions.

F/T Data Acquisition (sampling rate). The JR3 sensor is able to provide decoupled F/T data at 8 kHz per channel. There are 16 channels to be transmitted, however only six corresponding to the F/T values are utilised. At 8 kHz in the worst case the whole F/T decoupled data is available every 2 ms. When using the digital filters, additional considerations apply as indicated below.

Signal delay during filtering. During operations, noise on the F/T readings were present. In addition to this intrinsic noise, when the arm power was ON, the end-effector had a chattering effect which amplified the noise. This effect was even worse when the arm was in a stretched pose (joint 3 at an obtuse angle). From this behaviour the importance of filtering out high frequencies using the built-in low-pass filters of the JR3 sensor was clear. The cut-off frequencies (f_{off}) available for the filters at 8 kHz are 500 Hz, 125 Hz, 31.25 Hz, 7.81 Hz, 1.95 Hz, and 0.48 Hz.

The use of a low-pass filter also involves a signal delay which has to be taken into account for processing. This delay is approximately $1/f_{off}$ [77]. Using a very low f_{off}, say $f_{off} = 0.488$ Hz would represent a delay of $\delta_{f_{off}} = 2.04$ s which is considerable if a fast response is desired. Through observation and considering the signal delay, it was found that a value of $f_{off} = 31.5$ Hz ($\delta_{f_{off}} = 32$ ms) was

[1] In reality, the exact time depends on how long does the robot controller take for sending the information to the host computer (up to 27.1 ms depending on the amount of data). Hence, the 28 ms have been computed in the worst case and the communication handshaking respects this timing.

the best frequency choice with a smooth signal response.

7.2 Data Availability for the NNC

Ideally, force data should be available to the NNC as soon as it requires it. However, there are elements such as those described above and the transient response of the arm that have to be considered to provide reliable data to the NNC. To understand better these elements let us refer to the plot in Figure 7.1.

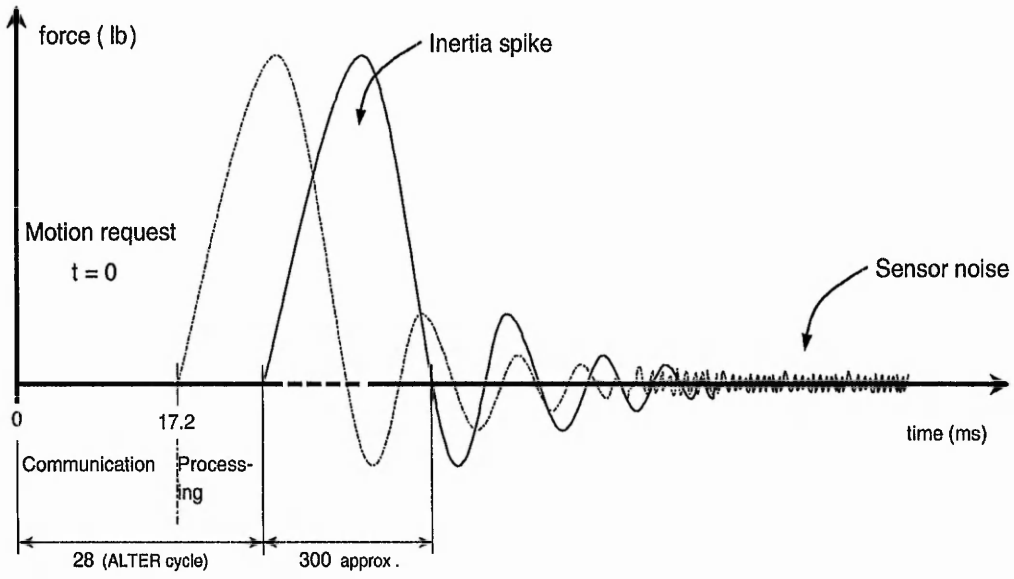


Figure 7.1: Time diagram

The diagram typically represents a complete incremental motion cycle. It starts at time $t = 0$ when the incremental motion is requested via ALTER mode. The communication with the robot controller is completed approximately at $t = 17.2\,\text{ms}$. Immediately after the robot controller processes the kinematics for the arm. The duration for this stage depends on the requested motion and the amount of data provided to VAL. The arm will actually start moving when the data processing is finished. Therefore, the motion can start at any time during the interval $17.2\,ms < t < 28\,ms$. Both extreme posibilities are represented by the dotted and continuous curve in the plot. During constrained motions, including direct impact, there were observed spikes that lasted in the worst case about 0.3 s. Immediately after the spike, an underdamped response was observed. The reason for

this behaviour is the combination of inertia, friction and the combined stiffness of the end-effector, gripper and sensor.

From the discussed so far, it is clear that the NNC has to bypass the sensor transient state in order to acquire reliable F/T information in steady state. In addition to the transient stage it is also important to consider the ALTER cycle, the delay produced by signal filtering and the worst case for the F/T sensor in providing the data. Roughly, these times result in 300, 28, 32 and 2 ms respectively which indicates that the NNC should consider a delay in requesting information, during constrained motion, of at least 362 ms. After this time, the NNC can either start learning the new pattern or predict the action from the mapping previously stored in memory.

How fast the insertion can be made is clearly restricted by this data request time and the NNC processing time. The data request time must be respected and cannot be changed since it depends on the physical system and operations. This fact leaves the NNC processing as the *only* modifiable factor to speed up assembly operations. Having defined the time needed to acquire the F/T information, the next step is to assess the learning time for the ART network, which is described in the next sections.

7.3 Temporal and Spatial Patterns Recognition

The neural controller explained in section 5.3.1 consists of two sections, adaptation and decision. The first section includes a pre-processing stage. For this stage *temporal* patterns, like those presented in Figures 5.3, 5.4 and 5.7, must be first converted to *static* or *spatial* patterns to be processed by the network. Although the patterns occurring during operations are continuous, they need to be input to the network in a discret manner to allow time for processing as highlighted by Nigrin [68].

Temporal patterns extracted from experimental insertion data were decomposed into spatial patterns by taking the final value of the corresponding signal (force or moment) after each incremental motion. This was important to assess the

recognition section of the NNC. Reading the values in this manner eliminated any transient response in the input data due to arm inertia or noise. An example of moment values during insertion is shown in Figure 7.2.

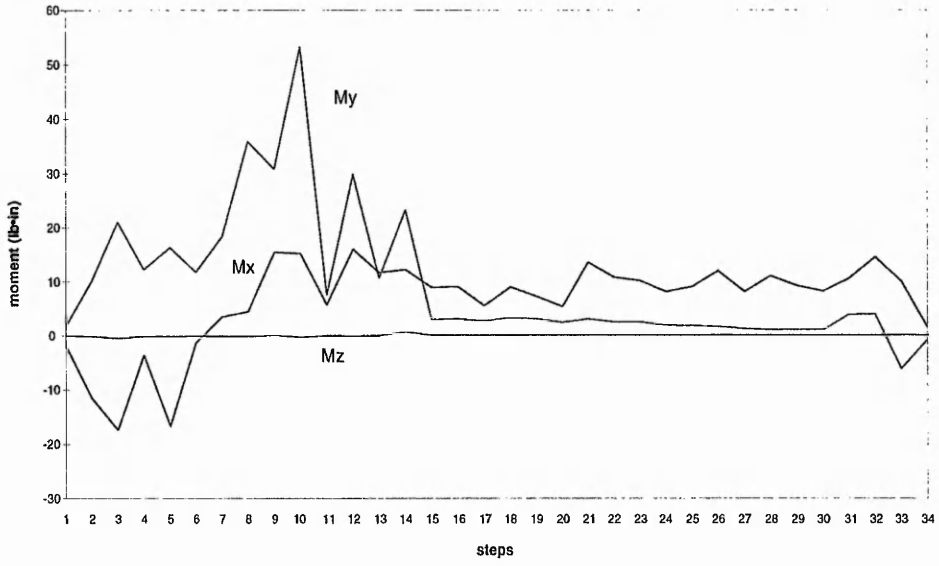


Figure 7.2: Temporal and spatial patterns

This graph corresponded to moment values during chamfered insertion of the circular peg at an angle of 180° and with a clearance of 0.1 mm. It is important to mention that although the graph is shown for moment values only, the analysis for the force signals will be similar. The insertion was made moving the arm step by step using the teach pendant. The insertion lasted 34 time steps (i.e., 34 spatial patterns) before reaching the depth end-condition. After 20 alignment steps, the peg had already passed the chamfer and accommodated well into the hole, so the alignment ceased and only motions in the vertical direction were needed to reach the end-condition. For clustering and recognition purposes only these 20 alignment motions were considered in the analysis. The patterns were presented to the network one after the other.

It can be observed that there are positive and negative *slope* values for each spatial pattern and the encoding of all patterns would be very large. A number of simplifications were made to facilitate the encoding of this information as

follows:

1. Type of slope. Three slopes were considered: slope positive (/), negative (\) or flat (—).

2. Slope location. That is, if the slope was placed above or below zero level.

Note: A slope was considered flat if its value was lower than five degrees.

Ambiguous situations in terms of slope location were solved by considering the slope as being placed in both halves of the graph. This situation occurred for instance with Mz, whose value remained approximately zero during the whole insertion (see Figure 7.2).

Following the above simplifications, the binary encoding for the graph in Figure 7.2 is given in Table 7.1.

STEP	MOMENT VALUE > 0									MOMENT VALUE < 0								
	Xp	Xn	X	Yp	Yn	Y	Zp	Zn	Z	Xp	Xn	X	Yp	Yn	Y	Zp	Zn	Z
1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1
3	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1
4	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1
5	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1
6	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
7	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
8	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1
9	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
10	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1
11	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
12	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1
13	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
14	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1
15	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1
16	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1
17	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
18	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1
19	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1
20	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1

Table 7.1: Binary encoding

where,

X_p, Y_p, Z_p represent the positive slope values in their respective axis.

X_n, Y_n, Z_n are the negative slope values.

X, Y, Z represent flat slopes.

The first three set of values for X, Y and Z corresponded to signals above zero level, whereas the last ones corresponded to signals below zero.

7.3.1 Pattern recognition and classification

The ART-1 algorithm was implemented in a 'C' program for fast learning [32]. The above binary pattern was presented twice to the network and the output results are given in Figure 7.3. The rows represent the patterns presented to the network. The set of 20 patterns were presented twice, i.e. two epochs. Groups or category nodes are formed during learning. For instance, when the first pattern is presented there are no groups then the pattern has to create its own group. Any subsequent pattern if similar to the previously recognised patterns (at this time only pattern 1), will be allocated in the same group then at this point the system is said to be in resonance. When the second pattern arrives, this is classified into group 1 due to its similarities and the network's vigilance.

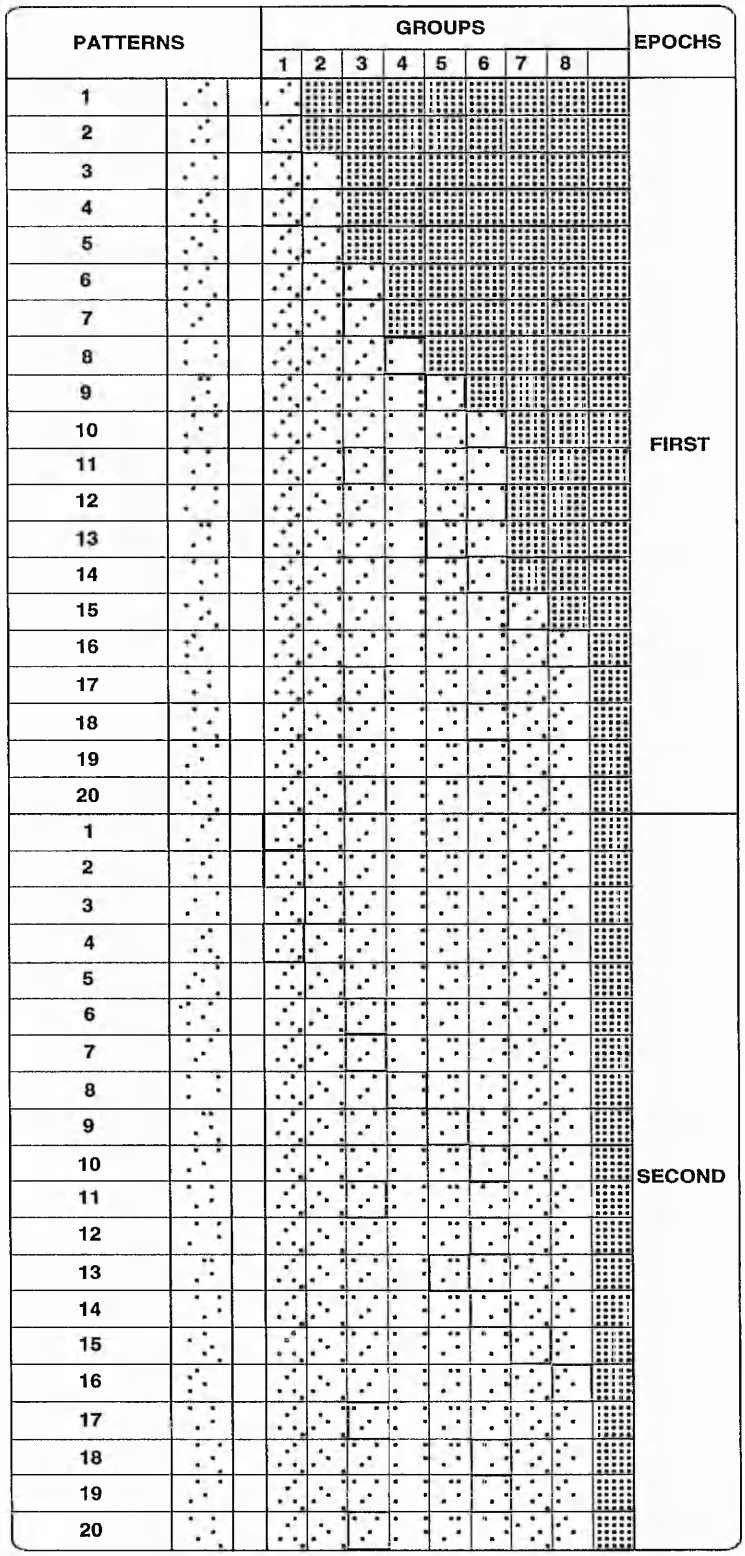


Figure 7.3: Clustered patterns

The mechanics of recognition and classification for the 20 patterns is as follows:

Pattern 1: This new pattern is classified into group 1.

Pattern 2: It is recognised by previous category node (1); it resonates.

Pattern 3: This new pattern is classified into group 2.

Pattern 4: It is recognised by previous category node (1); it resonates.

Pattern 5: It is recognised by previous category node (2); it resonates.

Pattern 6: This new pattern is classified into group 3.

Pattern 7: It is recognised by previous category node (3); it resonates.

Pattern 8: This new pattern is classified into group 4.

Pattern 9: This new pattern is classified into group 5.

Pattern 10: This new pattern is classified into group 6.

Pattern 11: It is recognised by previous category node (3); it resonates.

Pattern 12: It is recognised by previous category node (6); it resonates.

Pattern 13: It is recognised by previous category node (5); it resonates.

Pattern 14: It is recognised by previous category node (6); it resonates.

Pattern 15: This new pattern is classified into group 7.

Pattern 16: This new pattern is classified into group 8.

Pattern 17: It is recognised by previous category node (3); it resonates.

Pattern 18: It is recognised by previous category node (6); it resonates.

Pattern 19: It is recognised by previous category node (6); it resonates.

Pattern 20: It is recognised by previous category node (3); it resonates.

It can be seen from the clustered patterns that the network learnt these patterns at the second time they were presented to the network (direct access to the clustered groups). The vigilance parameter ρ [2] was 0.7 and with this level of vigilance the contact states were classified into eight groups as shown in the Table 7.2.

Different values of ρ were used. It was noted that for $0.5 < \rho < 1.0$ the network behaved in a consistent manner, the number of clustered groups were the same. With $\rho \leq 0.5$ the network was able to classify only two groups. These results

[2] ρ is a non-dimensional value within the interval $0 < \rho < 1$ that regulates how much mismatch is tolerated between current inputs and prototypes previously stored.

Groups	Contacts states
(a)	1,2,4
(b)	3,5
(c)	6,7,11,17,20
(d)	8
(e)	9,13
(f)	10,12,14,18,19
(g)	15
(h)	16

Table 7.2: Formed groups with $0.5 < \rho < 1.0$

corroborate the fine/coarse control of the vigilance parameter. At high values the network is more selective, forming more groups whereas with low values its discriminatory ability is reduced, classifying patterns in fewer groups.

7.3.2 Learning time

The number of patterns were increased gradually up to 200 to train further the network and verify its learning time. The simulation was implemented on a Pentium PC running at 120 MHz. The graph illustrating these results is shown in Figure 7.4.

The significance of having trained the network with more patterns was that the network took more epochs to converge, typically two. However, with a higher vigilance ($\rho > 0.9$) the network classified again all patterns at the first epoch. At the second epoch the access was direct since the patterns were already learnt. The effect of order was also tested by presenting the patterns randomly. In all cases, the learning times were very similar with a disparity of ± 0.1 s.

From Figure 7.4, it can also be seen that the curve tends to follow an exponential form, hence a small number of patterns will always be preferred. To achieve faster insertions, the network should also utilise the information as efficiently as possible avoiding redundant information. The network is expected to acquire knowledge as required by the insertion and increment this when new contact

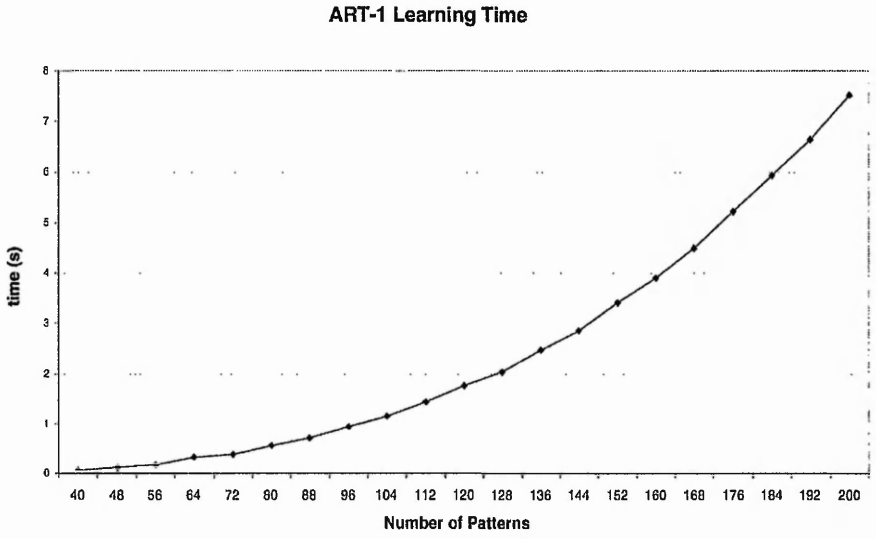


Figure 7.4: Learning time measurement for the ART-1 network

states are involved leading may be to new actions. This will certainly increase the learning times, but once the information is learnt, the required time is only needed to retrieve the action mapping pair.

The network took 7.52 s to learn the 200 different patterns and accessed them directly at the second time they were presented to the network. The fast learning showed by the ART1 network clearly outperforms the simulation results by M. Howarth using backpropagation when there were used 300 learning cycles[3] to achieve a minimum error level for the circular peg insertion and 3750 cycles for the square peg insertion [3].

The above results shows the incremental learning capability and stability of ART. The number of contact force patterns that can be handled by the network is only limited by the memory capacity of the supervisory computer. The knowledge base can increase dynamically along with the *"expertise"* of the robot. This brings the possibility of developing effectively an insertion in real-time. If the processing time of 7.52 s considers the time to learn 200 patterns and access each of them individually, then is likely that the total insertion times are also low. However, this can only be confirmed when testing the performance of the NNC in real operations.

[3] A cycle meant to be an actual arm motion.

7.4 Conclusions

Features such as *adaptation* and *decision* were identified as requirements for the NNC design in previous Chapter. The adaptation process, consisting of classifying the different contact state forces, was solved by using the ART1 algorithm during simulations. If the state was previously "known", its group was accessed directly otherwise a new group was created and dynamically incorporated into the knowledge base. During the simulations, the algorithm showed the incremental learning capability and stability necessary to be implemented in the NNC. Learning times were found to be short since the learning was achieved in only one epoch.

From the gained experience during experiments and simulations, it was decided to continue and extend the capabilities of the ART1 algorithm to implement the decision stage. It was then decided to implement this by using the supervised ARTMAP algorithm in conjunction with a dynamic knowledge base. A complete description of the final NNC implementation is given in Chapter 8.

Chapter 8

On-line Learning via Predictive ART

Initial investigations into Adaptive Resonance Theory (ART), regarding learning speed and classification were presented in Chapter 7. Results showed the feasibility of using ART in the NNC learning mechanism.

In this Chapter, it is described how a predictive ART network[1] in conjunction with a dynamic knowledge base can provide on-line learning and predictive capability to the NNC.

8.1 Introduction

Let us consider a generic cognitive system associated to a certain control process as shown in Figure 8.1. The cognitive system consists basically of three subsystems, the PROCESS to be controlled, the KNOWLEDGE about the process and the DECISION subsystem. The PROCESS subsystem needs to be controlled based on its interaction with the DECISION and KNOWLEDGE subsystems. In simple terms, an effective strategy to control the process is twofold. It should be based on monitoring reliably its operational parameters as well as accurately modifying them.

Information from the PROCESS is stored in a knowledge base. The knowledge can

[1] Also known as ARTMAP systems.

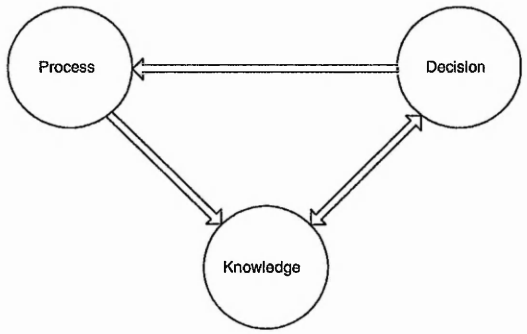


Figure 8.1: Cognitive Architecture

either be built by *hand* or *empirically* as suggested by Towell and Shavlik [78]. *Empirical* knowledge can be thought of as giving examples on how to react to certain stimuli without any explanation. On the other hand, *hand-built* knowledge is acquired by only giving explanations but without examples. For instance, if a large number of objects and their corresponding names are shown to a person, e.g. stapler, mug, pencil, etc. After the training process has finished, this person should, based on the example-based knowledge, be able to recognise a randomly presented object. On the other hand, a person who has only received explanations on how to recognise a particular object should also be able, without examples, to recognise the object.

The third element in the cognitive system is the DECISION subsystem. Here, based on the current state of the process and the information stored in the knowledge base, the final assumption is made regarding the modification of the parameters in the PROCESS.

The above description is a generic architecture for an intelligent system, which can be applied to the automation of the robotic assembly process. Issues regarding an assembly operation were considered in Chapter 5 in terms of monitoring the contact forces during the assembly. In this Chapter, issues regarding the knowledge acquisition and prediction are addressed.

8.1.1 Building the knowledge

The first situation to address for the NNC is the knowledge acquisition. From the explanation above, two approaches can be distinguished clearly. The first

approach would be to give the robot plenty of examples in the form of training sets. That is, building its knowledge empirically. The second approach would be to hard code a rule-based system into a robot program (hand-built knowledge).

In the particular case of part assembly, the use of a training set can be useful to teach the robot how to react to constraint forces during similar insertions. However, when the geometry of the components or working conditions are significantly different, the experience already held in the knowledge base may not be suitable for the operation and leading potentially, to wrong motions. On the other hand, building the knowledge by using rules and implementing it into a robot program would only be a problem-specific solution. The knowledge will also require to be adapted by the user if the geometry or the location of the mating pairs change.

In both approaches above, the importance of having an adaptable system to build the knowledge into the robot is clear. From the experience gained during experimentation, it was determined that a suitable strategy should include a combination of both methods[2].

The strategy to build the knowledge base consists in initially showing the robot how to recognise different assembly stages. Later on, the learning will be reinforced by many examples related to particular assemblies. The idea is to develop a generic knowledge base that can be enhanced further for every particular insertion, therefore providing a skill acquisition capability to the robot.

The framework for building up this information can be better understood by using Figure 8.2 as suggested by Shavlik [79].

In this scheme, the learner first inserts symbolic information into the ANN. Once the knowledge is in neural representation[3], then by using training examples the initial knowledge can be *refined.* The resulting ANN can then be used, if required, for extracting symbolic information. In this manner the use of ANN can improve the performance of rule-based systems.[4]

[2]Furthermore, this idea is supported by psychologic evidence that suggests that theory and examples interact closely during human learning [78].

[3]Sometimes this is referred to as numeric or subsymbolic representation.

[4]These systems are also known as ruled-based expert systems, or simply expert systems.

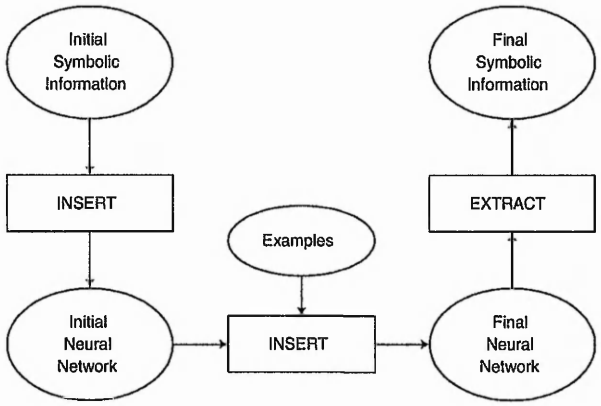


Figure 8.2: Framework for combining symbolic and neural learning

8.1.2 Prediction and decision making in the NNC

The cognitive architecture proposed in this thesis results in a novel NNC that combines both an empirical and a hand-built knowledge base. The idea is to enhance the capability of the ART network by providing an initial Knowledge Base. This is basically the development of ART as a Knowledge Base Artificial Neural Network (KBANN)[5]. In KBANNs, the knowledge is inserted into the network and subsequently refined by ANN training. Furthermore, research has demonstrated that the refinement of knowledge in KBANN is better than using purely symbolic systems [80].

Also, in the KBANN literature, the knowledge is embedded into the network by means of using an inferring symbolic system.[6] However, the core idea here is to form a Primitive Knowledge Base (PKB) by the manipulator itself under constraint motion therefore, the information will be numerical.

The purpose of using a dynamic knowledge base is to provide the robot with the capability of recognising primitive forces[7] during early stages of assembly, so that

[5]Although most current research in KBANN is based on symbolic representation to present and embed knowledge into the ANN, for the purpose of this thesis the network will still be considered as KBANN following the KBANN's language description by Towell and Shavlik [78].

[6]The term symbolic refers to the use of if-then rules which are used in rule-based systems. In counterpart the so-called subsymbolic system is related to the ANN representation of knowledge and associated to its weights.

[7]forces corresponding to the 6 DOF of the manipulator.

initial conditions can be started. During knowledge refinement, and by giving more examples, this knowledge is expected to be enhanced and improved.

An overview of the assembly system based on the NNC is shown in Figure 8.3. The Fuzzy ARTMAP (FAM) is the heart of the NNC. The controller includes three additional modules. The Knowledge Base, the Pattern-Motion Selection and the Automated Motion module.

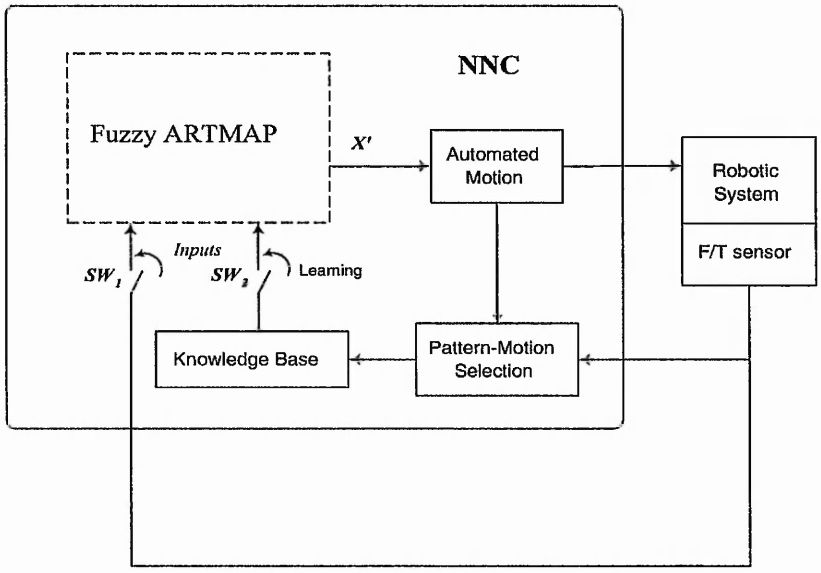


Figure 8.3: System Structure

The Knowledge Base stores the initial or PKB information about the environment. This information is used only during the first stage of training. In this stage the switch SW_1 will be open and the switch SW_2 closed since the initial training is made only using the PKB. After passing this initial state, the FAM network will predict the next motion based on the current input from the sensor (SW_1 closed and SW_2 open). Later if appropriate, the PKB will be enhanced by patterns that favoured the assembly and whose inclusion is regulated by the Pattern-Motion module. This module keeps track of the F/T patterns and verifies whether the action was good enough to allow the FAM network to be re-trained. If this is the case, the switch SW_2 is closed and the corresponding pattern-action provided to the FAM for on-line retraining. Future predictions will be based on this newly trained FAM network. The Automated Motion module is basically in charge of sending the incremental motion request to the robot controller and

handling the communication between the slave computer and the FAM network output. External components to the NNC in the Robotic System are the robot controller, the manipulator itself and the F/T sensor that provides the pattern information.

The strategy for the formation of the PKB is to gather the minimum information necessary for the NNC to start learning the assembly. The information is acquired by using the 6 DOF of the robot and moving the manipulator against a surface in such a way to generate the corresponding signals, namely fx, fy, fz, mx, my and mz. The training scheme is explained in detail in section 8.2.2.

8.2 Robotic Assembly Controller

The Robotic Assembly Controller (RAC) software, which is the integration of the NNC with the sensing system and the robot arm, was developed using Visual C++ 5.0 for PC Windows environment. This program is resident in the host computer and their main components are the knowledge formation and the NNC processing. The flowchart for this program is given in Figure 8.4 and the following sections explain each stage in detail.

8.2.1 Settings

During this first stage the user selects the appropriate parameters in terms of rotation and translation of the sensor coordinate frame as it was explained in section 3.6.3. The origin of the coordinate frame needs to transferred from the default location at the centre on the sensor unit to an appropriate location depending on the physical dimensions of the gripper. In the case of the PUMA 761 robot, the World, Tool and Sensor coordinate frames were all superimposed during the insertion operation. In this manner, using only one reference frame, the analysis and interpretation of the data is simplified greatly.

The acquisition of the PKB is required for the first assembly operation to bias the initial motions of the arm. After the first insertion is completed, the user has the choice to either use the same PKB or acquire a new PKB. The same PKB

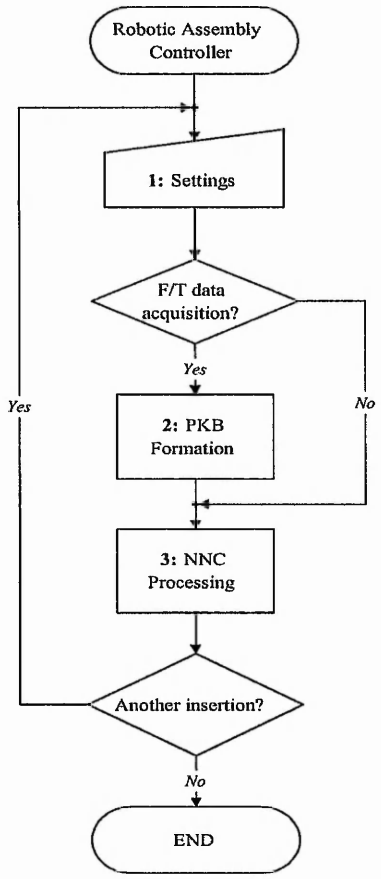


Figure 8.4: Flowchart of the assembly task

can be used if the parts geometry for the next assembly operation are the same or similar. On the other hand, if the parts geometry change another PKB will be required. E.g. PKB for chamferless and chamfered parts.

8.2.2 PKB Formation and robot training

.

The assembly information available to a blindfolded human operator are the contact forces experienced while attempting to insert the workpiece. In addition, there would be an associated knowledge on how to react to *primitive* constraint forces. In other words, the intrinsic attitude of moving a workpiece opposite to the constraint forces that impede its insertion. Similarly, these ideas are put into practice with the robot in order to form its PKB. The idea is to provide the primitive contact forces and the corresponding reaction motion for all the main

directions with a 6 DOF manipulator.

Basically, the formation of the PKB is to teach the robot how to react to every component of the F/T signal vector. The influence of each vector component requires a motion opposite to the direction of the applied force to diminish its effect. The procedure is illustrated in Figure 8.5. For simplicity, only the lower arm of the manipulator has been shown.

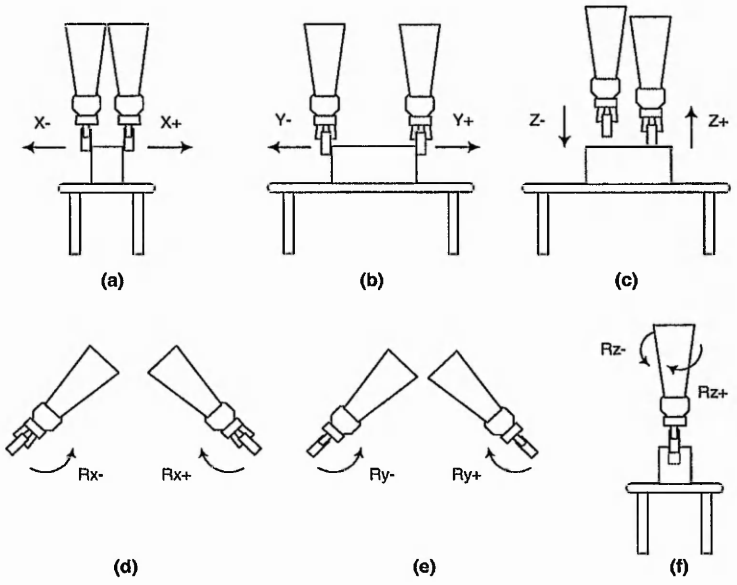


Figure 8.5: Training Procedure

Every motion of this type is referred to as a Primitive Motion (PM) and the idea is to teach the robot *where* to move when single F/T components, i.e. fx, fy, fz, mx, my, or mz are applied to the workpiece. Figure 8.5(a), 8.5(b) and 8.5(c) illustrate the PM needed to diminish the corresponding constraint force in the X, Y or Z axis. Note that in Figure 8.5(c), when the arm is in free-space the PM will be in -Z direction since this is the condition (minimum constraint forces) to proceed downwards during this assembly operation. Generally speaking, the training consists of moving the workpiece against a rigid object to produce the appropriate component force and consequently determine the corresponding PM. The magnitude of the constraint forces applied to the workpiece is bounded by the force limit selected in the Robotic Assembly Controller Program. The PM corresponding to the rotation in X and Y axis (Figure 8.5(d) and 8.5(e)) were assigned after rotating the arm in free-space at an angle so that a single mx or my

component was produced. The PM, Rz, was given to the network using a square peg into a square hole producing a moment around the Z axis as illustrated in Figure 8.5(f).

Once the arm is in constrained motion, the user is able to select the appropriate motion as indicated in Figure 8.6

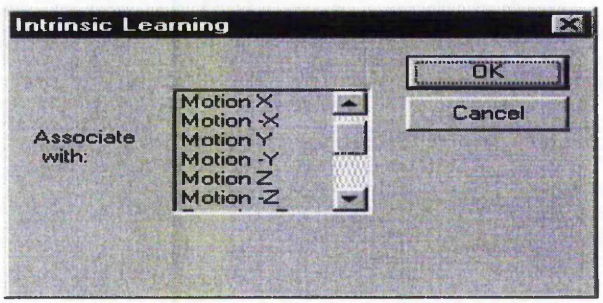


Figure 8.6: Motion Selection Dialogue

At this time, the F/T pattern will be acquired in the knowledge base and associated with the selected motion. The storage of the F/T vector and the PM will form the PKB that is required to start the assembly for the very first time. Once the first insertion has been completed, the robot may possibly have increased its knowledge. If so, the PKB is enhanced and an enhanced Knowledge Base (EKB) version will be used during the following insertion.

8.3 NNC processing

The mechanics of the NNC processing is explained below with the aid of Figure 8.7 as follows:

At the start of the operation the PKB is formed as it was explained in section 8.2.2. This is illustrated in Figure 8.7(a). In this stage, the robot is moved via the Automated Motion module and the force-action mappings are inputted into the PKB.

The next stage is to train the FAM with this knowledge as shown in Figure 8.7(b), so that the internal representation and initial categorical representation of the PKB mapping can be made. Note that both inputs I_a and I_b are fed into the FAM network during off-line training.

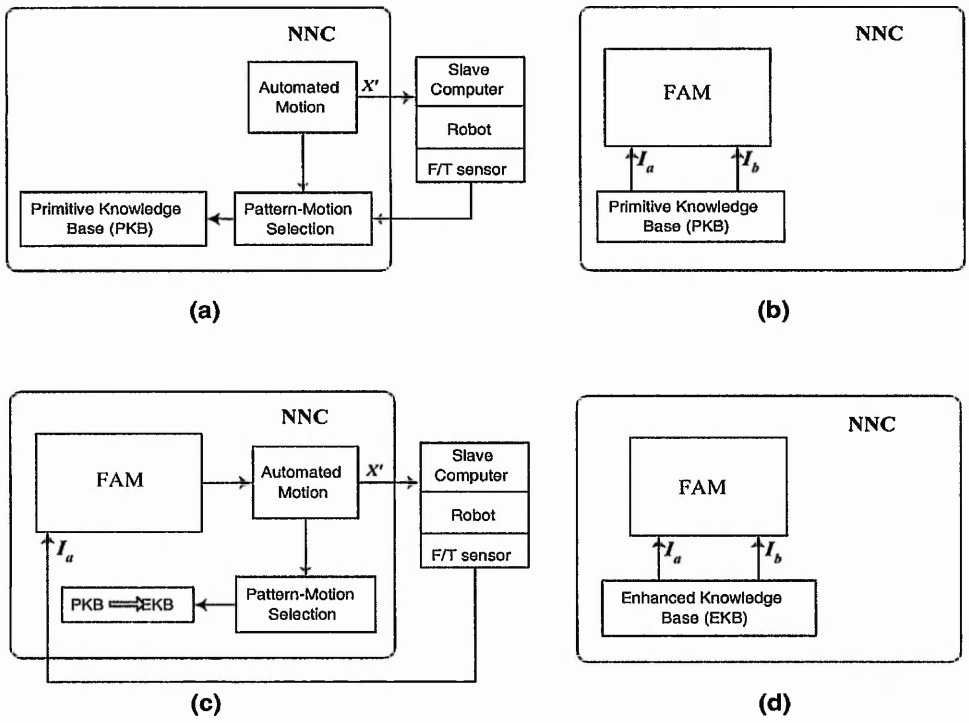


Figure 8.7: NNC Mechanics

During testing only, the I_a that represents the current F/T pattern is presented to the FAM network. This is shown in Figure 8.7(c). The first prediction is made at this time and the incremental motion is executed via the Automated Motion module. If the incremental motion was successful and reduced the constraint forces significantly then this pattern is added to the PKB. This enhanced knowledge will hold information associated to the specific geometry and henceforth be referred to as EKB.

If a new pattern in the EKB has just been added, the FAM network will be re-trained using this knowledge as shown in Figure 8.7(d) in order to modify its categorical representation.

If no new knowledge has been added to the knowledge base, then step 8.7(c) will be repeated until the end-condition is satisfied. Otherwise, both steps 8.7(c) and 8.7(d) will repeated. At the end of the insertion the EKB will actually contain information related to the assembly geometry, e.g. circular, square, etc.

8.3.1 NNC dynamics

The NNC processing, as explained in the above section, involves the use of a Knowledge Base that modifies its structure depending on the particular assembly geometry. The internal representation of this knowledge is also modified within the FAM network since it is re-trained on-line according to the dynamics of the NNC. In the following sections, the NNC processing is explained describing the functionality of the knowledge allocation mechanisms. The NNC processing is illustrated in the flowchart shown in Figure 8.8.

8.3.2 NNC settings

At the beginning of the operations the following settings are provided:

- Information about the type of coordinate system. This information is required by the Automated Motion module in order to move the arm either in World or Tool coordinates.
- End-condition. In this field, the number of incremental motions during a successful insertion are given. It should be mentioned that only incremental motions developed in the insertion direction are counted.
- FAM parameters. Information regarding learning rate, size of the input vectors (I_a and I_b), vigilance parameters, number of epochs, etc. are given at this time. A complete description of the ART and FAM algorithms and parameters are given in Appendix B.

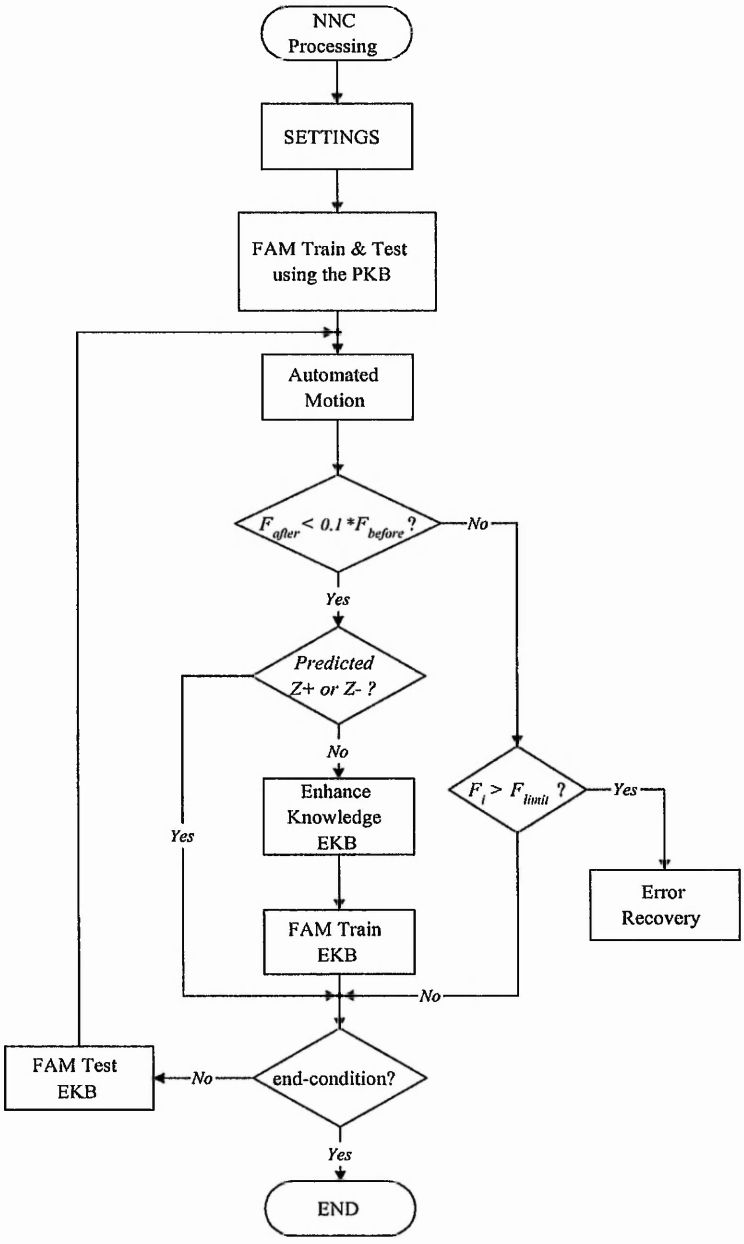


Figure 8.8: Flowchart of the NNC processing

8.3.3 FAM train & test using the PKB

Learning is initiated in the NNC by training the FAM using the PKB. This is illustrated in Figure 8.9

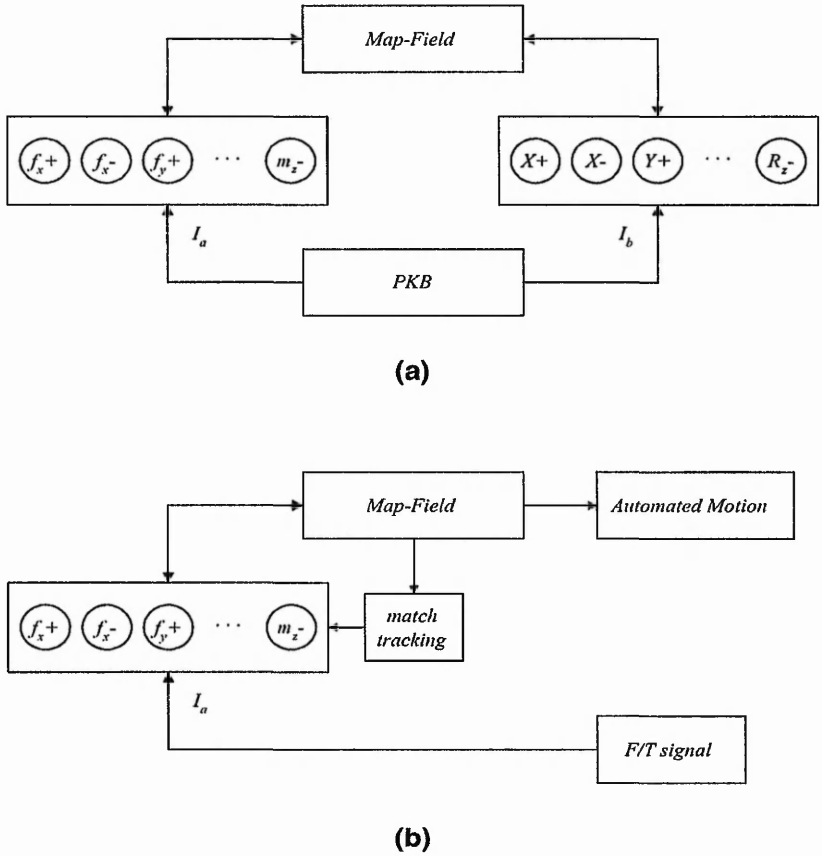


Figure 8.9: FAM train & test using the PKB

Initially, both inputs I_a and I_b are presented to the network as shown in Figure 8.9(a)[8]. Input I_a represents the normalised F/T information whereas I_b provides the corresponding motion direction. Categorical representations (or nodes) are created in modules ARTa and ARTb and the learning of the mapping occurs in the "Map-field" via its weight connections with the ARTa module. As can be seen in Figure 8.9(a), the maximum number of nodes in both modules will be twelve which corresponds to the possible number of motions/forces using a 6 DOF robot.

[8]Both inputs also contain their corresponding complement coding values. See Appendix B for details.

Immediately after training, the network is tested in order make a 'prediction' about the next incremental motion. The input information this time is solely provided by the F/T sensor. This is illustrated in Figure 8.9(b). The categorical representation of the input vector is compared with the previously learnt categorical representation of the output vector in the Map-field. If a mismatch occurs between these two categorical representations then the Match tracking system becomes active and modifies the vigilance parameter (ρ_a) in module ARTa so that another category can be tested until the system's vigilance parameter (ρ_{map}) is met. The output prediction is read directly at the Map-field and sent to the Automated Motion module to command the motion.

At this initial stage, the arm will most probably be in free-space and consequently a prediction in the Z direction is expected. However, the prediction can also be made in constraint motion in any direction.

8.3.4 Automated motion

The main function of this module is to move the robot arm incrementally. The communication is achieved at two levels as explained in Chapter 3. Higher level communication is achieved by sending a command to the robot controller to initialise the ALTER mode. The communication is then established in low level (ALTER mode) via the Automated motion module and the slave computer.

In this module, the magnitude of the F/T vector is evaluated using the following equation:

$$F = \sqrt{fx^2 + fy^2 + fz^2 + mx^2 + my^2 + mz^2} \tag{8.1}$$

The magnitude before and after the incremental motion (F_{before} and F_{before}) are sent to the Pattern-Motion Selection module for assessing the incremental force change.

8.3.5 Pattern-Motion selection and knowledge enhancement

There are potential problems associated with the learning mechanism which are solved by the Pattern-Motion Selection module. The robot should continue moving in the insertion direction if, and only if, a minimum force value has been reached. This situation should trigger the learning mechanism in order to allow the acquisition and learning of the pattern-action pair that produced such a situation. In the event of continual learning after having reached this point, the performance of the NNC might decay. This situation is similar to what is known as overtraining, overfitting or overlearning in ANNs. At this point the learning should be stopped because if the robot learns other patterns under the above circumstances, eventually the minimum force value will be different leading to wrong motions. The same applies to the condition when the end-effector meets a force higher than the force limit. There should not be any further learning during this situation since learning a higher force would probably damage the sensor. The above situations can be resumed in three fundamental questions:

1. What is a good motion?
2. How to recover from errors?
3. Which motions should or should not be learned?

Having an assembly system which is solely guided by contact force states, the criterion to decide whether the motion was good enough to be learnt is based on the following expression:

$$F_{after} < 0.1 * F_{before} \tag{8.2}$$

F_{after} and F_{before} are computed using Equation 8.1. Expression 8.2 means that if the total force after the incremental motion is significantly reduced then that pattern-action will be considered good to be included in the knowledge base.

Experiments showed that if this threshold value is set higher the network become very sensitive and eventually showing overtraining behaviour.

Forces that are higher than the value given by $0.1 * F_{before}$ and lower than the F_{limit} are still good values. However, the corresponding pattern-action pair will only be used during network recall. This situation is illustrated in Figure 8.10 that shows three possible situations: learning, recall and error recovery.

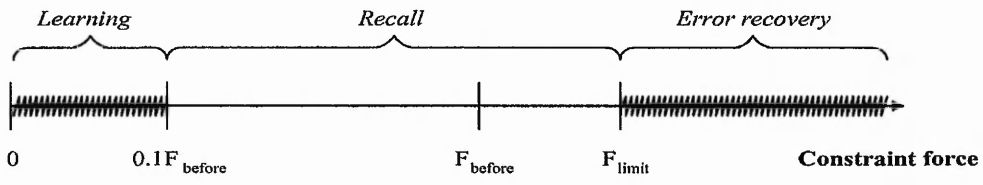


Figure 8.10: Learning, Recall and Error Recovery

The third area is a situation where $F \geq F_{limit}$. In this situation the user is alerted and asked to reposition the arm.

There will be ambiguous situations in which learning should not be permitted. This applies to patterns in the insertion direction (usually Z direction). Consider downward movements in the Z- direction. At the time the peg makes contact with the female block, there may well be a motion prediction in the Z+ direction. This recovery action will certainly diminish the contact forces and will satisfy the condition given by the expression 8.2 in order to learn the force-action pair. However, this situation is redundant since it was given when the PKB was formed and it is likely that it will corrupt the PKB. Similarly, learning should not be allowed when the arm is in free-space. In this situation, F_{after} and F_{before} will be very similar and again learning another pattern in the Z- direction will be redundant. Both situations were tested experimentally by the author and revealed that an unstable situation may appear if further learning is allowed in the insertion direction.

The situations discussed here regarding the Pattern-Motion Selection were implemented in the NNC processing as it is shown in lower part of the flowchart in Figure 8.8. After the pattern-action has satisfied expression 8.2 and the prediction direction is not in the Z direction, the pattern is allowed to be included in the

new "expertise" of the robot, the EKB. Patterns that do not satisfy expression 8.2 and whose values are lower than the F_{limit} will only be used to recall previous knowledge.

The knowledge refinement process will continue in the NNC until the end-condition is satisfied. If this is the case, the NNC processing will be terminated and another insertion can be initiated as indicated in the lower part of the flowchart given in Figure 8.4.

8.4 Summary and Conclusion

Throughout the development of this project two essential requirements for the NNC were identified: on-line learning and knowledge enhancement. A novel NNC architecture was proposed in this Chapter to fulfil these requirements. The architecture includes the use of Primitive Knowledge Base (PKB) which is integrated in the NNC by using the manipulator under constrained motions. After having formed this PKB, the knowledge can then be enhanced during assembly operations. The criteria for allowing new information into the knowledge base is basically centred on how much the force-action mapping favoured the insertion. Results that corroborate the author's approach are given in the following Chapter where the NNC performance is assessed under different working conditions.

Chapter 9

Performance Assessment of the NNC and Discussions

In this Chapter the assessment result of the NNC performance is presented. Several insertions were made using different working conditions, i.e. different part geometry and positional offset. Also the knowledge discovery capability of the NNC was verified by stopping/inhibiting its learning during operations. The graphs resulting from the tests are presented in this Chapter as well as in Appendix A where appropriate.

9.1 Prior Settings

The objective of this research, as established earlier, was the creation of a NNC to enable the PUMA 761 robot arm to be autonomous and self-adaptive. The autonomy of the arm consists in having its motions solely directed by the NNC without having to write a specific program for different assembly tasks. The arm is able to be self-adapting since the system is designed to recognise particular geometry features, discover knowledge associated to the assembly and, based on this knowledge, enhance its current behaviour.

In order to assess the performance of the NNC a series of tests were carried out. Broadly speaking, these tests looked at aspects such as assembly speed, generalisation power and knowledge discovery. The tests were performed using

different part geometry and positional offset. In the following sections system parameters such as part geometry, clearances, learning rate, etc. are described. The PKB employed during operations is described and finally, the presentation of results explained.

9.1.1 System Parameters

The assembly operations were carried out using the following assembly parameters:

1. **Mating pairs.** The male and female components are shown in Figure 9.1. The components were manufactured from aluminium using three different geometrical shapes. Two of them were symmetrical: Circular and Square, while the third one was asymmetric and referred to as Radiused-Square. The term Radiused-Square was used because it was basically a square component with one corner rounded to a 12.5 mm radius.

2. **Insertion Direction.** The direction of insertion in all operations was downwards in the vertical direction (Z-).

3. **End-Condition.** This is the termination condition for the assembly and it was established to be 3/4 of the insertion depth of the peg into the hole. Since the body of the peg is 15 mm long, the insertion was stopped approximately when 12 mm of its body was inside the hole. In terms of incremental motions, this represented 50 motion steps in the assembly direction.

4. **Clearance.** The total clearance between the peg and the hole for each type of insertion was as shown in the following Table:

Type	Peg-Hole clearance (mm)		
	Circular	Square	Radiused-Square
Chamfered	0.1	0.1	0.1
Chamferless	0.09	0.1	0.09

Table 9.1: Peg-Hole clearance

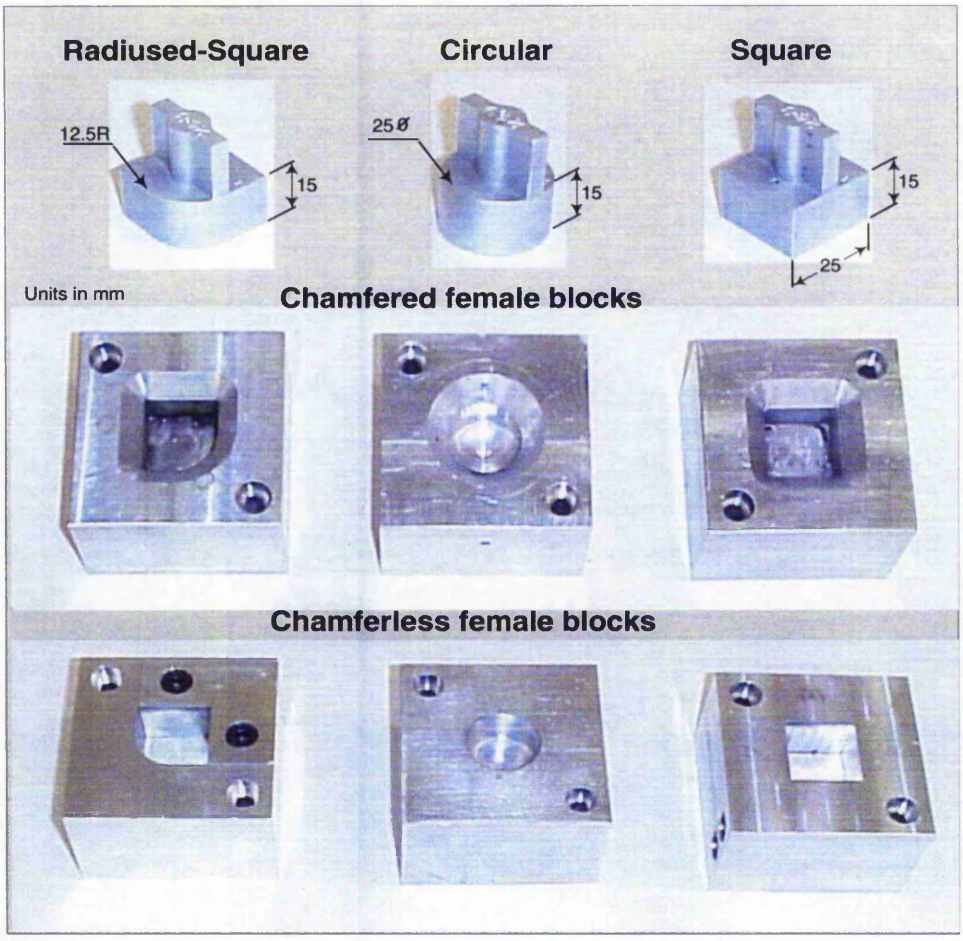


Figure 9.1: Assembly components

5. **Chamfer.** The chamfer was present on the female blocks only and set to 45°.

6. **Offset.** The offset of the peg was measured with respect to the centre of the hole and given in the X and Y directions. In the square and radiused-square peg insertions angular misalignments about the Z axis were also given.

7. **Force threshold limit.** A force limit value of 15 lb was established to prevent any damage to the F/T sensor. When a higher force value was generated, the arm was stopped immediately.

8. **Servoing time delay.** This is the time that the robot was halted between incremental motions. A time delay of 1 second was used. This time can be further reduced and is only limited by the communication overhead between

the robot's controller and host PC.

9. **NNC Parameters.** The Fuzzy ARTMAP network parameters were set for fast learning (learning rate = 1) and the following vigilance:

$$\overline{\rho_a} = 0.2 \ (\textit{base vigilance})$$

$$\rho_{map} = 0.7$$

$$\rho_b = 0.9$$

The inputs were first pre-processed using their complement code value[1] and presented twice (i.e. two epochs) to the FAM network.

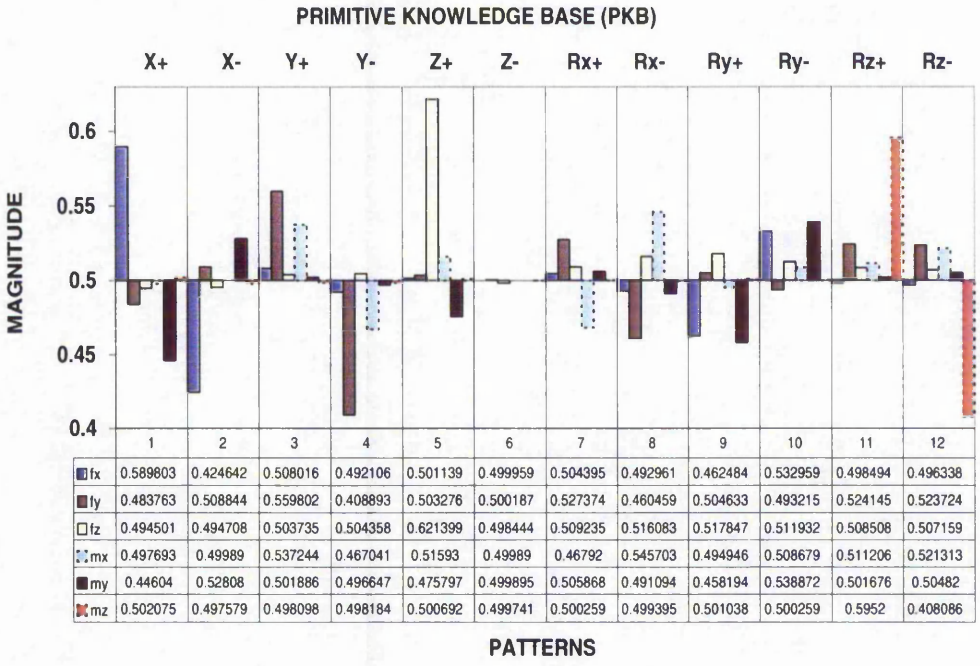
9.1.2 Knowledge of the environment –Primitive Knowledge Base (PKB)

The robot requires a basic knowledge of the environment, that is, an associated knowledge about how to react to primitive constraint forces. The procedure was straightforward since it only required "touching" a block using the 6 DOF robot's motion. This was accomplished as described in Section 8.2.2. Figure 9.2 shows the PKB.

The F/T data from the sensor was scaled to the range [0,1], where the extreme values 0 and 1 corresponded to a force of 15 lb and -15 lb respectively. Negative values were assigned to the interval [0,0.5) and positive values were assigned to the interval (0.5,1]. Every column of data below the graph corresponds to the input vector I_a to the network. An incremental motion was assigned to each input vector as it is shown at the top of the graph forming in this manner the output vector I_b of the network during the training phase. It should be noted that the origin in the graph is set to 0.5, where positive and negative values are represented in the upper and lower halves of the graph respectively.

The above knowledge base was used to bias the learning at the start of the operation and it was common to all three geometrical shapes during chamfered type insertions.

[1] see Appendix B for details.



	1	2	3	4	5	6	7	8	9	10	11	12
fx	0.589803	0.424642	0.508016	0.492106	0.501139	0.499959	0.504395	0.492961	0.462484	0.532959	0.498494	0.496338
fy	0.483763	0.508844	0.559802	0.408893	0.503276	0.500187	0.527374	0.460459	0.504633	0.493215	0.524145	0.523724
fz	0.494501	0.494708	0.503735	0.504358	0.621399	0.498444	0.509235	0.516083	0.517847	0.511932	0.508508	0.507159
mx	0.497693	0.49989	0.537244	0.467041	0.51593	0.49989	0.46792	0.545703	0.494946	0.508679	0.511206	0.521313
my	0.44604	0.52808	0.501886	0.496647	0.475797	0.499895	0.505868	0.491094	0.458194	0.538872	0.501676	0.50482
mz	0.502075	0.497579	0.498098	0.498184	0.500692	0.499741	0.500259	0.499395	0.501038	0.500259	0.5952	0.408086

Figure 9.2: PKB

9.1.3 Presentation of Results

The history of every insertion was tabulated as shown below:

Insertion	Offset (δx,δy,δRz) (mm,mm, °)	Learning	New Patterns	Alignment Motions	Total Motions	Process. time (s)	Comments	Figure

Figure 9.3: Presentation of results

where:

- Insertion. This is the record of complete insertion operations in chronological order.
- Offset. This corresponds to the offset with respect to the centre of the hole. Please note that two linear increments were used in X and Y axes and one angular increment around Z axis. Both angular and linear offsets had negative or positive values.
- Learning. In this field, the status of the learning is indicated. An ON or START state meant that the network was allowed to learn new information

from the assembly. Conversely, an OFF state indicated that new patterns were not allowed to be learnt. In most of the insertions the learning switch was kept ON to allow the progressive acquisition of knowledge. When the learning was inhibited (OFF state) only the PKB was used. In certain circumstances a STOP state was used to prevent the knowledge from increasing any further.

- New Patterns. In this field, the number of learned patterns during the insertion is reported.
- Alignment Motions. These are the corrective motions used by the arm to reach the end-condition. Any type of motion could be included here except a motion in Z-, which was the insertion direction.
- Total Motions. The total number of motions to reach the end-condition.
- Processing Time. This is the total processing time during the full insertion that includes, data acquisition and training/testing of the NNC, but excludes the elapsed time during the serial communication with the robot controller and the delay between motions. Roughly speaking, the insertion time is the servoing time delay plus the communications overhead multiplied by the number of arm motions.
- Comments. Important notes for each insertion are included here.
- Figure. Indicates the corresponding Figure number either within this Chapter or in the Appendix A.

The analysis of the retrieved data during operations has been divided into two sections. The first section corresponds to chamfered insertions while the second corresponds to the chamferless insertions. Selected graphs are included in the present Chapter while the remaining figures can be consulted in Appendix A.

9.2 Chamfered Peg-in-hole Insertion

9.2.1 Circular chamfered peg insertion

The signal patterns during the first insertion using the PKB are shown in Figure 9.4.

The offset was set to X = -0.8 mm and Y = -0.4 mm with respect to the centre of the hole. The upper graph represents the force traces during the insertion whereas the middle graph represents the moment signal. The motion directions commanded by the NNC at every time step are given in the lower graph.

In the Motion Direction graph, the horizontal axis was selected to correspond with the Z- direction. By using this convention, bars above the horizontal axis represent linear alignments and bars below the horizontal axis represent angular alignments during the insertion. It can be observed from Figure 9.4 that the arm moved down until step 20, before the first alignment motion occurred as indicated by the bar at step 21 in the X+ direction. Subsequent alignment motions were developed as shown until step 58, where the end-condition was reached. A total of 8 alignment motions were required to complete the insertion. Note that the insertion direction Z- is not considered an alignment motion.

By the end of this first insertion, the network had autonomously learned three new patterns within the chamfer hole: X+, X- and Y-. This additional information represented the force-action mapping within the chamfer and the knowledge discovery was related to the circular geometry. It is worth mentioning that as soon as one pattern was acquired into the knowledge base, the next time the training took place this pattern was accounted for. Therefore, the internal representation of the force-action map within the NNC was modified enhancing the expertise of the robot. After learning new patterns, the PKB became the new Enhanced Knowledge Base (EKB).

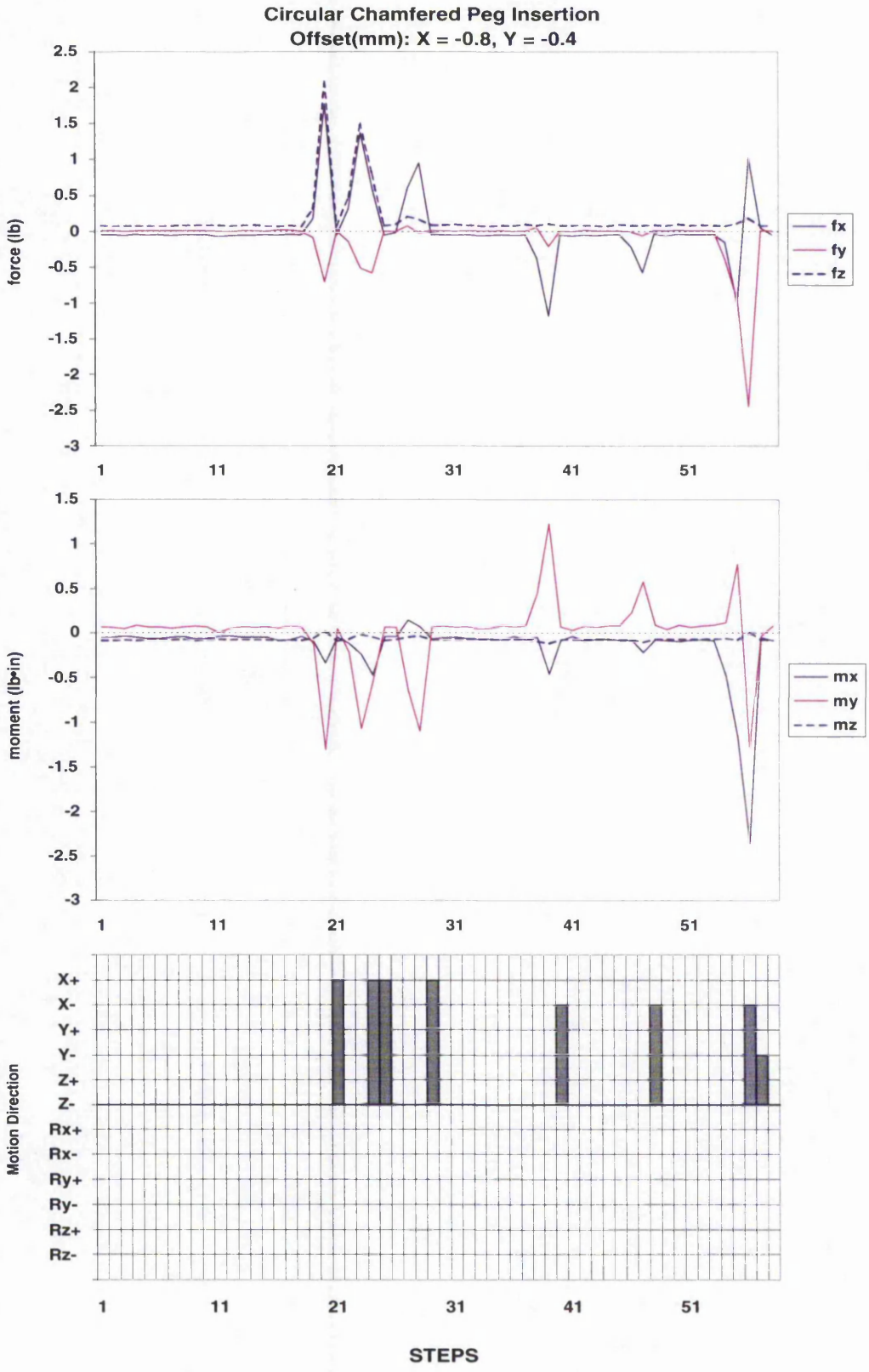


Figure 9.4: First Circular Chamfered Peg Insertion

In subsequent insertions, the EKB was used and additional patterns were learnt as illustrated in Table 9.2

Circular Chamfered Peg Insertion

Insertion	Offset (δx, δy, δRz) (mm,mm, °)	Learning	New Patterns	Alignment Motions	Total Motions	Process. time (s)	Comments	Figure
1	(-0.8, -0.4, 0.0)	ON	3	8	58	5.17	OK X+,X-,Y-	9.4
2	"	"	2	8	58	5.23	OK X+,Y-	A.1
3	"	"	0	4	54	4.70	OK	A.2
4	"	"	1	4	54	5.00	OK X+	A.3
5	"	"	1	6	56	4.94	OK X+	A.4
6	"	"	2	6	56	5.18	OK X+, Y-	A.5
7	"	"	1	6	58	4.93	OK X+	A.6
8	"	"	1	6	56	5.09	OK X+	A.7
9	"	"	1	6	56	4.84	OK X+	A.8
10	(-2.5, -2.5, 0.0)	ON	1	14	64	5.47	Incl.Rx- Y+	A.9
11	"	"	0	13	63	5.44	OK	A.10
12	"	"	0	16	66	5.60	OK	A.11
13	"	"	0	14	64	5.42	OK	A.12
14	"	"	0	13	63	5.49	OK	A.13
15	"	OFF	0	25	85	7.35	Z+(14) Ry-(3)	9.5
16	"	"	0	24	84	7.27	Z+(15) Ry-(3)	9.6

Table 9.2: Circular Chamfered Peg Insertion

During the insertions the offset from the centre of the hole was as indicated and given towards X- and Y- direction. During the first insertion the network learned 3 new patterns and this operation required 58 incremental motions and only 8 alignment motions. The learned patterns were X+, X- and Y- as indicated in the comments field. The processing time for the whole insertion was 5.17 s. This time considered only the processing of the patterns, training and testing of the network. The actual insertion time was longer since a delay of 1 s was added to avoid the transient stage after every incremental motion. Considering this delay, the first assembly was accomplished in approximately 63.17 s. Subsequent assemblies were carried out at the same offset and the number of learned patterns decreased to only one. The processing time showed only small fluctuations.

Table 9.2 also shows the "expertise" acquired by the robot during the operations. After nine insertions the NNC had learnt 12 additional patterns. This implied that only 12 patterns were good enough to be learned. This new knowledge reinforced the prediction capability of the network since the new patterns were actually generated by the particular geometry of the parts, i.e. circular. The type of learned pattern at every insertion is indicated in the comments field.

A second set of insertions were carried out using a larger offset (insertions 10 to 14). The results are shown in the lower part of the Table 9.2. What is remarkable about the performance of the NNC is that it only learned one additional pattern indicating that the network had already acquired the necessary knowledge about the chamfer and used this information effectively. As the starting point was further from the end-condition, the time to complete the insertion was proportionally longer. This is reflected in both the number of alignment motions and the total number of actions. Another interesting result is that the NNC also predicted rotation about the X axis, which indicates that information from the original PKB was still used if appropriate, as occurred in this situation.

Expertise test

A further test was conducted using the EKB and the incremental learning capability inhibited. The performance of the NNC during this situation can be seen in the last two rows of Table 9.2 where the learning was switched OFF. Despite that the offset was the same as in insertions 10-14, the number of alignment motions and insertion time were higher. The corresponding graphs are given in Figures 9.5 and 9.6.

The robot was not allowed to learn contact states within the chamfer hence the NNC generated motions based only on its initial PKB. This resulted in motions that produced an excessive fz. As a result, the NNC predicted a series of compensatory movements in Z+ and Ry- to recover from these situations. The robot was ultimately able to insert the workpiece successfully, however the performance was poorer in terms of alignment and consequently speed.

It can clearly be observed that the same operation with the same offset can be achieved more efficiently and faster if the robot uses the EKB. In other words, the robot has demonstrated its *dexterity* when it is allowed to use its expertise.

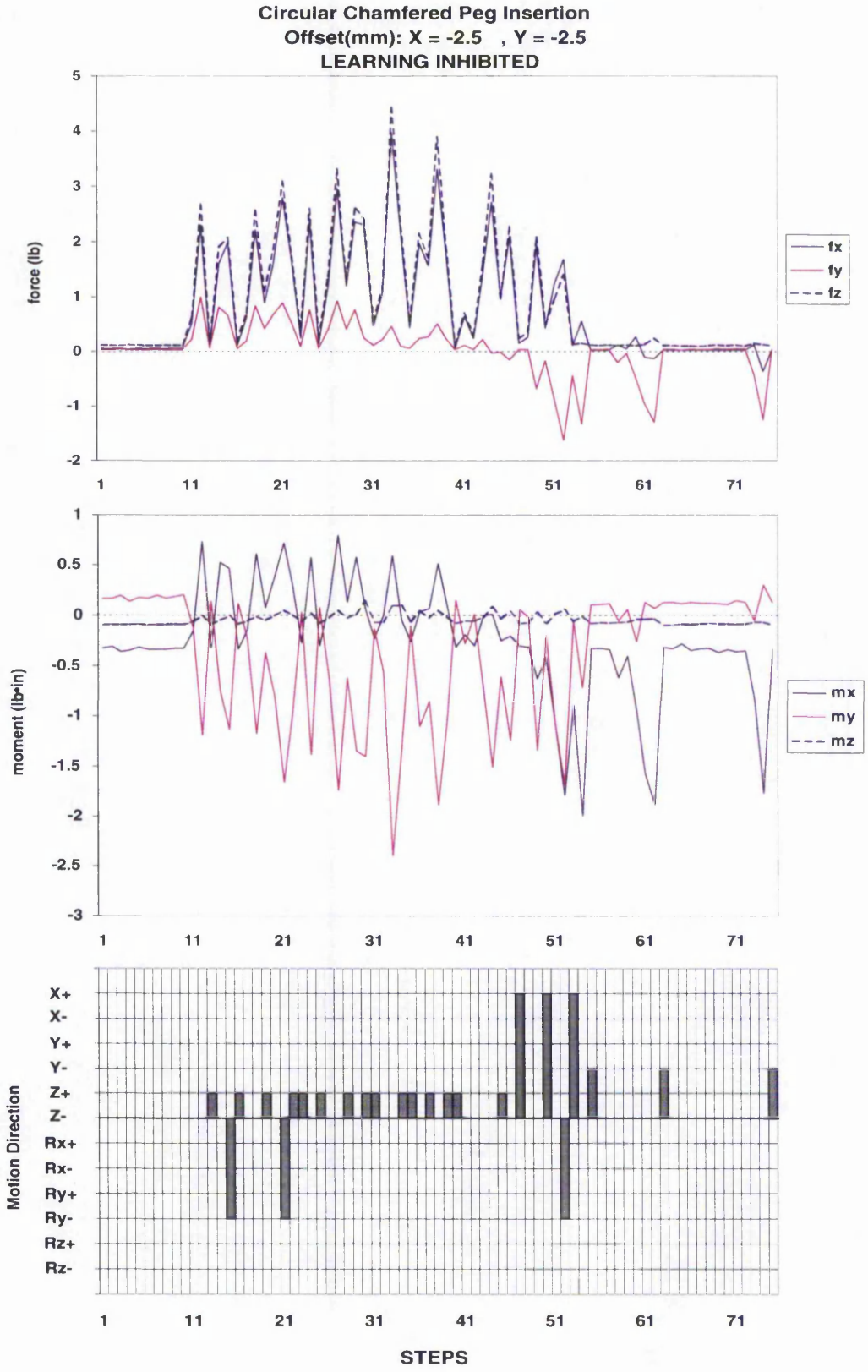


Figure 9.5: First insertion with learning inhibited

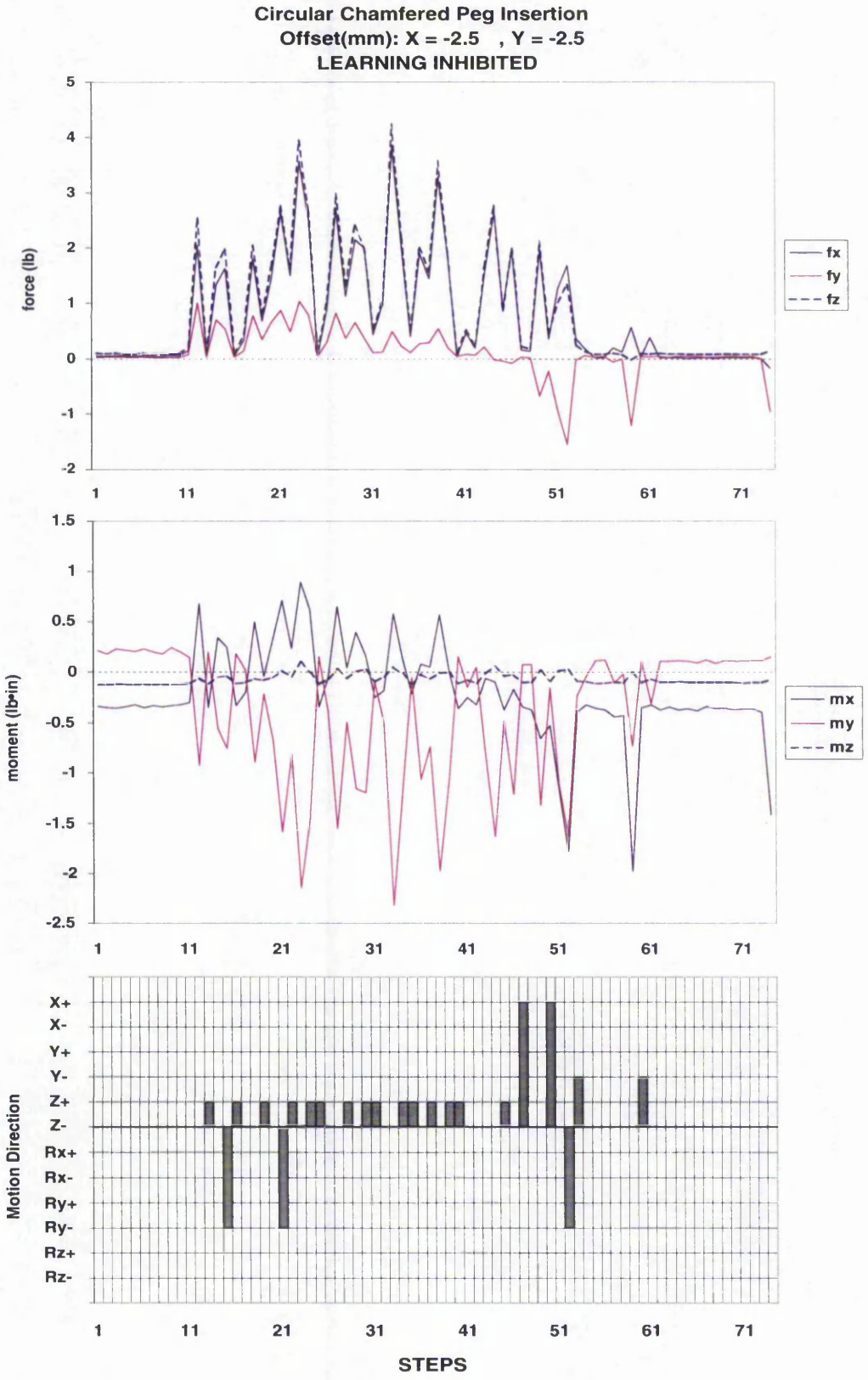


Figure 9.6: Second insertion with learning inhibited

9.2.2 Square chamfered peg insertion

The procedure to assess the performance of the NNC for the chamfered square peg insertion was similar to the circular insertion. At the start of the operation the knowledge base for the network was the same PKB and several offset values were given to the peg with respect to the centre of the hole. The results are summarised in Table 9.3[2].

Square Chamfered Peg Insertion

Insertion	Offset (δx, δy, δRz) (mm,mm, °)	Learning	New Patterns	Alignment Motions	Total Motions	Process. time (s)	Comments	Figure
1	(-2.5, -2.5, 0.0)	START	1	39	89	7.76	OK, X+	9.7
2	"	ON	12	41	91	8.17	OK, Y+, Y-, X+, X-	9.8
3	"	ON	6	15	65	5.66	OK, X+(1),Y+(3),Y-(2)	9.9
4	(-0.37, -0.25, 0.0)	OFF	0	45	105	n/a	Oscillation	A.14
5	(-0.37, -0.25, -7.96)	OFF	0	81	191	n/a	Oscillation	A.15
6	"	START	0	8	24	n/a	resumed from insertion 3	A.16
7	"	ON	0	4	22	n/a	High Forces	A.17
8	(0.0, 0.0, -12.9)	ON	2	4	18	n/a	High Forces	A.18
9	"	ON	0	3	16	n/a	High Forces	A.19

Table 9.3: Square Chamfered Peg Insertion

During the first insertion the NNC learnt only one new pattern and it was assigned to the X+ direction. When comparing the results in the second insertion, it can be seen that the number of learned patterns and processing time were much higher.

Graphs during insertion 1 and 2 are shown in Figures 9.7 and 9.8 respectively. Compare the number of motions in the Z+ direction in both insertions. This direction was predicted by the NNC in response to the high forces developed in the chamfer and which were not part of the knowledge. The number of motions in the Z+ direction diminished in the second insertion due to the learning of the patterns corresponding to the X+ direction. This behaviour clearly indicated that the robot's compliance for the task had improved.

During the third insertion shown in Figure 9.9, the performance improvement was noticeable. Note that after learning 13 patterns, no alignment motions in the Z+ direction occurred. This implied that the chosen trajectory was optimal.

[2]In the processing time field, n/a stands for not applicable.

The contact forces also decreased from approximately 5 lb during the first and second insertion to 1.5 lb approximately during the third insertion.

Another important observation is that the number of rotations about the Y axis during the first insertion also diminished and in the third insertion no rotation occurred at all. This situation clearly illustrates that at the beginning of the operation the NNC, having no contact information on the chamfer, commanded rotational motions about Y axis. The rotation about Y reduced the constraint forces, however this action was not good enough to be learnt by the NNC. As the expertise of the robot increased, as is seen in the graphs corresponding to the third insertion, the NNC commanded 15 alignment motions from which 11 motions perfectly compensate for the offset. That is, 5 motions in X+ direction and 6 motions in Y+ direction. It should be mentioned that during this alignment, the peg entered into the hole and the remaining alignments were made inside the hole.

Learning inhibition and insertion failure

Learning was inhibited during the 4^{th} insertion (learning state OFF), therefore the PKB was used. Despite the fact that the offset was small, the NNC could not cope with this situation and the system became unstable. This situation is shown in Figure A.14. The values of fz+ were high and needed to be compensated for with a Z+ motion. This compensation effectively reduced the force to 2 lb approximately. However the NNC, with its learning inhibited, repeatedly predicted another motion in Z- direction, which consequently produced a high fz+ making the system unstable.

During the fifth insertion, an angular misalignment was added to observe the robot's behaviour. Keeping the learning inhibited, the robot again failed to make the insertion. Insertions 6 to 9 were carried out with angular offset only and with the learning capability enabled. The robot still failed to perform the insertion due to high forces appearing in the Z direction, eventually, the insertion had to be stopped (see Appendix A for details on these insertions).

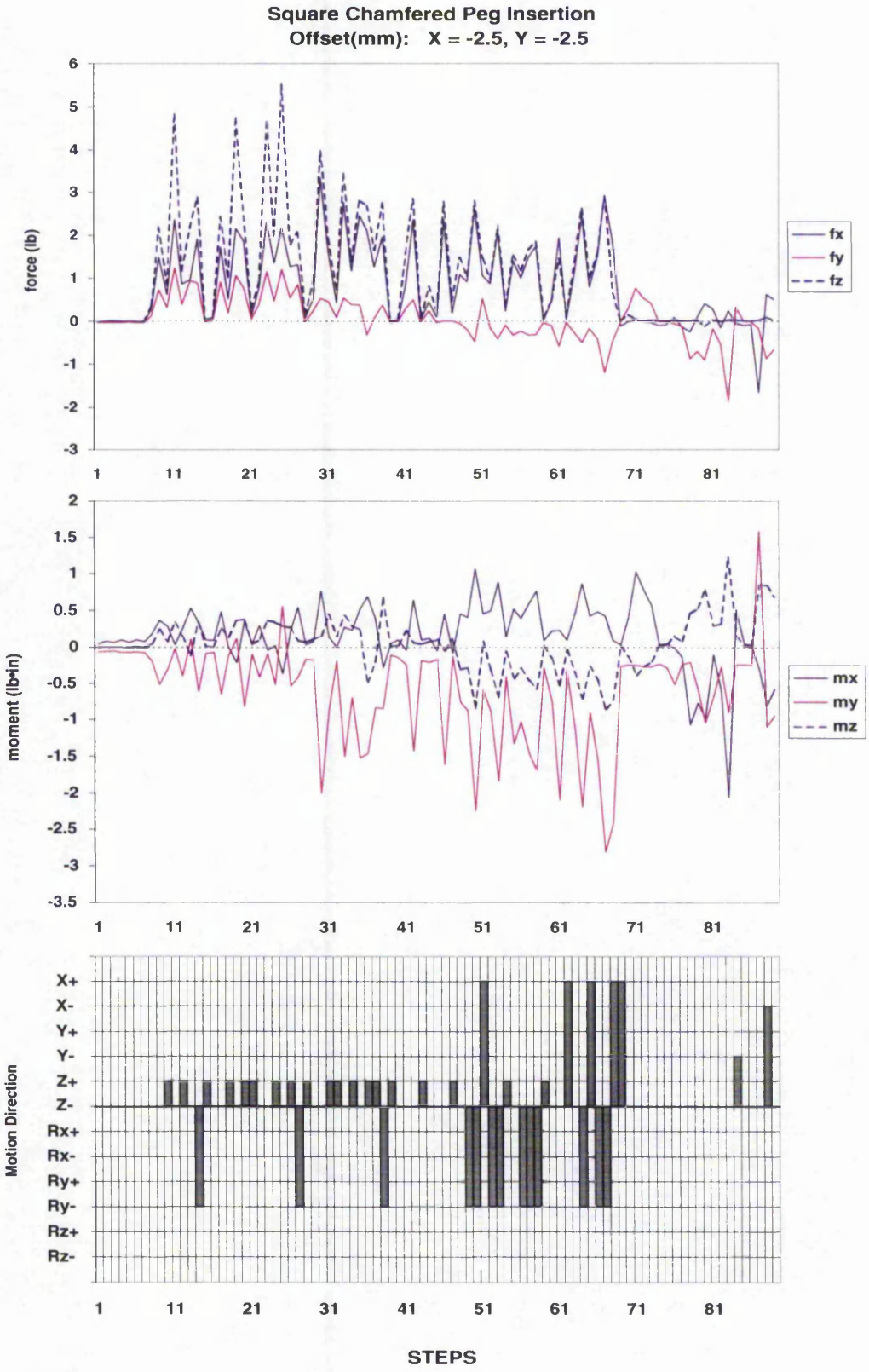


Figure 9.7: First square chamfered peg insertion

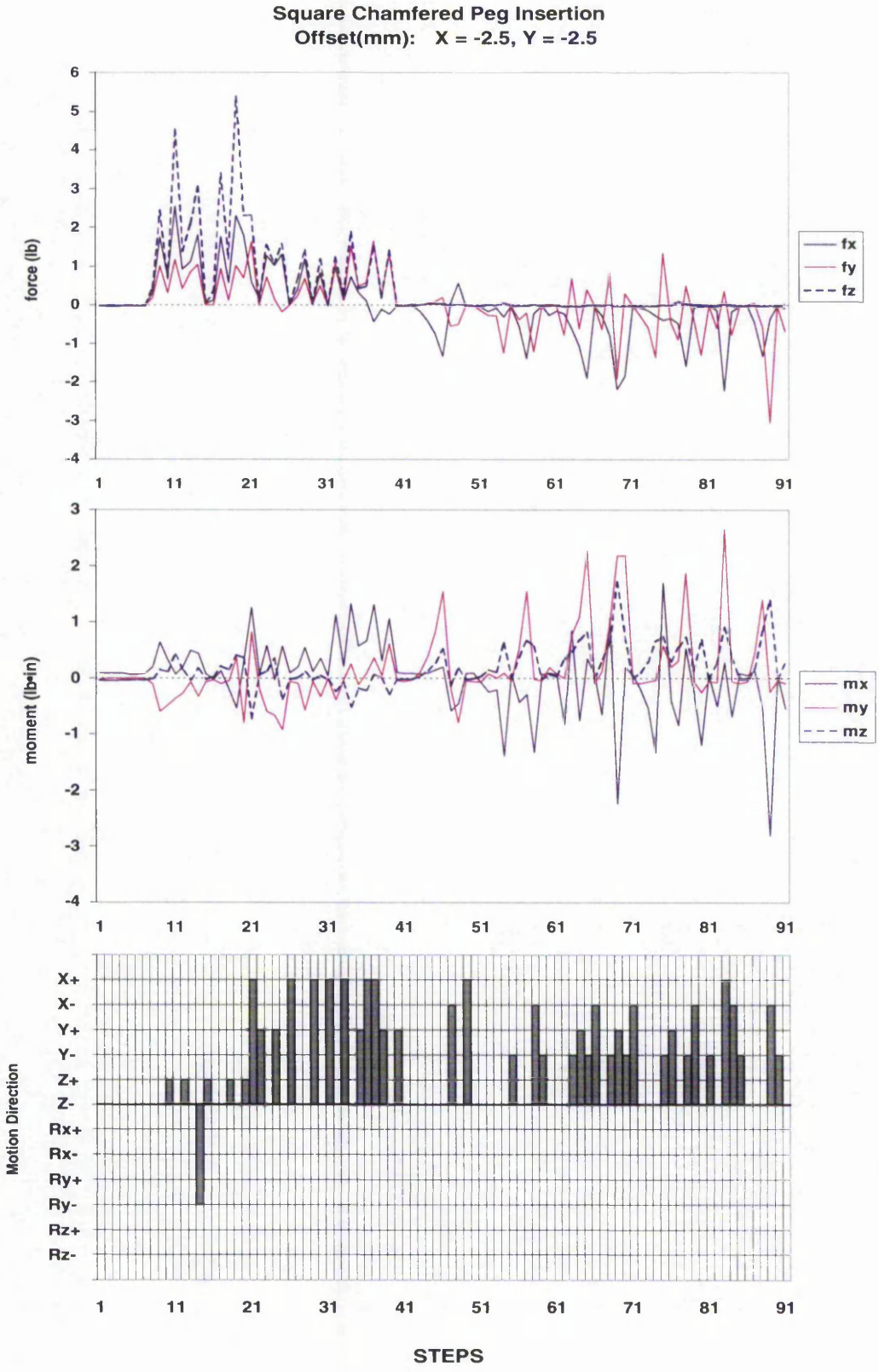


Figure 9.8: Second square chamfered peg insertion

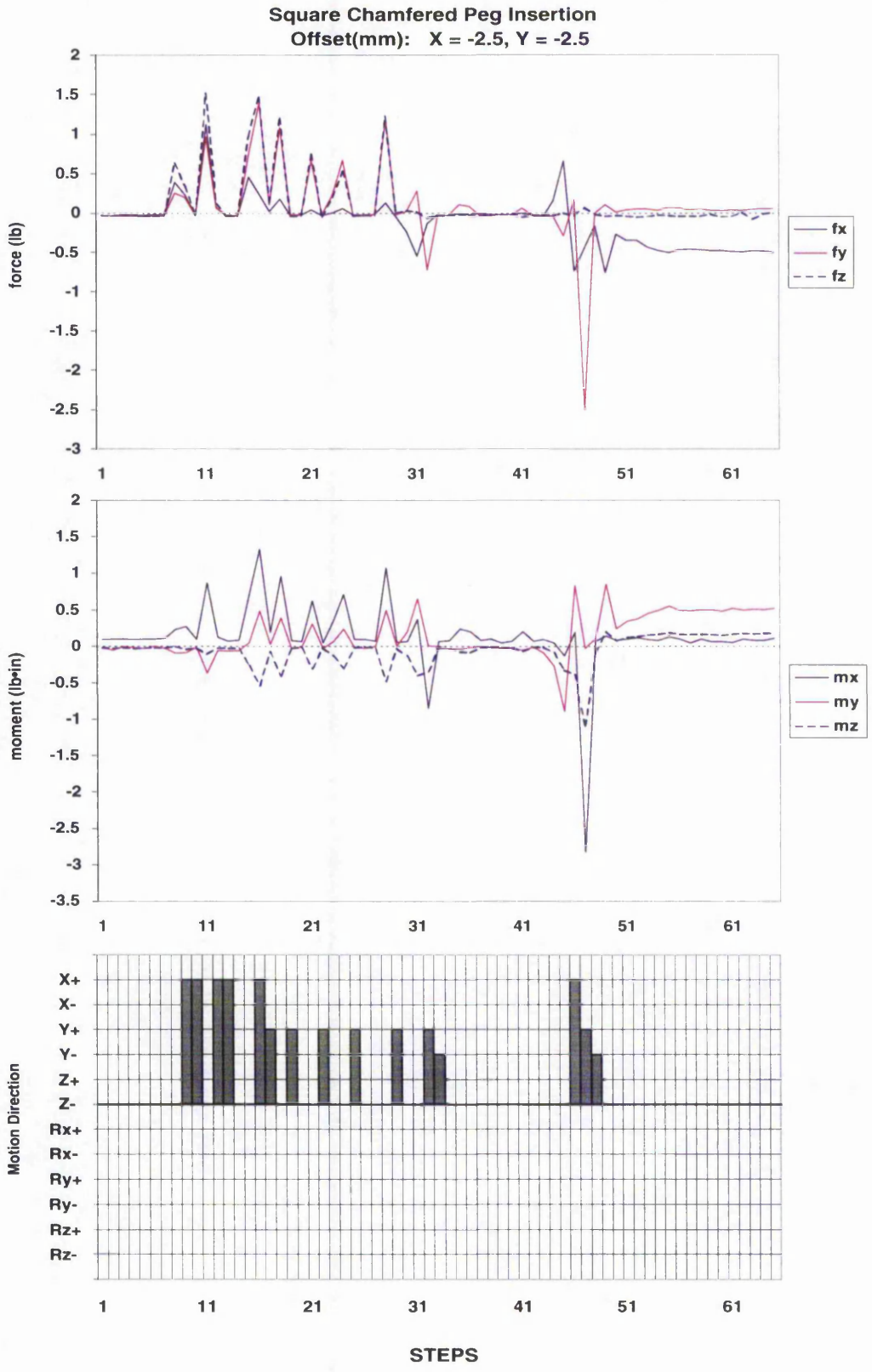


Figure 9.9: Third square chamfered peg insertion

Results also revealed that when the insertion had to be stopped because of the development of high forces, sometimes the followed trajectory indicated the correctness of the motion prediction. From Figures A.16, A.17 and Table 9.3 it can be seen that the peg moved correctly. During insertion A.18 the NNC learnt 2 patterns. The square geometry of the chamfer produced abrupt changes in force that reached the force limit. These results suggest that finer motions are required. It is believed that the robot's positional error may have led to this situation and that additional information within this area may improve the pattern selectivity mechanism in the NNC.

One important conclusion can be drawn from the above results: That is the robot can only make the insertion having its learning capability enabled and using only linear offsets and no angular misalignments. The situation suggests that more information is needed at the start of the operation. Further tests were conducted to investigate this situation and which will be discussed later in Section 9.4.

9.2.3 Radiused-square chamfered peg insertion

The knowledge base for the radiused-square peg insertion was the original PKB used in the previous insertion. Different offset were used as shown in Table 9.4

Radiused-Square Chamfered Peg Insertion

Insertion	Offset (δx, δy, δRz) (mm,mm, °)	Learning	New Patterns	Alignment Motions	Total Motions	Process. time (s)	Comments	Figure
1	(2.44, 0.19, 0)	ON	3	14	64	5.56	OK, Learnt: X-, Y-	9.10
2	(2.44, 0.19, 0)	ON	5	17	67	5.95	OK, Learnt:X-, Y-, X+	9.11
3	(-1.56, 2.38, 0)	ON	2	22	72	6.17	OK, Learnt: Y+	9.12
4	(0, 0, -6.34)	ON	1	18	46	n/a	Osc. Learnt: X+	9.13

Table 9.4: Radiused-Square Chamfered Peg Insertion

The offset used during the first and the second insertion was the same. The corresponding graphs are shown in Figure 9.10 and 9.11, respectively. Although the number of total motions were higher in the second insertion with respect to the first one, the followed trajectory was better. Note that the first alignment motion during the first insertion was a rotation Ry+ when no rotational offset was given about this direction. Luckily, this alignment motion ultimately speed

up the insertion. However, based on the given information, the arm should not have moved in that direction since the offset was given only in the linear direction and along X and Y axes. During the second insertion, no rotation occurred and there were only observed motion alignments to compensate the corresponding offset in the X and Y axes.

After completing both insertions, the NNC have already learnt motions in the X+, X-, and Y- directions. With this new information the NNC is expected to perform better even when different and higher offsets are used. A third insertion was tested to validate this assumption, for which the corresponding graphs are given in Figure 9.12.

During this third insertion the robot performance was satisfactory and the NNC learnt patterns in the Y+ direction.

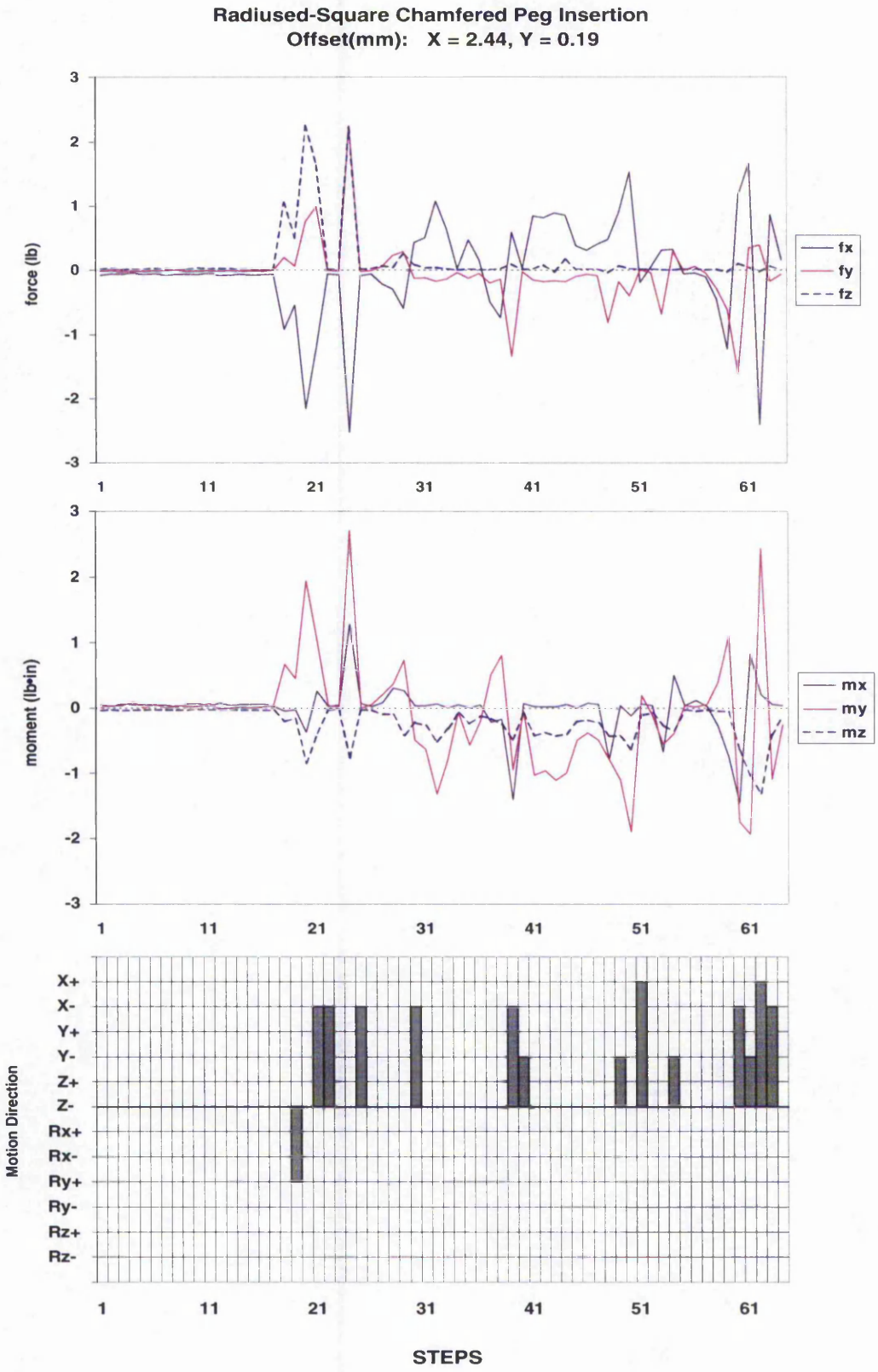


Figure 9.10: First radiused-square chamfered peg insertion

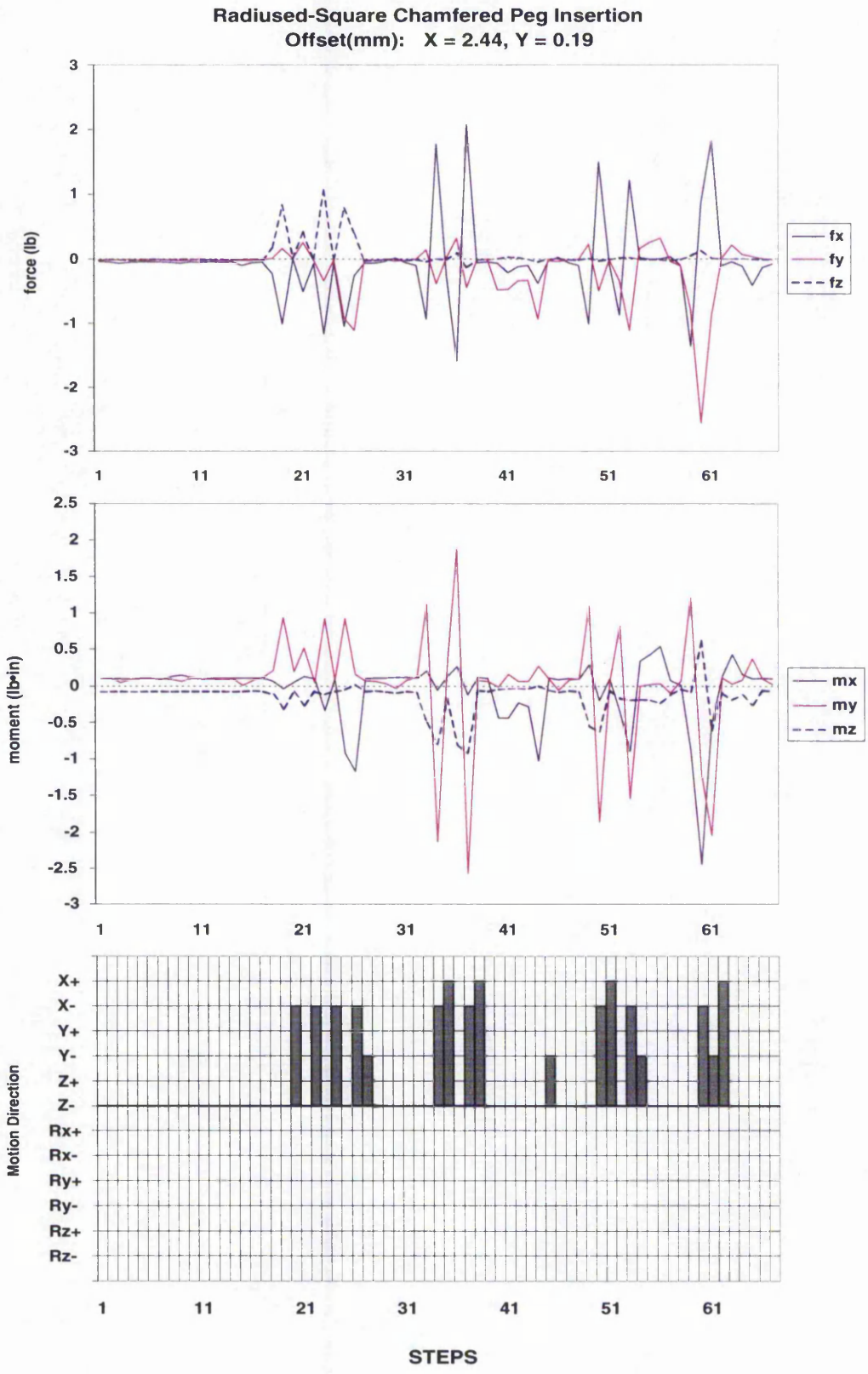


Figure 9.11: Second radiused-square chamfered peg insertion

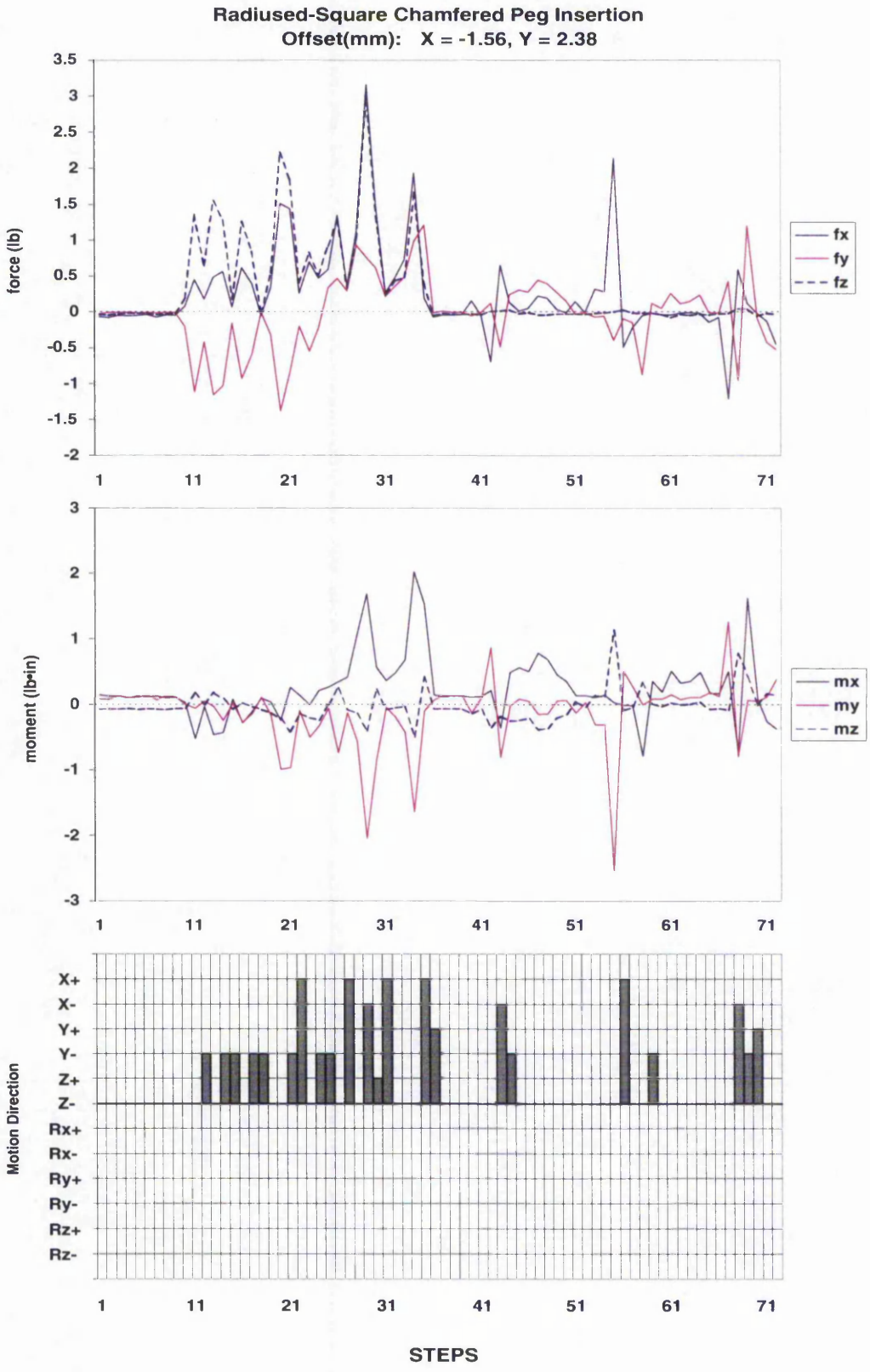


Figure 9.12: Third radiused-square chamfered peg insertion

Insertion failure

Insertion number four looked at compensatory movements for angular misalignment. A rotation of $-6.3°$ about the Z axis was given and the performance observed. The corresponding graphs can be seen in Figure 9.13.

Similar to the case during the square peg insertion, this time the NNC was unable to insert the workpiece since the system entered into an unstable stage produced by Z+ and Z- motions.

One pattern was learnt. The knowledge was identified as a pattern corresponding to the X- motion. However, this knowledge was irrelevant since what it was expected was a rotation about Z. This again identifies clearly the lack of information at the time of contact during angular misalignment and suggests that more information is needed to aid the NNC to identify this state while the corners of the peg contact the chamfer.

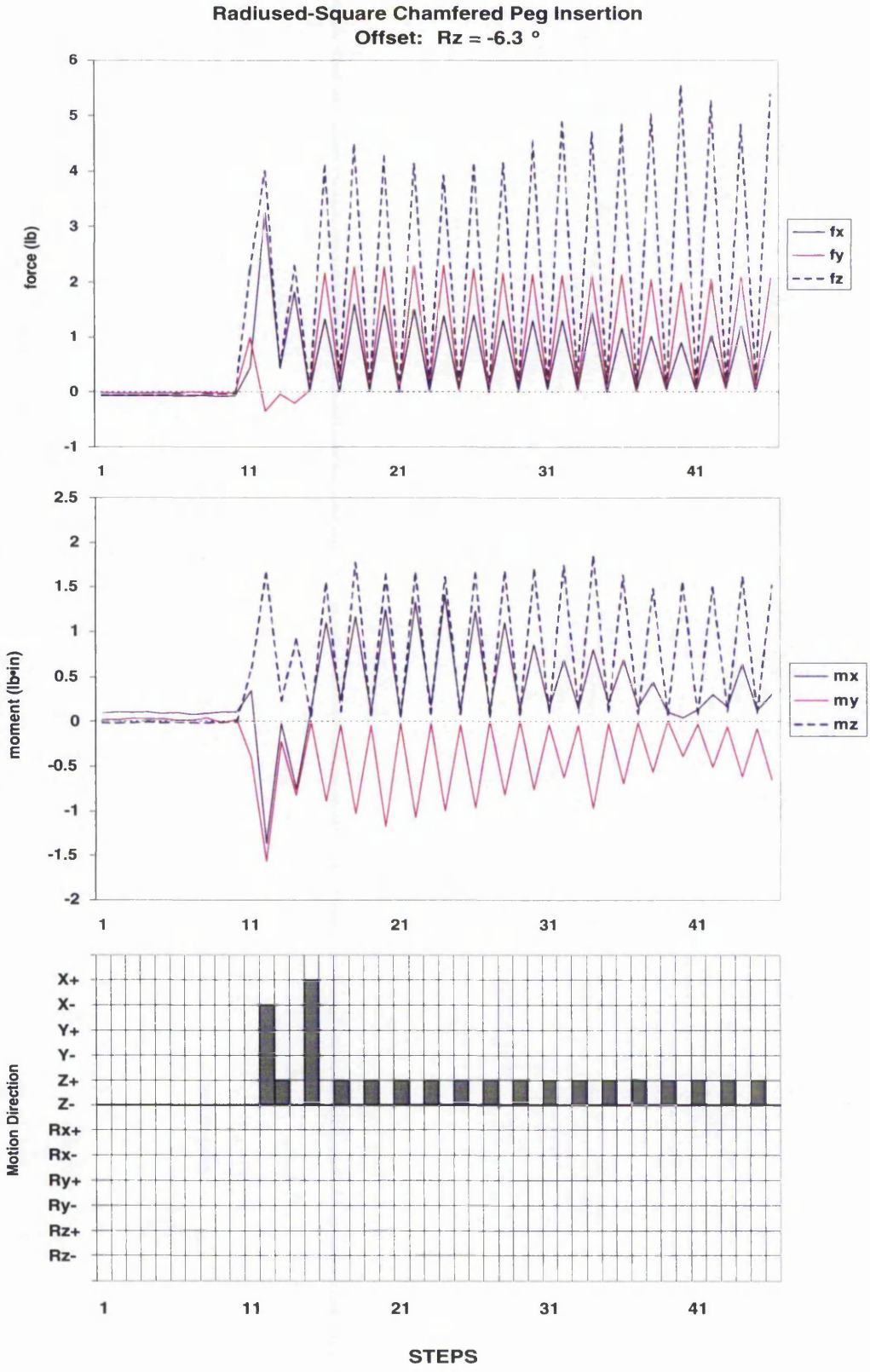


Figure 9.13: Fourth radiused-square chamfered peg insertion

9.3 Chamferless Peg-in-hole Insertion

The use of the chamfer as an aid for assembly resulted very effective in generating the required information for the NNC. In this section, the NNC performance is assessed through the assembly of chamferless parts.

The experience from the previous insertions suggested that eliminating the chamfer in the female component could affect the learning and therefore additional information on the environment would be needed. New information was supplied to the NNC consisting in additional patterns about the rim area of the female components. This is depicted in Figure 9.14(a)

The chamferless circular insertion has been used to illustrate the procedure. However, the same approach was employed for the square and radiused-square geometries.

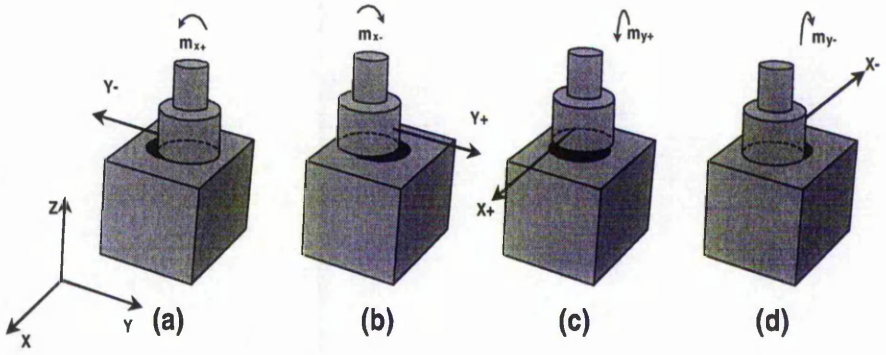


Figure 9.14: Patterns for chamferless insertion

The peg is shown when contacting the rim of the hole. This contact produces a moment with respect to the centre of the peg and about the X axis as indicated. This is a contact state that typically occurs when there is a misalignment at the start of the insertion and for which the NNC should compensate. As it can be seen from the figure, the optimal action to alleviate the contact forces and to move the peg towards the centre of the hole would ideally be a motion in the Y-direction.

The same assumption can be extended to the situation when the peg contacts the rim of the hole in the other three regions producing the mx-, my+ and my- values. This can be seen in Figures 9.14(b), 9.14(c) and 9.14(d). The compensatory

motions are given by the arrow as indicated in each case.

Three data bases were built for each geometry that included the original PKB plus 4 additional patterns that were obtained as explained above.

9.3.1 Circular Chamferless Peg Insertion

The performance during the circular peg insertion was tested with an offset in the X and Y axis as shown in Table 9.5.

Circular Chamferless Peg Insertion

Insertion	Offset (δx, δy, δRz) (mm,mm, °)	Learning	New Patterns	Alignment Motions	Total Motions	Process. time (s)	Comments	Figure
1	(-1.12, -0.13, 0)	ON	6	20	70	6.42	OK Y-(2), X+(4)	9.15
2	(-1.12, -0.13, 0)	ON	1	13	63	5.71	OK Y-	9.16
3	(-1.12, -0.13, 0)	OFF	0	11	23	n/a	High Forces	9.17

Table 9.5: Circular Chamferless Peg Insertion

The first and second insertion were successful and the number of learned patterns decreased from 6 to only 1. The processing time was also lower during the second insertion. The type of learned patterns were as indicated in the comments field. The graphs for these two insertions are shown in Figure 9.15 and 9.16. Note that the trajectory followed during the second insertion was more efficient in terms of the number of alignments.

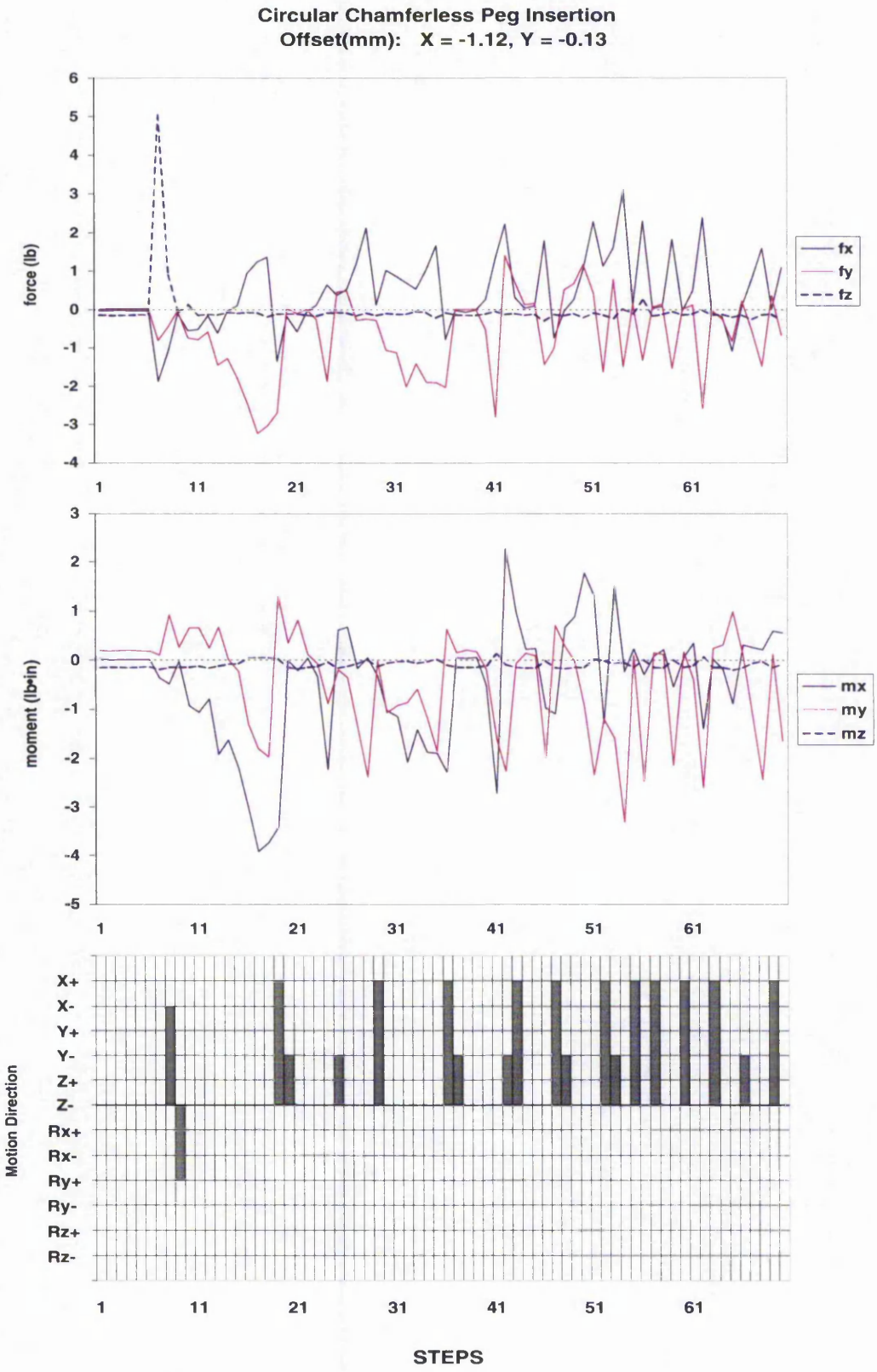


Figure 9.15: First circular chamferless peg insertion

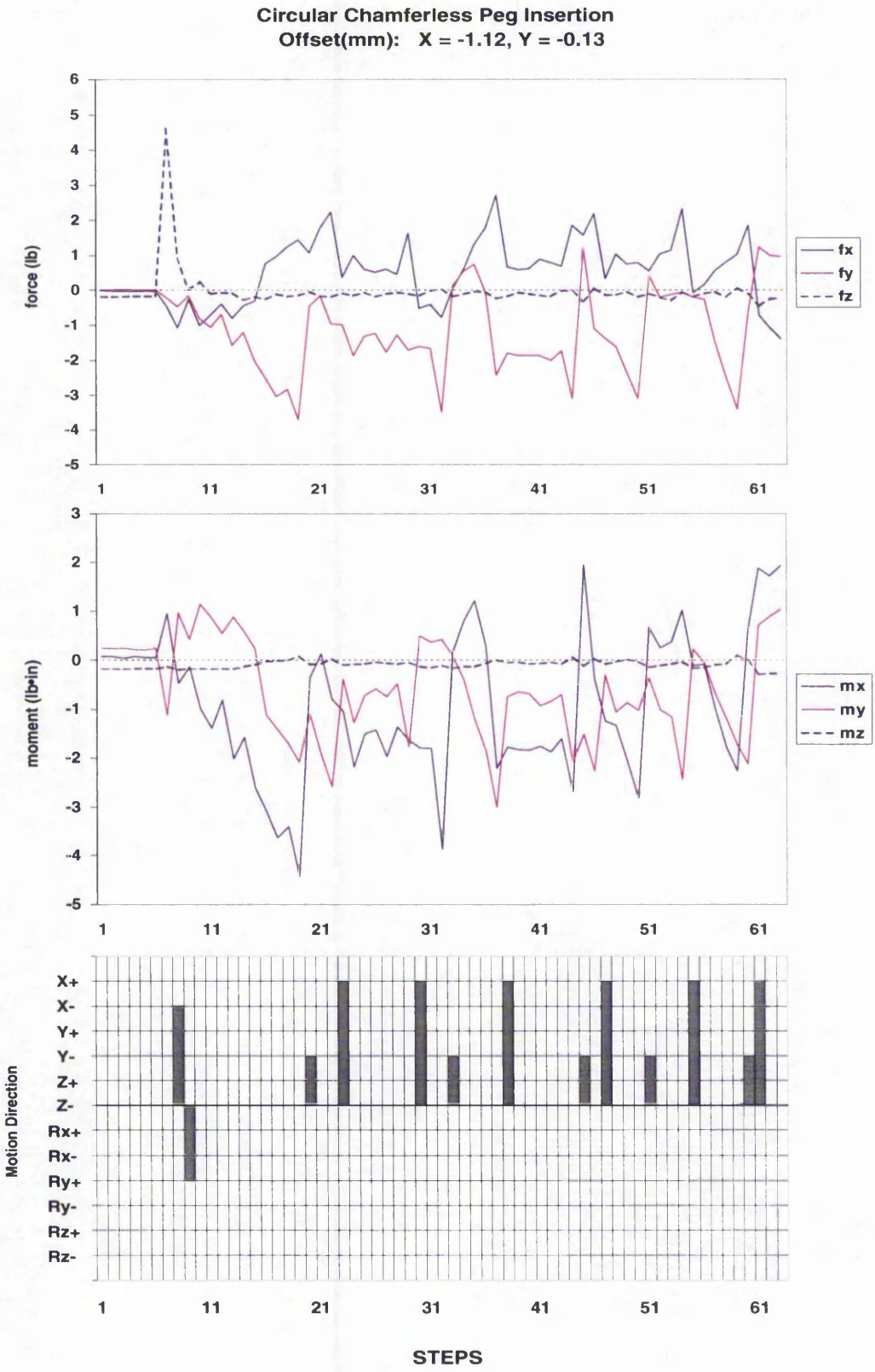


Figure 9.16: Second circular chamferless peg insertion

Please note step motions 8 and 9 in both graphs when the peg first contacts the rim of the hole. At the time of impact, the peg is moved towards X- direction to compensate for the combination of high values in fz+ and the my- and mx+ values. The NNC predicted after this motion Ry+ that made the peg to slid into the hole. The following alignments were all developed inside the hole.

These first set of motions are important to observe since the first expected motion was towards X+ rather than the X- direction. This situation reveals that the peg was not perfectly aligned with the vertical axis and it was slightly tilted. The NNC ultimately coped well with the situation by predicting a motion in X- followed by a rotation Ry+. Further alignment continued inside the hole as expected in the X+ direction. The continued alignment in the Y- direction is attributed to the contribution of the rotation about the Y axis at the start of the operation more than the small offset given in Y (0.13 mm). This offset was very small and approximately 10 times lower than the offset given in the X axis.

A third insertion was carried out using the same offset but this time with the learning capability inhibited. The corresponding graphs are shown in Figure 9.17. The learning state was OFF, therefore the PKB was used. This condition resulted in the insertion being stopped due to high forces during insertion. Once again this fact corroborates the ability of the robot to insert the peg when it is allowed to use its acquired skill.

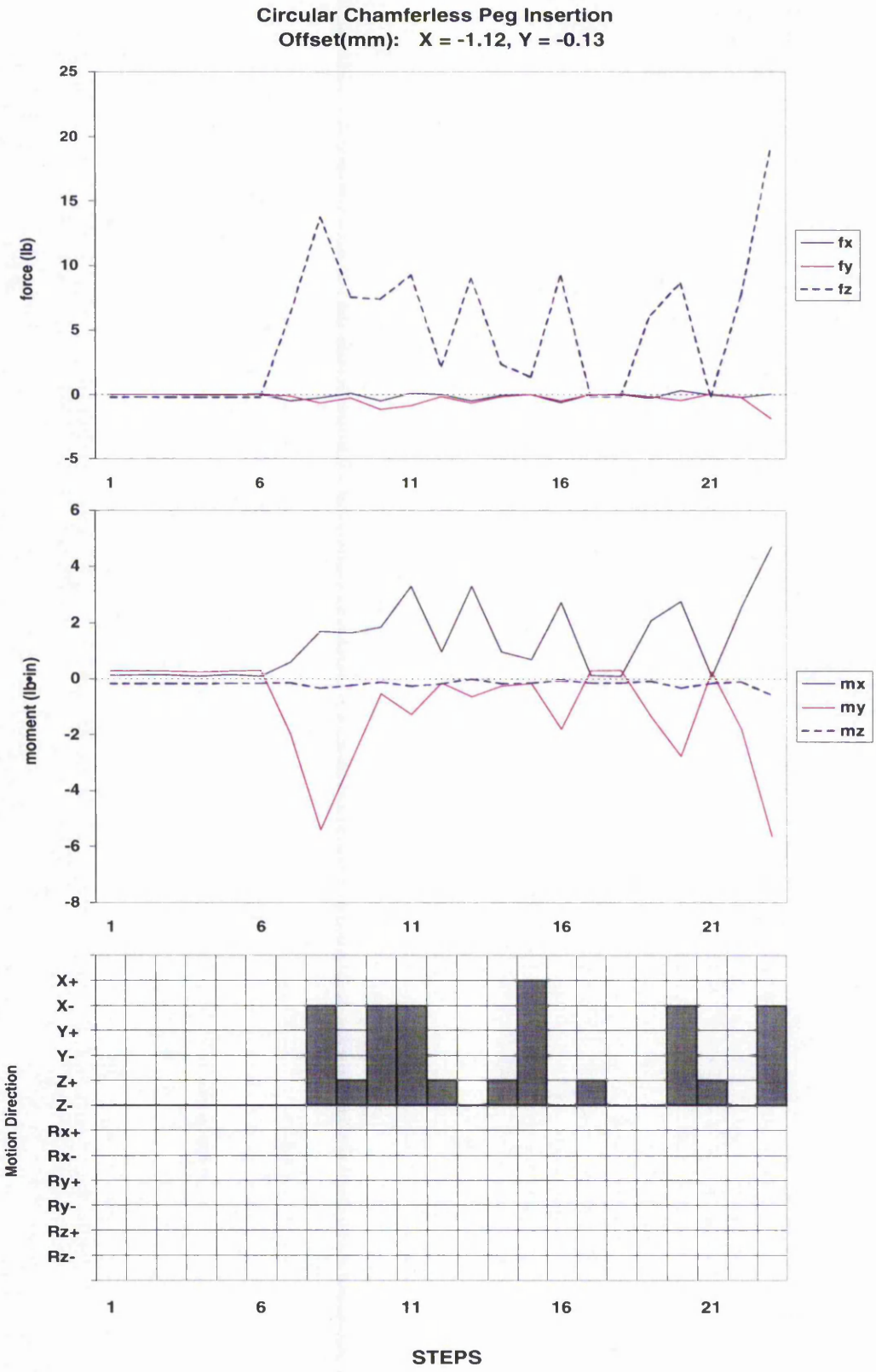


Figure 9.17: Third circular chamferless peg insertion

9.3.2 Square chamferless peg insertion

A specific PKB was used for the square peg insertion that also included information about the edge of the hole in the female component. Four operations were conducted using different offsets, the results are summarised in the Table 9.6.

Square Chamferless Peg Insertion

Insertion	Offset (δx,δy,δRz) (mm,mm, °)	Learning	New Patterns	Alignment Motions	Total Motions	Process. time (s)	Comments	Figure
1	(-0.75, 0.0, 0)	ON	0	7	57	5.37	OK	9.18
2	(-0.25, -0.68, 0)	ON	0	6	56	5.22	OK	9.19
3	(0.94, 1.25, 0)	ON	0	18	68	6.17	OK	9.20
4	(-1.44, 0.75, 3.9)	ON	0	8	18	n/a	High Forces	9.22

Table 9.6: Square Chamferless Peg Insertion

The learning was enabled during the four insertions however no patterns were learnt as can be seen form the results. This suggests that the information provided to the NNC using the PKB was good enough to predict the following motions so that high forces were not developed. It also indicates that the overall forces were not low enough for a given contact state to be learnt within the knowledge base. Details about insertion 1 to 3 can be seen in Figures 9.18, 9.19 and 9.20.

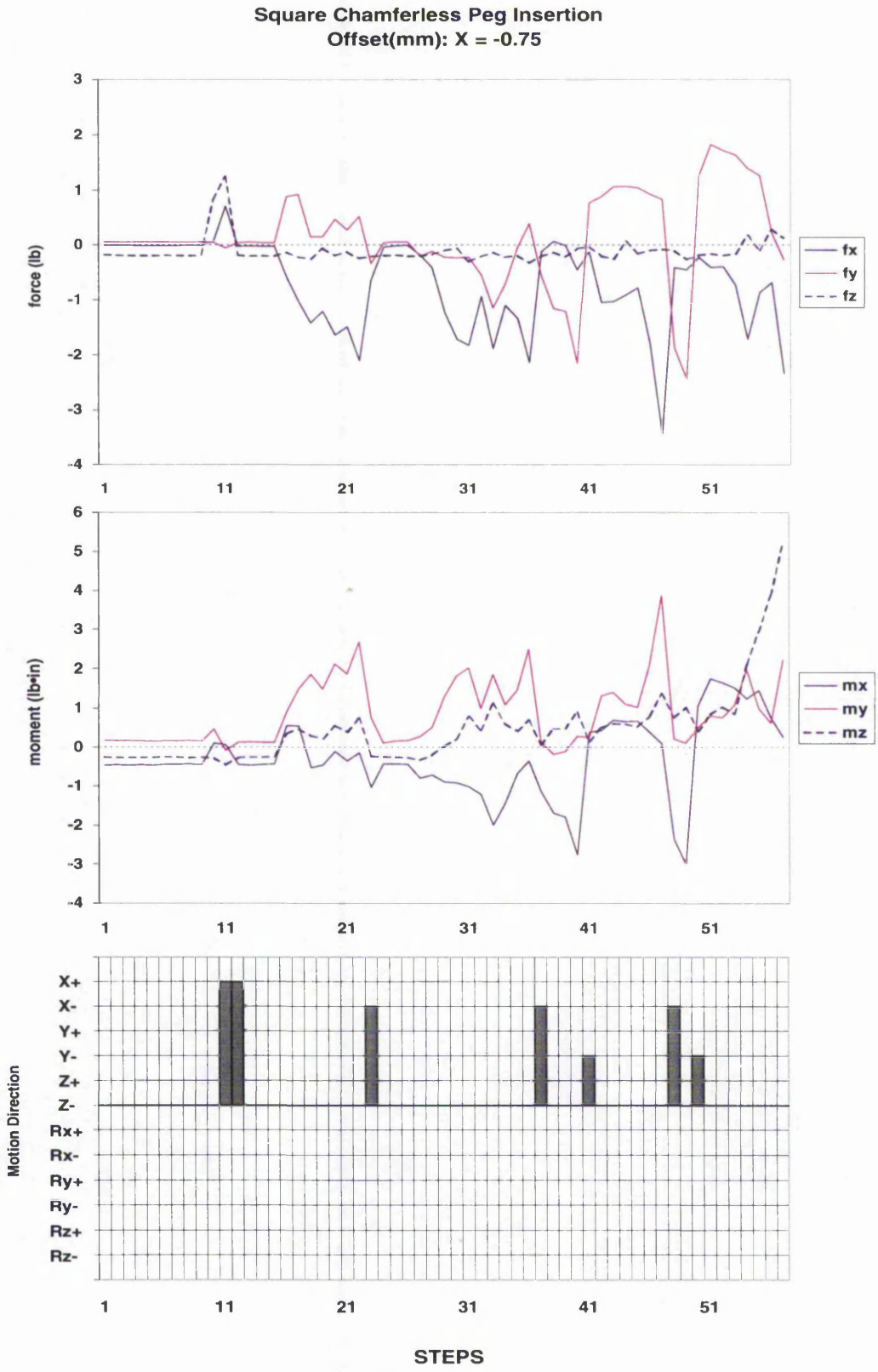


Figure 9.18: First square chamferless peg insertion

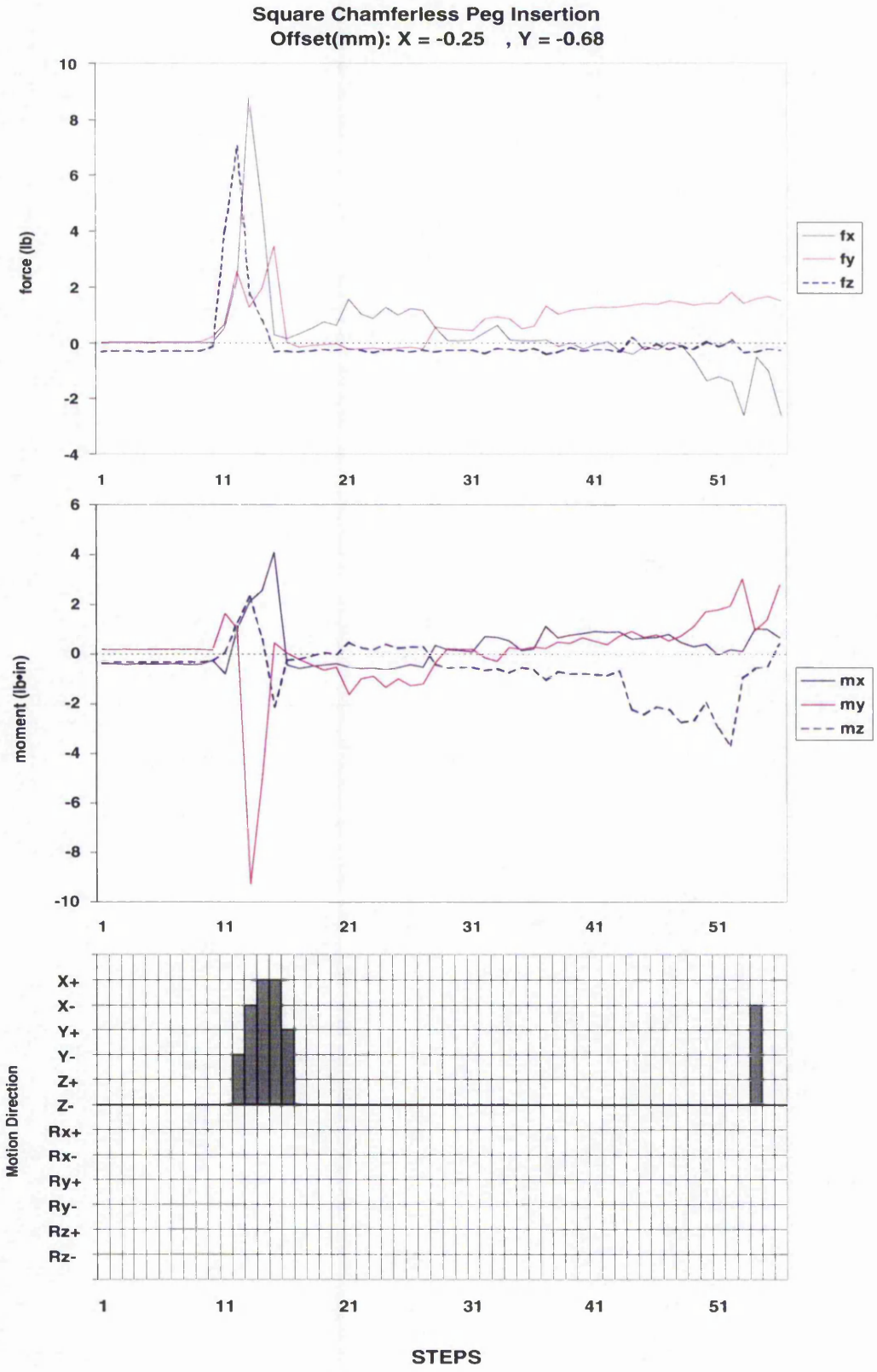


Figure 9.19: Second square chamferless peg insertion

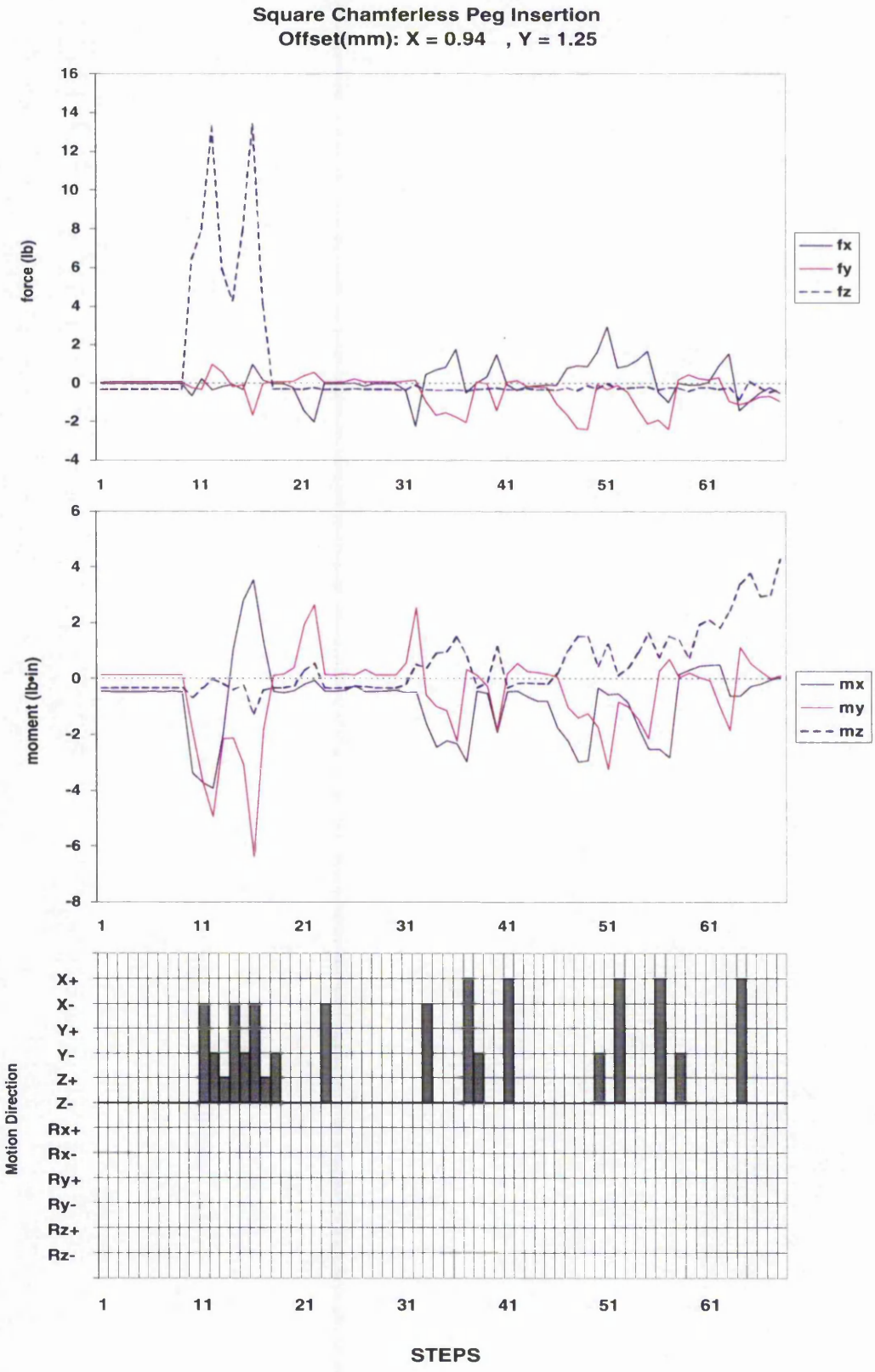


Figure 9.20: Third square chamferless peg insertion

Failure due to rotational offset

An additional insertion was attempted by giving a small linear and angular misalignment, so that one of the corners of the peg lied inside the hole. This is illustrated in Figure 9.21, where a top view of both components is given.

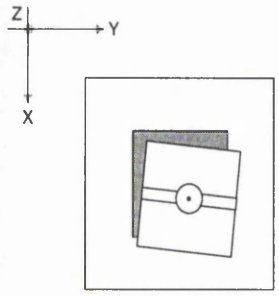


Figure 9.21: Linear and angular misalignment

Results from this operation are given in Figure 9.22. Considering the offset given to the peg, it is reasonable to think about motion in X+, Y-, and Rz- as appropriate motions towards the centre of the hole. It can be observed that the arm moved eight times before the insertion was stopped due to the reach of the force limit. However, the trajectory followed by the peg shows that the arm moved effectively towards the centre of the hole as expected. One out of 8 movements was in the X+ direction and 1 in the Y- direction.

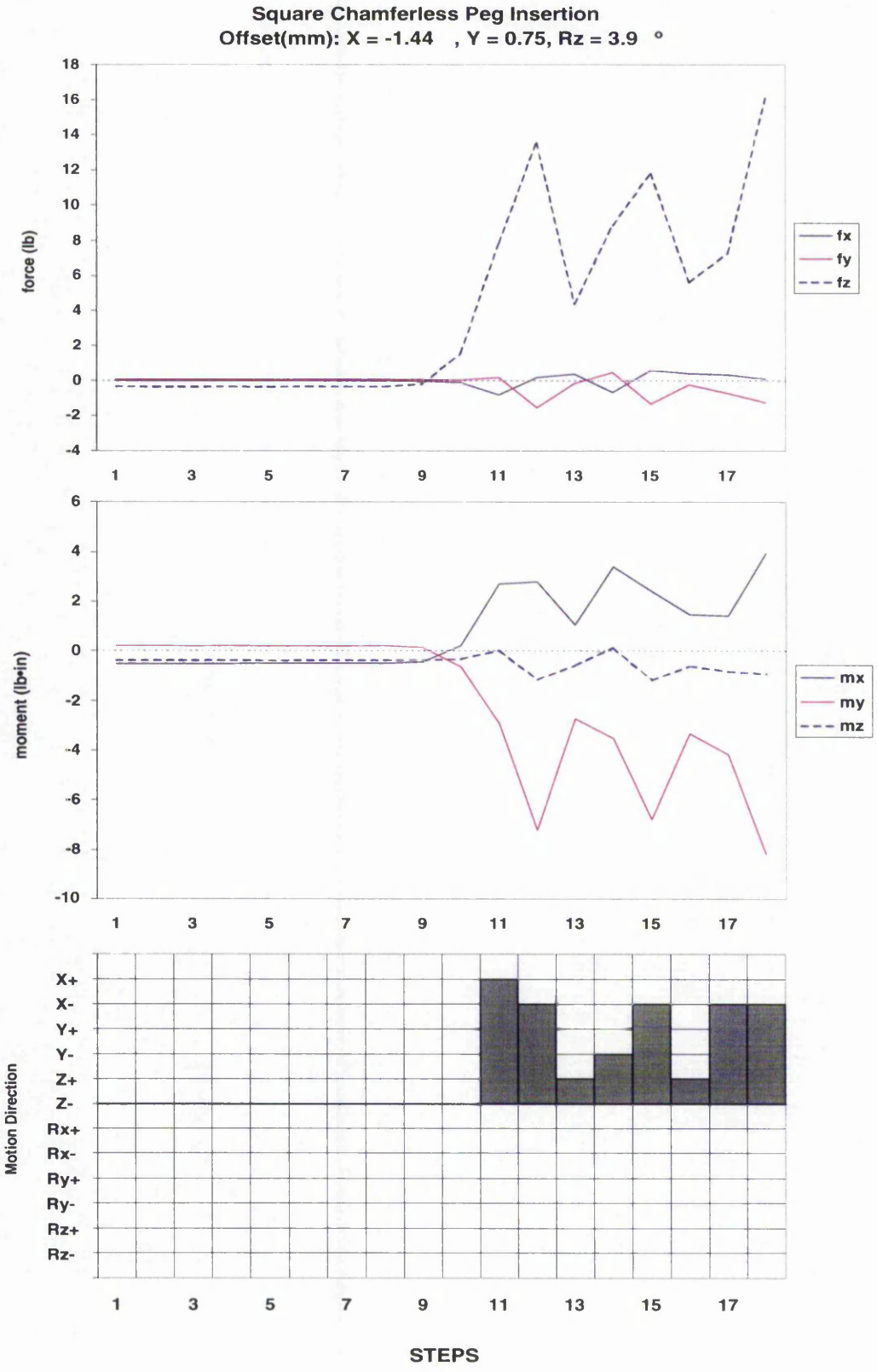


Figure 9.22: Fourth square chamferless peg insertion

9.3.3 Radiused-Square Chamferless Insertion

The NNC performance during the insertion of the radiused-square peg was assessed using a variety of offsets as shown in Table 9.7.

Radiused-Square Chamferless Peg Insertion

Insertion	Offset (δx,δy,δRz) (mm,mm, °)	Learning	New Patterns	Alignment Motions	Total Motions	Process. time (s)	Comments	Figure
1	(1.25, 0.0, 0)	ON	0	9	59	5.08	OK	9.23
2	(-1.38, 0.0 ,0)	ON	0	11	61	5.50	OK	9.24
3	(-0.94, 0.56,0)	ON	0	50	94	n/a	High Force at end	9.25
4	(-0.94, -1.44,0)	ON	1	39	89	8.07	OK, Learnt: Y+	9.26
5	(0.0, -5.0, 0)	ON	0	5	5	n/a	High Force	9.27

Table 9.7: Radiused-Square Chamferless Peg Insertion

Only one pattern was learnt during the operations. An especial case was observed during insertion five, which will be addressed later in this section.

Graphs corresponding to insertions 1 to 4 are given in Figures 9.23, 9.24, 9.25 and 9.26.

The time when the peg fell into the hole is indicated in the Motion Direction graph in each insertion. After this time, all alignments were executed inside the hole. Note that in most of the insertions (except insertion 2), rotational alignment occurred within the hole. Also it is important to note the pick value of the forces at the moment of impact. This value is shown in the respective force graphs.

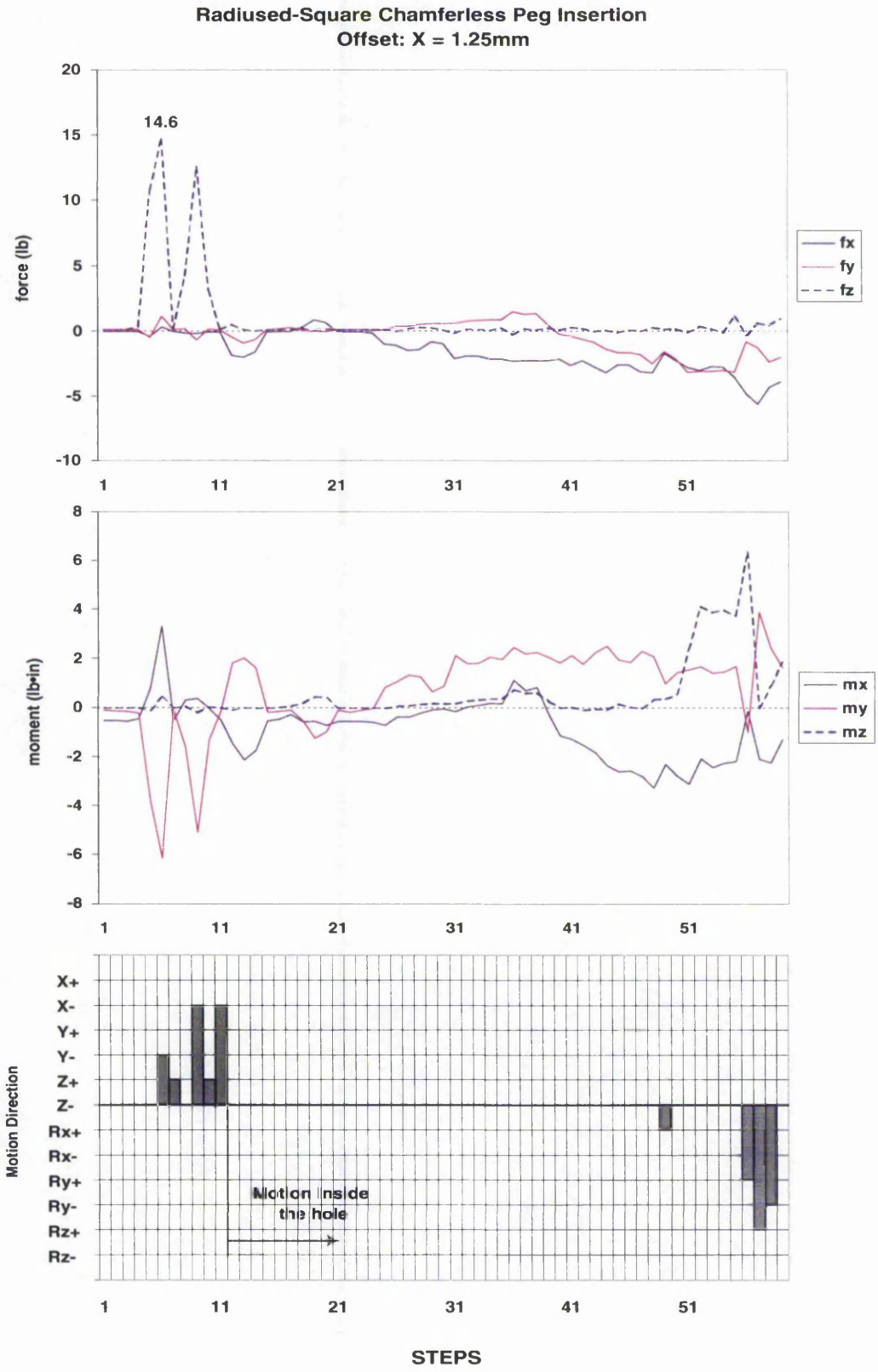


Figure 9.23: First radiused-square chamferless peg insertion

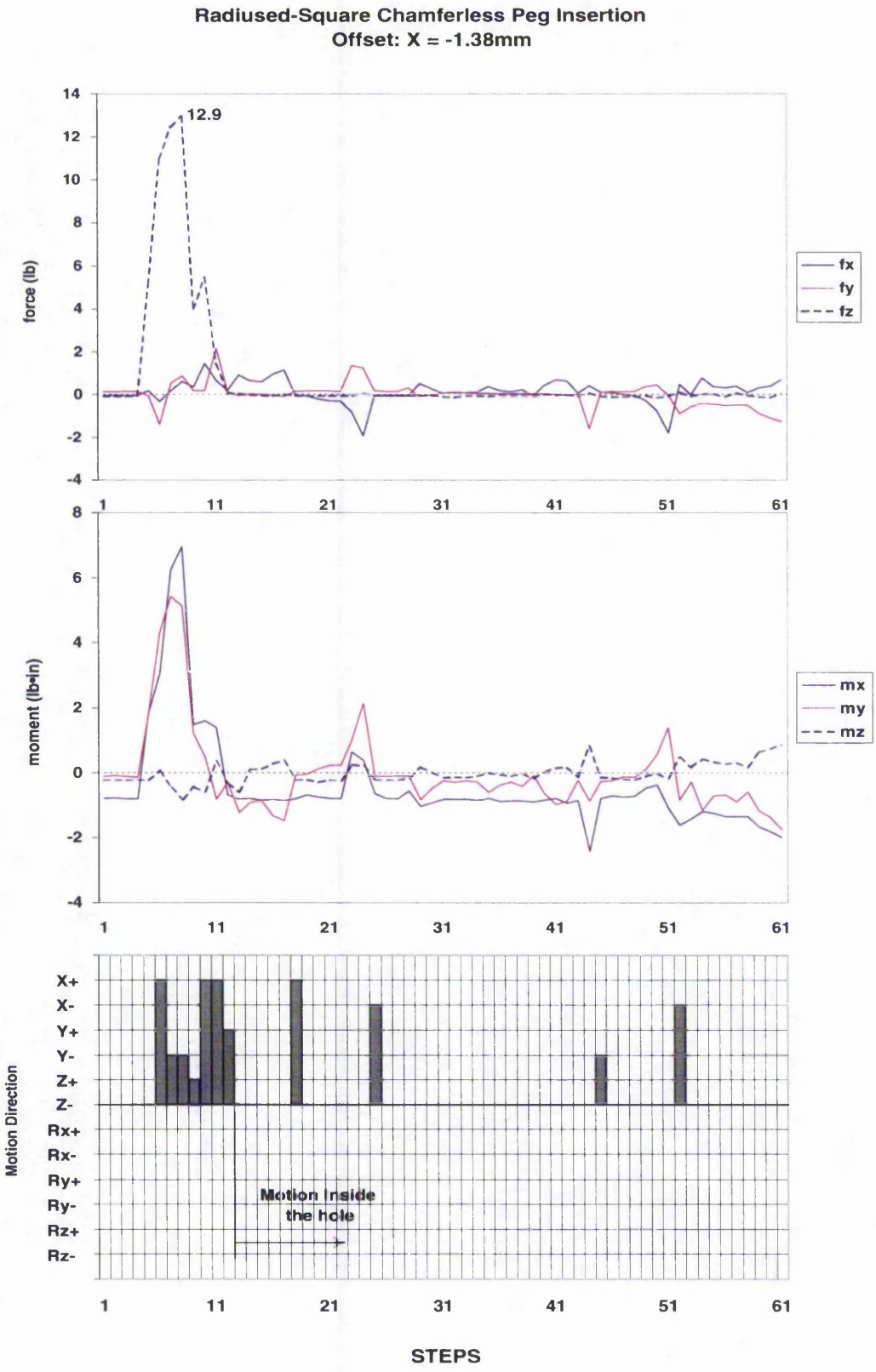


Figure 9.24: Second radiused-square chamferless peg insertion

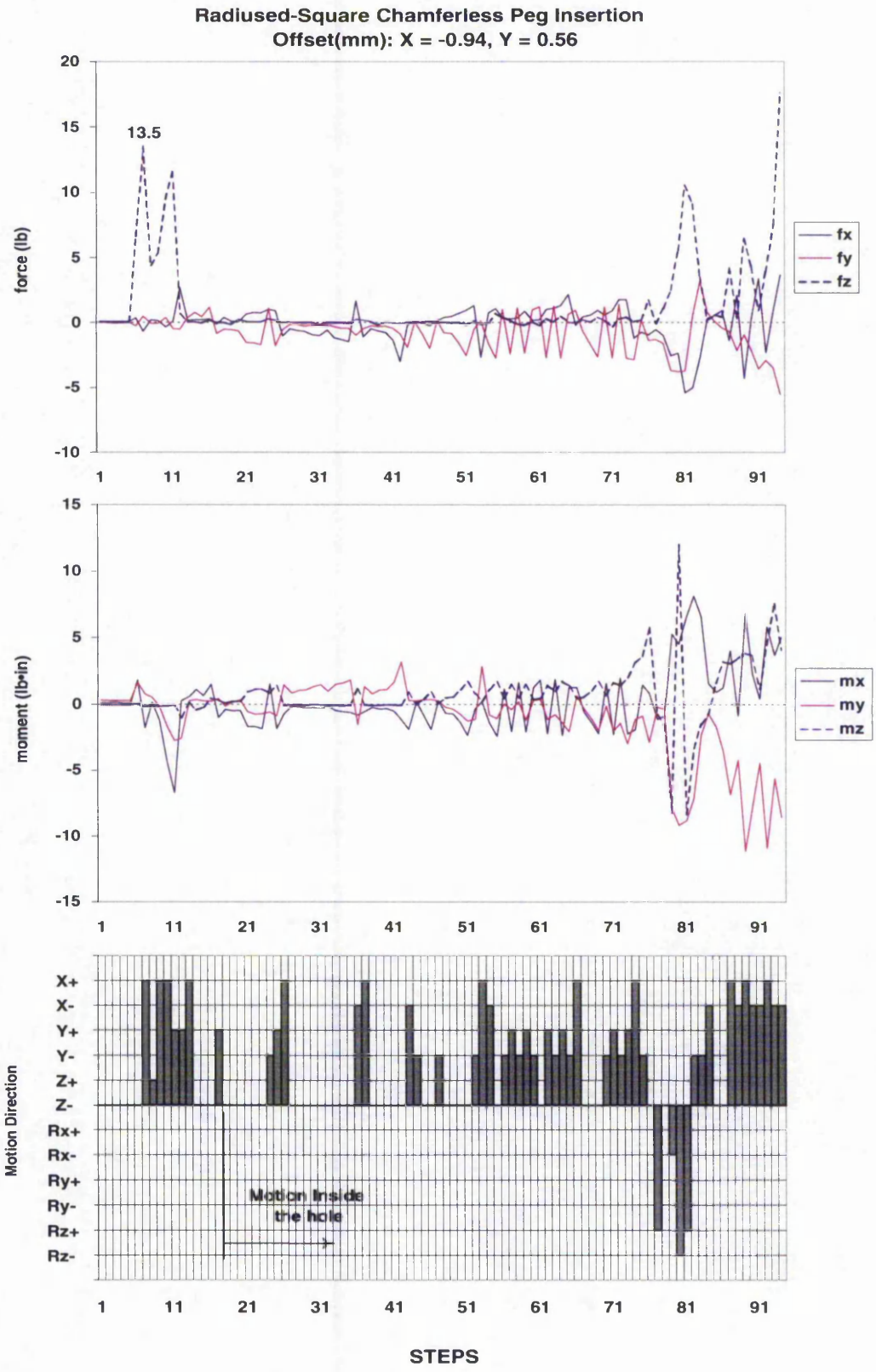


Figure 9.25: Third radiused-square chamferless peg insertion

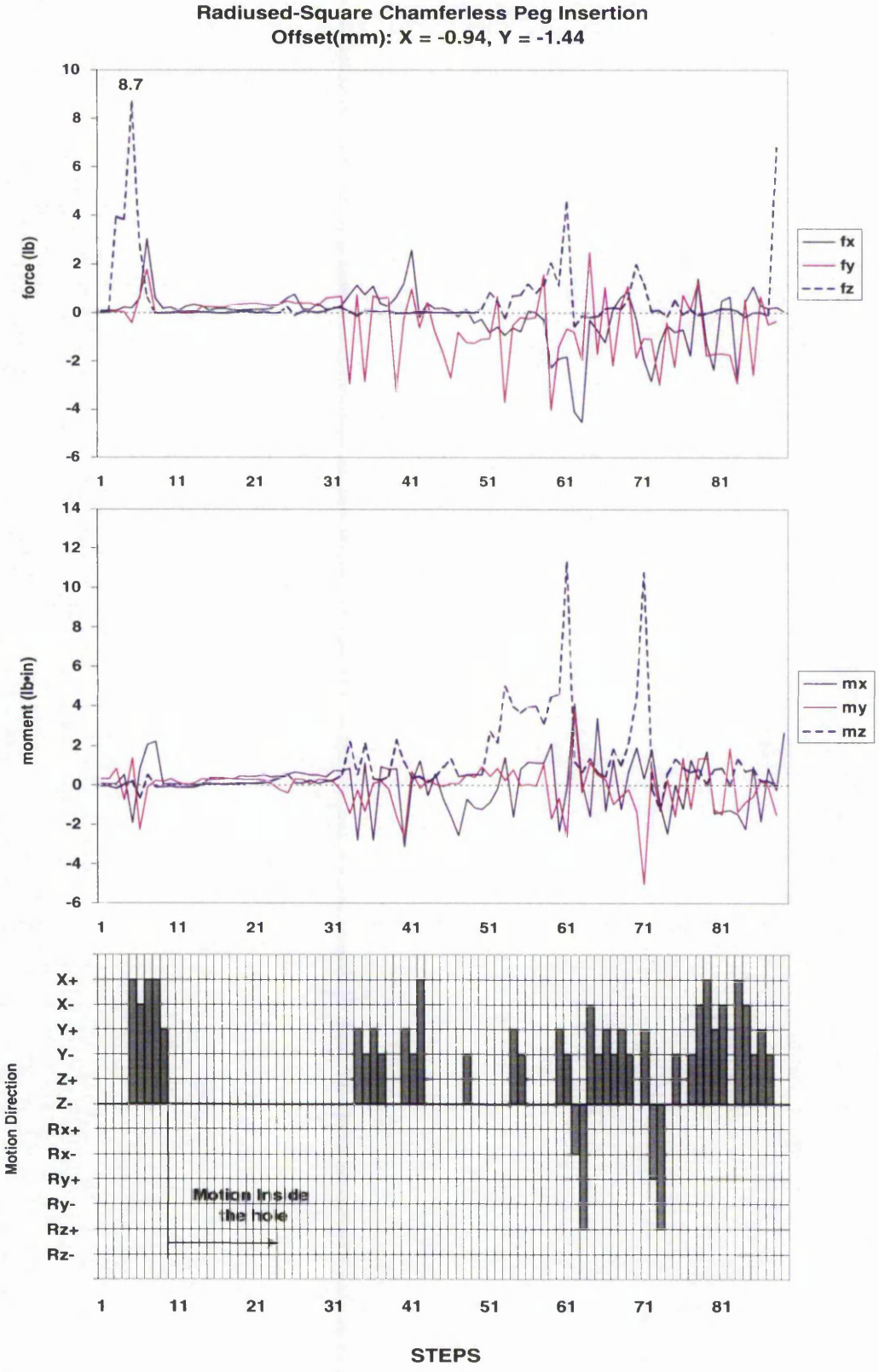


Figure 9.26: Fourth radiused-square chamferless peg insertion

Ambiguous situation

An ambiguous situation occurred during insertion number four. The NNC learnt one pattern and this pattern was identified as a pattern corresponding to the Y+ direction. The pattern was learnt when the peg fell into the hole and it was a peculiar situation. The learning of this pattern did not affected further behaviour since this was very similar to a the pattern previously stored in the PKB, so that it did not altered the robot's behaviour.

Handling higher offsets

The offset given to the peg at the start of the operation has been about ten times the value of the clearance. That is, 1 mm offset for a clearance of 0.1 mm. Several combinations have been used about that range and the overall behaviour resulted to be satisfactory.

An additional insertion with a higher offset was tested. The offset was given only towards the Y axis with a magnitude of 5 mm. The corresponding graphs are shown in Figure 9.27.

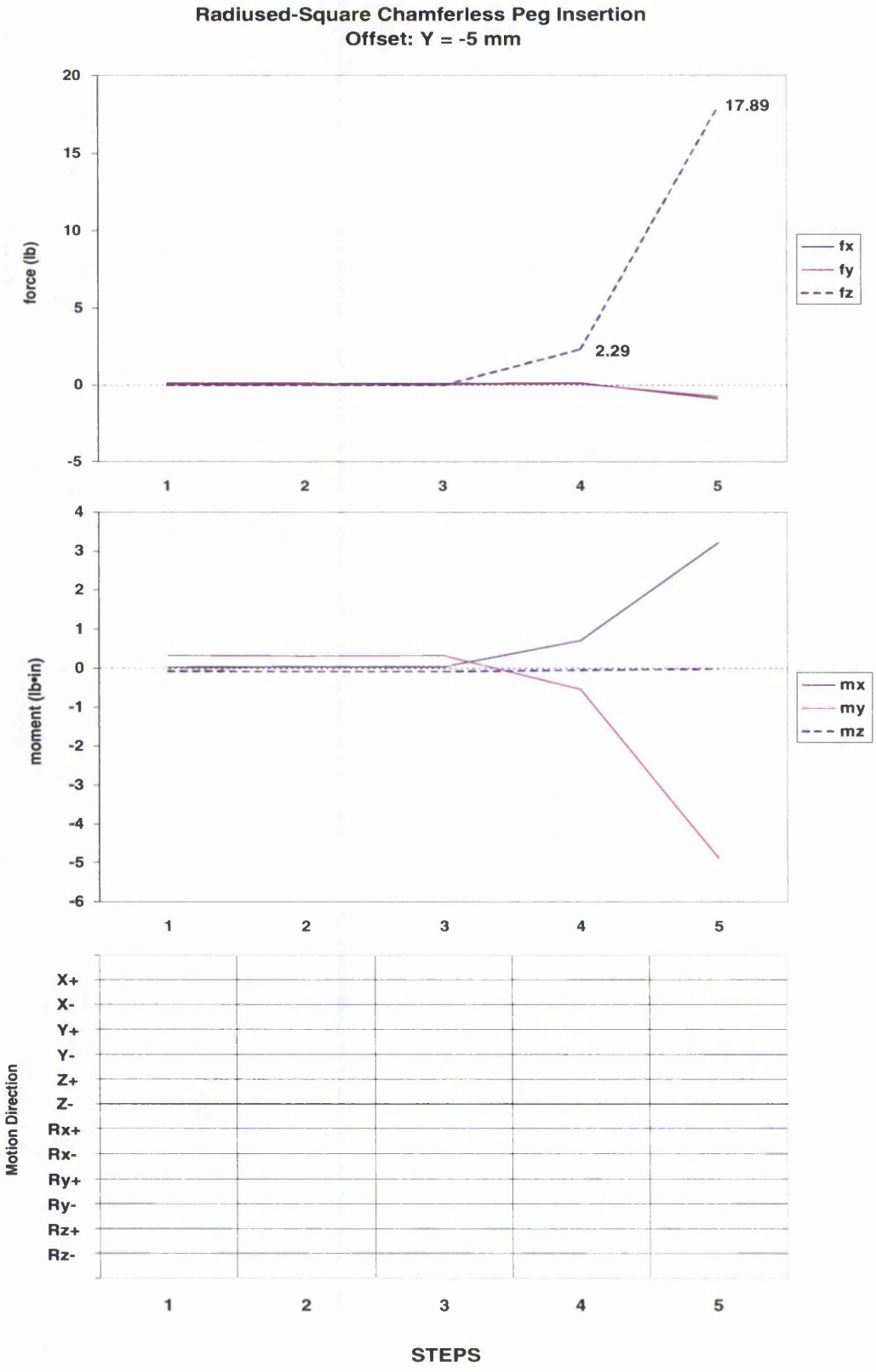


Figure 9.27: Fifth radiused-square chamferless peg insertion

The insertion had to be stopped due to the high swing in force that appeared in the Z direction and reached the force limit. Several issues can be addressed in regard of this result.

First of all, it is evident that the farther the offset from the centre of rotation, the higher the fz component vector and vice versa. This is illustrated in Figure 9.28.

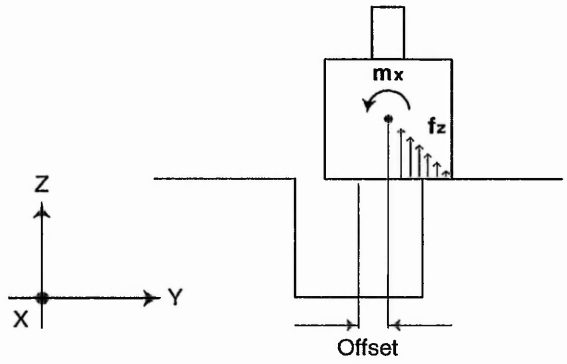


Figure 9.28: fz and offset relationship

With a higher offset the value mx is proportionally lower and therefore is less likely that the NNC predicts a motion in Y-. The NNC may predict a motion in the Z+ direction to reduce fz. With the current working parameters the system is able to cope with offsets of approximately 1.5 mm.

It is worth remembering that the current development is expected to be integrated into an intelligent cell with vision ability. The general idea is to use visual feedback to deal with gross misalignments to approach and align the peg to the hole. Once the peg is vertical and aimed at the hole, it will only be required force feedback for fine motion alignment.

9.4 Further Tests

Results demonstrated the ability of the NNC to compensate for linear positional errors. Results also showed that the manipulator failed to insert the square and radiused-square pegs when an angular misalignment was given. These facts suggested that for the manipulator to compensate for these errors additional

information should be included in the PKB, i.e. information within the chamfer. Eight patterns were added to the PKB to cope with the above situation. This is illustrated in Figure 9.29 showing a top view of the peg making contact at eight different points within the chamfer.

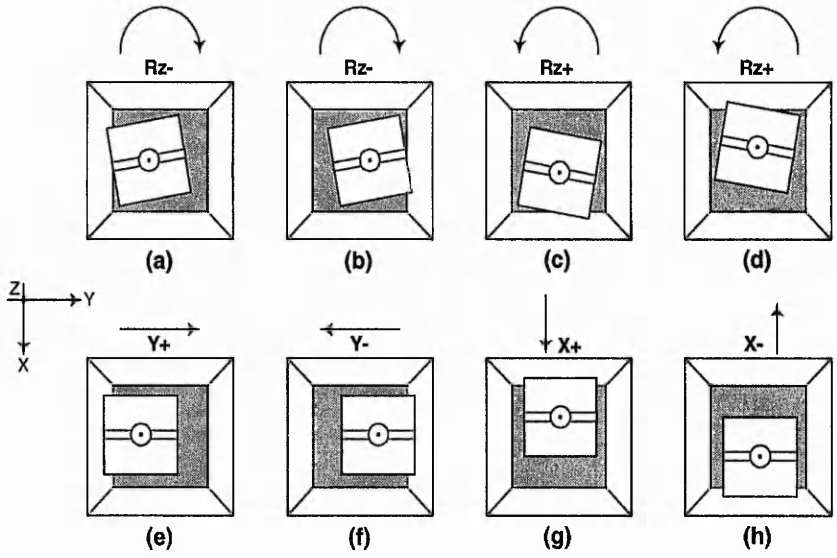


Figure 9.29: Teaching additional information

Note that the female component has been enlarged to highlight the point of contact. The compensatory actions are angular actions in the case of Figure 9.29(a) to 9.29(d) and linear motion for contacts represented in Figures 9.29(e) to 9.29(h). The sign of the rotation was assigned following the right hand rule respect to the reference coordinate frame. The linear motion sign was given using the same coordinate frame.

This knowledge about the chamfer was given explicitly to the NNC. The eight patterns were added on top of the previous PKB. For this reason this new knowledge base will henceforth be named as PKB'.

A series of insertions were conducted with the PKB' with the purpose of observing the rotational capability and learning of the NNC. A summary of the results is given in Table 9.8. It was decided to test both the radiused-square and square peg insertion using the same PKB' and test the NNC generalisation properties.

Particular details for each insertion can be consulted in the corresponding Figure in the Appendix A as indicated in the Table.

Radiused-Square Chamfered Peg Insertion									
Insertion	Offset (δx,δy,δRz) (mm,mm, °)	Learning	New Patterns	Alignment Motions	Total Motions	Process. time (s)	Comments		Figure
1	(0.0, 0.0 -1.0)	START	2	30	80	7.49	OK,	X+, Y-	A.20
2	"	ON	2	21	71	6.51	OK,	X+(2)	A.21
3	"	ON	2	35	85	7.93	OK,	Rz+, Y-	A.22
4	"	ON	0	24	74	7.01	OK,		A.23
5	"	ON	2	25	75	7.11	OK,	Y+, Rz+	A.24
6	"	ON	0	25	75	7.20	OK,		A.25
7	"	ON	1	15	65	5.93	OK,	X+	A.26
8	"	ON	0	27	77	7.04	OK,		A.27
9	"	ON	0	24	74	7.00	OK,		A.28
10	(-1.06, 0.0, -1.0)	ON	1	32	82	7.81	OK,	Rz+	A.29
11	"	ON	0	33	83	7.37	OK,		A.30
12	"	ON	1	51	101	9.17	OK,	Rz+	A.31
13	"	START	2	42	92	8.43	OK,	X+, Y-	A.32
14	"	ON	2	49	99	8.77	OK,	X+, Y-	A.33
Square Chamfered Peg Insertion									
15	"	ON	4	44	94	8.91	OK,	X+(2), Y-(2)	A.34
16	"	ON	0	40	90	7.97	OK,		A.35
17	"	ON	1	44	94	8.63	OK,	Rz+	A.36
18	"	ON	0	40	90	8.38	OK,		A.37
19	"	ON	0	34	84	7.93	OK,		A.38
20	(-1.06, 0.0, -1.0)	START	3	24	74	7.26	OK,	X+, Y-(2)	A.39
21	"	ON	2	38	88	8.84	OK,	Y-, Rz+	A.40
22	(1.1, 0.0, -3.4)	ON	2	55	105	10.61	OK,	X+, X-	A.41
23	"	START	4	46	96	10.12	OK,	X+(3), X-	A.42
24	"	ON	2	55	105	12.91	OK,	X-, Rz+	A.43
25	"	ON	2	54	104	10.71	OK,	Y-, Rz+	A.44

Table 9.8: Rotational offset results

Insertions 1-9 —Angular misalignment

A small misalignment of $-1°$ was given to the peg and the insertion was successful. It can be seen from the Table 9.8 that the trend in the number of the learned patterns during insertions diminished and eventually they were zero at insertions 8 and 9.

In terms of processing time, it can be concluded that this time is directly related to the chosen trajectory during insertion. Note that the best insertion was made with only 15 alignment motions and corresponded to the 7^{th} insertion.

An analysis of the motion types for all these insertions revealed that a brief oscillation occurred along the insertion path, mostly in the Y+ and Y- directions. That is, the arm moved back and forth between these two directions. These oscillations delayed the movement towards the end-condition and consequently the insertion times were longer. Note the oscillatory behaviour did not appear during the 7^{th} insertion. It is clear from these observations that the NNC was gaining more information about the geometry and it ultimately predicted the right motion. The slight change in the force traces during repeated contacts made the NNC to pick the right motion direction exiting the oscillatory period. Also note that during the first insertion, the contact forces were high and in the

rest of the insertions the forces were substantially reduced, which demonstrates the dexterity acquired by the robot.

Insertions 10-12 — Angular and linear misalignment

An additional offset was given in the X direction and the same angular misalignment applied. The robot's behaviour was acceptable and the number of alignment motions increased as expected. Two patterns corresponding to the Rz+ direction were learnt.

Insertions 13 & 14 — Restarting the learning

The learning was started again so that the acquired knowledge during the previous insertions was discarded in order to observe the robot's behaviour. The PKB' was used and the same offset applied. This time, as expected, the NNC learnt more patterns as it did in the insertions 10-12 however it learnt patterns corresponding to the X+ and Y- directions instead in the Rz+ direction. Note that these type of patterns (X+, Y-) were also learnt at the very first insertion. Also note that the forces during the first stages of insertion in these two operations were much higher, which demonstrates the dexterity which the robot had acquired from previous operations before restarting the learning.

Insertions 15-19 — Same EKB different geometry

The operations were carried out using the square mating pair. From the Table 9.8 it can be seen that four patterns were learnt during insertion 15 and this figure reduced to zero at insertion 18 and 19. This implies that the NNC acquired most of the required knowledge for the square geometry during the first insertion. The overall performance was satisfactory.

Insertions 20 & 21 — Restarting the learning

The learning was restarted by using the PKB' and letting the NNC form its own knowledge about the square geometry.

Insertions 22-25 — Increasing the angular misalignment

The offset was changed from (-1.06, 0, -1) to (1.1, 0, -3.4) and the results were acceptable. The learning was also restarted at insertion 23. After this, the increase in the number of patterns was repeated again and the assemblies were carried out successfully.

9.4.1 Comments on these results

Important conclusions can be drawn from the above experiments.

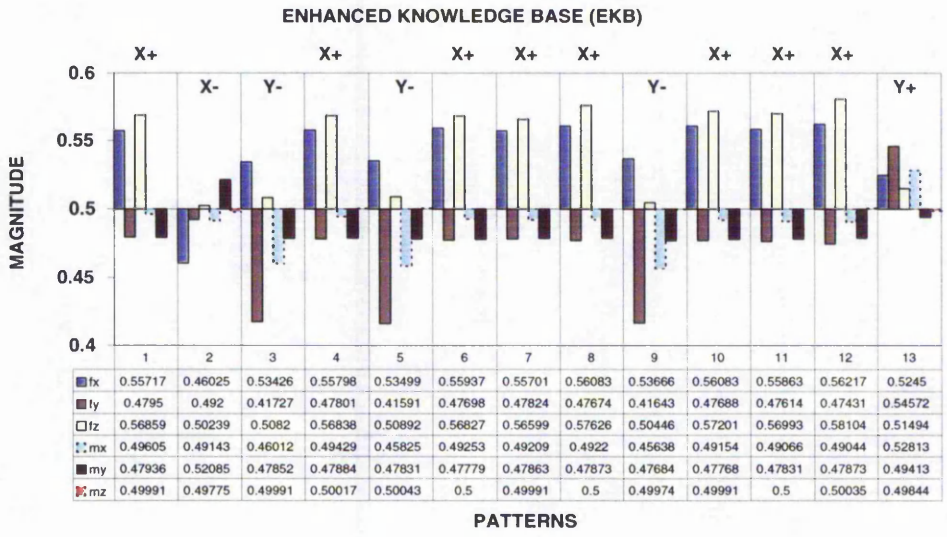
- The NNC can operate under small angular misalignment during chamfered insertions provided that the PKB contains information about the chamfer area.
- The robot enhances its dexterity most during operations in the early stages of learning.

9.5 Discussions

9.5.1 Density of data and knowledge acquisition

The capability of generalisation and knowledge acquisition of the NNC has been demonstrated. Patterns that reduce significantly the contact forces during manipulations were acquired into the knowledge base and learnt. A representative learning example was shown in Section 9.2.1 with the circular chamfered insertion. In this example, the network was initially trained with the PKB containing the 12 possible patterns associated with the robot's 6 DOF. This information biased the initial learning by creating 12 categories to allocate every possible motion direction. From these results, it was verified that subsequent patterns corresponding to contact states within the chamfer were effectively allocated into these categories. However, the pattern population within certain categories produced high density of data within regions in the feature space.

During the chamfered circular peg insertion only four patterns were learnt. These patterns corresponded to the X+, X-, Y+ and Y- (see Table 9.2). The new patterns were valuable to speed up the insertion and to improve the insertion trajectory as it was shown during the test. However, these patterns were present within the data more than once and a total of 13 patterns were acquired after 14 insertions which implied that certain categories were more populated. This can be appreciated in Table 9.30 that shows the nature of learned patterns.



	1	2	3	4	5	6	7	8	9	10	11	12	13
fx	0.55717	0.46025	0.53426	0.55798	0.53499	0.55937	0.55701	0.56083	0.53666	0.56083	0.55863	0.56217	0.5245
fy	0.4795	0.492	0.41727	0.47801	0.41591	0.47698	0.47824	0.47674	0.41643	0.47688	0.47614	0.47431	0.54572
fz	0.56859	0.50239	0.5082	0.56838	0.50892	0.56827	0.56599	0.57626	0.50446	0.57201	0.56993	0.58104	0.51494
mx	0.49605	0.49143	0.46012	0.49429	0.45825	0.49253	0.49209	0.4922	0.45638	0.49154	0.49066	0.49044	0.52813
my	0.47936	0.52085	0.47852	0.47884	0.47831	0.47779	0.47863	0.47873	0.47684	0.47768	0.47831	0.47873	0.49413
mz	0.49991	0.49775	0.49991	0.50017	0.50043	0.5	0.49991	0.5	0.49974	0.49991	0.5	0.50035	0.49844

Figure 9.30: Learned patterns during the circular chamfered insertion

As it can be seen, patterns belonging to the same category were very similar. The patterns corresponding to the X+ direction were allowed to be learnt 8 times. This implied that the contact forces were significantly reduced in eight occasions. This high number of patterns populated more the feature space in that area. In the feature space this is represented in Figure 9.31 as follows:

For simplicity, only four major areas of action (X+, X-, Y+ and Y-) are represented. Initially, the main groups are formed, this is represented by the big black dots as illustrated at the beginning of the operation using what has been termed PKB. The smaller dots represent additional patterns that have been clustered within the same major region. As it is observed, the region belonging to the X+ direction was more populated than in the others. The high density of data only implies that there are more data in the region and the cost is memory space.

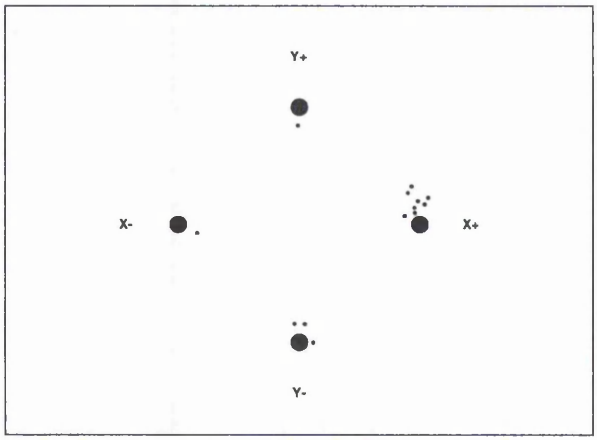


Figure 9.31: Data density

However, since the criteria to learn new patterns was the condition given by the expression $F_{after} < 0.1 * F_{before}$ in Chapter 8, then as the learning progresses, a reduction in contact forces is expected, as it was demonstrated during the experiments, since the robot became more skillful. Being this statement true, it is also true that the knowledge acquisition becomes more strict. This obeys to the fact that forces are smaller as the robot is more skillful and from the above expression forces have also to be smaller to be accepted into the EKB.

Also, as the robot's dexterity improved, the trend in the number of patterns that were accepted into the EKB decreased as it was shown in the corresponding Tables. The above expression for allowing the patterns to be learnt resulted to be a criterion to stop automatically the learning.

With this reasoning in mind, it can be demonstrated that the density does not corrupt the selectivity of the NNC, but only affects the memory resources to allocate the learned patterns.

9.6 Conclusions

The results presented in this Chapter demonstrate the acquisition of assembly skills by the robot. The *expertise* was demonstrated in a number of situations under different working conditions. The robot's behaviour has shown its dexterity by assembling parts faster using the acquired new knowledge, namely, the EKB. If

the robot's learning capability is inhibited then the same task is still accomplished however, the time the robot takes to complete it is longer as more alignment motions are required.

The generalisation capability of the NNC is demonstrated by allowing the robot to assemble parts with different cross-sectional geometry.

Assembly failures were also identified. To compensate for these failures the PKB was increased with appropriate information about the chamfer of the female component.

The findings clearly define the capability of the assembly system and its limitations. To overcome such limitations additional sensory information is needed. Current research is being undertaken in the areas of object recognition and language understanding with potential use in the intelligent assembly cell.

Chapter 10

Conclusions

The aim of this research was to provide robots with self-adapting capability so that assembly operations could be performed in poorly structured environments, i.e. under extreme uncertainty. It should be noted however that real-world operations have an innumerable range of contact conditions. To deal effectively with this variety of conditions, a requirement for learning was identified. This requirement provided the inspiration for this research and the desire to create intelligent robots for autonomous assembly using on-line incremental learning.

With regard to the above, it is suggested that the research has been largely successful, as it was demonstrated by the implementation of the assembly strategy into the robotic system. The results which report the performance of the Neural Network Controller (NNC) and validate the approach were presented in Chapter 9.

Many issues had to be addressed before an on-line learning capability could ultimately be demonstrated by the robot. The design methodology followed a bottom-up approach which consisted of two fundamental aspects:

The first was an accurate control of the manipulator as well as an effective evaluation of its actions. This stage involved the design of a host-slave computer system for real-time control and the implementation of the force sensing system.

The second aspect was embedding the necessary intelligence into the robotic system so that the robot could cope with extreme uncertainty, learn from experience and improve its assembly dexterity.

10.1 What was achieved

Robot force control

The inclusion of the slave computer in the control system enhanced the overall performance by separating the low level communication task from the host computer. This strategy provided more time to the host computer for data processing and avoided the necessity of interrupting its processing by the robot controller for data requests during the robot's ALTER mode. The communication task was delegated completely to the slave computer. Additionally, with this host-slave architecture, force can be monitored continuously during arm motions. It should be mentioned however, that the force during incremental motions was monitored only after the motion was completed. This facilitated the assessment of its contribution to the assembly. The capability was not fully exploited since the force information was evaluated after completing the compensatory motion. However, the architecture can also be used effectively in other industrial applications, for instance, in metal removal processes.

Learning and skill acquisition

The dexterity of the manipulator and its capability to acquire assembly skills was demonstrated in a number of situations. The overall approach consisted of the design of a novel Neural Network Controller (NNC) based on the Adaptive Resonance Theory (ART) and in combination with a dynamic knowledge base, whose knowledge was regulated by the assembly tasks.

The capability of ART networks to learn incrementally was assessed early during the research using the ART-1 network, which demonstrated that the new contact conditions occurring in an assembly could effectively be learnt without any degradation of previous knowledge. Furthermore, the learning was achieved in a single epoch, i.e. data presented once to the network. These findings suggested the appropriateness of the algorithm to be implemented in the NNC.

The NNC was enhanced by the use of the Fuzzy ARTMAP network and a Primitive Knowledge Base (PKB). The PKB was obtained by commanding the manip-

ulator against a rigid body utilizing the robot's 6 DOF. The information retrieval was straightforward. The information was necessary at the start of the operations to bias the initial learning in the manipulator.

The *expertise* of the manipulator was demonstrated in a number of assembly situations. This was demonstrated by inhibiting the use of its acquired expertise in the Enhanced Knowledge Base (EKB) (See Table 9.2). The same operation was tested using the starting conditions with both, the PKB and the EKB. In both situations the insertion was achieved. However, the number of alignment motions and consequently, time using the PKB was higher compared with the expertise embedded into the EKB. The robot's dexterity deteriorates since the constraint forces were much higher using the PKB. In general, whilst the learning progresses and the robot learns more patterns, the magnitude of forces also diminishes. On average, the time that the robot took to learn a new insertion was 1 minute approximately.

The *generalisation* capability of the NNC was also tested using symmetric and non-symmetric parts. The robot learned to assemble parts in chamfered female blocks using the same PKB and created an EKB for every part geometry. Every particular EKB contained new information from each geometry. In the case of chamferless insertion, different PKBs were needed for each part geometry.

The robot also demonstrated *dexterity* by avoiding any jamming or wedging conditions. Results showed that the manipulator continued to align the peg when inside the female block, effectively preventing this condition.

In addition, failures were detected during rotational alignment that clearly recognised the need to embed more information into the PKB. Further insertions were tested providing additional information about the chamfer. This solved the ambiguity and more importantly, revealed the areas that can improve the performance of the NNC. The improvements are highlighted in the next section.

10.2 Directions for Future Work

Further work has been envisaged to allow the robot to react autonomously and build its own PKB. This step is feasible and the design would involve contact localisation related to the parts to be assembled and based on their geometry. This relationship would generate the corresponding vector motion to diminish the constraint forces. These reflexes are needed only at the beginning of a new operation. The primitive motion conditions will be generated in a similar way the PKB was created. Once these reflexes have been used, they will not be necessary unless the robot is required to learn a completely new operation.

This idea is similar to building a "primitive reflex system" analogous to the reflexes in a human being. Human reflexes are involuntary responses that occur automatically in the presence of certain stimuli. Many of these reflexes are critical for survival, and, they unfold naturally as a part of the infant's development. For instance, the rooting reflex that causes newborn babies to turn their heads toward things that touch their cheeks or; the Babinski reflex, which is the faning out the baby's toes that happens when the outer edge of the sole of the foot is stroked. These primitive reflexes are lost after few months of life since new knowledge is being acquired during development which allows babies to develop complex motions. In a similar manner, a step forward in the robotic assembly system should look at developing the PKB based on these primitive reflexes.

Currently, the PKB has been formed by the user and embedded into the NNC. The order of the categorical representation during learning has implicitly been given by the PKB training order. That is, by initialising the learning in this way the categorical representation were formed by fx followed by fy, then fz and so on until 12 categories were formed. Having the automatic inclusion of this knowledge by the manipulator itself raises other issues in terms of providing a "teacher" to show the robot which action to take.

The automatic generation of the PKB will involve the use of richer sensory information from the environment. Tactile sensing for further exploration around the hole in conjunction with a vision system will be necessary. Information regarding the part geometry will be required to tell the robot how to automatically relate

sensation with actions.

Further understanding of the knowledge discovery has also been envisaged. This would involve knowledge extraction from the learned experience in a form of rules. In this manner, this information can later be implemented in the NNC. By using these rules, it is suggested that it is possible to implement a reasoning capability in the robot's controller.

Finally, it is important to mention that the work developed during this research is intended to be implemented into an intelligent robotic cell that comprises the areas of object recognition and language understanding. These areas are currently being investigated in the Manufacturing Automation Research Group in the Nottingham Trent University. In this direction, the research presented in this thesis provides a solid foundation towards the development of autonomous robots for industrial applications.

fin,

References

[1] L. A. Martin-Vega; H. K. Brown; W. H. Shaw; T. J. Sanders. Industrial perspectives on research needs and opportunities in manufacturing assembly. *Journal of Manufacturing Systems*, 14(1):45–58, 1995.

[2] V. Balendran. *Cosmetic quality of surfaces: a computational approach.* PhD thesis, The Nottingham Trent University, 1994.

[3] Martin Howarth. *An Investigation of Task Level Programming for Robotic Assembly.* PhD thesis, The Nottingham Trent University, January 1998.

[4] Vijaykumar Gullapalli; Judy A Franklin; Hamid Benbrahim. Acquiring robot skills via reinforcement learning. *IEEE Control Systems*, pages 13–24, February 1994.

[5] H. Asada. Skill acquisition from human experts through pattern processing of teaching data. *Proc of IEEE Conf on Robotics and Automation*, pages 1302–1307, 1989.

[6] Doyoung Jeon; Masayoshi Tomizuka. Learning hybrid force and position control of robot manipulators. *Proc of the 1992 IEEE Int Conf on Robotics and Automation*, 2:1455–1460, May 1992.

[7] L. Liu; B. J. Ulrich; M. A. Elbestawi. Robot grinding force regulation: Design, implementation and benefits. *Proc of the 1990 IEEE Int Conf on Robotics and Automation.*, 1:258–265, 1990.

[8] Sukhan Lee; Hank Sung Lee. Generalized impedance of manipulators: Its application to force and position control. *91 ICAR 5th IEEE Int Conf on Advaced Robotics*, 2:1477–1480, 1991.

[9] James M. Hyde; Mark R. Cutkosky. Controlling contact transition. *IEEE Control Systems*, pages 25–30, February 1994.

[10] L. S. Wilfinger; J. T. Wen; S. H. Murphy. Integral force control with robustness enhancement. *IEEE Control Systems*, pages 31–40, February 1994.

[11] Shahram Payandeh; Andrew A. Goldenberg. A robust force controller: Theory and experiments. *Proc of the 1991 IEEE Conf on Robotics and Automation*, 1:36–41, April 1991.

[12] Seul Jung; T. C. Hsia; R. G. Bonitz. On force tracking impedance control with unknown environment stiffness. *IASTED Conference on Robotics and Manufacturing*, pages 181–184, June 1995.

[13] H. Qiao; B. S. Dalay; R. M. Parkin. Robotic peg-hole insertion operations using a six-component force sensor. *Proc Instn Mech Engrs, Part: C: Journal of Mechanical Engineering Science*, 207:289–306, 1993.

[14] H. Qiao; B. S. Dalay; R. M. Parkin. Precise robotic chamferless peg-hole insertion operation without force feedback and remote centre compliance (rcc). *Proc Instn Mech Engrs. Part C: Journal of Mechanical Engineering Science*, 208:89–104, 1994.

[15] H. Bruyninckx; S. Dutre; J. De Schutter. Peg-on-hole: A model based solution to peg and hole alignment. *Proc of the 1995 IEEE Int Conf on Robotics and Automation*, pages 1919–1924, 1995.

[16] Enrique Cervera; Angel P. del Pobil; Miguel A. Serna. A sensor-based approach for motion in contact task planning. *Proc 1995 IEEE/RSJ Int Conf on Intelligent Robots and Systems*, 2:468–473, August 1995.

[17] Susan N. Gottschlich; Avinash C. Kak. A dynamic approach to high-precision parts mating. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(4):797–810, July/August 1989.

[18] D. E. Whitney. Quasi-static assembly of compliantly supported rigid parts. *J. of Dynamic Systems, Measurement, and Control*, 104:65–77, March 1982.

[19] S. Joo and F. Miyazaki. Development of variable rcc and its application. *Proc. of the 1998 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2:1326–1332, October 1998.

[20] Tomas Lozano-Perez; Matthew T Mason; Russell H Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24, Spring 1984.

[21] J. De Schutter; H. Van Brussel. Compliant robot motion i. a formalism for specifying compliant motion tasks. *The Int Journal of Robotics Research*, 7(4):3–17, August 1988.

[22] J. De Schutter; H. Van Brussel. Compliant robot motion ii. a control approach based on external loops. *The Int Journal of Robotics Research*, 7(4):18–33, August 1988.

[23] H. Asada. Teaching and learning of compliance using neural nets. *Proc 1990 IEEE Int Conf on Robotics and Automation*, pages 1237–1244, 1990.

[24] Enrique Cervera; Angel P. del Pobil. Learning and classification of contact states in robotic assembly tasks. *Proc of the 9th Int. Conf. IEA/AIE*, pages 725–730, June 1996.

[25] Vijaykumar Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3:671–692, 1990.

[26] Enrique Cervera; Angel P. del Pobil. Programming and learning in real-world manipulation tasks. *Proc 1997 IEEE/RSJ Int Conf on Intelligent Robot and Systems*, 1:471–476, September 1997.

[27] Vijaykumar Gullapalli. Learning control under extreme uncertainty. *In Advances in Neural Information Processing Systems (Eds.) C. L. Giles, S. J. Hansan, J.D. Cowan*, pages 327–334, 1993.

[28] N. Chandler; V. Balendran; L. Evett; K. Sivayoganathan. From hearing to understanding. *Manufacturing Engineering*, pages 65–68, April 1996.

[29] N. Chandler; V. Balendran; L. Evett; K. Sivayoganathan. Using art neural networks to enable robots to understand language. In A. N. Bramley; A.R. Mileham; G. W. Owen, editor, *Advances in Manufacturing Technology X*, pages 171–175, Bath, England, September 1996. University of Bath.

[30] C. D'Souza; K. Sivayoganathan; D. Al-Dabass; V. Balendran; J. Keat. Machine vision for robotic assembly: Issues and experiments. In David K Harrison, editor, *Advances in Manufacturing Technology XI*, pages 114– 118, Glasgow, U.K., September 1997. Glasgow Caledonian University.

[31] C. D'Souza; I. Lopez-Juarez; K. Sivayoganathan; M. Howarth; V. Balendran. Skill acquisition for robotic assembly via integration of vision and force sensing. In R. Baines; A. Taleb-Bendiab; Z. Zhao, editor, *Advances in Manufacturing Technology XII*, pages 349–354, Derby, UK, September 1998. Professional Engineering Publishing Ltd.

[32] Gail A. Carpenter; Stephen Grossberg. *Computer Vision, Graphics, and Image Processing*, chapter A Massively Parallel Architecture for a Sel-Organizing Neural Pattern Recognition Machine, pages 54–115. Academic Press, Inc., 1987.

[33] Gail A. Carpenter; Stephen Grossberg; John H Reynolds. Artmap: Supervised real-time learning and classification of nonstationary data by self-organizing neural network. *Neural Networks*, pages 565–588, 1991.

[34] Gail A. Carpenter; Stephen Grossberg; Natlya Markunzon; John H. Reynolds; David B. Rosen. Fuzzy artmap: A neural network architecture for incremental learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3(5):698–713, September 1992.

[35] M. Howarth; K. Sivayoganathan; A. Schonegge. Integrating force sensing for robotic assembly. *In Advances in Manufacturing Technology VII, Proc of the 9th Nat Conf on Manufacturing Research*, pages 361–365, September 1993.

[36] M. Howarth; K. Sivayoganathan;P. D. Thomas; C. R. Gentle. Task level programming using neural networks. *Proc 1995 IEE Int Conf on Artificial Neural Networks*, pages 262–267, 1995.

[37] Bernard Widrow; Michael A. Lehr. *The Handbook of Brain Theory and Neural Networks*, chapter Perceptrons, Adalines, and Backpropagation, pages 719–724. MIT Press, 1995.

[38] Andrew G. Barto. *The Handbook of Brain Theory and Neural Networks*, chapter Reinforcement Learning in Motor Control, pages 809–813. MIT Press, 1995.

[39] Andrew G. Barto. *The Handbook of Brain Theory and Neural Networks*, chapter Reinforcement Learning, pages 804–809. MIT Press, 1995.

[40] Trident Robotics and Research, Inc., 2516 Matterhorn Drive, Wexford, Pennsylvania. *TRC004 User's Manual*, April 1994.

[41] DEC. Retirement of the pdp family. http://www.digital.com/info/CU2902/, 1996.

[42] S. P. Praturu; J. N. Anderson. Position signal interface for the robotic arm control system. *Proc of the 23rd IEEE Southeastern Symposium on System Theory*, pages 168–72, 1991.

[43] R Bukowski; L S Haynes; N Coleman; A Santucci; K Lam; A Paz; M DeVito. Robot hand-eye coordination rapid prototyping environment. *Proc Int Symp on Industrial Robots*, pages 16.15–16.28, 1991.

[44] Unimation, Inc., Danbury, Connecticut. *User's Guide to VAL II*, version 2.0 edition, February 1986.

[45] George V. Paul; Katsushi Ikeuchi. Modelling planar assembly paths from observation. *IROS 96*, 1996.

[46] John J. Craig. *Introduction to Robotics Mechanics and Control.* Addison-Wesley Publishing Company, 2nd edition, 1986.

[47] Keith A. Morris. Has force/torque sensing gained factory acceptance? *Proc 1992 Int Conf on Robots and Systems*, 1992.

[48] Adept. Adept and staubli announce new products and marketing partnership. Press Releases. Adept Technology, Inc., May 1995.

[49] DEC. *DECnet DIGITAL Network Architecture DDCMP Functional Specification.*, phase iv, version 4.1 edition, August 1994.

[50] DEC, Danbury, CT. USA. *700 Series Models 761/762 Equipment Manual 398Z1 For VAL II Operating System*, 398z1 edition, October 1986.

[51] (Southern Technical Editor) Robert N. Boggs. Transducer's three beams outperform four. *Design Application Note, Assurance Technologies, Inc.*, 1995.

[52] Richard M. Voyles; J. Dan Morrow; Pradep K. Khosla. A comparison of force sensors,. *Advanced manipulators Laboratory, The Robotics Institute.*, July 1994.

[53] Mikell P. Groover; Mitchell Weiss: Roger N. Nagel. *Industrial Robotics Technology, Programming and Applications.* Mc Graw-Hill, 1986.

[54] Yoram Koren. *Robotics for Engineers.* Mc Graw-Hill, 1985.

[55] I Lopez-Juarez. Diseno y Construccion de un Medidor de Presion Barometrica. Bachelor thesis, Engineering Faculty-National Autonomous University of Mexico, 1994.

[56] Michael Erdmann. Using backprojections for fine motion planning with uncertainty. *The Int. Journal of Robotics Research*, 5(1):19–45, Spring 1986.

[57] Vijaykumar Gullapalli. Skillful control under uncertainty via direct reinforcement learning. *Robotics and Autonomous Systems*, 15:237–246, 1995.

[58] M. R. Popovic; A. A. Goldenberg. Modeling of friction using spectral analysis. *IEEE Trans. on Robotics and Automation*, 14(1):114–122, February 1998.

[59] Steven C. Chapra; Raymond P. Canale. *Numerical methods for engineers.* Mc Graw-Hill, 2 edition, 1988.

[60] Ian M. Berry; M. Paul Gough. A comparison of art and som artificial neural networks for unsupervised classification of remote sensing data. *Unpublished*, 1999.

[61] Stephen Grossberg. Adaptive pattern classification and universal recoding ii: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics*, 23:187–202, 1976.

[62] Stephen Grossberg. Non-linear difference-differential equations in prediction and learning theory. *Proc. of the National Academy of Science*, 58:1329–1334, 1967.

[63] Stephen Grossberg. Some psychological and biochemical consequences of psychological postulates. *Proc. of the National Academy of Sciences*, 60:758–765, 1968.

[64] Stephen Grossberg. Some non-linear networks capable of learning a spatial pattern of arbitrary complexity. *Proc. of the National Academy of Sciences*, 59:368–372, 1968.

[65] Gail A. Carpenter; Stephen Grossberg; David B. Rosen. Art 2-a: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, 4:493–504, 1991.

[66] James R. Williamson. Gaussian artmap: A neural network for fast incremental learning of noisy multidimensional maps. *Neural Networks*, 9(5):881–897, 1996.

[67] Gail A. Carpenter; William D. Ross. Art-emap: A neural network architecture for object recognition by evidence accumulation. *IEEE Trans on Neural Networks*, 6(4):805–818, July 1995.

[68] Albert Nigrin. *Neural Networks for Pattern Recognition.* MIT Press, 1993.

[69] A. G. Andreou T. Serrano-Gotarredona, B. Linares-Barranco. *Adaptive Resonance Theory Microchips.* Kluwer Academic Publishers, 1998.

[70] Robert S. Feldman. *Understanding Psychology.* McGraw-Hill, Inc., 3rd. edition, 1993.

[71] Susan E. Gathercole. The development of memory. *J. of Child Psychol. Psychiat.*, 39,(1,):3–27., 1998,.

[72] N. McCloskey, M.; Cohen. *The Psychology of Learning and Motivation*, volume 24,, chapter Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem., pages 109–164. Academic Press., 1989,.

[73] Robert M. French. Using pseudo-recurrent connectionist networks to solve the problem of sequential learning. *Proc. of the 19th Annual Cognitive Science Society Conference*, 1997,.

[74] Robert M. French. Catastrophic forgetting in connectionist networks: Causes, consequences and solutions. *Trends in Cognitive Sciences*, 3(4):128–135., 1999.

[75] Stephen Grossberg. *Self-organizing neural networks for stable control of autonomous behaviour in a changing world.* Elsevier Science Publishers,, 1993,.

[76] Gail A. Carpenter; Stephen Grossberg; David B. Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.

[77] Alan V. Oppenheim; Ronald W. Schafer. *Digital Signal Processing.* Prentice-Hall, London, 1986.

[78] Geoffrey G. Towell; Jude W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70:119–165, 1994.

[79] Jude W. Shavlik. A framework for combining symbolic and neural learning. Technical Report 1123, University of Wisconsin-Madison, November 1992.

[80] G.G. Towel. *Symbolic Knowledge and Neural Networks: Insertion, Refinement, and Extraction.* PhD thesis, University of Wisconsin, 1992.

Appendix A

Results

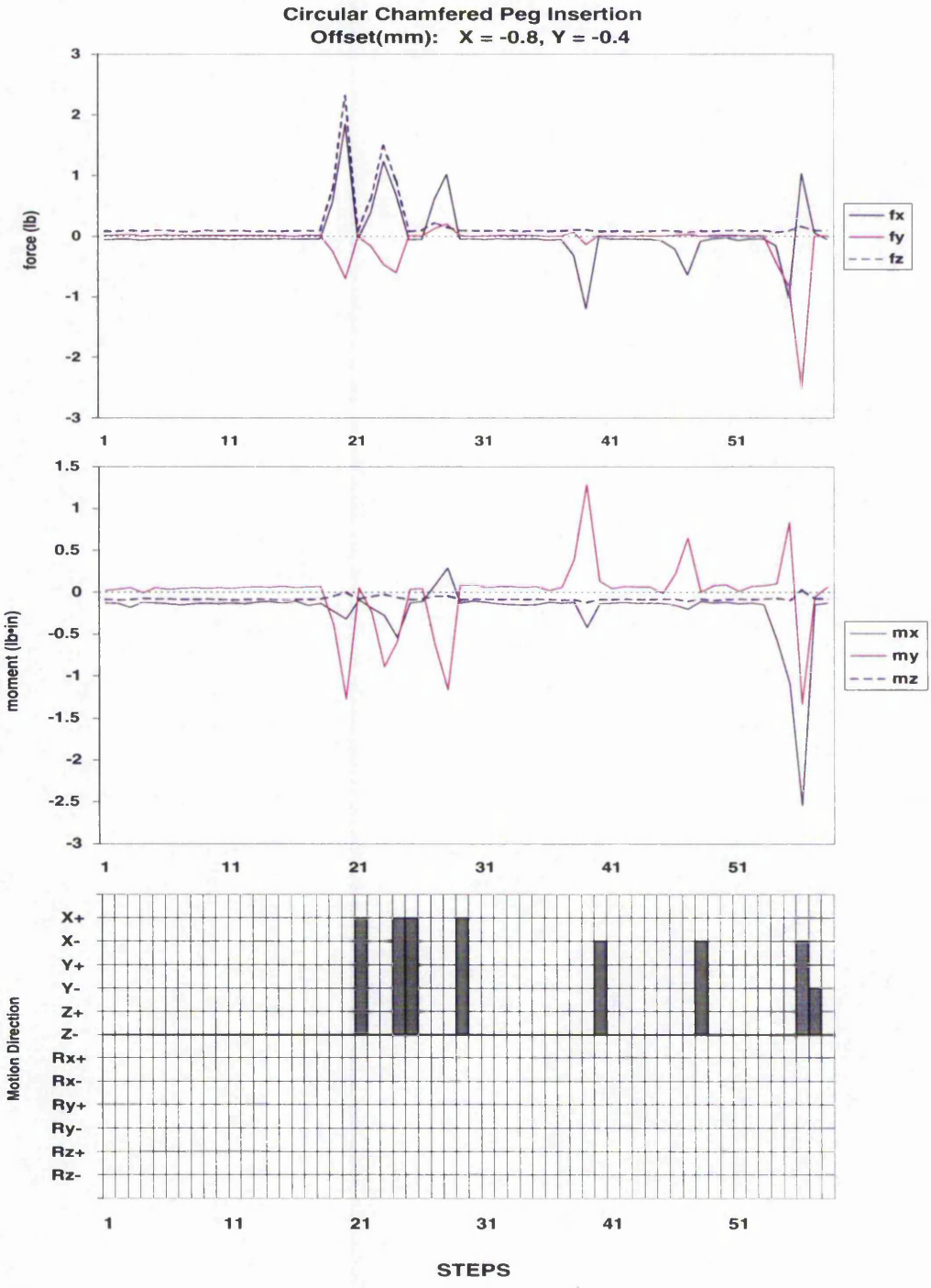


Figure A.1: Second Circular Chamfered Peg Insertion

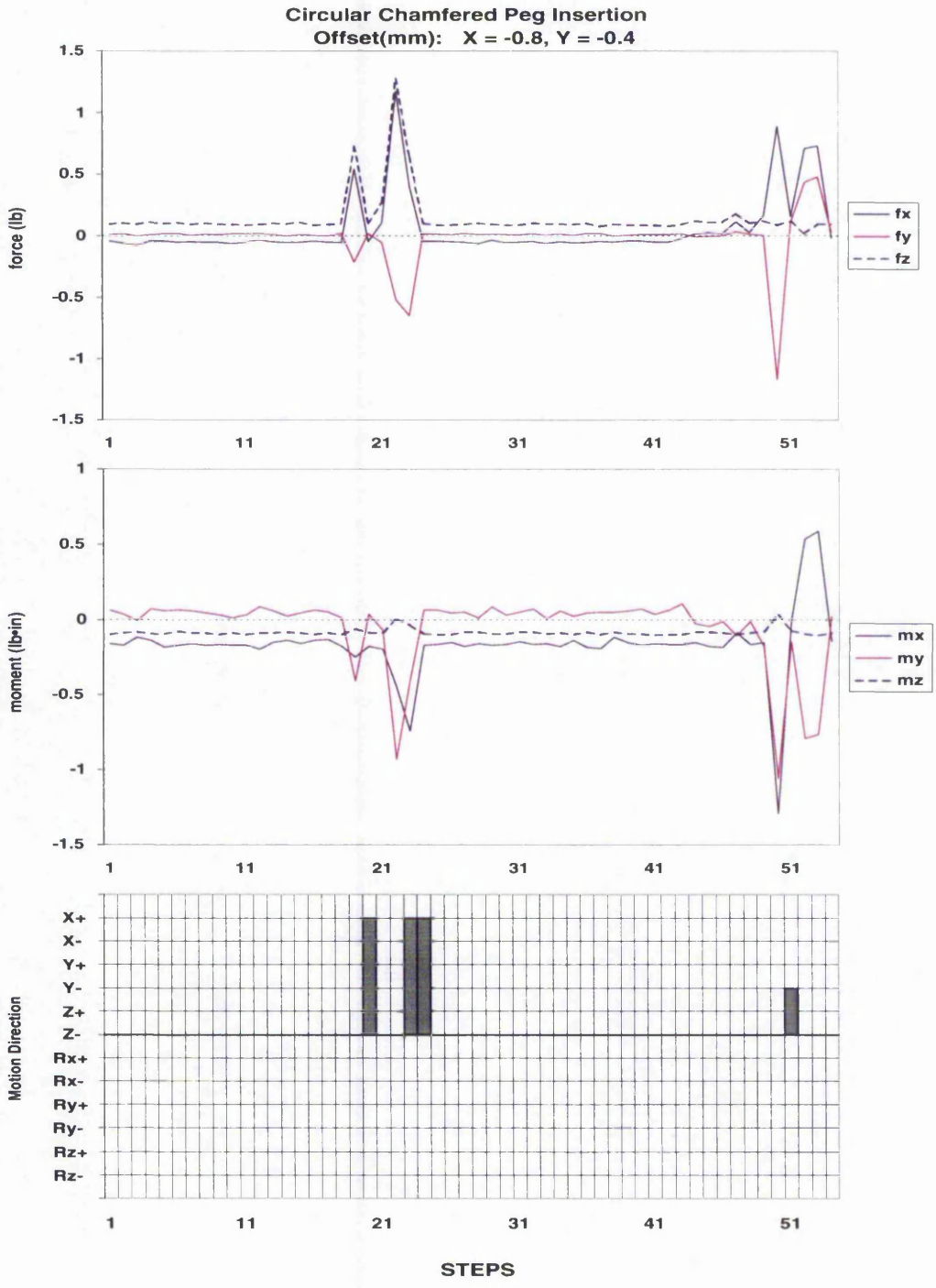


Figure A.2: Third Circular Chamfered Peg Insertion

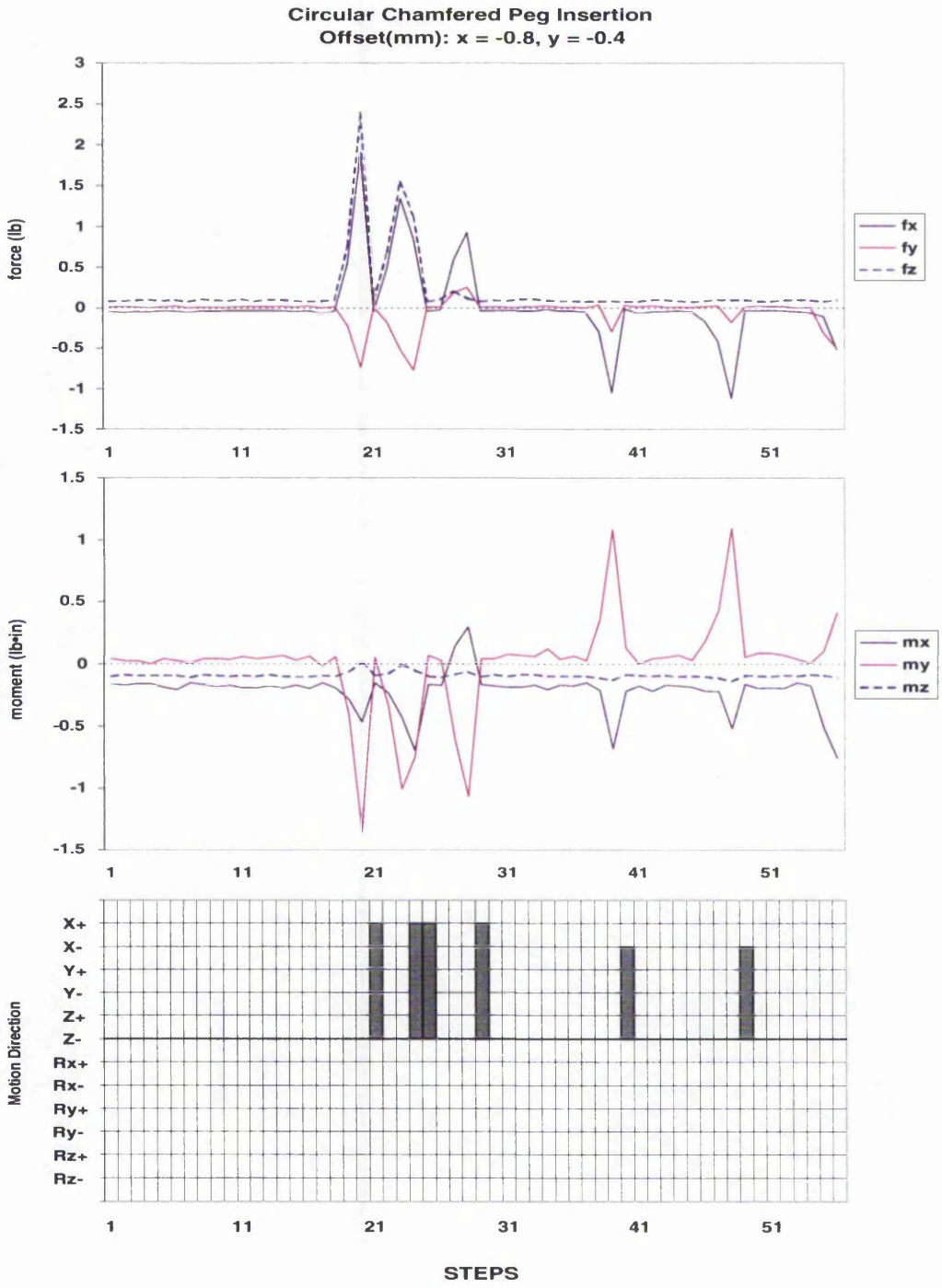


Figure A.3: Fourth Circular Chamfered Peg Insertion

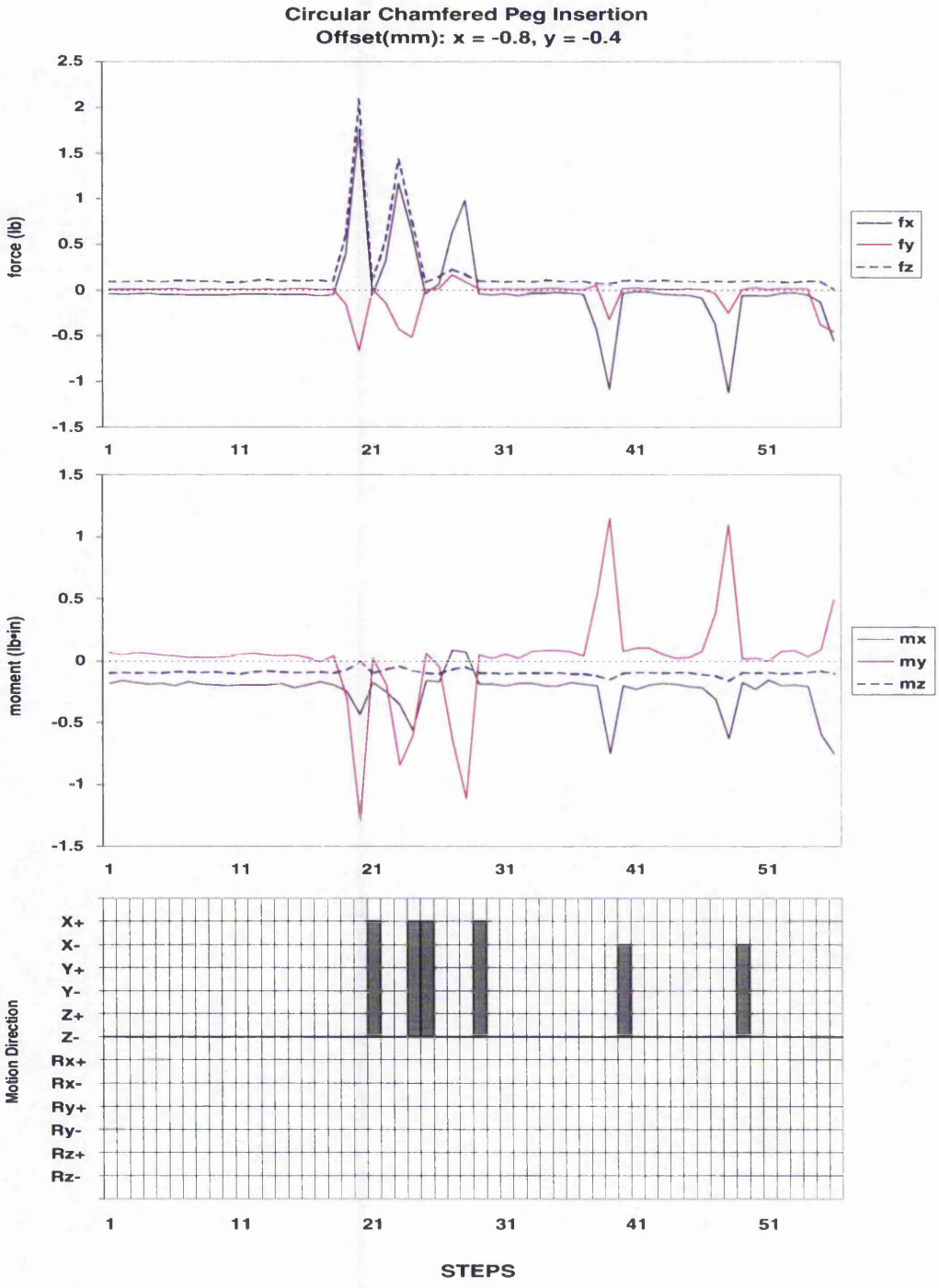


Figure A.4: Fifth Circular Chamfered Peg Insertion

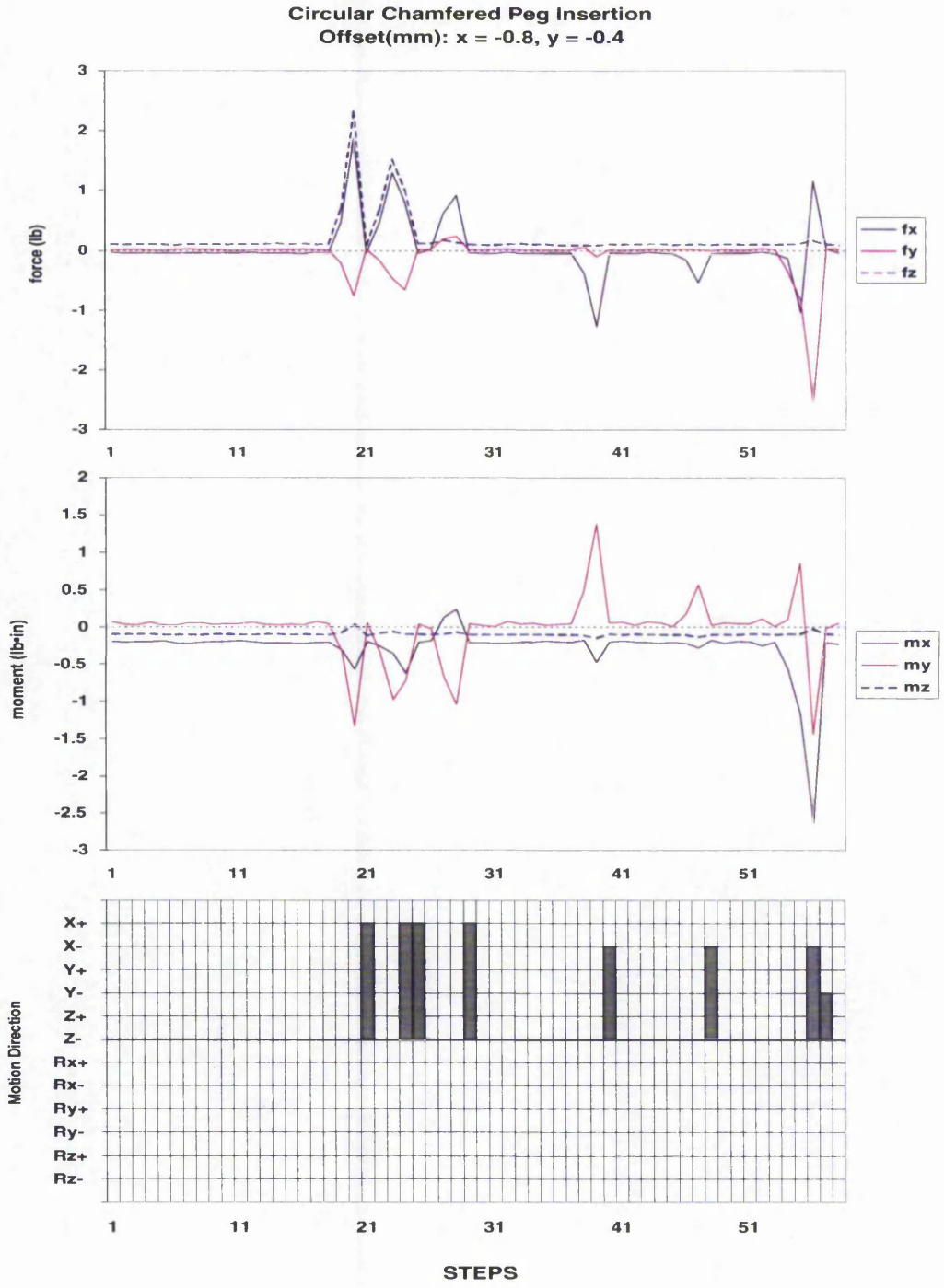


Figure A.5: Sixth Circular Chamfered Peg Insertion

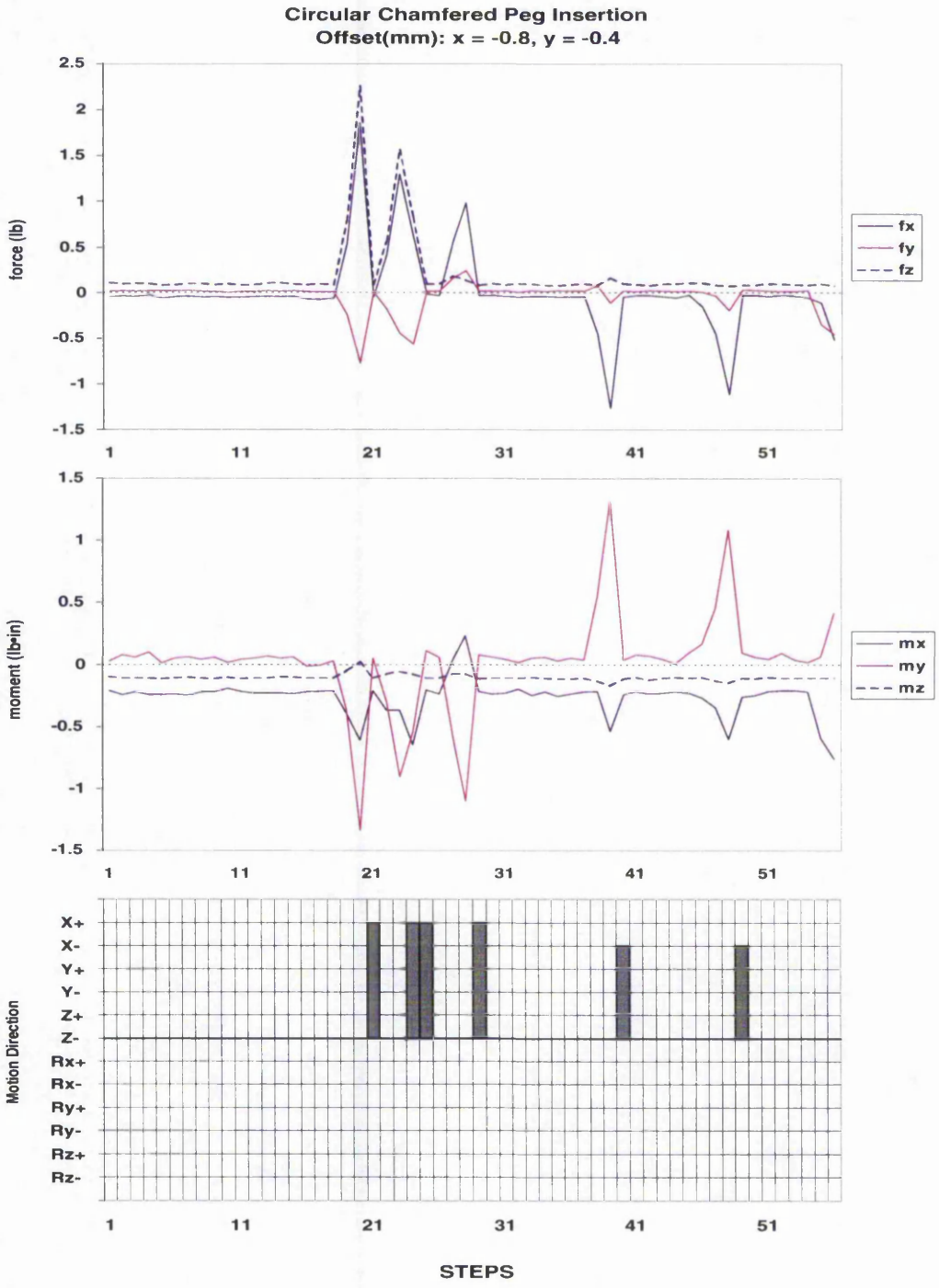


Figure A.6: Seventh Circular Chamfered Peg Insertion

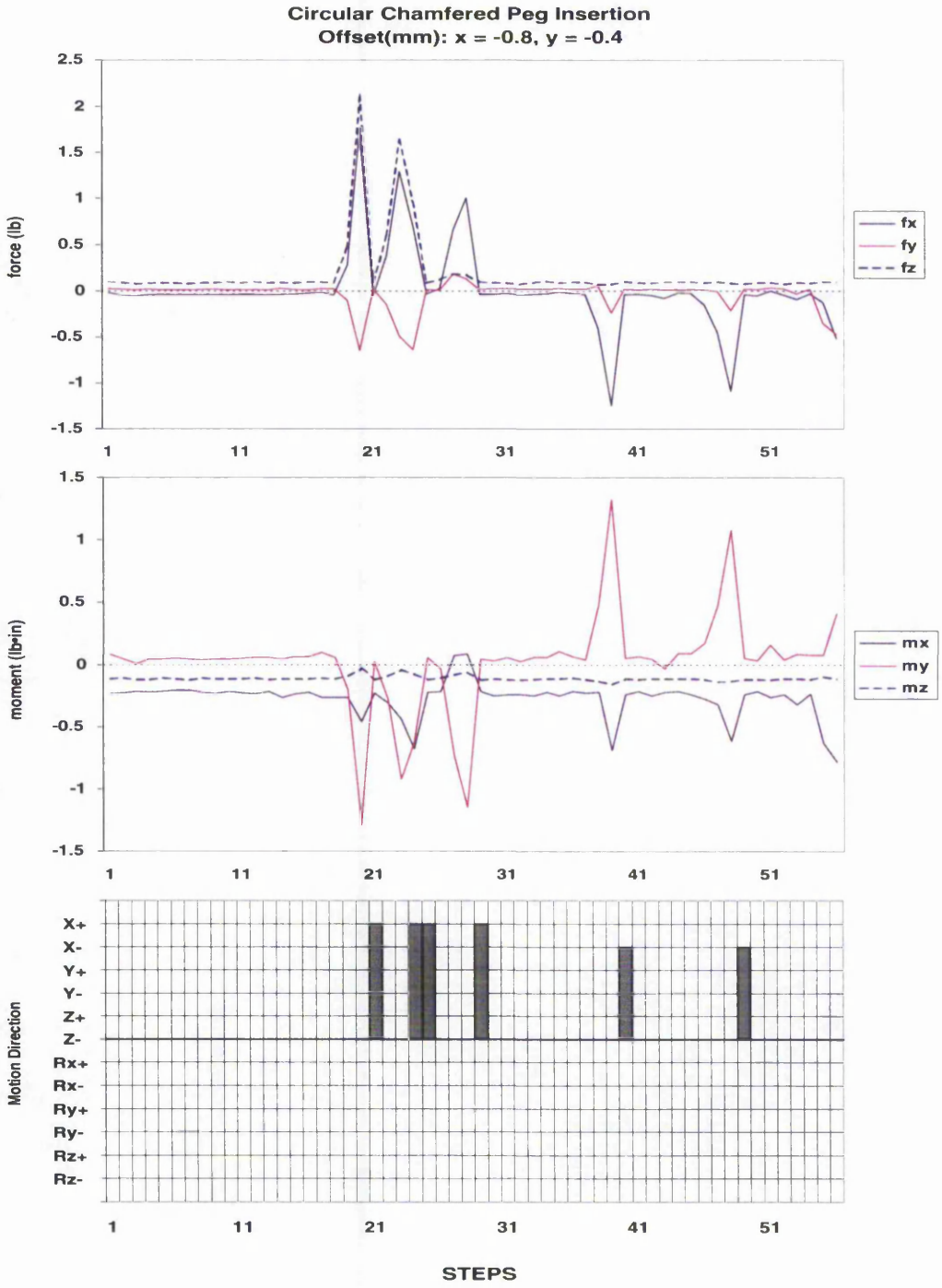


Figure A.7: Eighth Circular Chamfered Peg Insertion

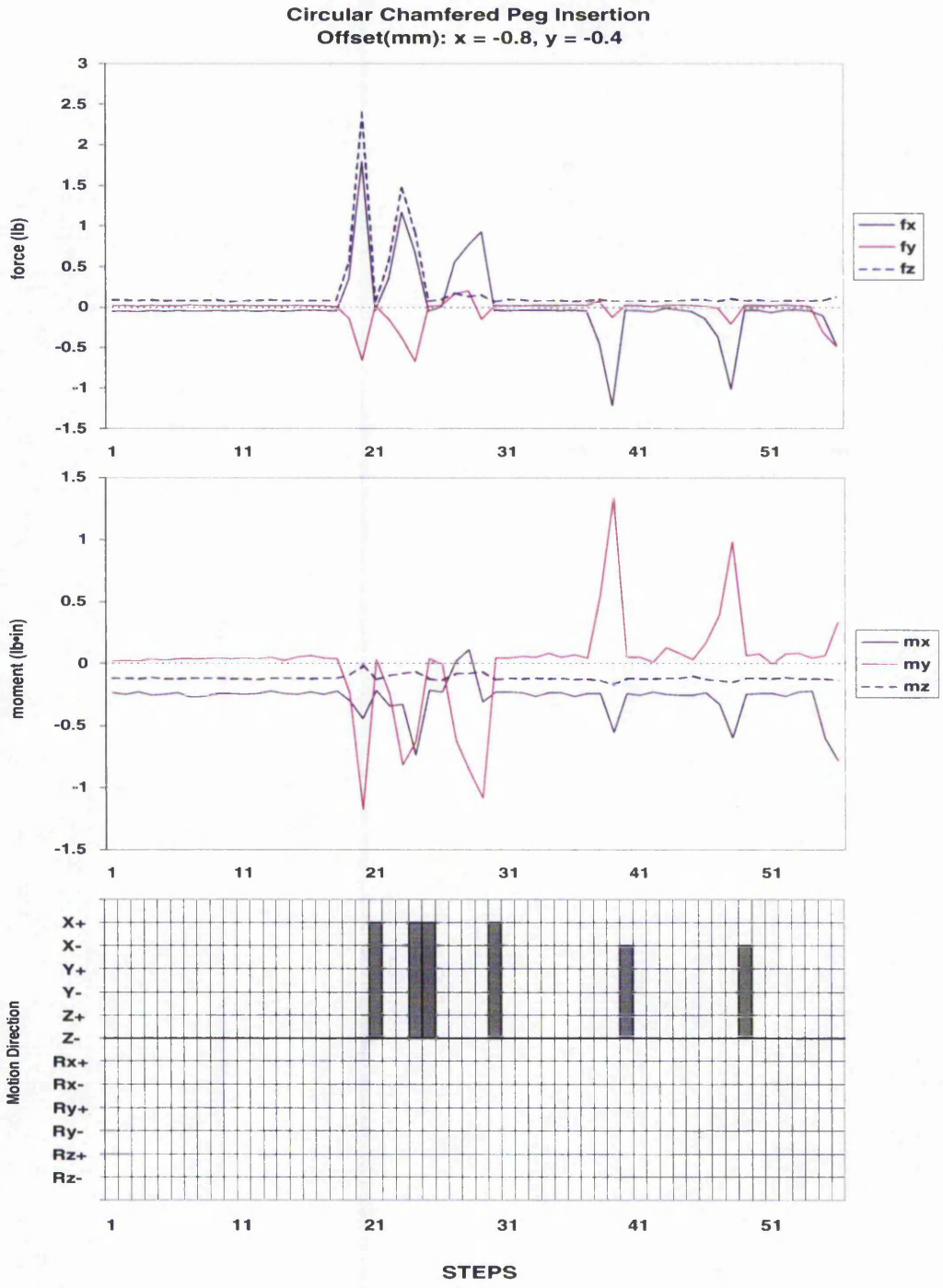


Figure A.8: Nineth Circular Chamfered Peg Insertion

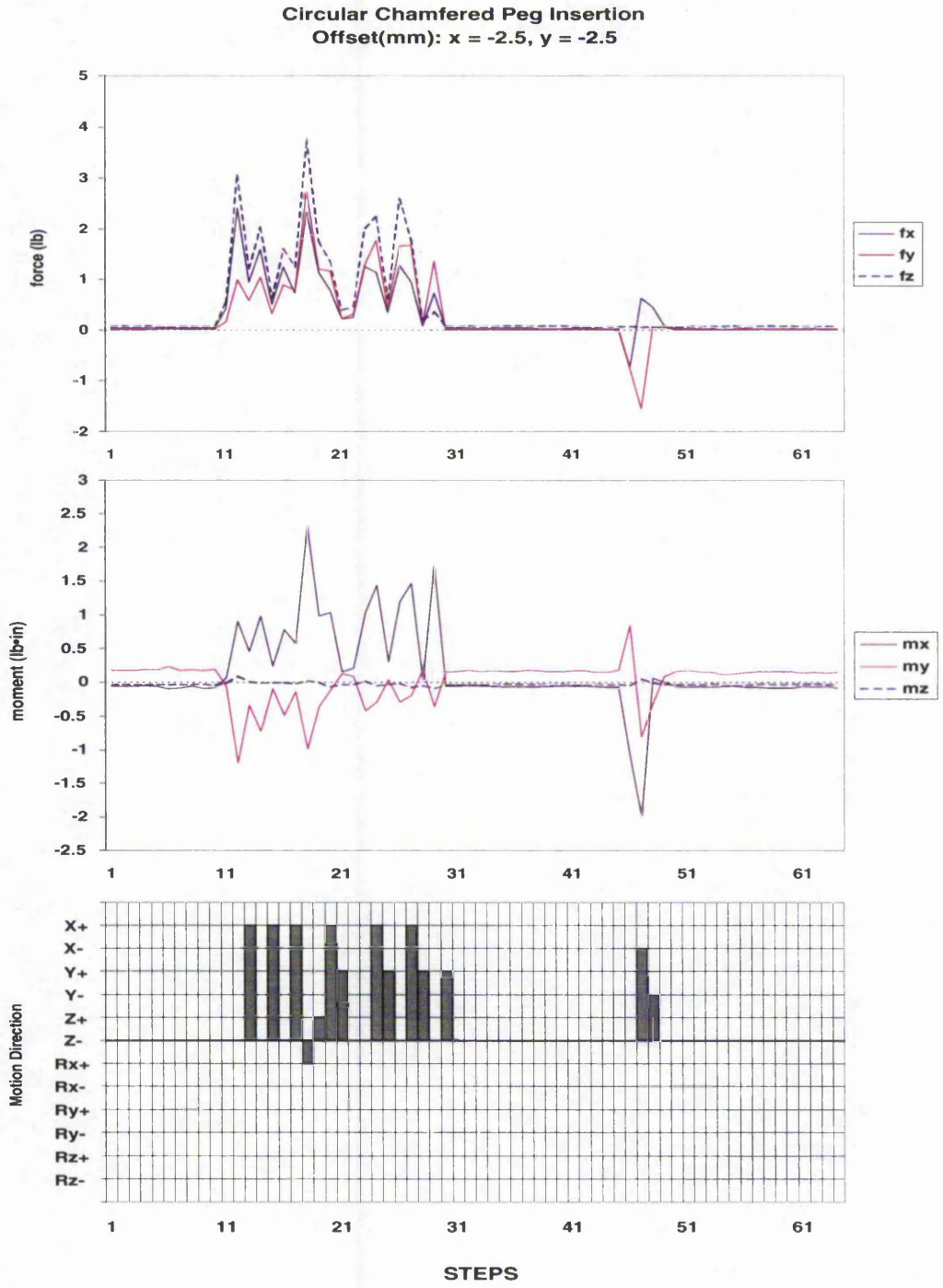


Figure A.9: 10th Circular Chamfered Peg Insertion

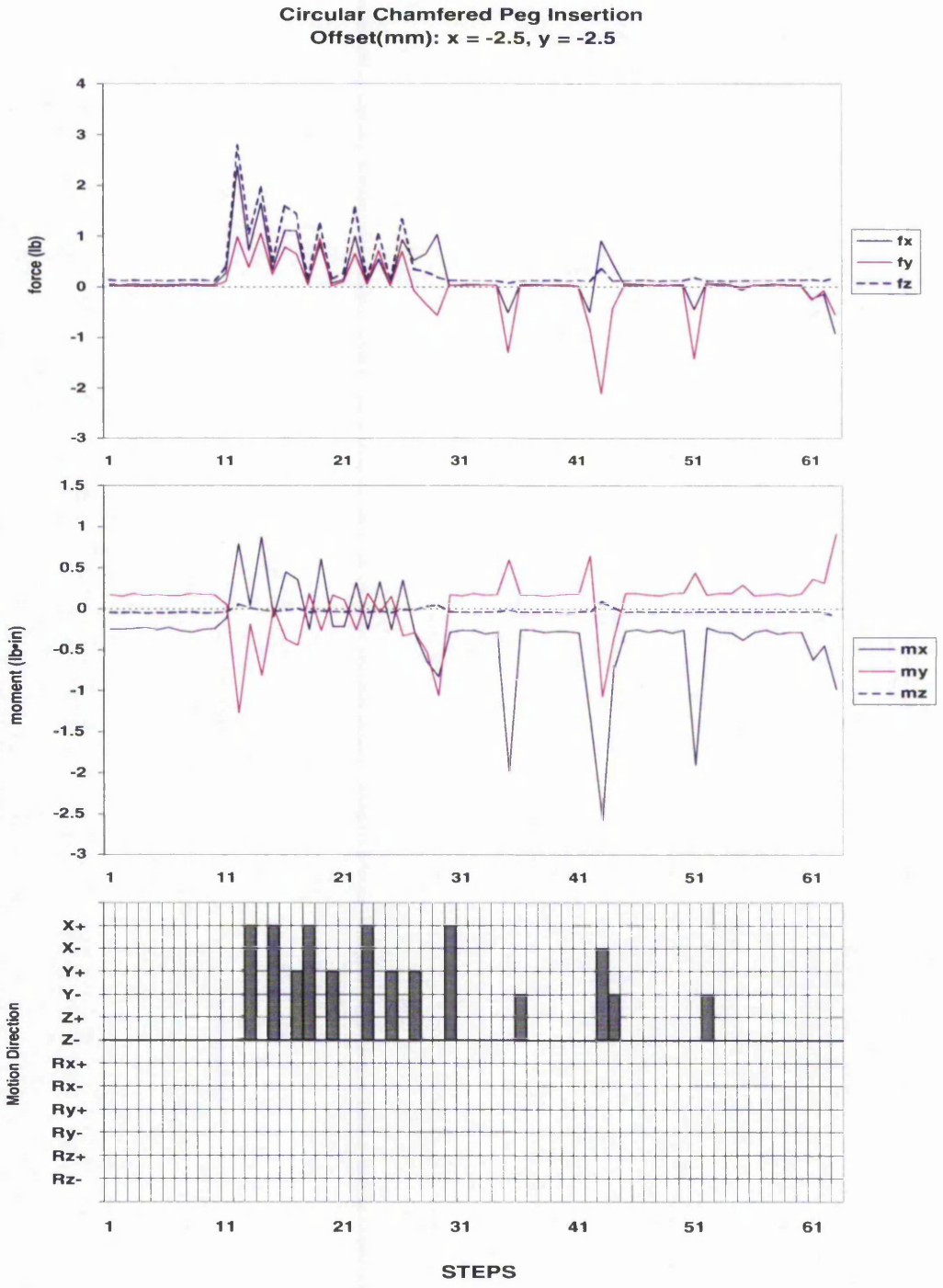


Figure A.10: 11th Circular Chamfered Peg Insertion

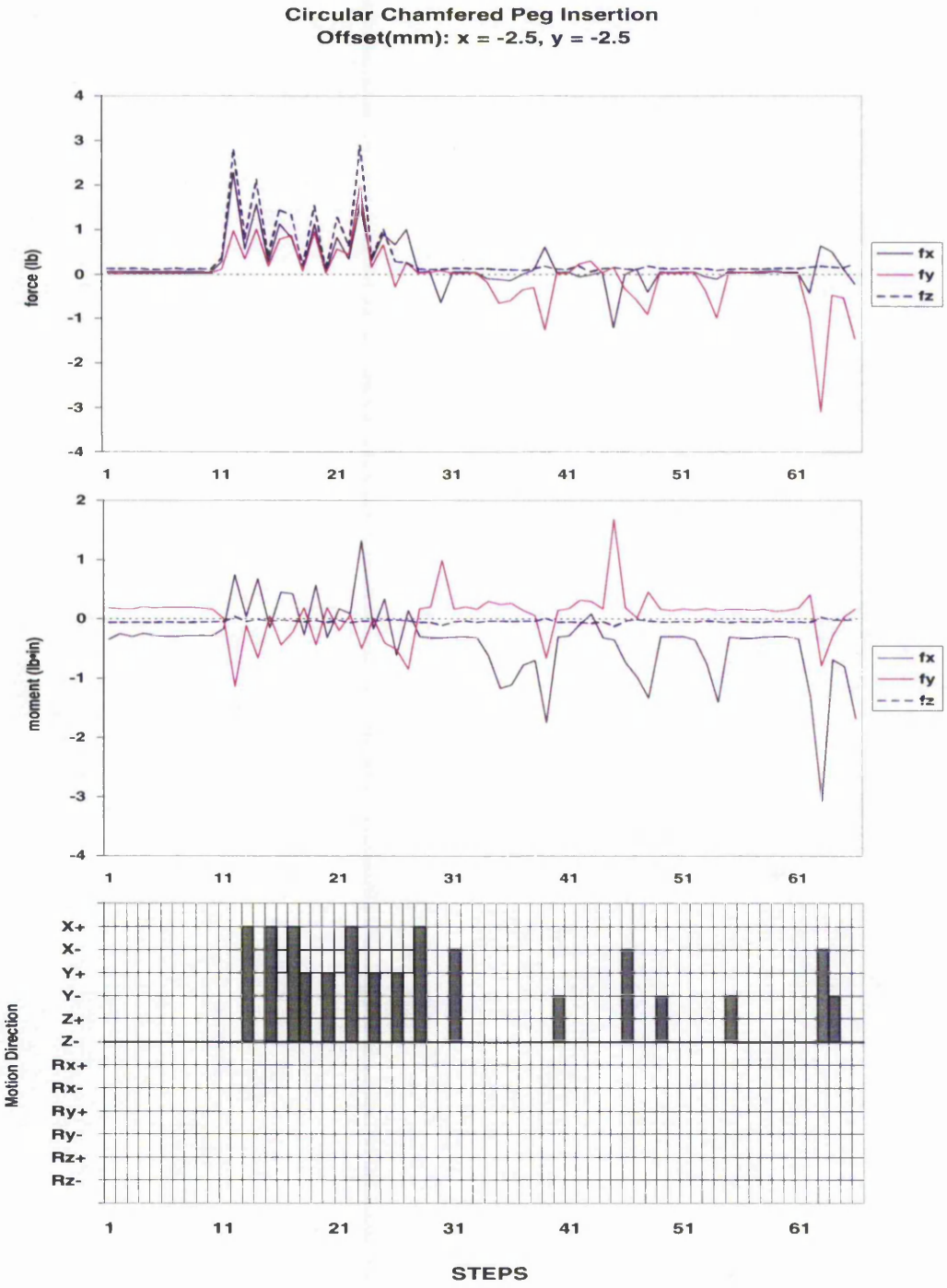


Figure A.11: 12th Circular Chamfered Peg Insertion

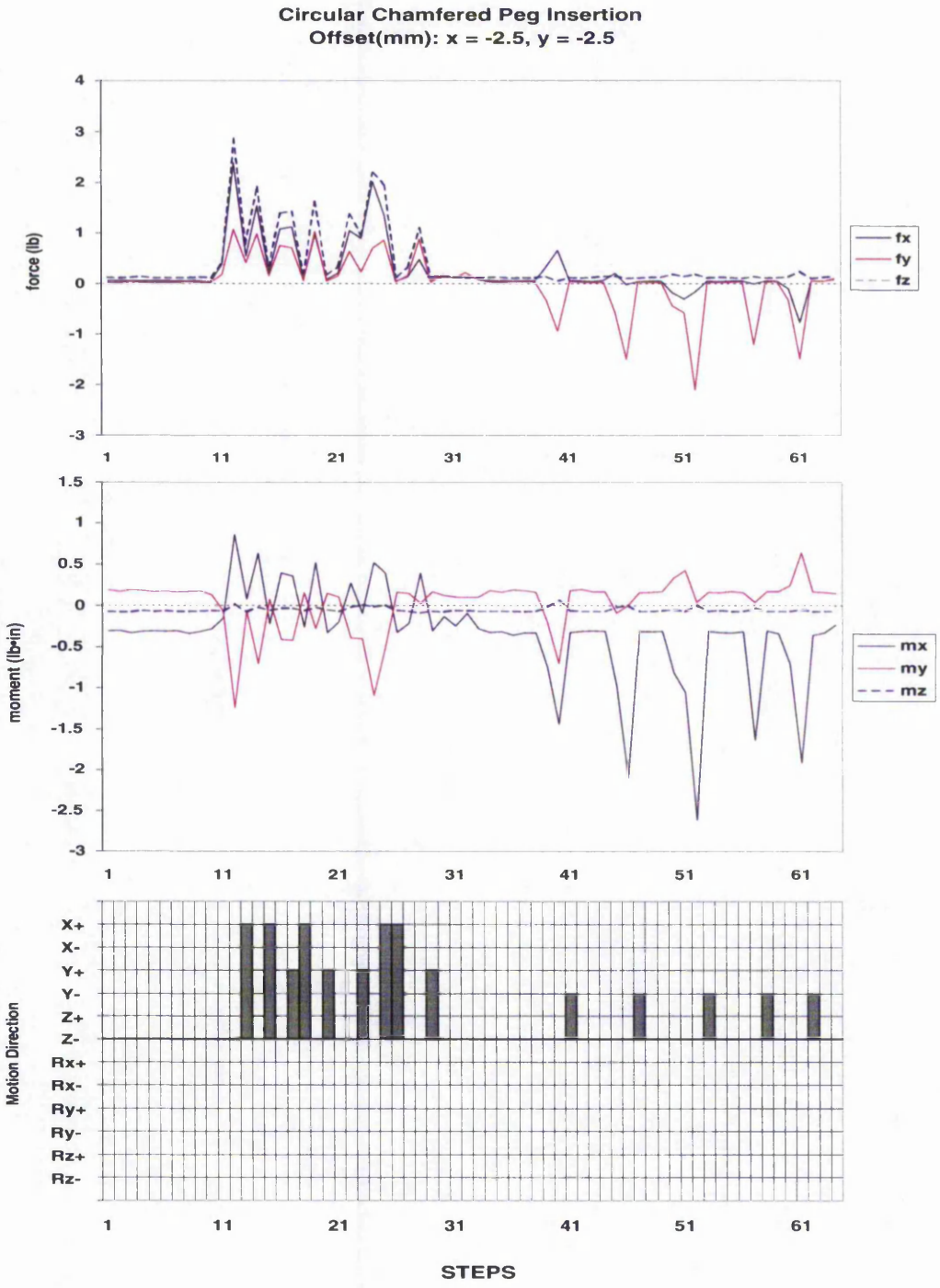


Figure A.12: 13rd Circular Chamfered Peg Insertion

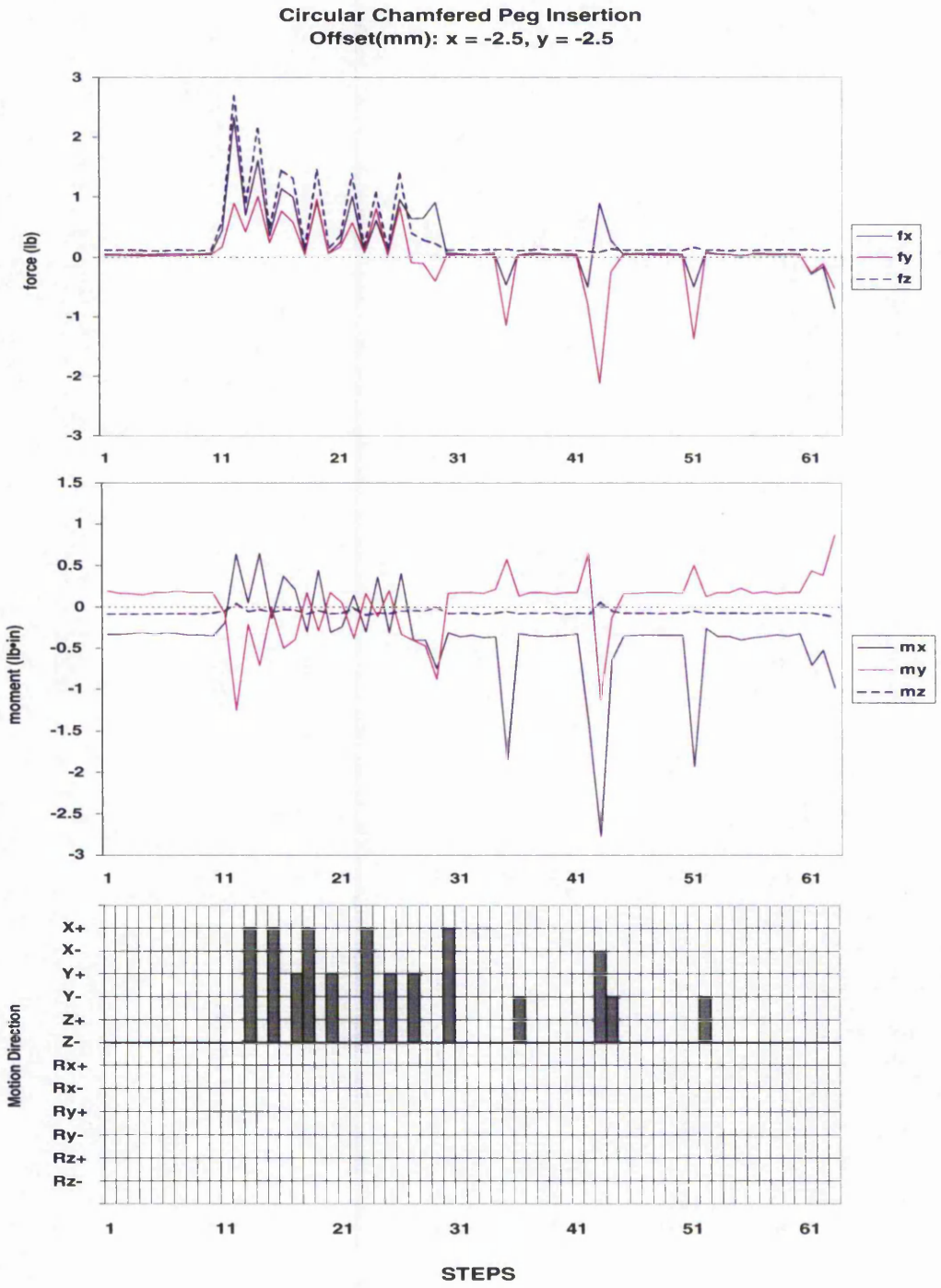


Figure A.13: 14th Circular Chamfered Peg Insertion

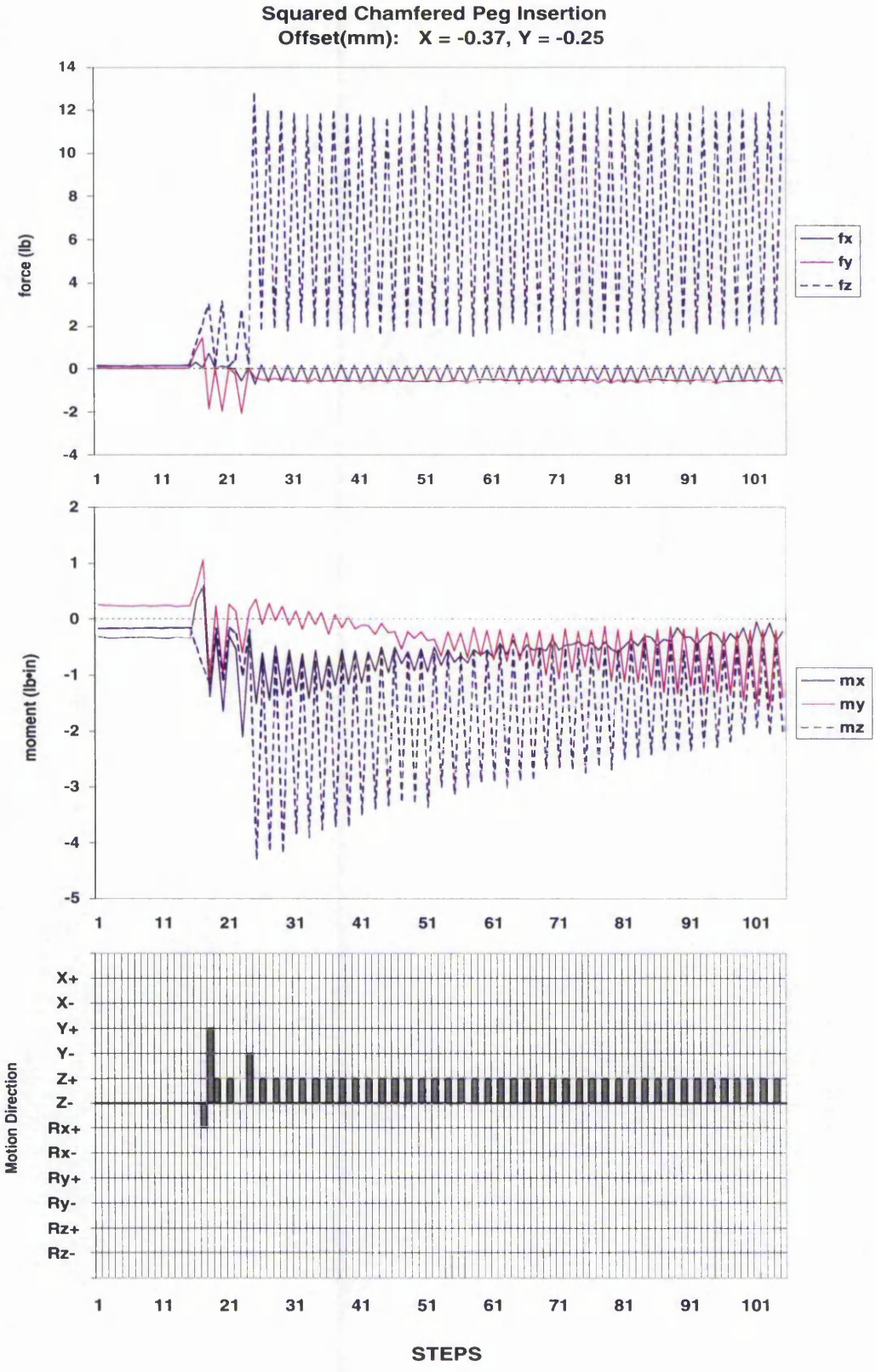


Figure A.14: Fourth Square Chamfered Peg Insertion

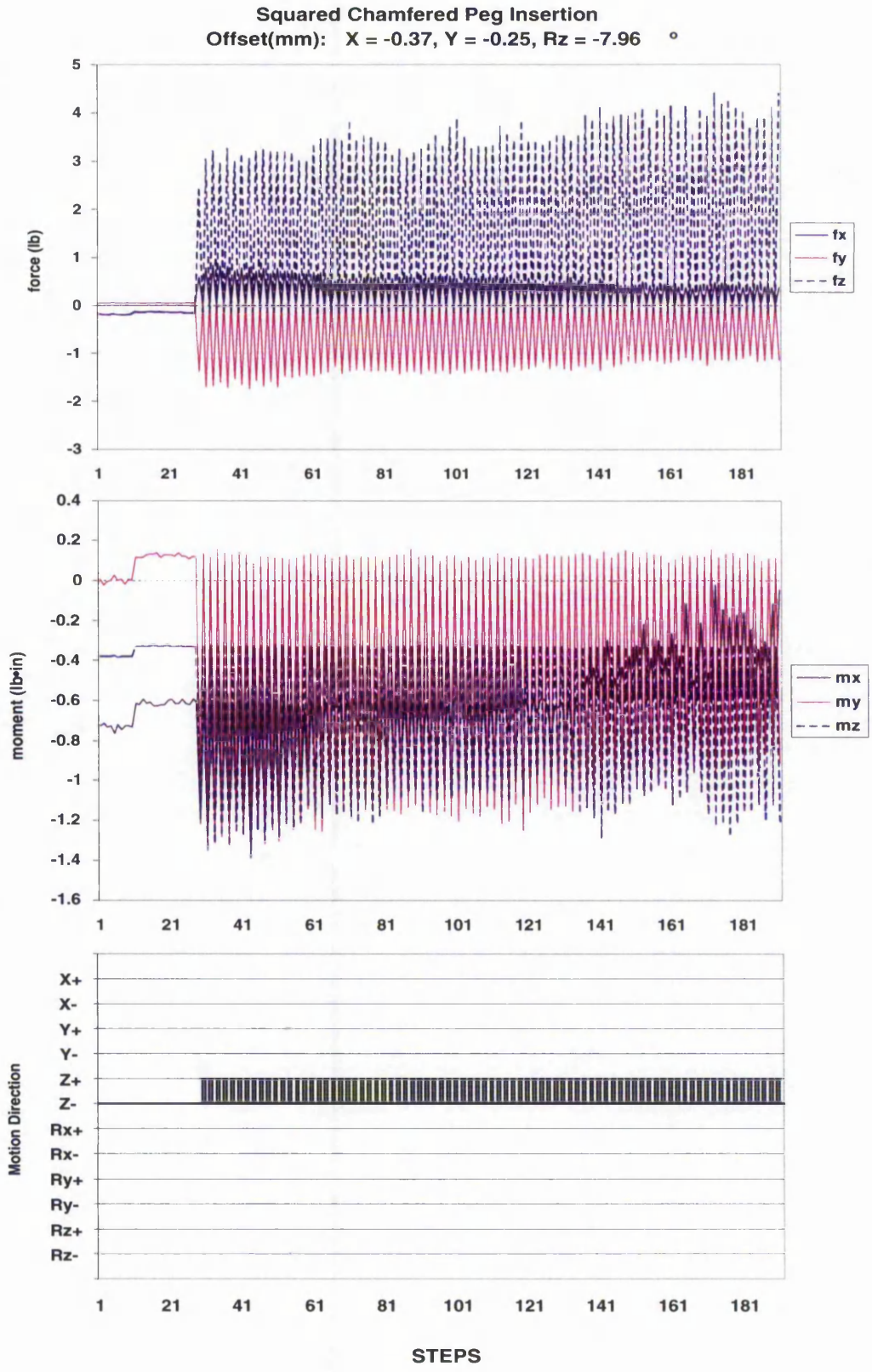


Figure A.15: Fifth Square Chamfered Peg Insertion

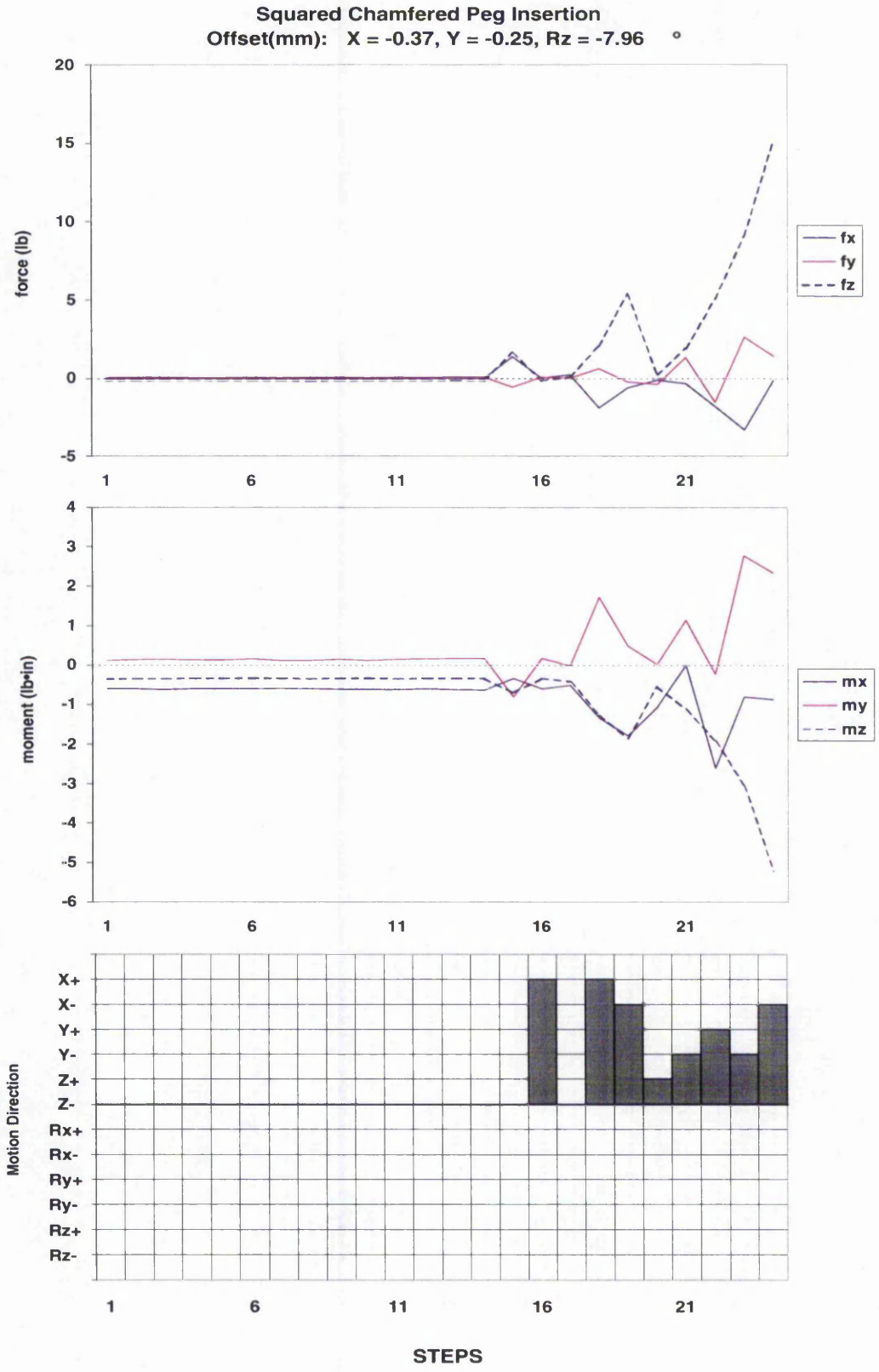


Figure A.16: Sixth Square Chamfered Peg Insertion

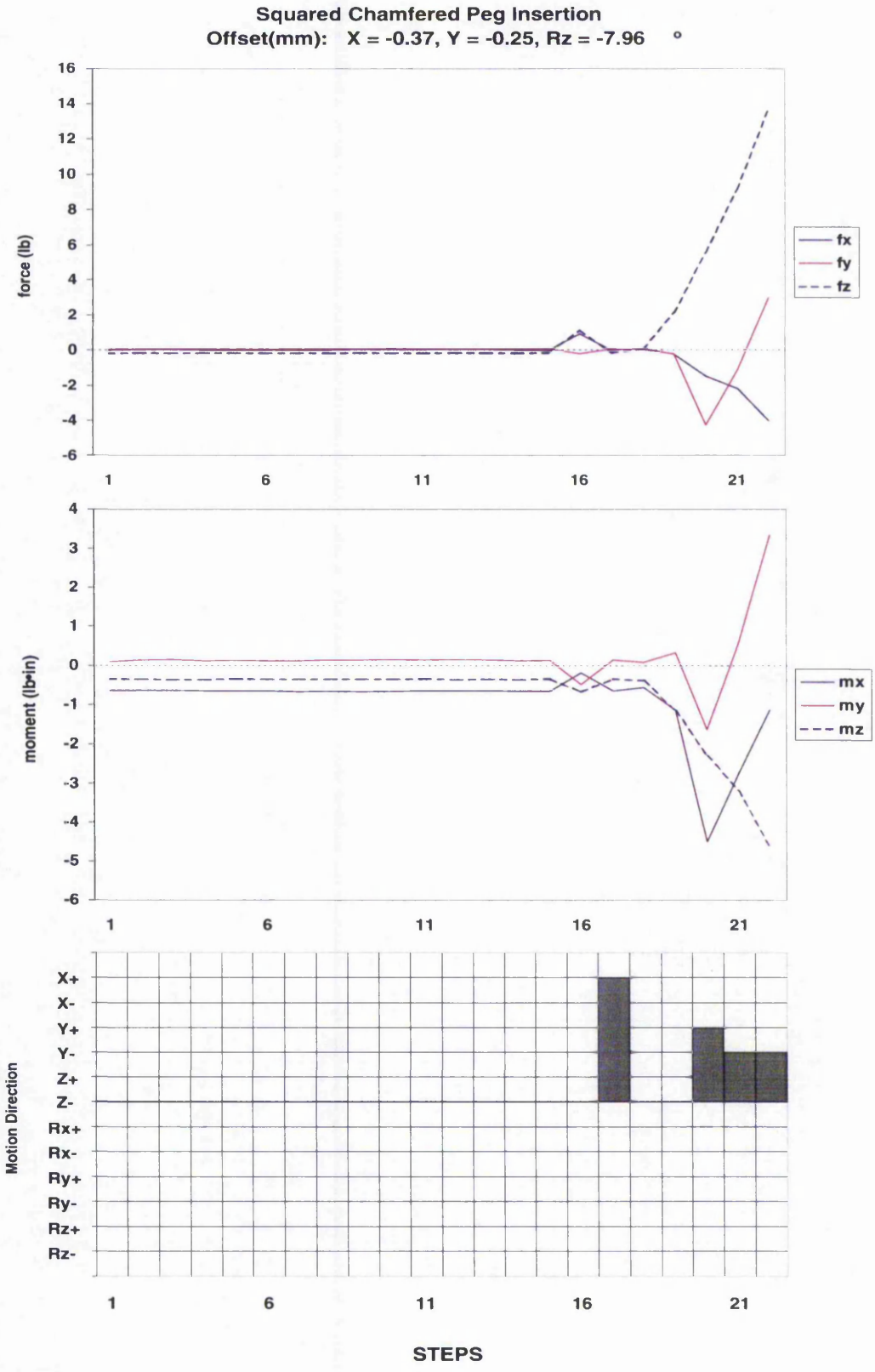


Figure A.17: Seventh Square Chamfered Peg Insertion

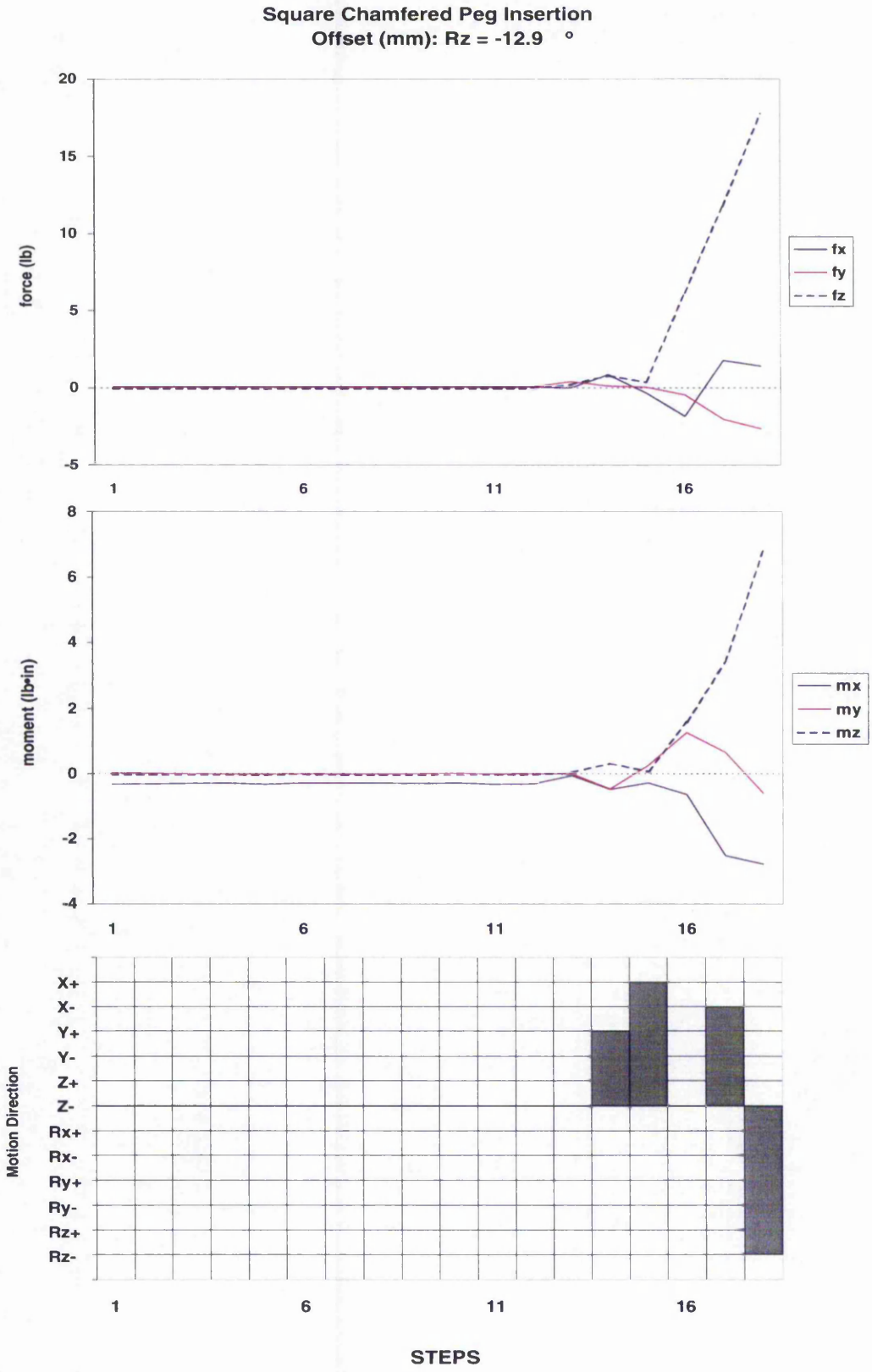


Figure A.18: Eighth Square Chamfered Peg Insertion

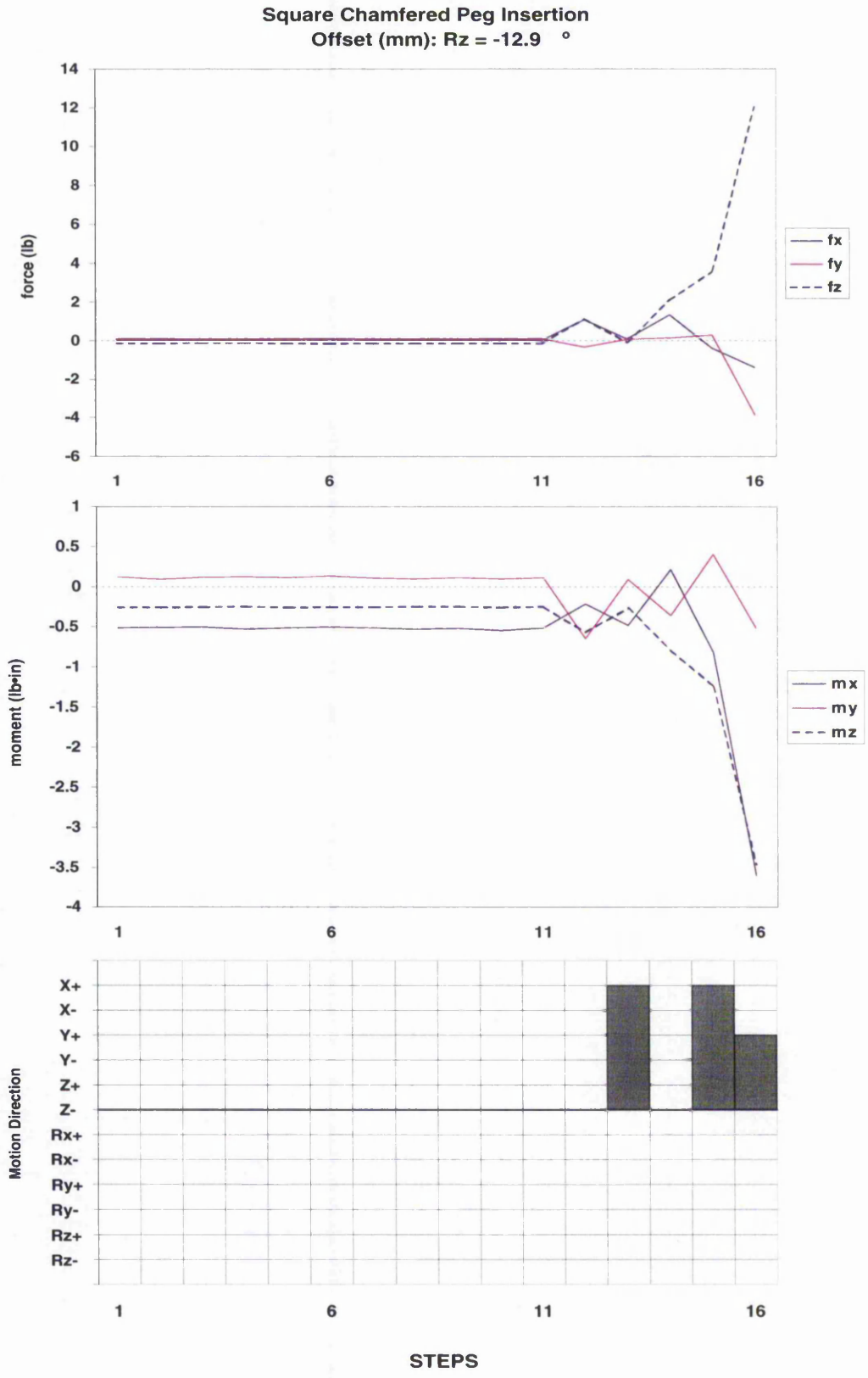


Figure A.19: Nineth Square Chamfered Peg Insertion

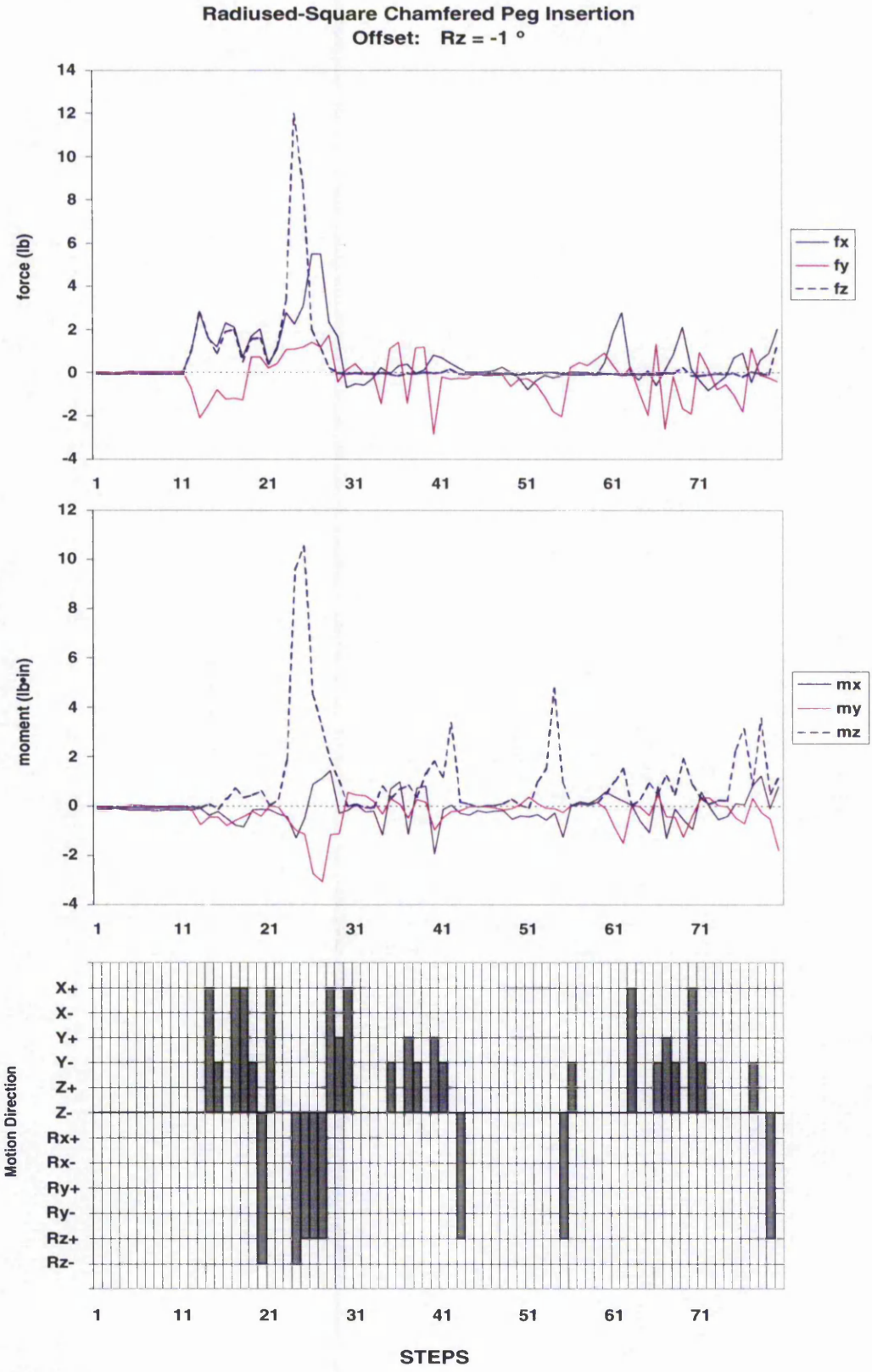


Figure A.20: 1st Angular misalignment

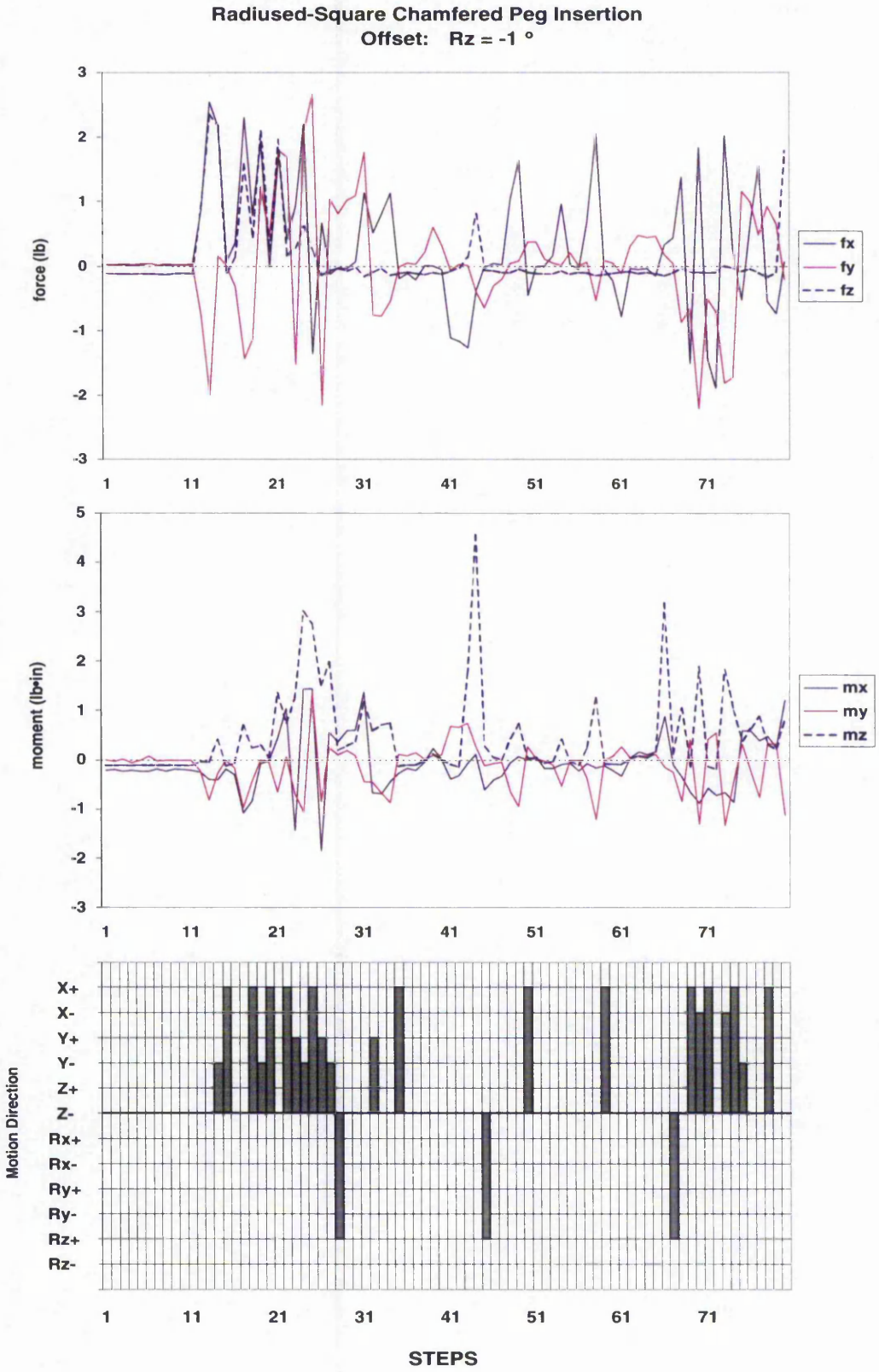


Figure A.21: 2nd Angular misalignment

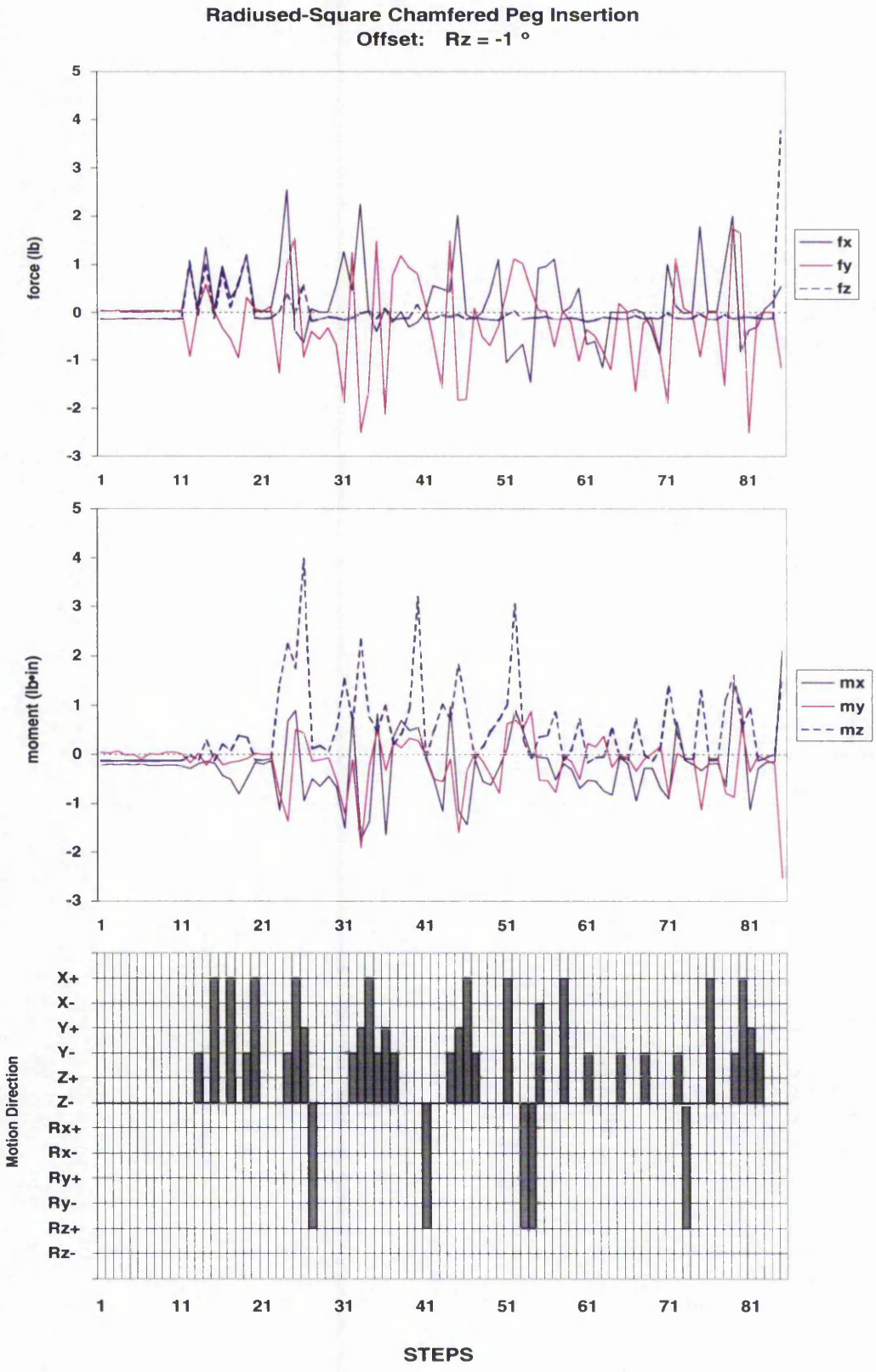


Figure A.22: 3rd Angular misalignment

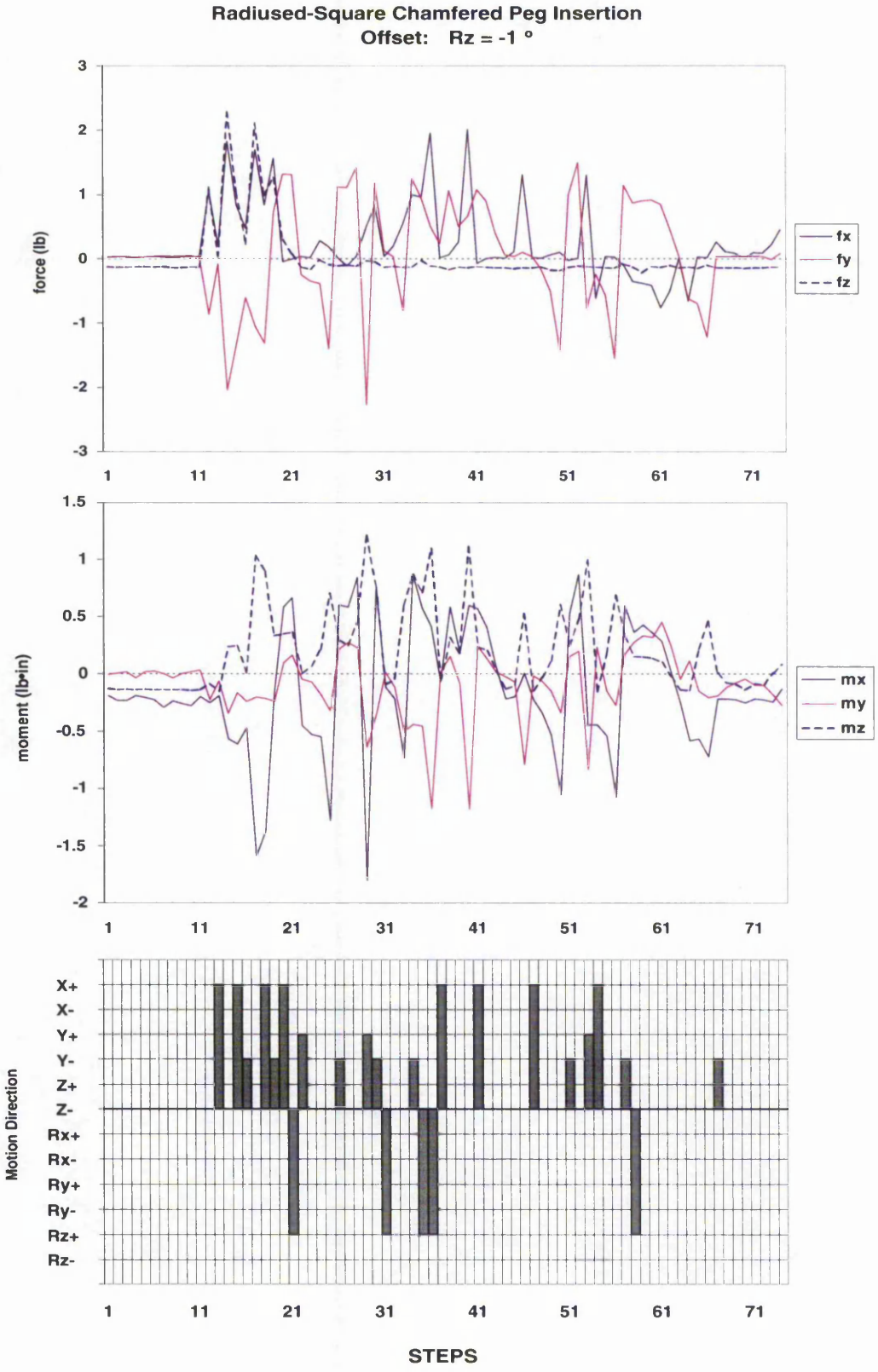


Figure A.23: 4th Angular misalignment

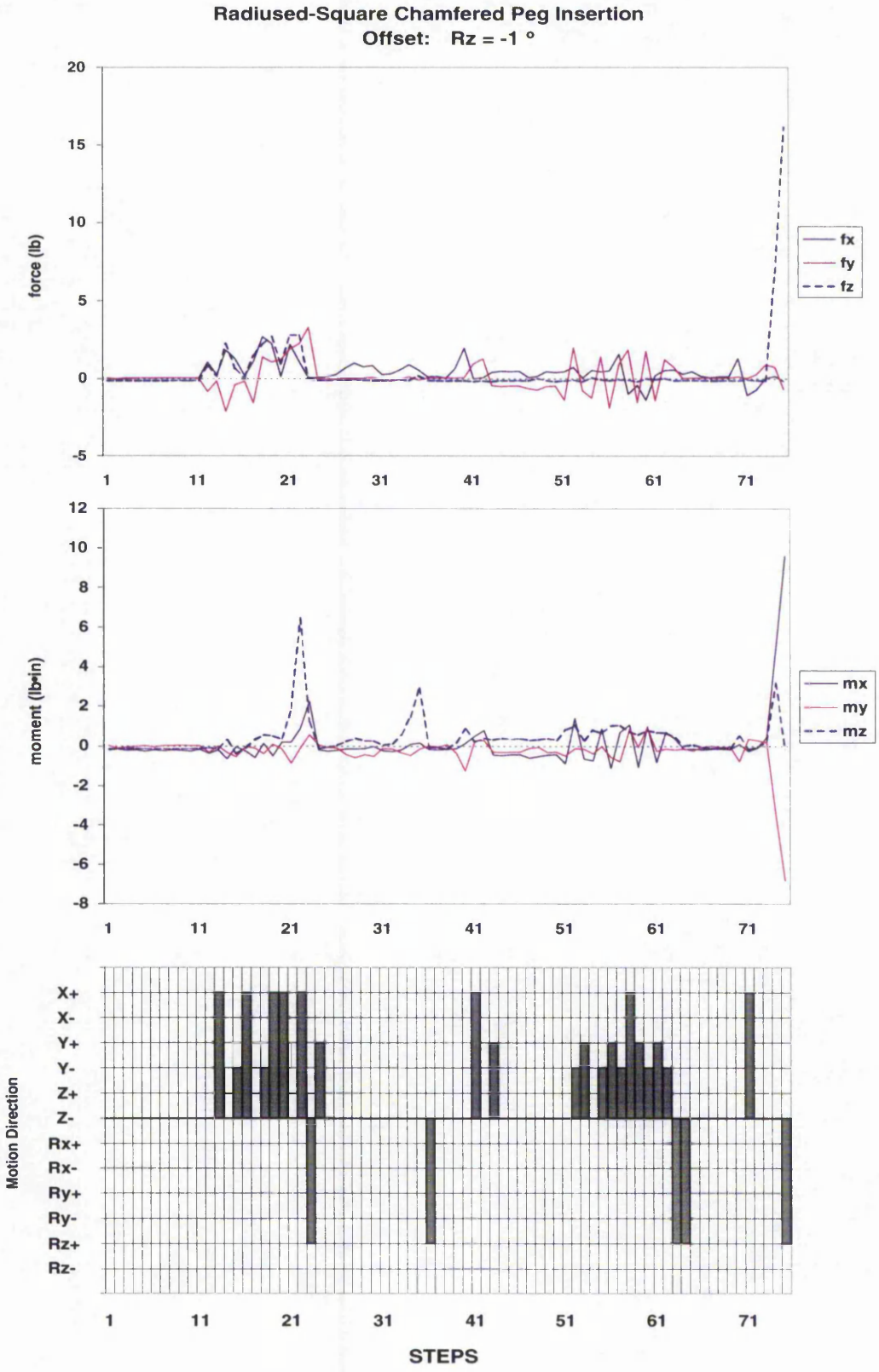


Figure A.24: 5th Angular misalignment

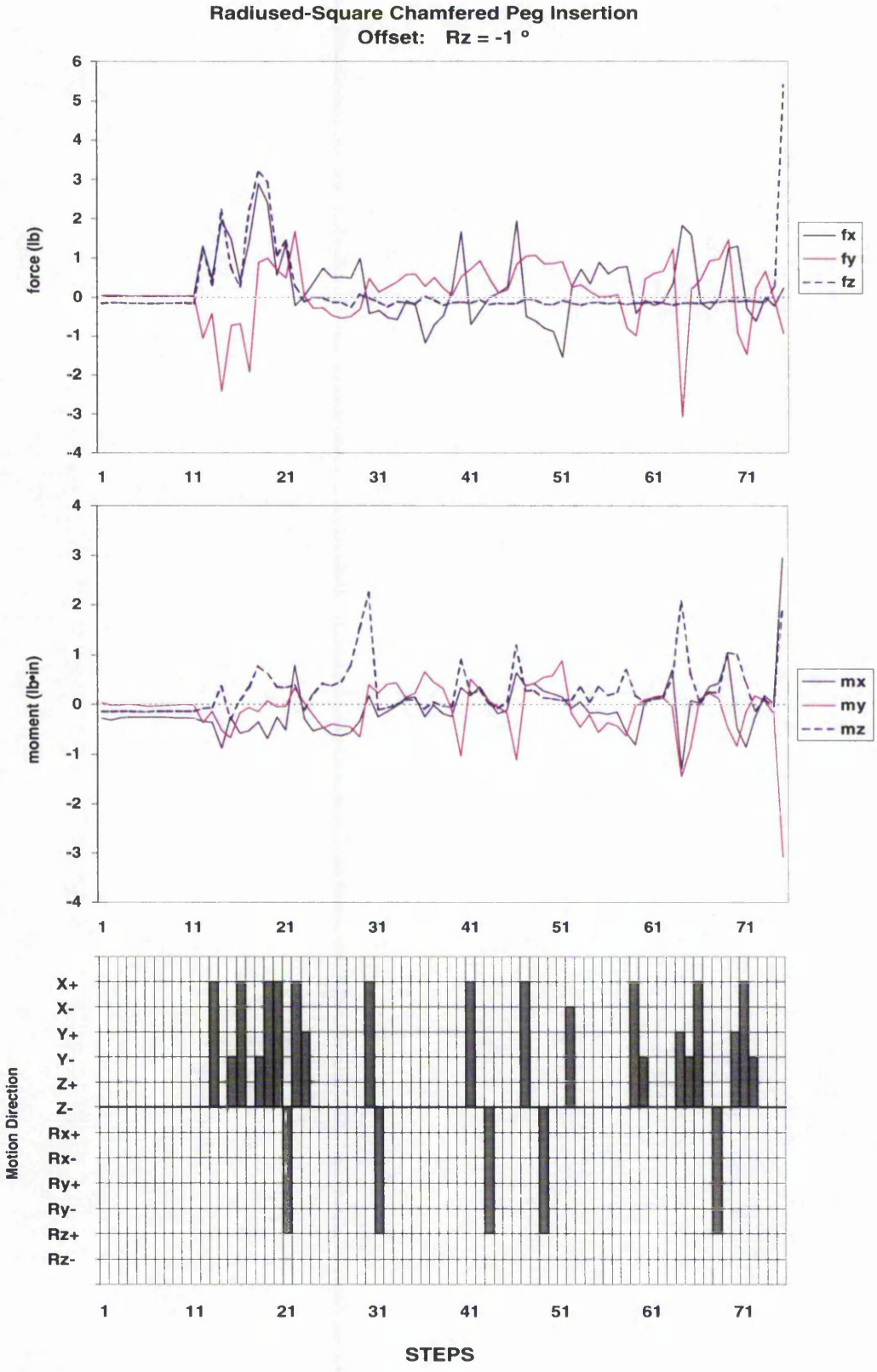


Figure A.25: 6th Angular misalignment

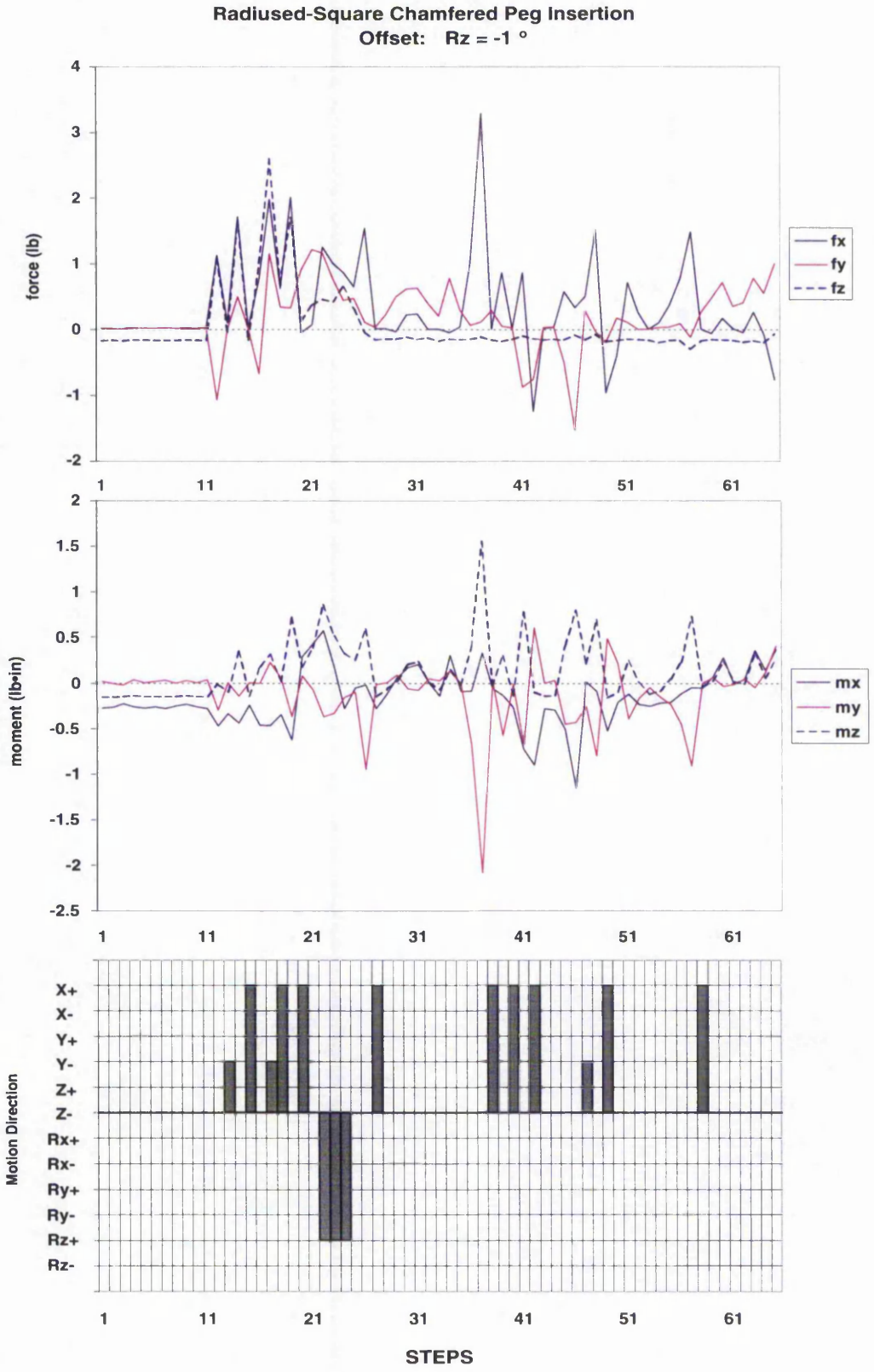


Figure A.26: 7th Angular misalignment

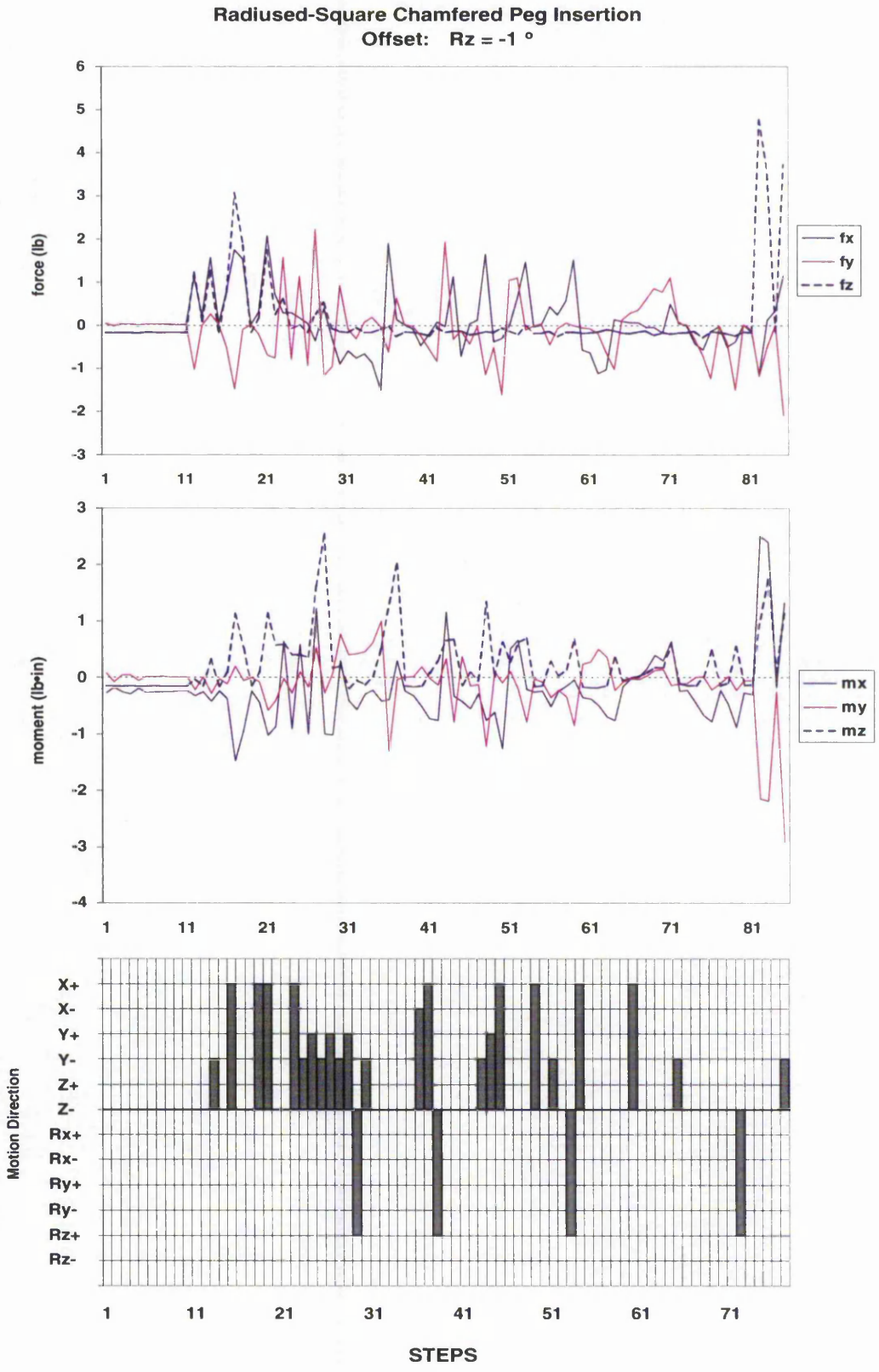


Figure A.27: 8th Angular misalignment

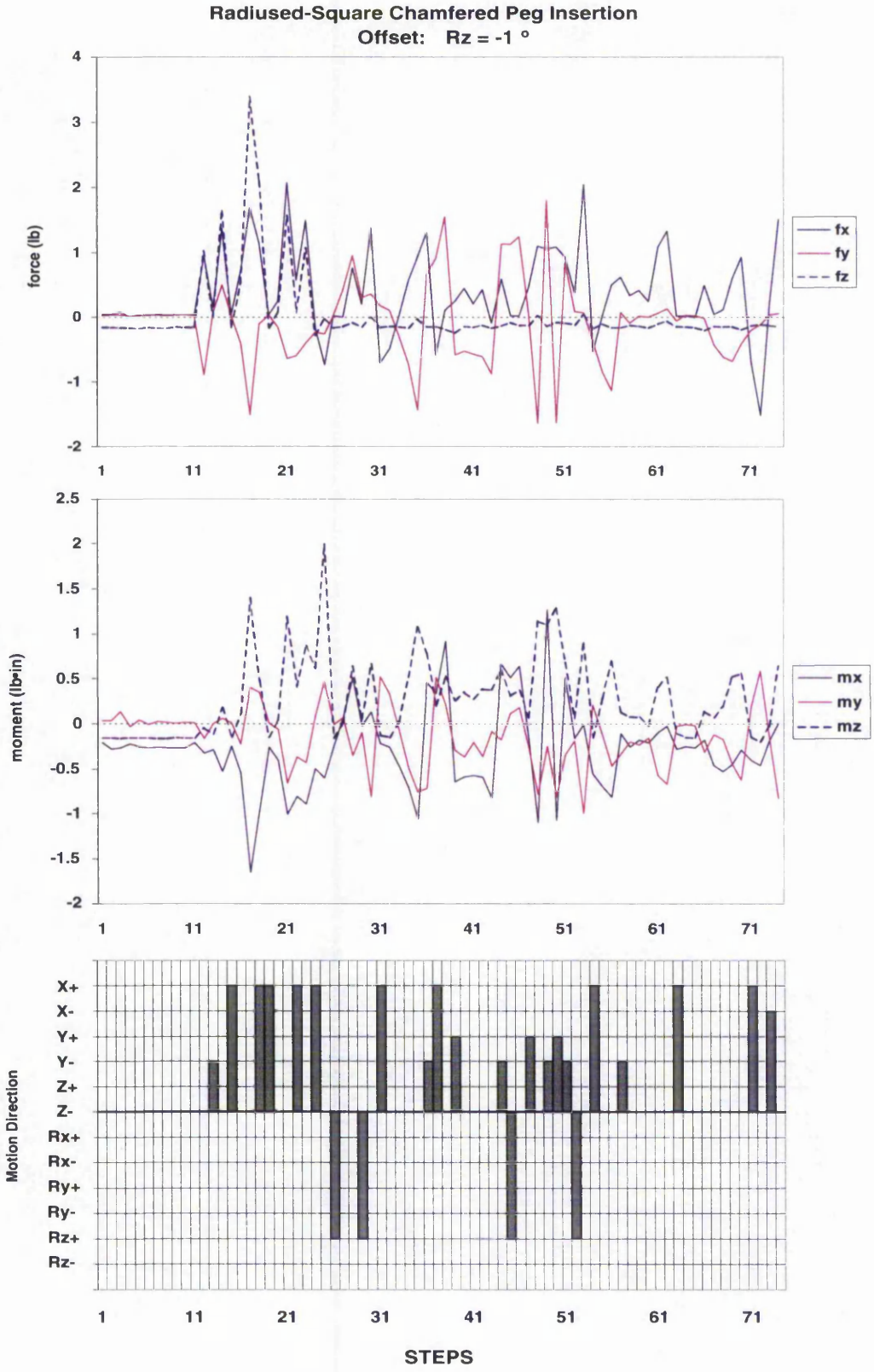


Figure A.28: 9th Angular misalignment

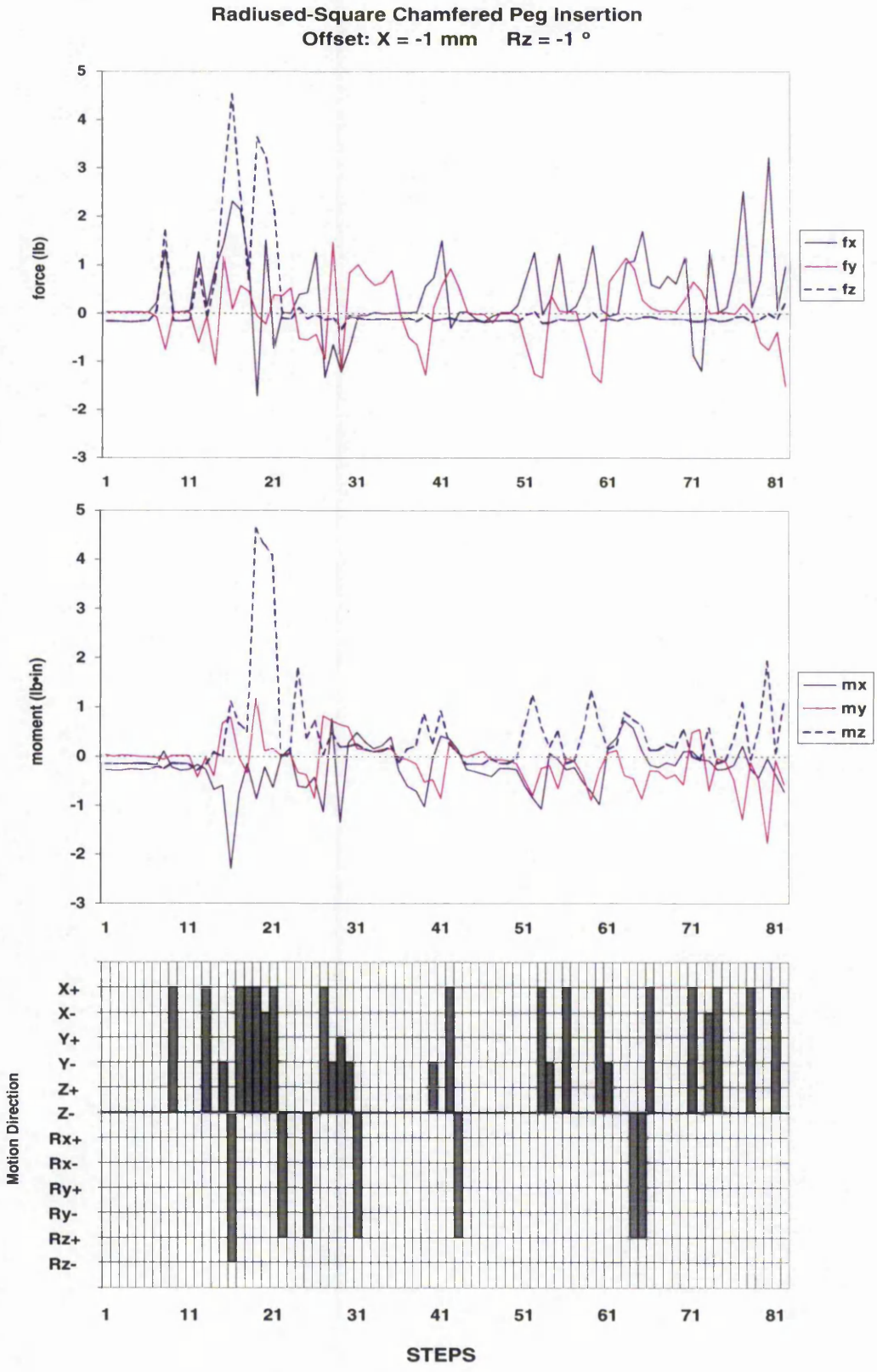


Figure A.29: 1st Angular and linear misalignment

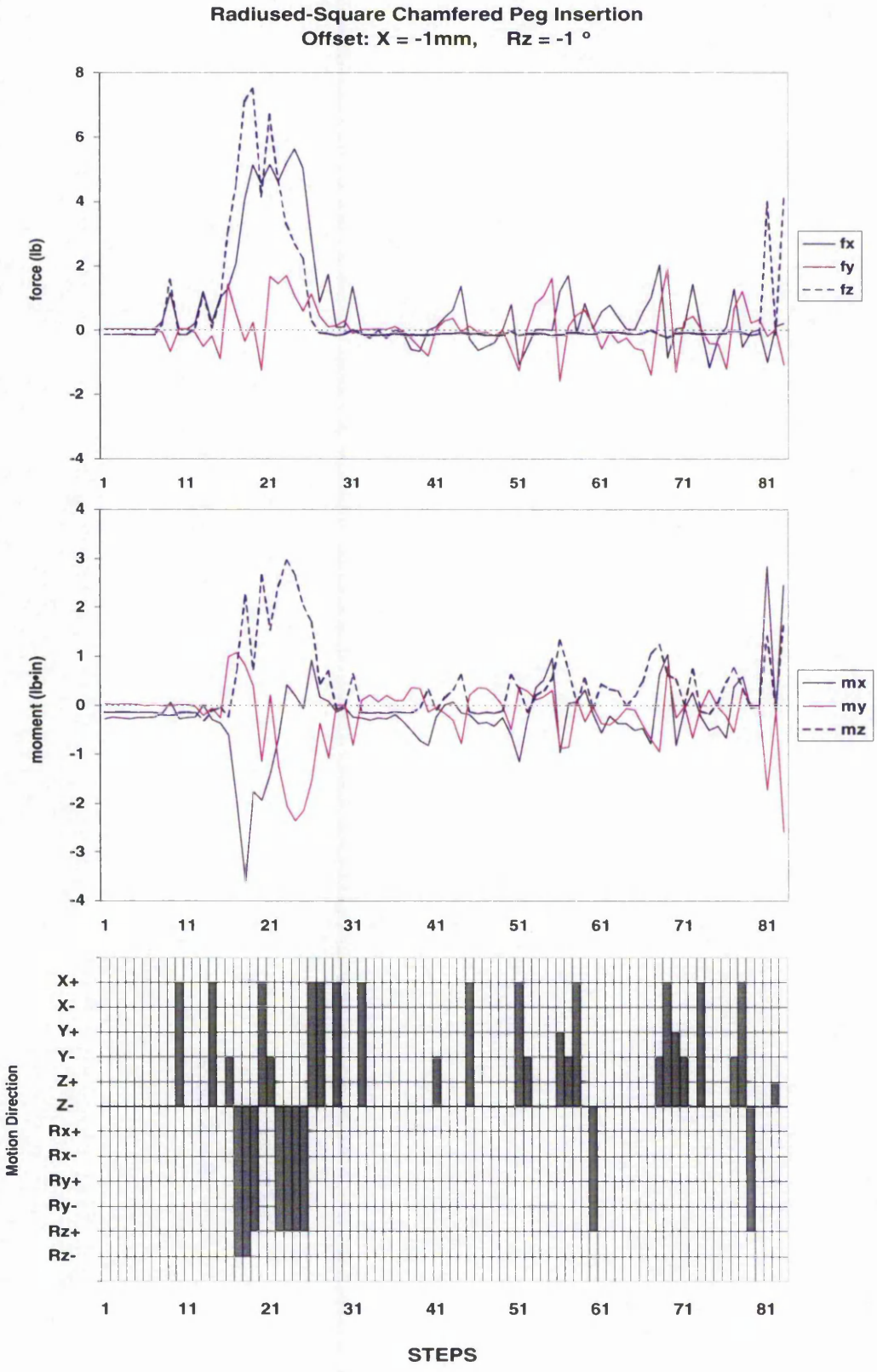


Figure A.30: 2nd Angular and linear misalignment

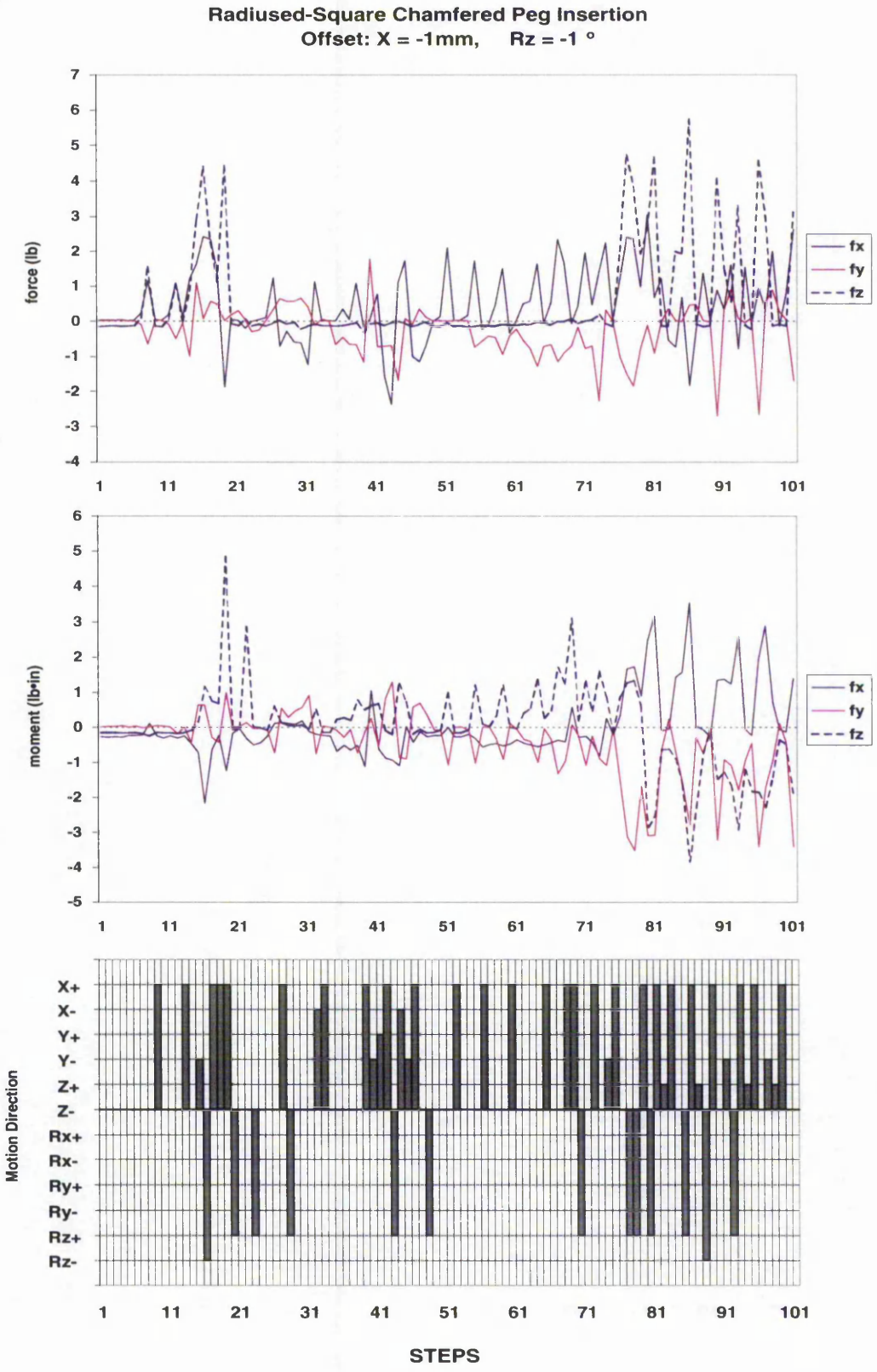


Figure A.31: 3rd Angular and linear misalignment

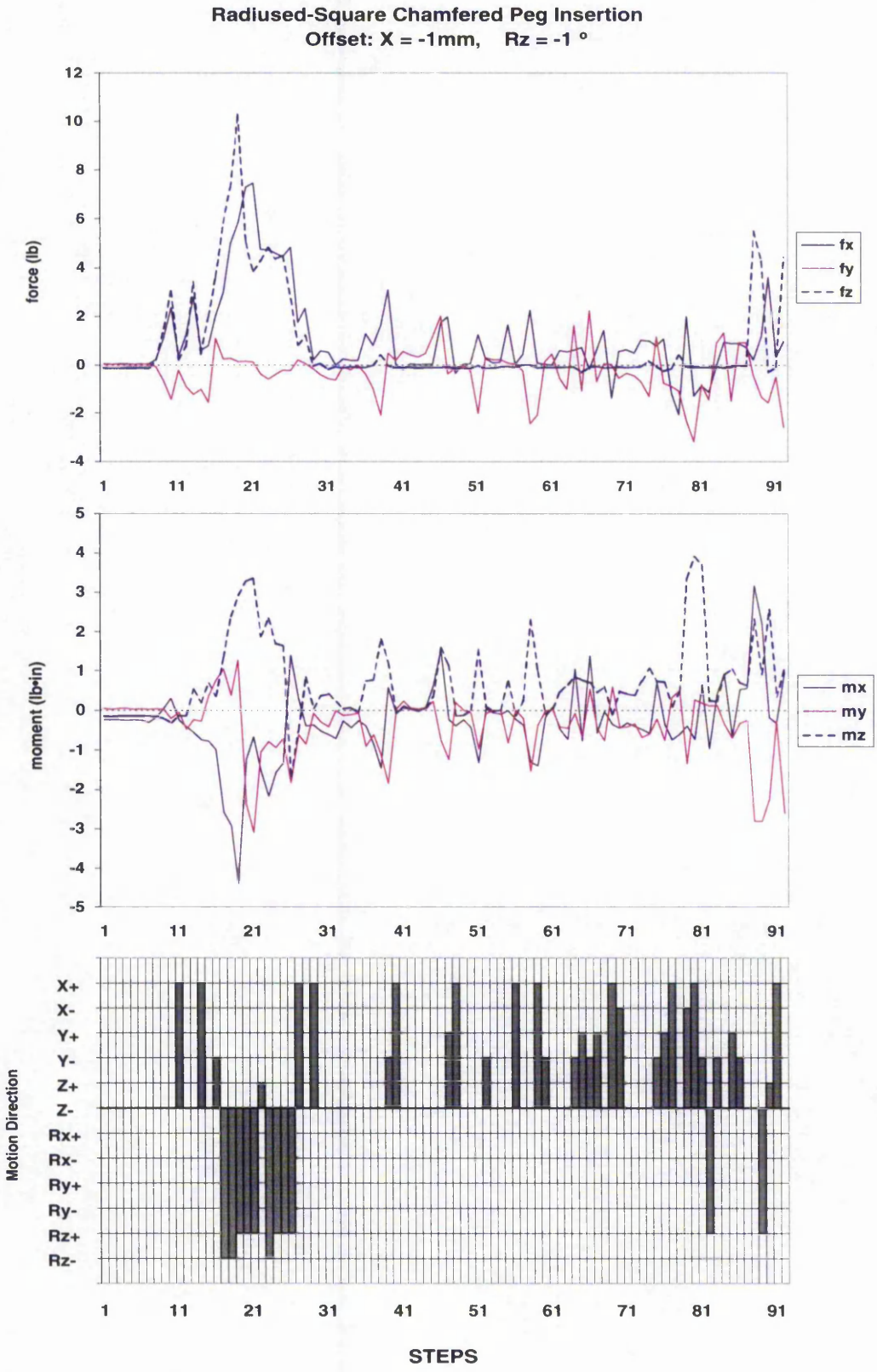


Figure A.32: 1st insertion, Restarting the learning

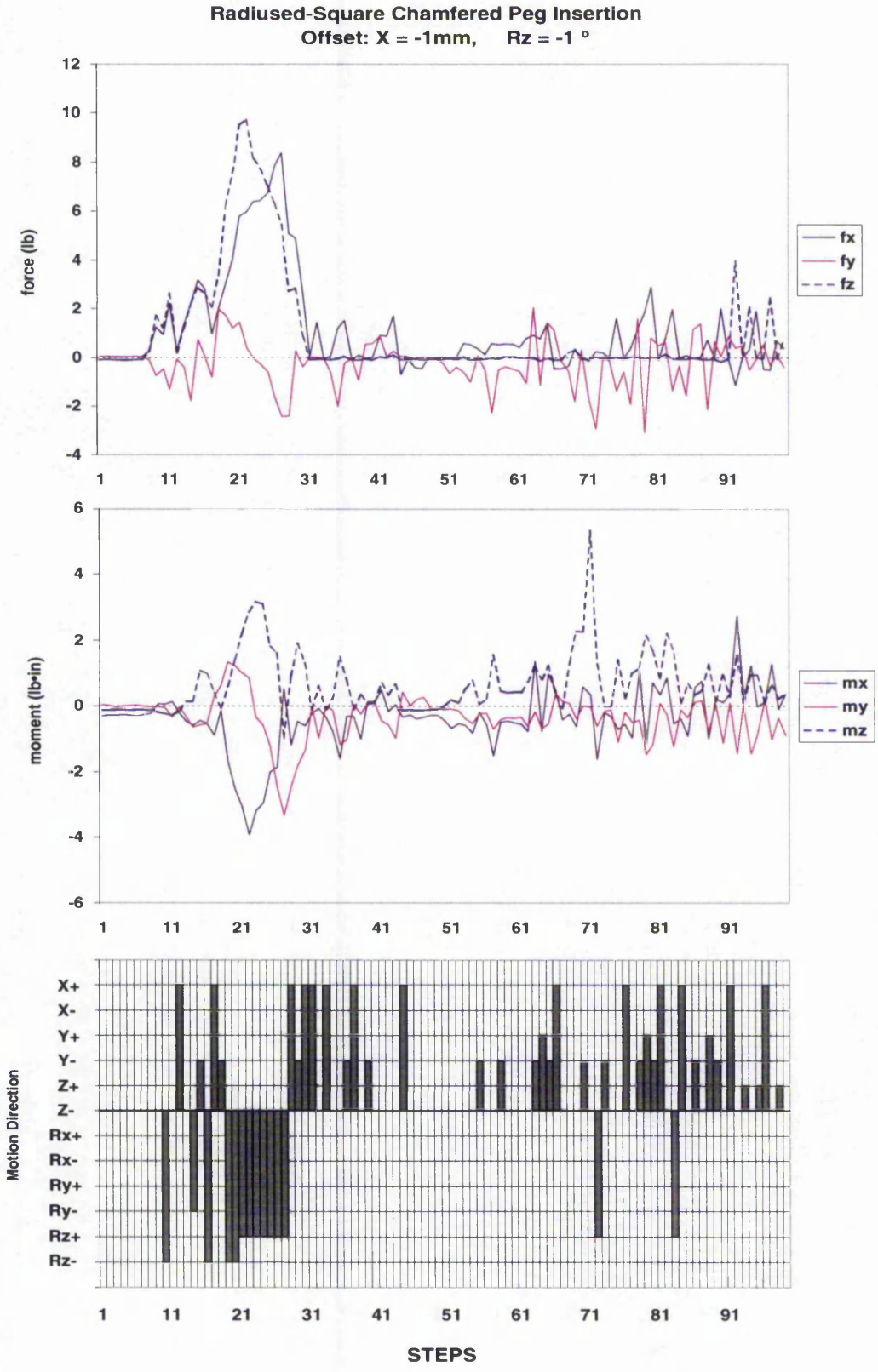


Figure A.33: 2nd insertion, Restarting the learning

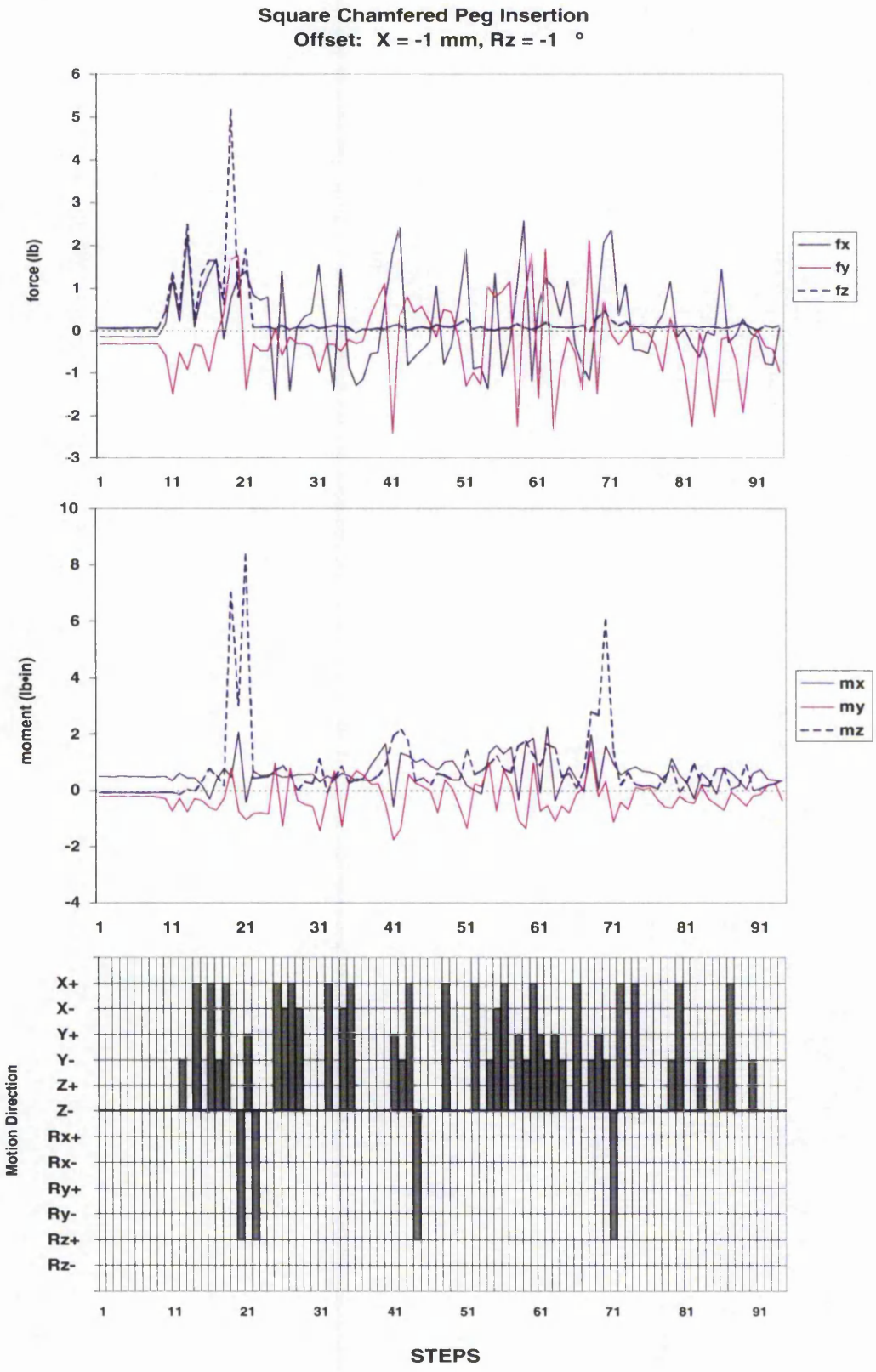


Figure A.34: Same EKB different geometry

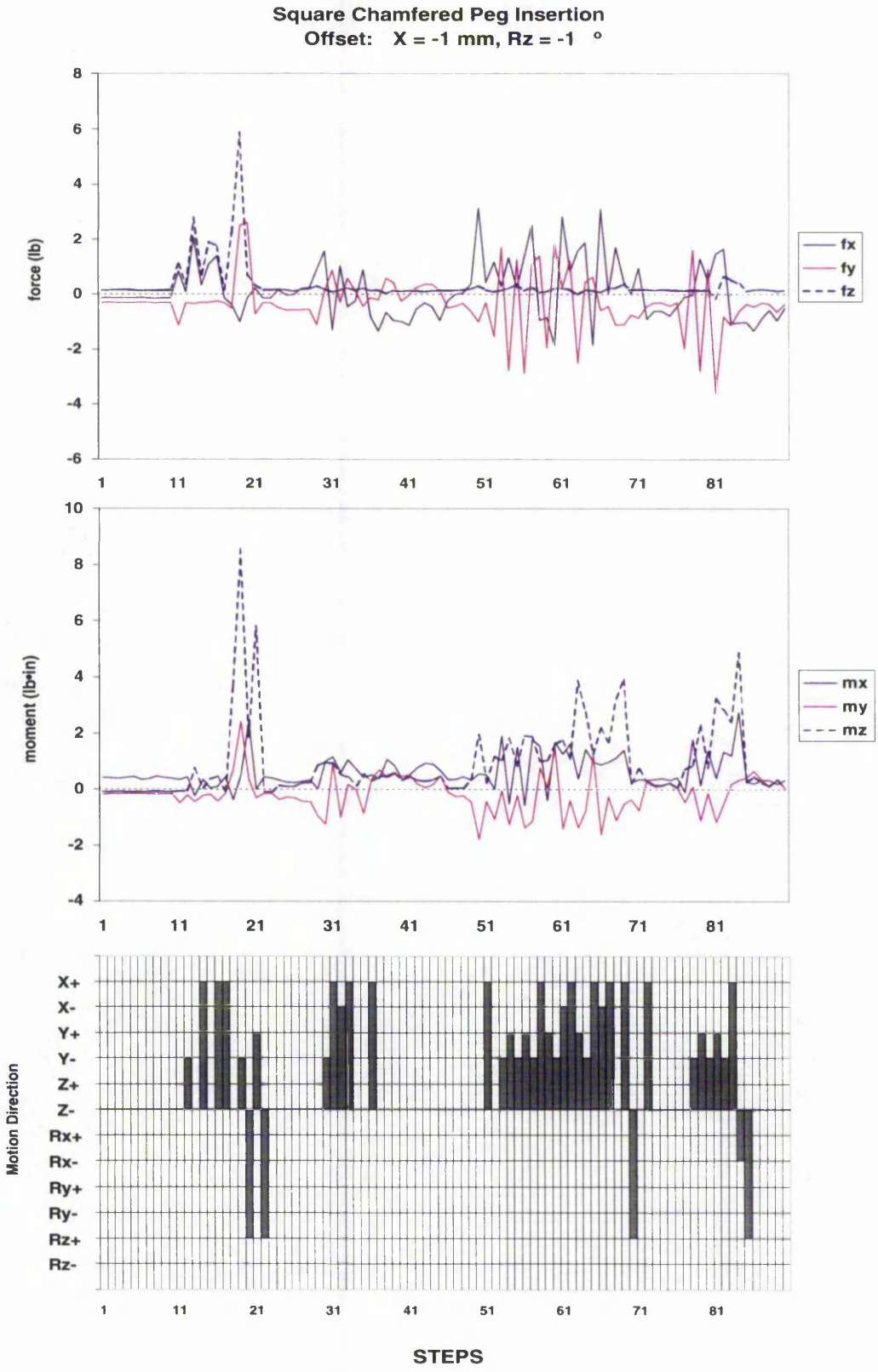


Figure A.35: Same EKB different geometry

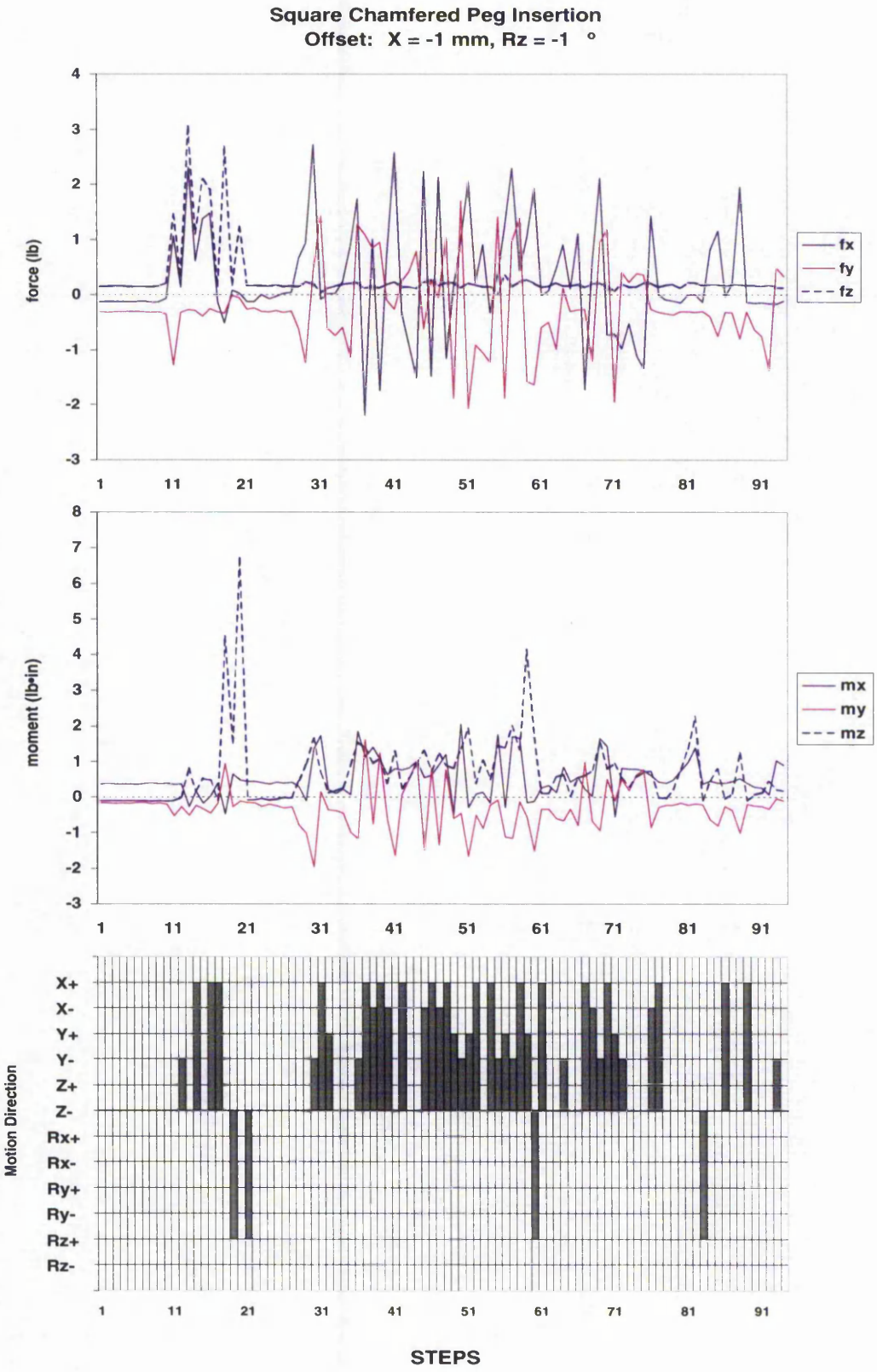


Figure A.36: Same EKB different geometry

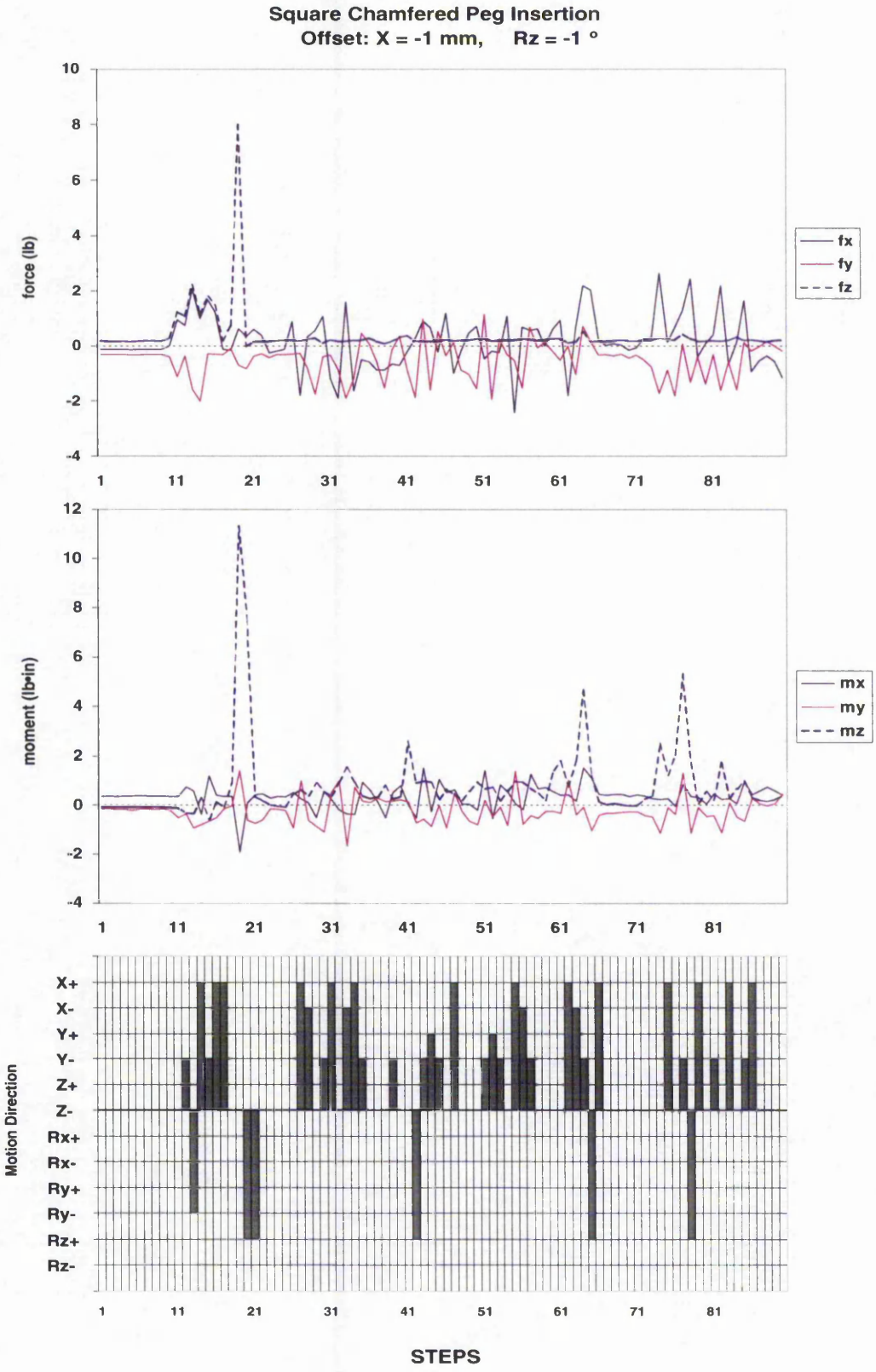


Figure A.37: Same EKB different geometry

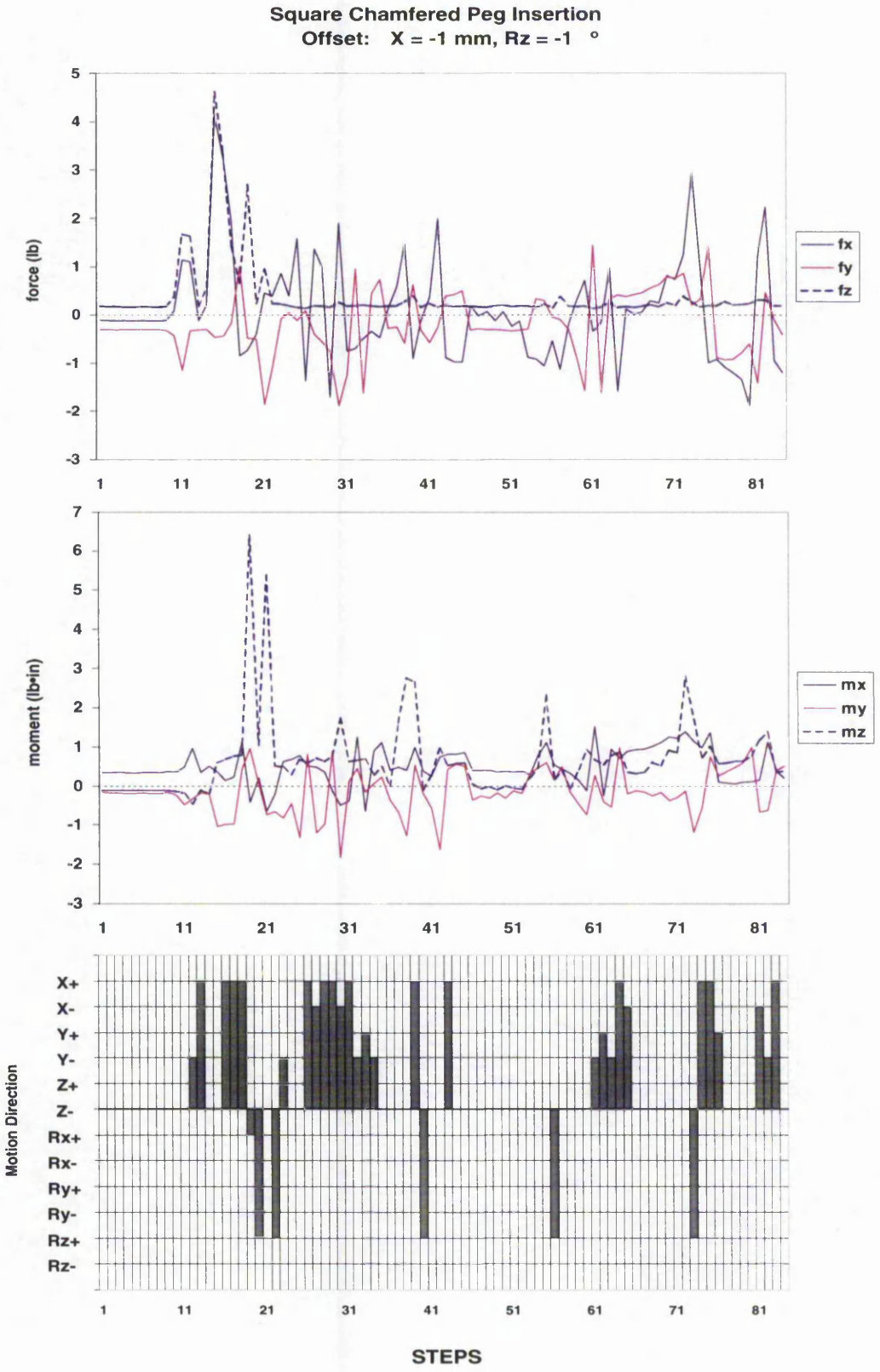


Figure A.38: Same EKB different geometry

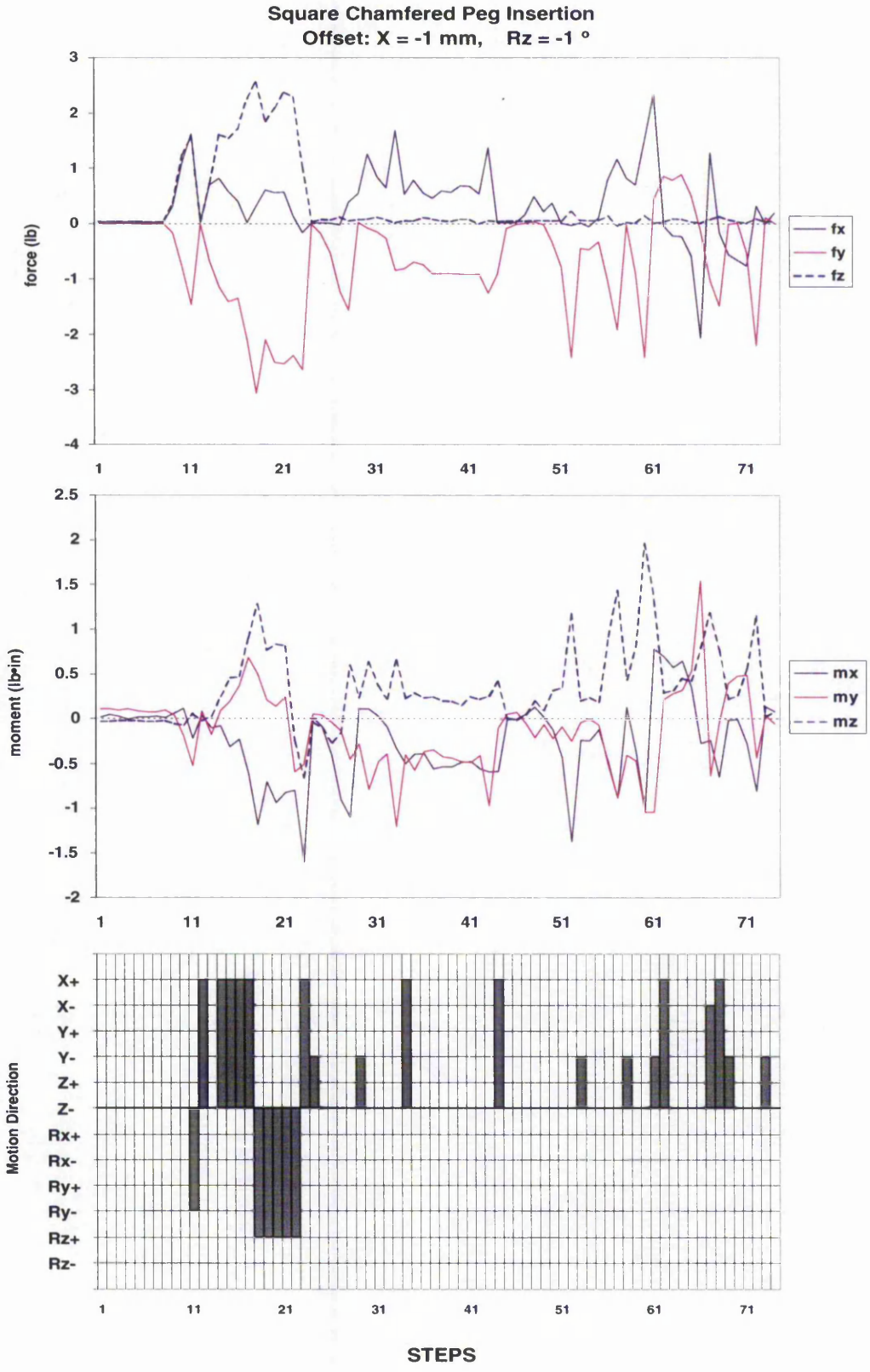


Figure A.39: Restarting the learning

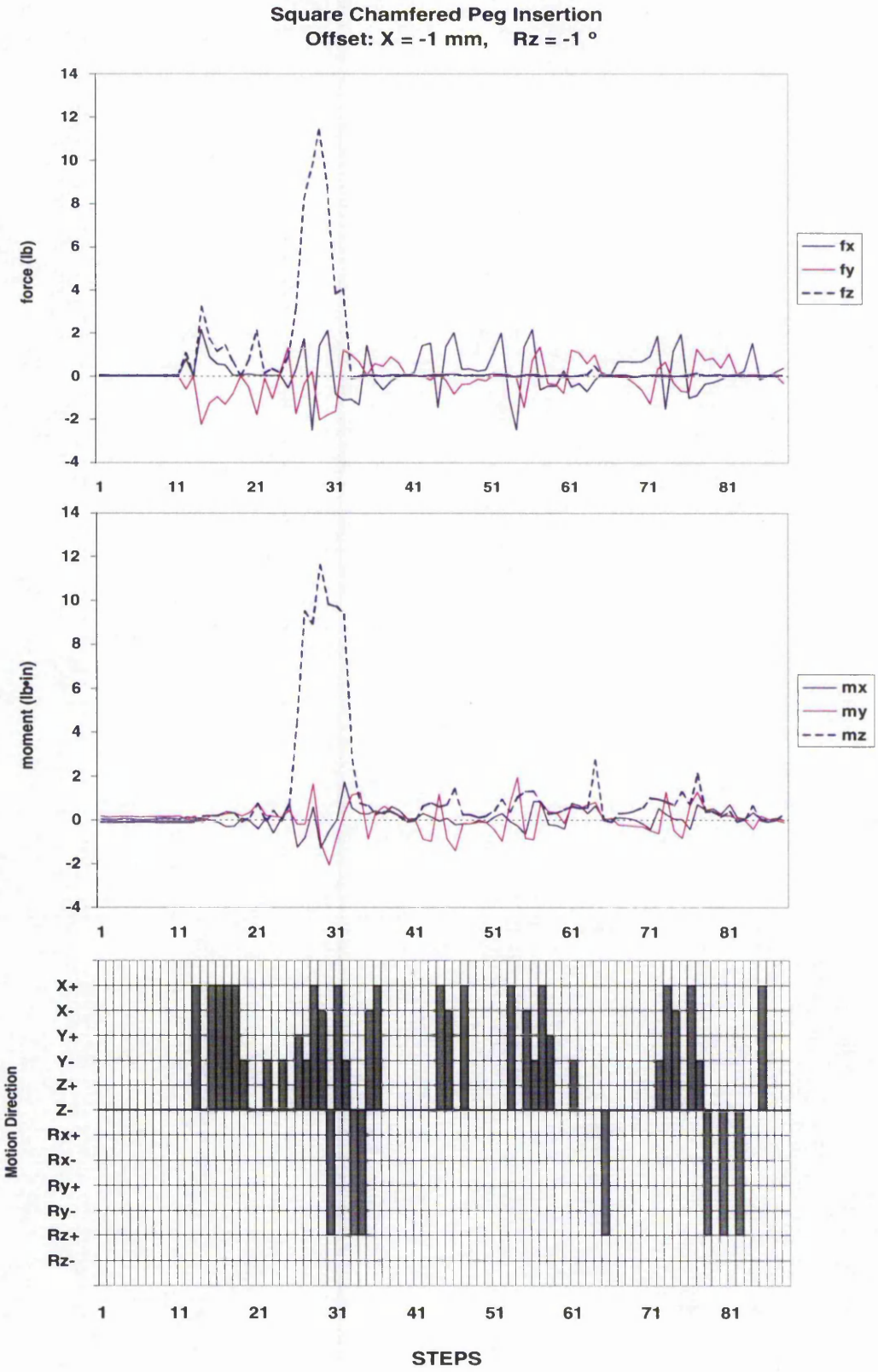


Figure A.40: Restarting the learning

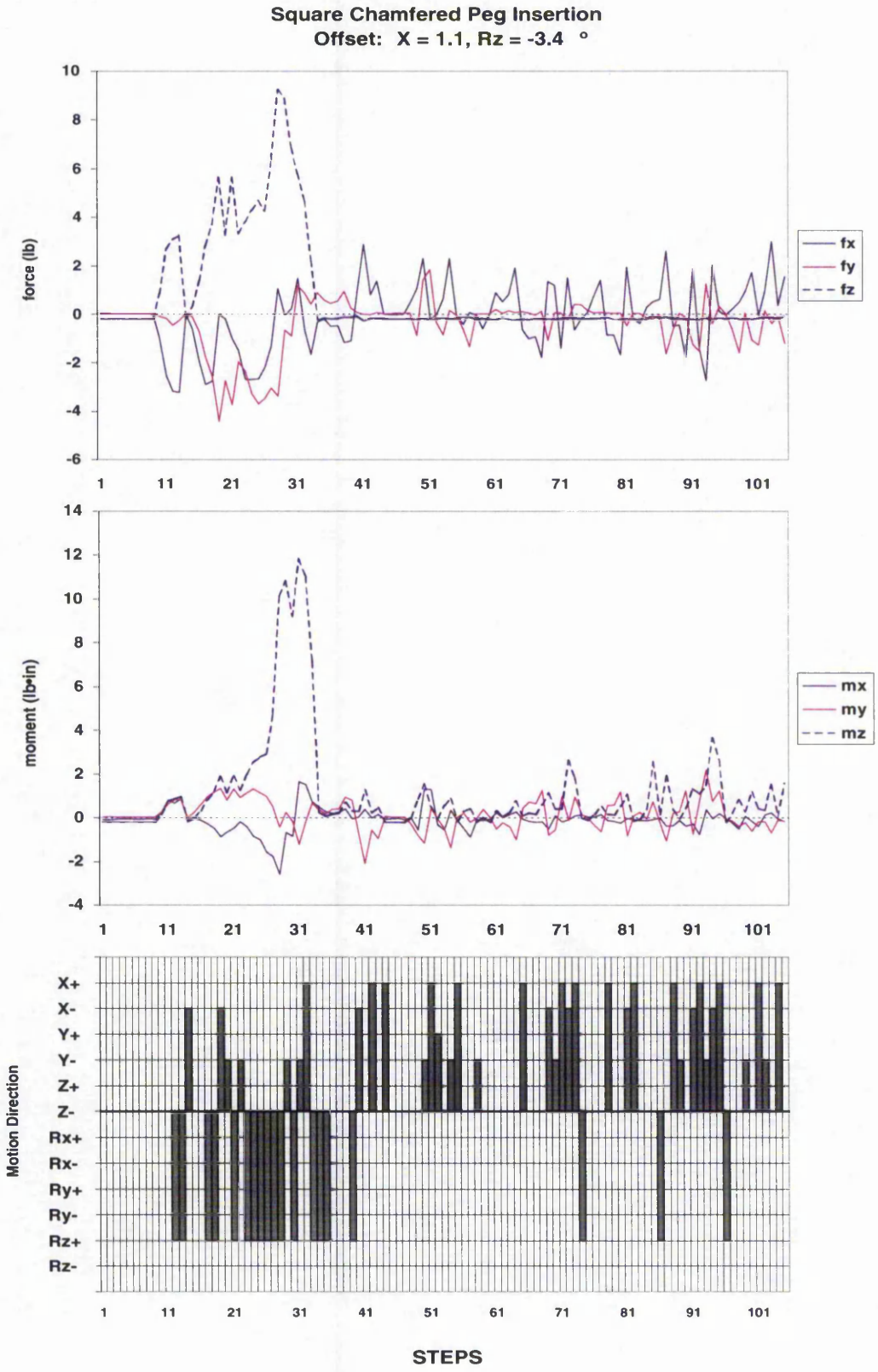


Figure A.41: Increasing the angular misalignment

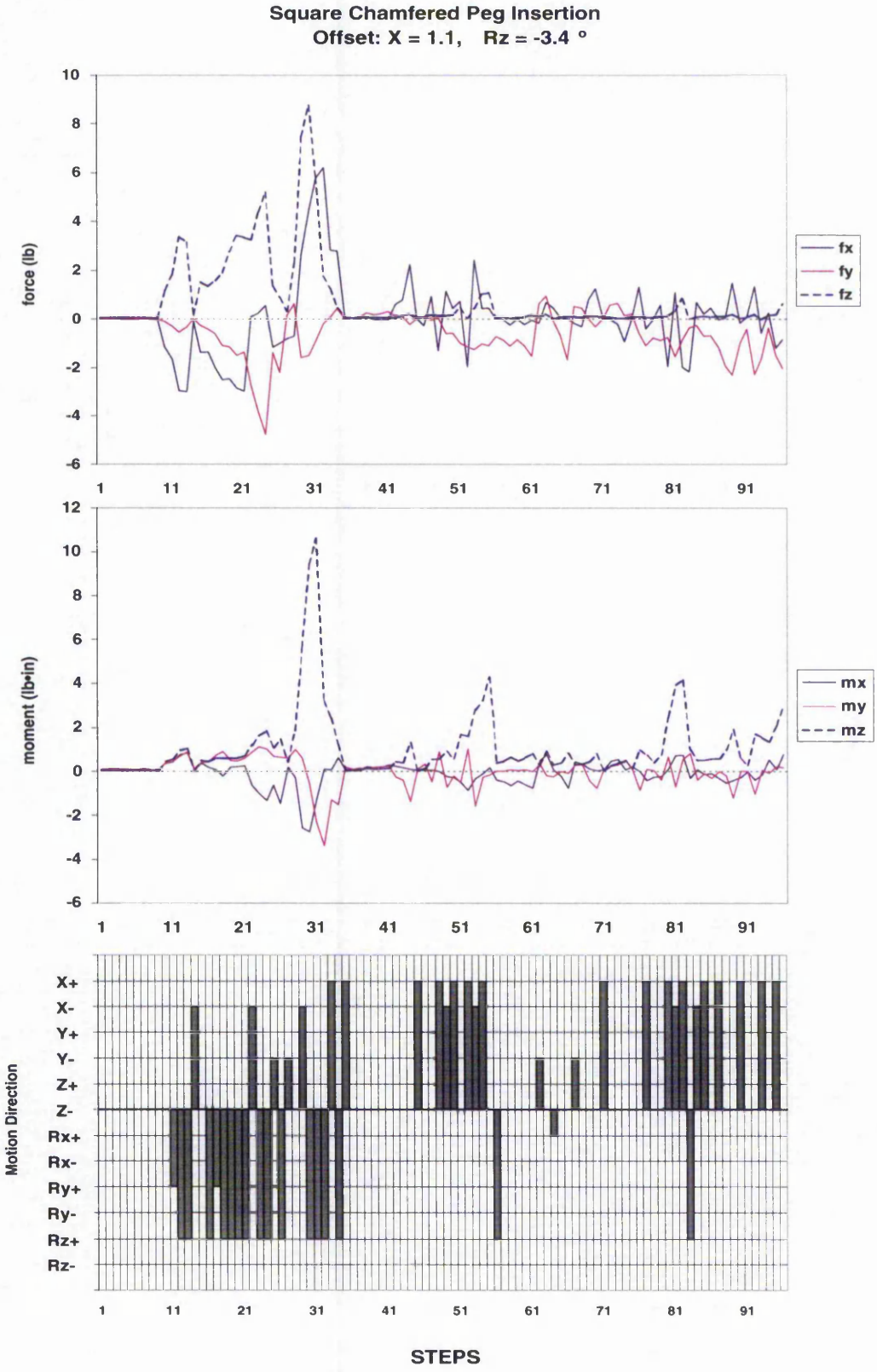


Figure A.42: Increasing the angular misalignment

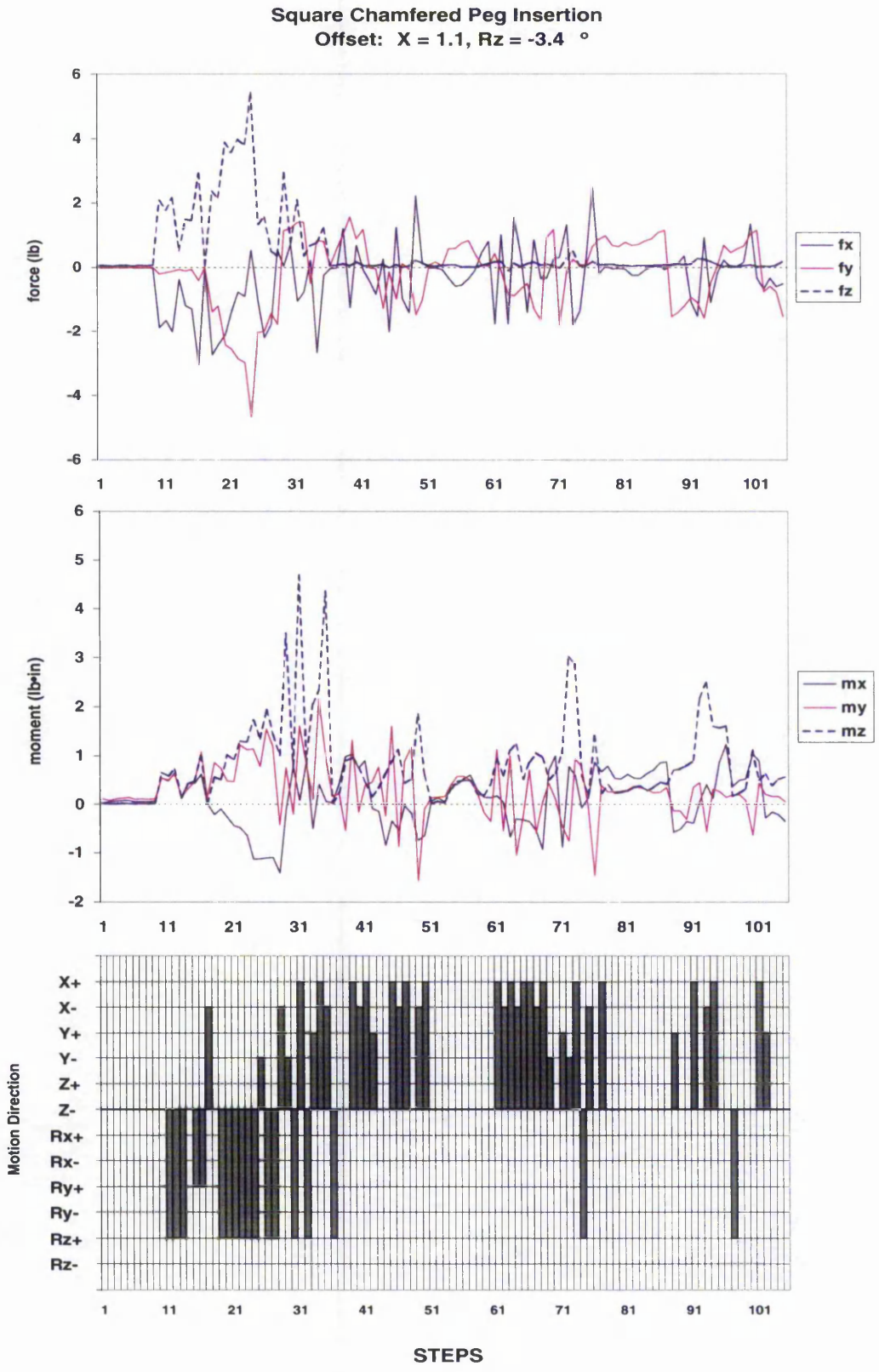


Figure A.43: Increasing the angular misalignment

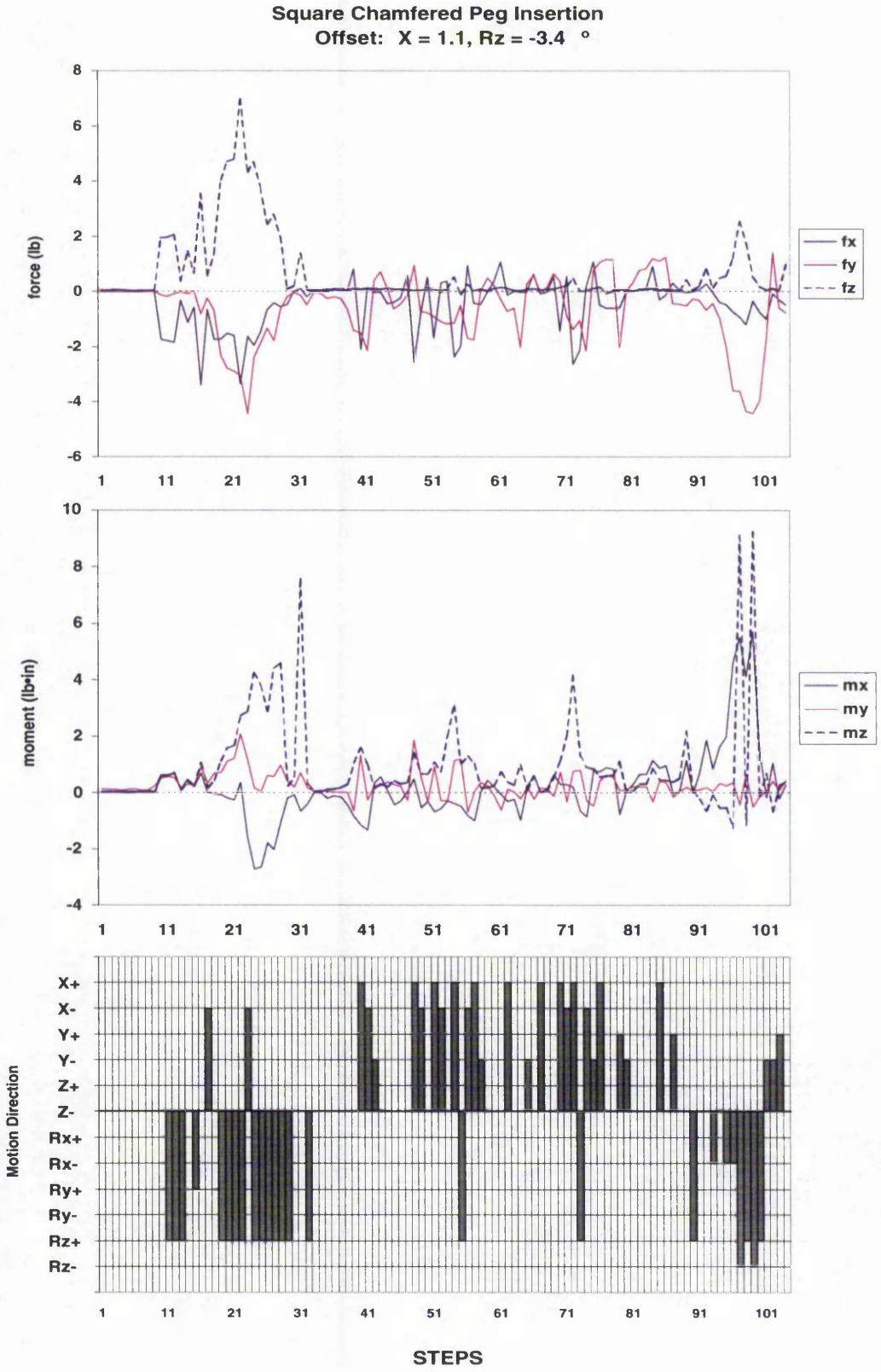


Figure A.44: Increasing the angular misalignment

Appendix B

Adaptive Resonance Theory Algorithms

The algorithms that describe the dynamics of the ART models and upon which the NNC was built are included in this appendix, e.g. ART-1, FuzzyART and Fuzzy ARTMAP.

ART-1

Figure B.1 illustrates the main components of an ART-1 module. It consists of two layers of neurons or nodes. F_1 with output vector $X \equiv (x_1, ..., x_M)$, registers the $F_0 \rightarrow F_1$ input vector $I \equiv (I_1, \ldots, I_M)$. Each neuron in the layer F_1 is connected to every neuron in the F_2 layer through the bottom-up synaptic adaptative weights Z_{ij}. The index i indicates that the connection goes from the i_{th} neuron in F_1 to the j_{th} neuron in the F_2 layer.

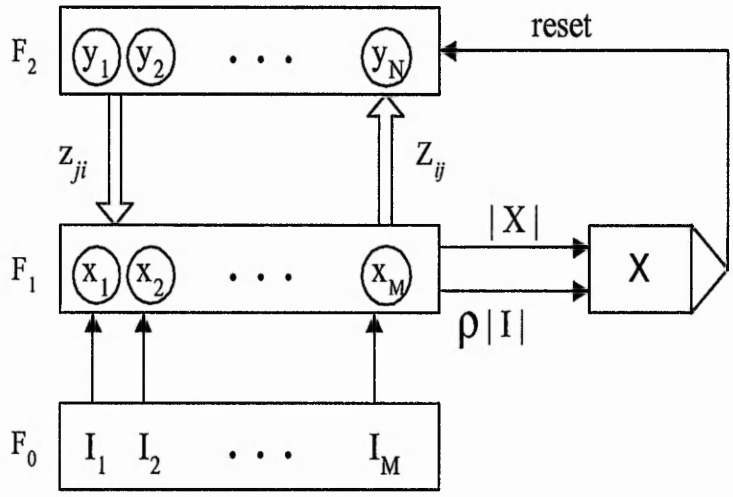


Figure B.1: ART-1 Architecture

$\mathbf{F_2}$ Choice

Let T_j denote the total input from F_1 to the j_{th} F_2 node given by,

$$T_j = \sum_{i=1}^{M} x_i Z_{ij}, \qquad j = 1, \ldots, N \tag{B.1}$$

If some $T_j > 0$, then the F_2 choice index J is:

$$T_J = max\{T_j : j = 1 \ldots N\}. \tag{B.2}$$

J is uniquely defined so that the output of the F_2 layer is 'zero' except for the node with maximum activation. In this manner, $y = (y_1, \ldots, y_N)$ has an output

$$y_j = \begin{cases} 1 & \text{if } j = J \\ 0 & \text{if } j \neq J \end{cases}$$

Resonance or reset

Each node of F_2 is connected to all F_1 nodes through the top-down connections of strength z_{ji}, which contain binary values. Thus, the i_{th} F_1 node input from the F_2 layer is

$$V_i = \sum_{j=1}^{N} z_{ji} y_j, \qquad i = 1, \ldots, M \tag{B.3}$$

There is a vigilance subsystem formed by the comparator in Figure B.1, which controls how much mismatch is tolerated between the bottom-up activity and the top-down learned expectations. In other words it compares the norm of vector X to the norm of vector $\rho\ \mathbf{I}$, where $\rho \in [0,1]$ is the *vigilance* parameter.

The norm of a vector $a = (a_1, \ldots, a_M)$ can be obtained by:

$$\mid a \mid = \sum_{i=1}^{M} \mid a_i \mid \tag{B.4}$$

vector X is defined as:

$$x_i = V_i I_i \qquad or \qquad X = V \cap I = z_J \cap I \tag{B.5}$$

Depending on the result of this comparison the vigilance subsystem may reset the actual active F_2 category and search for another category or update the LTM traces (weights).

Learning

Learning then ensues if the vigilance parameter is met for the chosen category J. That is, if

$$\frac{|\mathbf{I} \cap \mathbf{z}_J|}{|\mathbf{I}|} \geq \rho \tag{B.6}$$

The connections in the bottom-up and top-down weights are updated as follows:

$$z_J^{new} = I \cap z_J^{old} \tag{B.7}$$

$$Z_J^{new} = \frac{L z_J^{new}}{L - 1 + \mid z_J^{new} \mid} \tag{B.8}$$

The above equations are for *fast learning* that use an algebraic form of the non-linear differential equations for the LTM [1]. Note that only two parameters are needed for the implementation of ART-1, ρ and L. L can take a value larger than 1. The above steps are summarised in the flowchart of the algorithm given in Figure B.2.

[1] The reader is referred to [32] for a mathematical proof.

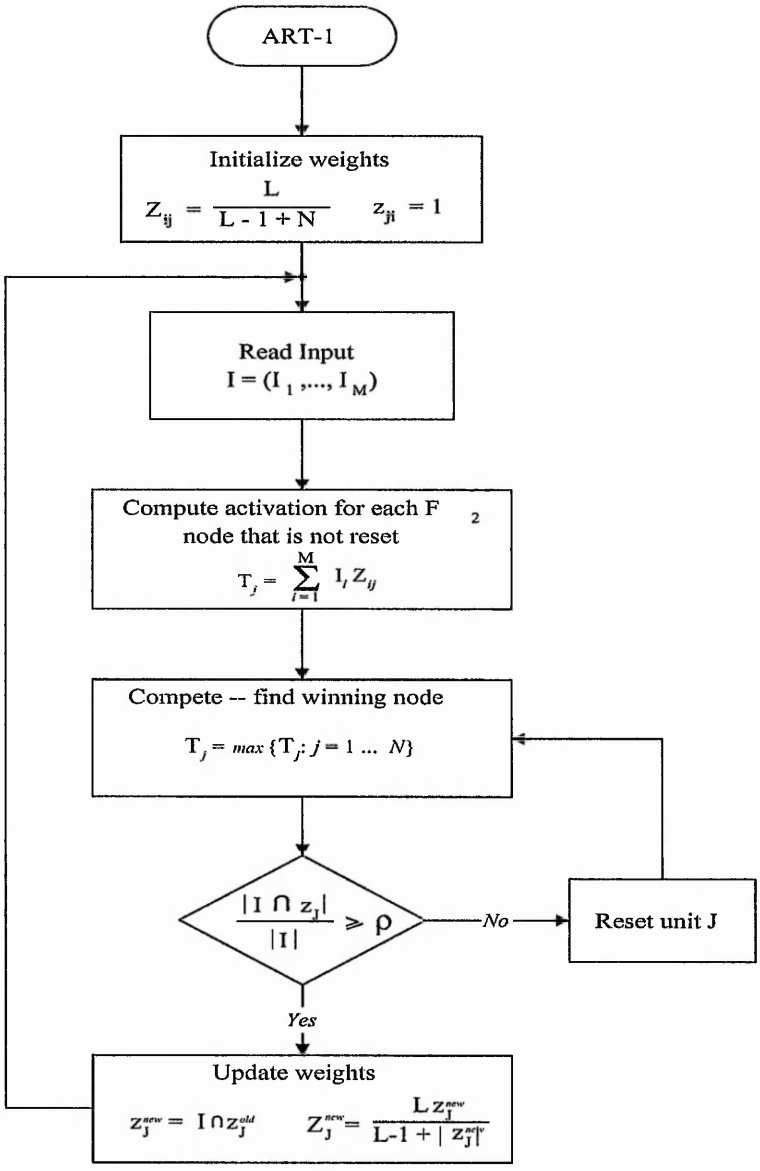


Figure B.2: Flowchart of the ART1 algorithm

Fuzzy ART (FA)

The FAM system incorporates two FA modules, ART_a and ART_b and it makes sense to describe first the key elements in the FA system, which are given in this section.

Each fuzzy ART subsystem includes a field, F_0, of nodes that represent a current input vector; a field, F_1, that receives both bottom-up input from F_0 and top-down input from a field; F_2, that represents the active code, or category. The F_0 activity vector is denoted $\mathbf{I} = (I_1, ...I_M)$, with each component I_i in the interval [0,1](i=1,...,M). The F_1 activity vector is denoted $\mathbf{x} = (x_1, ..., x_M)$, and the F_2 activity vector is denoted $\mathbf{y} = (y_1, ..., y_N)$. The number of nodes in each field is arbitrary.

Weight Vector: Associated with each F_2 category node j(j=1,..., N) is a vector $\mathbf{w_j} \equiv (w_{j1}, ..., w_{jM})$ of adaptive weights, or LTM(long-term memory) traces. Initially, when each category is said to be uncommitted

$$w_{j1}(0) = ... = w_{jM}(0) = 1 \tag{B.9}$$

After a category is selected for coding it becomes committed. Each LTM trace w_{ji} is monotonically nonincreasing through time and hence converges to a limit. The fuzzy ART weight vector $\mathbf{w_j}$ formally represents both the bottom-up and top-down weight vectors of ART 1.

Parameters: Fuzzy ART dynamics are determined by a choice parameter $\alpha > 0$ a learning rate parameter $\beta \in [0, 1]$ and a vigilance parameter $\rho \in [0, 1]$.

Category Choice: For each input $\mathbf{I}$ and F_2 node j, the choice function T_j is defined by

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge w_j|}{\alpha + |w_j|} \tag{B.10}$$

where the fuzzy AND, or intersection, $operator(\wedge)$ is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i) \tag{B.11}$$

and where the norm |.| is defined by

$$|\mathbf{p}| = \Sigma_{i=1}^{M} |p_i| \tag{B.12}$$

for any M-dimensional vectors **p** and **q**. For notational simplicity, $T_j(\mathbf{I})$ is written as T_j when the input **I** is fixed.

The system is said to make a *category choice* when at most one F_2 node can become active at a given time. The category choice is indexed by J, where

$$T_j = max\{Tj : j = 1...N\} \tag{B.13}$$

If more than one T_j is maximal, the category j with the smallest index is chosen. In particular, nodes become committed in order j = 1,2,3,...When the Jth category is chosen, $y_J = 1$ and $y_j = 0$ for $j \neq J$. In a choice system, the F_1 activity vector **x** is characterized by the equation

$$x = \begin{cases} \mathbf{I} & \text{if } F_2 \text{ is inactive} \\ \mathbf{I} \wedge \mathbf{w}_j & \text{if the } J\text{th } F_2 \text{ node is active.} \end{cases} \tag{B.14}$$

Resonance or Reset: Resonance occurs if the match function $|\mathbf{I} \wedge \mathbf{w}_J|/|\mathbf{I}|$ of the chosen category J meets the vigilance criterion

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho \tag{B.15}$$

That is, when the Jth category is chosen, resonance occurs if

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{w}_J| \geq \rho|\mathbf{I}| \tag{B.16}$$

Learning them ensues, as defined below. Mismatch reset occurs if

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} < \rho \tag{B.17}$$

That is, when

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{w}_J| < \rho|\mathbf{I}| \tag{B.18}$$

the value of the choice function T_j is set to zero for the duration of the input presentation to prevent the pesistent selection of the same category during search.

A new index J is then chosen by eqn. B.13. The search process continues until the chosen J satisfies eqn.B.15.

Learning: Once search ends, the weight vector w_J is updated according to the equation

$$\mathbf{w}_J^{(new)} = \beta(\mathbf{I} \wedge \mathbf{w}_J^{(old)}) + (1 - \beta)\mathbf{w}_J^{(old)} \tag{B.19}$$

Fast learning corresponds to setting $\beta = 1.0$

Complement Coding: This is a preprocessing step that uses on-cell and off-cell responses to prevent category proliferation. Complement coding normalizes input vectors while preserving the amplitudes of individual feature activations. (see [76] for a detailed discussion). The preprocessed input vector I that is output from F_0 is twice the size of the raw input **a**, where

$$I = (\mathbf{a}, \mathbf{a}^c)$$

and $\mathbf{a}^c$ is the complement of **a** (i.e. $\{1 - a_1, \ldots, 1 - a_M\}$). Thus if the M-dimensional vector

$$a = \{0, 1, 1\}$$

then $\mathbf{a}^c$ = {(1 - 0), (1 - 1), (1 - 1)} and thus the $2M$-dimensional vector I is as follows:

$$I = \{0, 1, 1, 1, 0, 0\}$$

Fuzzy ARTMAP (FAM)

In the FAM system, ART_a and ART_b are linked via an inter-ART module, F^{ab}, called a map-field. This is illustrated in Figure B.3. The explanation of the dynamics of the architecture is given in the following paragraphs.

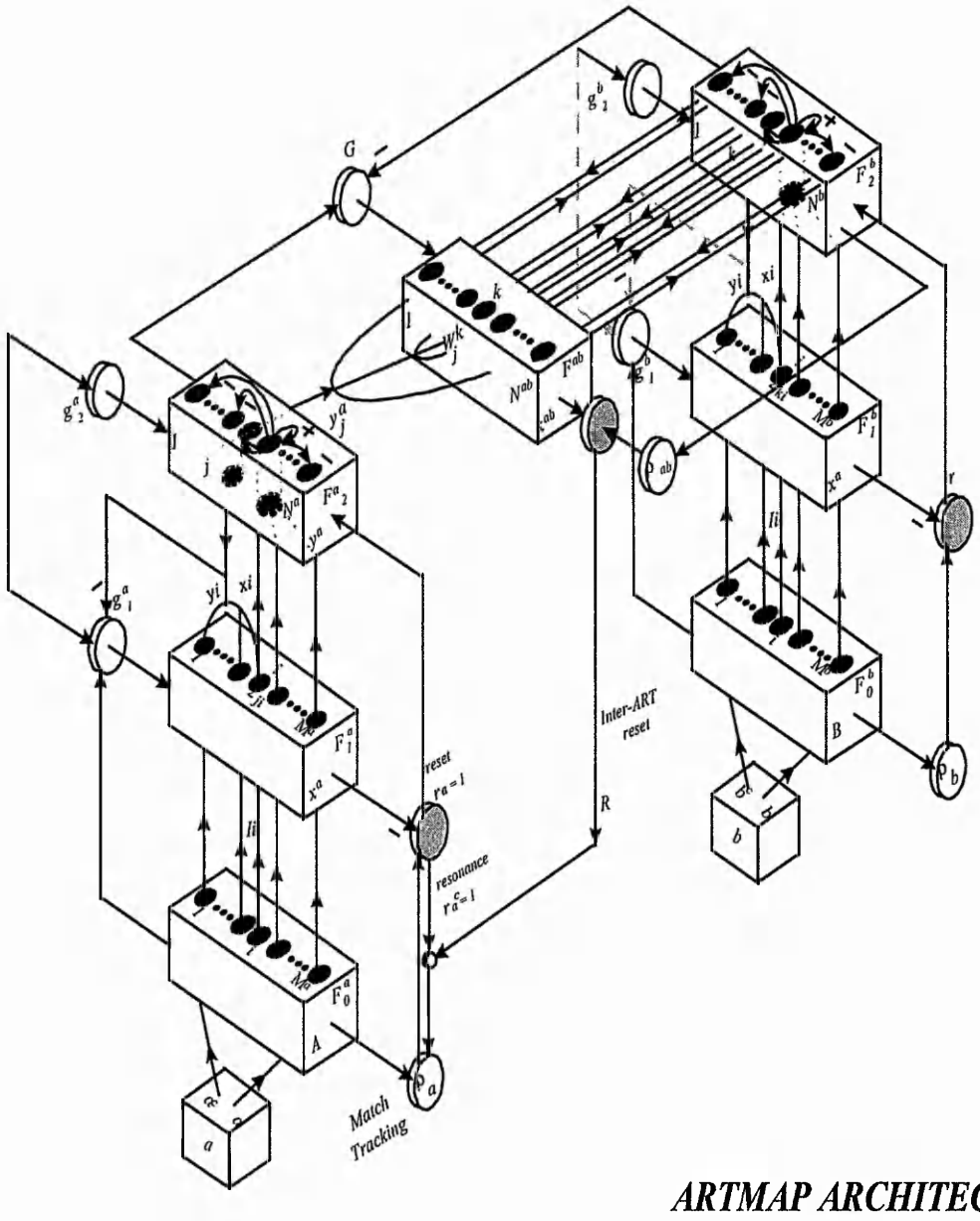


Figure B.3: Fuzzy ARTMAP architecture

$\mathbf{ART_a}$ and $\mathbf{ART_b}$: Inputs to ART_a and ART_b are in the complement code for: For ART_a, input $\mathbf{I} = \mathbf{A} = (a, a^c)$; and for ART_b, input $\mathbf{I} = \mathbf{B} = (b, b^c)$. Variables in ART_a or ART_b are designated by superscripts a or b.

For ART_a, $\mathbf{x}^a \equiv (x_1^a, ..., x_{2M_a}^a)$ denotes the F_1^a output vector; $\mathbf{y}^a \equiv (y_1^a, ..., y_{N_a}^a)$ denotes the F_2^a output vector; and $\mathbf{w}_j^a \equiv (w_{j,1}^a, 1...w_{j,2M_a}^a)$ denotes the jth ART_a weight vector. For ART_b, $\mathbf{x}^b \equiv (x_1^b, ..., x_{2M_b}^b)$ denotes the F_1^b output vector; $\mathbf{y}^b \equiv (y_1^b, ...y_{Nb}^b)$ denotes the F_2^b output vector; and $\mathbf{w}_k^b \equiv (w_{k,1}^b, ..., w_{k,2M_b}^b)$ denotes the kth ART_b weight vector. For the map field, $\mathbf{x}^{ab} \equiv (x_1^{ab}, ..., x_{N_b}^{ab})$ denotes the F^{ab} output vector and $\mathbf{w}_j^{ab} \equiv (w_{j1}^{ab}, ..., w_{jN_b}^{ab})$denotes the weight vector from the jth F_2^a node to F^{ab}. Components of vectors $\mathbf{x}^a$, $\mathbf{y}^a$, and $\mathbf{x}^{ab}$ are reset to zero between input presentations. Initially, each weight is set equal to one. Note, that $|\mathbf{A}| = M_a$ and $|\mathbf{B}| = M_b$ for all input vectors a and b.

Map Field Activation: Map field F^{ab} is activated when one of the ART_a or ART_b categories becomes active. When the Jth F_2^a node is chosen, $F_2^a \longrightarrow F^{ab}$ input is proportional to the weight vector $\mathbf{w}_j^{ab}$. When the Kth F_2^b node is chosen, the F^{ab} node K is activated by one-to-one pathways between F_2^b and F^{ab}. If both ART_a and ART_b are active, as in supervised learning, then F^{ab} activity reflects the degree to which a correct prediction has been made. With fast learning, F^{ab} remains active only if ART_a predicts the same category as ART_b, via the weight vector $\mathbf{w}_j^{ab}$, or if chosen ART_a category J has not yet learned an ART_b prediction. In summary, the F^{ab} output vector $\mathbf{x}^{ab}$ obeys

$$x_{ab} = \begin{cases} y^b \wedge w_j^{ab} & \text{if the } J\text{th } F_2^a \text{ node is active and } F_2^b \text{ is active} \\ w_j^{ab} & \text{if the } J\text{th } F_2^a \text{ node is active and } F_2^b \text{ is inactive} \\ y_b & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is active} \\ 0 & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is inactive.} \end{cases} \tag{B.20}$$

If the prediction $\mathbf{w}_j^{ab}$ is disconfirmed by $\mathbf{y}^b$, this mismatch event triggers an ART_a search for a new category, as follows.

Match Tracking: At the start of each input presentation the ART_a vigilance parameter ρ_a equals a baseline vigilance, $\bar{\rho}_a$. The map field vigilance parameter is ρ_{ab}. Match tracking is triggered by a mismatch at the map field F^{ab}, that is, if

$$|\mathbf{x}^{ab}| < \rho_{ab}|\mathbf{y}^{b}| \tag{B.21}$$

then ρ_a is increased until it is slightly larger than the ART_a match value, $|\mathbf{A}\wedge\mathbf{w}_j^a||\mathbf{A}|^{-1}$, where A is the input to F_1^a and J is the index of the active F_2^a node. After match tracking, therefore

$$|\mathbf{x}^{a}| = |\mathbf{A}\wedge\mathbf{w}_j^a| < \rho_a|\mathbf{A}| \tag{B.22}$$

When this occurs, ART_a search leads either to ARTMAP resonance, where a newly chosen F_2^a node J satisfies both the ART_a matching criterion

$$|\mathbf{x}^{a}| = |\mathbf{A}\wedge\mathbf{w}_j^a| \geq \rho_a|\mathbf{A}| \tag{B.23}$$

and the map field matching criterion

$$|\mathbf{x}^{ab}| = |\mathbf{y}^{b}\wedge\mathbf{w}_j^{ab}| \geq \rho_{ab}|\mathbf{y}^{b}| \tag{B.24}$$

or, if no such F_2^a node exists, to the shutdown of F_2^a for the remainder of the input presentation. Since $w_{ij}^a(0) = w_{jk}^{ab}(0) = 1$ and $0 \leq \rho_a, \rho_{ab} \leq 1$, ARTMAP resonance always occurs if J is an uncommitted node.

Map Field Learning: A learning rule determines how the map field weights w_{jk}^{ab} change through time, as follows. Weights w_{jk}^{ab} in $F_2^a \longrightarrow F^{ab}$ paths initially satisfy

$$w_{jk}^{ab}(0) = 1 \tag{B.25}$$

During resonance with the ART_a category J active, $\mathbf{w}_j^{ab}$ approaches the map field vector x^{ab}. With fast learning, once J learns to predict an ART_b category K, that association is permanent; ie., $w_{JK}^{ab} = 1$ and $w_{Jk}^{ab} = 0(k \neq K)$ for all time.

The algorithmic process for the FAM learning is given in Figure B.4. During learning both inputs I_a and I_b are required in their complement code form. The base vigilance ρ_a is reset to the base vigilance value $\bar{\rho}_a$. The activation is made for every uncommited or non-reset F_2 node in both modules. Competition then start in a winner- take-all fashion until the maximum activation is found. The top-down expectation is compared with the input to satisfy the respective vigilance level. If

the vigilance is satisfied then the vector activity X^{ab} is computed. Learning then ensues when the map vigilance value is satisfied. If it is not, then the current selected node or category in ARTa is reset and another category is selected. The learning cycle then continues if appropriate.

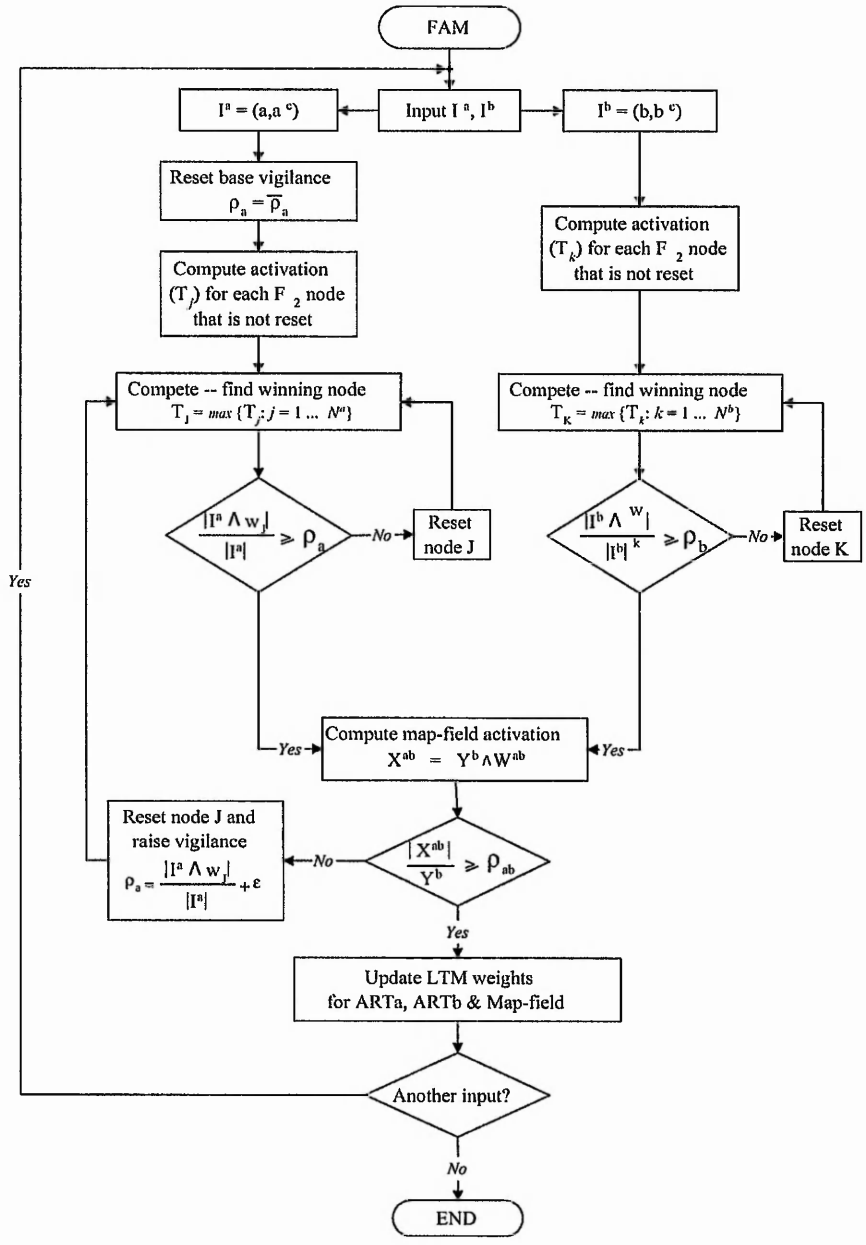


Figure B.4: FAM learning cycle

Appendix C

List of Publications

The details of the publications written by the author during the research for this thesis are as follows:

- I. Lopez-Juarez, L. Brignone, M. Howarth, K. Sivayoganathan, V. Balendran, D. Al-Dabass.
  Reducing Positional Errors Using Adaptive Resonance Theory (ART). Int. Conf. on Gearing, Transmissions and Mechanical Systems. Nottingham, U.K. July 3-6, 2000 (Accepted for publication).

- I. Lopez-Juarez, M. Howarth, K. Sivayoganathan.
  Acquisition of Assembly Skills by Industrial Robots. Fourth International Conference on Cognitive and Neural Systems. Boston University, U.S. May 24-27, 2000 (Accepted for publication‡).

- I. Lopez-Juarez, M. Howarth, D. Al-Dabass, K. Sivayoganathan.
  Contact Force Pattern Simulation Towards Real-Time Robotic Assembly Control. Proceedings of the UK SIM Workshop Simulation' 99. UK Simulation Society, UCL London. October 1999. pp 27-31.

- I. Lopez-Juarez, M. Howarth, K. Sivayoganathan, D. Al-Dabass.
  A Strategy to Control Contact Force in Robotic Assembly. Proc. of the Int Symp. on Robotics and Automation. pp. 177-181. Dec., 1998. Saltillo, Coahuila. Mexico.

- I. Lopez-Juarez, M. Howarth, K. Sivayoganathan.
  An adaptive learning approach to control contact force in assembly. Proceedings of the 1998 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. (IROS'98) Vol. 3 pp 1443-1448. ISBN0-7803-4465-0. Victoria, Canada. October, 1998.

- C. D'Souza, I. Lopez-Juarez, M. Howarth, K. Sivayoganathan, V. Balendran.
  Skill acquisition via integration of vision and force sensing. Proceedings of the XIV National Conference on Manufacturing Research (NCMR98). Derby, England. September, 1998. In Advances in Manufacturing Technology XII edited by R W Baines, A Taleb-Bendiab, Z Zhao. Pp 349-354. ISBN 1 86058 172 2.

- I. Lopez-Juarez, M Howarth, K Sivayoganathan.
  Using force data to control robotic assembly. Proceedings of the XIII National Conference on Manufacturing Research (NCMR97). Glasgow, Scotland. September, 1997. In Advances in Manufacturing Technology XI edited by David K. Harrison. Pp 300-304. ISBN 1 9012 4811 9.

- I Lopez-Juarez, M Howarth, K Sivayoganathan.
  Robotics and skill acquisition. Proceedings of the XII National Conference on Manufacturing Research (NCMR96). Bath, England. September, 1996. In Advances in Manufacturing Technology X edited by A N Bramley, A R Mileham and G W Owen. pp 166-170. ISBN 1 85790 031 6.

‡The author's work has been selected and the author has been awarded a student fellowship by the Boston University to present this work.