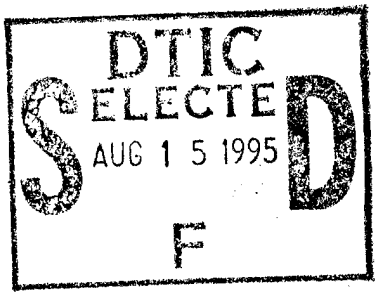
AFIT/DS/AA/95-01




# Teaching Accommodation Task Skills: From Human Demonstration to Robot Control Via Artificial Neural Networks

DISSERTATION
Paul Vincent Whalen
Captain, USAF

AFIT/DS/AA/95-01



Approved for public release; distribution unlimited

AFIT/DS/AA/95-01

# Teaching Accommodation Task Skills: From Human Demonstration to Robot Control Via Artificial Neural Networks

DISSERTATION

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air Education and Training Command

In Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Paul Vincent Whalen, B.S.M.E, M.S.A.E

Captain, USAF



March 1995

AFIT/DS/AA/95-01

Teaching Accommodation Task Skills:
From Human Demonstration
to Robot Control Via
Artificial Neural Networks

Paul Vincent Whalen, B.S.M.E, M.S.A.E
Captain, USAF

Approved:

Curtis H. Spenny, Chairman — 13 MAR 95

Matthew Kabrisky — 13 Mar 95

Mark E. Oxley — 13 Mar 95

Accepted:

Robert A. Calico, Jr.
Senior Dean

*Acknowledgements*

Competitors in the Tour de France cycling competition have elaborate support teams that enable them to compete in this grueling endurance event. Despite their incredible athletic talents, the competitors know they wouldn't stand a chance without a good support team. In many ways, completing this dissertation has been a grueling endurance event for me. Like the Tour de France competitors, my success is almost wholly attributable to my support team.

The *captain* of my support team has been my lovely wife, Shawna, who gave of herself, on my behalf, like no other can. Even when she was stressed out or lonely, she always encouraged me to work on my dissertation. In many ways, this dissertation has been more work for her than for me. An enormous part of Shawna's work load our two children, Erika Leigh and Matthew Ryan, who were both born during the arduous course of this dissertation. For their parts, Erika and Matthew have been a BIG source of joy, inspiration, and motivation. With no effort at all, they helped me keep my perspective.

My *coaching staff* consisted of Dr. Curtis H. Spenny, Major (Dr.) Michael B. Leahy Jr, Dr. Matthew Kabrisky, and Dr. Mark E. Oxley. As my chairman, Dr. Spenny provided the right-brain viewpoint to keep me oriented and caused me to rethink things on numerous occasions. His assistance in pulling the final piece of work together was pivotal. Major Leahy was my robot mentor as well as career counselor. It was Mike's influence that convinced me to do the experimental part of this dissertation and it was his assistance that saw me through it. Dr. Kabrisky is a wise man with an incredible breadth and depth of knowledge. I am grateful that he was kind enough to spoon-feed me. Dr. Oxley provided the mathematical ladder which enabled me to reach new personal heights. Occasionally I could even keep up with him as he derived something.

The crew chief of my *maintenance team* was the ever-helpful, chameleon-of-computer-operating-systems-and-equipment, Dan A. Zambon. I know I wouldn't be writing this if it wasn't for Dan's patient help with even the most irritating problems (it's okay if you want to do that `rm *.*` now, Dan). Also on my maintenance team were Nick Yardich, Jay Anderson, Dan Rioux, Andy Pitts, and Mark Derriso who helped me with a lot of my hardware even

though I was working in the "other" lab. Other members of my maintenance team were Gregory L. Tarr and Dennis W. Ruck who provided me with the initial software I used to train my networks and Kevin L. Priddy who was the only peer with whom I could discuss neural network problems. A final, but crucial member was my good friend Dr. Raymond E. Slyh. He started his dissertation after me, finished before me, and provided hours of helpful discussion and lots of encouragement that I could do it too.

Tim Hancock, John Brohas, Ron Ruley, and Jack Tiffany in the AFIT model shop made up my *fabrication team*. They did a fantastic job of bringing my concept drawings from vague ideas to finely crafted, precision pieces of hardware.

I had an incredible *cheering squad* consisting of my parents (Bill and Joann), my in-laws (Mike and Billye Ruth), my siblings (Susi, Julie, Mark, and Tony), and other relatives too numerous to mention. These are the folks who tried to understand why it took so long and (usually) patiently waited for me to come visit them during the few breaks I took over the six years of this effort. In addition to my relatives, there were many other encouraging and understanding *fans* such as the Pedro's, the Sizemore's, the Powell's, my boss (Dr. Nixon), and my mentor Major Ronald G. Julian. The list could go on from there, but suffice it to say, that my achievements are a testimony to their faith in me which often exceeded my own. To all of you who have grown fearful of asking, it's safe again because the answer is "YES!"

Of course, I cannot forget to thank the *Owner* and *Creator* of my, and all other, support teams; God. It's amazing what God can do with a stupid lump of clay. He deserves better than this.

Paul Vincent Whalen

*Table of Contents*

*List of Figures*

Figure Page

*List of Tables*

*List of Symbols*

| *Symbol* | *Description* | *Page* |
|---|---|---|
| $\mathcal{L}'$ | Lipschitz cutoff ratio, | 5-15 |
| $\mathcal{L}_{ij}$ | Lipschitz ratio between the $i$th and $j$th vectors, | 5-14 |
| ${}^{w}K_e$ | Environment stiffness matrix in world-frame coordinates, $(n \times n)$ | 5-26 |
| $\vec{\tau}_c$ | Vector of commanded torques for robot DC servomotors, $(6 \times 1)$ | 4-5 |
| ${}^{w}\mathcal{J}(\vec{q})$ | Jacobian matrix in world-frame coordinates, $(n \times n)$ | 5-5 |
| ${}^{T}\mathcal{J}(\vec{q})$ | Jacobian matrix in tool-frame coordinates, $(n \times n)$ | 5-3 |
| ${}^{T}\mathcal{J}(\vec{q})^{-1}$ | Inverse of ${}^{T}\mathcal{J}(\vec{q})$, $(n \times n)$ | 5-32 |
| $\varphi_f$ | Angle of planar force vector projection onto $xy$-plane | D-3 |
| $\lambda_f$ | Elevation angle of planar force vector above the $xy$-plane | D-3 |
| $\varphi_v$ | Angle of planar velocity vector projection onto $xy$-plane | D-3 |
| $\lambda_v$ | Elevation angle of planar velocity vector above the $xy$-plane | D-3 |
| $T$ | Time, in seconds, between commanded position trajectory updates | 5-9 |
| $\zeta$ | Scalar performance metric used to evaluate the various controller configurations tested | 3-11 |
| $\chi$ | Composite performance index used in computing $\zeta$ | 5-41 |
| $\tilde{\chi}$ | Composite performance index of best $\mathcal{A}_a$ | 5-41 |
| $(\rho, \beta, \gamma, \xi, \phi)$ | Weighting coefficients for $\chi$ | 5-41 |
| $\lvert\Delta P\rvert_{\max}$ | Maximum (peak) RMS position error after first alignment | 5-41 |
| $\lvert F\rvert_{\max}$ | Maximum (peak) RMS force amplitude after alignment | 5-41 |
| $\lvert F\rvert_{\text{avg}}$ | Average RMS force amplitude after alignment | 5-41 |
| $\mathcal{D}$ | RMS path length traversed from impact to initial alignment | 5-41 |
| $\mathcal{W}_h$ | Hidden layer weight matrix, $(m \times k)$ | A-1 |
| $\mathcal{W}_o$ | Output layer weight matrix, $(n \times m)$ | A-2 |
| $k$ | Number of input features (nodes) for ANN | 5-22 |
| $m$ | Number of hidden layer nodes in ANN | A-1 |
| $n$ | Number of output features (nodes) for ANN | 5-22 |
| $p$ | Number of input-output pairs applied to ANN during training | A-4 |
| $\vec{\varsigma}$ | Vector of hidden layer node outputs, $(m \times 1)$ | A-2 |
| $\vec{\psi}_h$ | Vector of biases (offsets) for hidden layer nodes, $(m \times 1)$ | A-2 |
| $\vec{\psi}_o$ | Vector of biases (offsets) for output layer nodes, $(m \times 1)$ | A-2 |
| $\eta$ | Training rate coefficient for ANN | 5-20 |
| $\alpha$ | Training momentum coefficient for ANN | 5-20 |
| $\Phi$ | Matrix of training data input-output pairs, $(p \times (k+ n))$ | A-4 |
| $\vec{\phi}_i$ | A training exemplar vector equal to $(\vec{F}^{*T}, \vec{V}^{*T})$, $(1 \times (k+ n))$ | A-4 |
| $\vec{\delta}_o$ | ANN output error vector $(\vec{V}^* - \vec{V})$, $(n \times 1)$ | A-5 |
| $\vec{\delta}_h$ | Error-related term used to adjust hidden layer weights, $(m \times 1)$ | A-6 |
| $\vec{q}$ | Vector of joint angles, $(6 \times 1)$ | 5-3 |
| $\vec{q}_m$ | Vector of measured joint angles, $(6 \times 1)$ | 5-34 |
| $\vec{q}_d$ | Vector of desired joint angles, $(6 \times 1)$ | 4-5 |
| $\dot{\vec{q}}$ | Vector of joint velocities, $(6 \times 1)$ | 4-5 |
| ${}^{T}R_w$ | Matrix transformation from world-frame to tool-frame coordinates | 5-6 |
| ${}^{w}R_T$ | Matrix transformation from tool-frame to world-frame coordinates | 5-26 |

| *Symbol* | *Description* | *Page* |
|---|---|---|
| $F_t$ | Threshold force vector magnitude for collision pruning | 5-17 |
| $V_t$ | Threshold velocity vector magnitude for velocity pruning | 5-12 |
| $\epsilon$ | Parameter controlling blending of ${}^{T}\vec{V}_n$ and ${}^{T}\vec{V}_c$ | 5-30 |
| $\varpi$ | Average (mean) value | 6-12 |
| $\sigma$ | Standard deviation | 6-12 |
| $\mu$ | Coefficient of friction | 5-27 |
| $\Upsilon_s$ | Scalar index of structural similarity between two matrices | 5-23 |
| $\Upsilon_g$ | Scalar index of magnitude similarity between two matrices | 5-23 |
| $\Upsilon_r$ | Scalar index of similarity between ratio of (3,3)/(2,2) elements of two matrices | 5-24 |
| $\Upsilon_\pm$ | Scalar index of sign similarity between two matrices | 5-23 |
| $\Psi$ | Included angle between $\vec{F}$ and $\vec{V}$ | 5-13 |
| $\Psi_t$ | Threshold value of $\Psi$ for hemisphere pruning | 5-13 |
| $\sigma_\Psi$ | Standard deviation of $\Psi$ for a training data set | 6-51 |

*List of Acronyms*

| *Acronym* | *Description* |
|---|---|
| ACC: | accommodation relation |
| ADC: | analog-to-digital converter |
| DAC: | digital-to-analog converter |
| DIO: | digital input-output |
| AFIT: | Air Force Institute of Technology |
| ANN: | artificial neural network |
| ARCADE: | AFIT Robotic Control Algorithm Development and Evaluation |
| ASN: | Associative Search Network |
| CBR: | chemical/biological/radioactive |
| CCW: | counter-clockwise |
| CG: | center of gravity |
| CMAC: | Cerebellar Model Articulation Controller |
| CW: | clockwise |
| DIDO: | Demonstration Input/Demonstration Output |
| DIO: | digital input-output |
| DISO: | Demonstration Input/Synthetic Output |
| DoF: | degree-of-freedom |
| FFT: | Fast Fourier Transform |
| HMM: | hidden Markov model |
| Hz: | hertz (cycles per second) |
| I/O: | input/output |
| Kg: | kilogram |
| MITL: | Man-In-The-Loop |
| MLP: | Multi-layered Perceptron |
| mm: | millimeter |
| ms: | milliseconds |
| NGMH: | next-generation munitions handler |
| PID: | position-integral-derivative |
| PIH: | peg-in-hole |
| PLIMMS: | planar low-impedance motion measurement system |
| PSD: | power spectral density |
| RCC: | remote center of compliance |
| RISO: | Real Input/Synthetic Output |
| RMS: | root-mean-squared |
| SISO: | Synthetic Input/Synthetic Output |
| SCARA: | Selective Compliant Articulated Robot for Assembly |
| TDANN: | time-delayed ANN |
| UFS: | universal force-moment sensor |
| USAF: | US Air Force |
| V-AI: | vertically-articulated, industrial (manipulator) |

*Abstract*

A simple edge-mating task, performed automatically by accommodation control, was used to study the feasibility of using data collected during a human demonstration to train an artificial neural network (ANN) to control a common robot manipulator to complete similar tasks. The 2-dimensional (planar) edge-mating task which aligns a peg normal to a flat table served as the basis for the investigation. A simple multi-layered perceptron (MLP) ANN with a single hidden layer and linear output nodes was trained using the back-propagation algorithm with momentum. The inputs to the ANN were the planar components of the contact force between the peg and the table. The outputs from the ANN were the planar components of a commanded velocity. The controller was architected so the ANN could learn a configuration-independent solution by operating in the tool-frame coordinates. As a baseline of performance, a simple accommodation matrix capable of completing the edge-mating task was determined and implemented in simulation and on the PUMA manipulator. The accommodation matrix was also used to synthesize various forms of training data which were used to gain insights into the function and vulnerabilities of the proposed control scheme.

Human demonstration data were collected using a gravity-compensated PUMA 562 manipulator and using a custom-built planar, low-impedance motion measurement system (PLIMMS). The raw demonstration data collected using both systems were found to be poor examples of accommodation mappings for reasons that are discussed. In addition to the problem of the existence of the desired mapping in the demonstration data, the sensitivity of the ANN paradigm to the richness of the training data was also determined. For the proposed controller training method, a key problem is one of matching the distribution statistics (mean and standard deviation) between the training data and what is to be encountered in the measurement stream when the trained ANN controller is implemented. Seven data processing algorithms were investigated independently and in combinations to determine if they could improve the quality of the demonstration data. None were found to produce very

robust results, although mirroring the raw data about all the axes to force a zero mean upon the training data set was found to improve controller performance significantly.

# Teaching Accommodation Task Skills: From Human Demonstration to Robot Control Via Artificial Neural Networks

## *I. Introduction.*

The mission of the US Air Force (USAF) is to fly and fight. To be an effective part of the United States' capability to deter war, the USAF must be capable of conducting its mission in a broad spectrum of environmental conditions. From the biting sub-zero temperatures of Thule, Greenland to the intense heat of the Saudi Arabian desert, mission sorties must be flown and flightline operations must continue. Several war scenarios predict even more hazardous environments for mission operations than temperature extremes. Proliferation of ballistic missile technology and chemical weapons indicates a real possibility that future conflicts will require the USAF to operate in a chemical/biological/radioactive (CBR) environment.

During an active air campaign, military aircraft will often undergo a process called *hot*-aircraft turnaround between mission sorties. They land at their airbase and taxi to a hot ramp where they are simultaneously refueled and reloaded with munitions so they can immediately be launched for another sortie. This whole operation occurs while the plane's engines are running, hence the term *hot*. In some cases, minor repairs such as modular component replacements can also be made on the hot ramp.

The threat of a war involving CBR weapons is not new to the USAF. In such a war the entire flightline may be contaminated with chemical or biological agents making it much more difficult to perform routine aircraft maintenance, let alone hot-aircraft turnarounds. Consequently, CBR defensive procedures and equipment have been under development for decades. However, despite the great improvements that have been made, contemporary protective garments are still bulky, lack adequate ventilation, and seriously degrade the

dexterity of the wearer. As a result, even simple aircraft maintenance and hot-aircraft turnaround procedures can become very difficult to perform. The added time it takes for an airman to connect arming lanyards or mate cannon plugs extends his/her exposure time to the hazardous environment and increases the hot-turnaround time of the aircraft being serviced. Increased turnaround time ultimately results in reduced mission capability.

Mission capability in a protracted war is often drastically affected by the efficiency of the logistics support. The logistics support of the USAF is provided by the Air Force Materiel Command (AFMC). Besides making sure that the "beans and bullets" are placed where they are needed, when they are needed, the AFMC is responsible for refurbishing and renovating aircraft, missile, propulsion, and munition systems. Since some of the weapon systems are very old, the AFMC also manufactures and assembles replacement parts for older systems at various Air Logistics Center (ALC) throughout the nation when the replacement parts are no longer available commercially. Thus, these ALC function as military manufacturing plants operating similar to those in civilian industry except that they often produce low volumes of a high variety of products. Consequently, flexible manufacturing and assembly systems potentially offer premium payoffs for the AFMC.

### *1.1 Motivation.*

Robotic systems promise to be a very beneficial part of the future USAF. Robotic systems can reduce the exposure of airmen to CBR environments and streamline the hot-aircraft turnaround process in hazardous environments. In addition, robotic manufacturing and assembly systems can help the ALC with their missions.

The process of turning a hot aircraft for another sortie involves three steps: fueling, loading munitions, and possibly making simple, module-replacement type repairs. Fundamental to all three of these steps is the ability to mate parts. In the case of fueling, the nozzle of the hose from the fuel truck must be inserted into the aircraft's fueling receptacle. In the case of munitions loading, the bomb lugs must be aligned with the bomb rack and the bomb jammed into place prior to latching the rack. In the case of the module-replacement repairs, one component is removed and another inserted in its place. Robots capable of performing these tasks with total autonomy are considered a distant reality. However, Man-In-The-

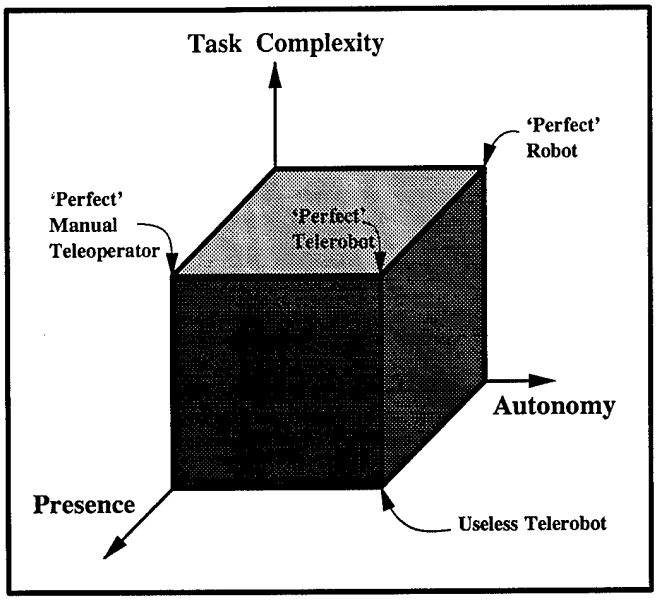



Figure 1.1 The spectrum of MITL robotic systems showing tradeoffs between local autonomy and operator control [47]

Loop (MITL) robotic systems, also known as telerobotic systems, are a viable, near-term alternative.

In the MITL scenario, a human operates a master unit to provide the high level of intelligence and experience required to perform complex tasks and a robotic system acts as a slave to the master unit's commands. Depending on the balance between control guidance provided by the human operator and autonomous control of the slave, an MITL robotic system can be categorized on a spectrum between *telepresence* and *autonomous* as shown in Figure 1.1. Although one may imagine that telerobotic systems are simpler to build than autonomous robots and require very little autonomous capability, they actually demand a fairly high degree of *local autonomy* in the slave robot unit. Local autonomy is encouraged by at least four important benefits.

- It requires less information to be sent back to the operator thereby lowering the transmission bandwidth requirements.
- Operator fatigue from concentrating on the task and trying to process the sensory feedback information is reduced so his/her efficiency is enhanced.

- For the autonomous functions, it eliminates delays in the control loop caused by communication and operator-induced time delays, both of which cause instability in the control system.
- Local autonomy can allow a simple operator command set to perform a large variety of relatively complex tasks by taking care of the details of task execution automatically.

Increased local autonomy in the slave unit means increased *intelligence*. Because parts mating is fundamental to aircraft turnaround operations, a robotic control system capable of intelligently mating parts would constitute a foundation on which an MITL robotic system could be developed to replace airmen on the flightline in hazardous environments. A robotic system capable of mating parts would also be useful in assembly and module-replacement repairs at the ALC since many of their operations also share the fundamental characteristics of parts mating. In the ALC the main motivation for implementing robotic systems would be increased efficiency rather than increased safety as in the case of flightline operations. A further beneficiary from intelligent parts mating technology would be the space industry. The National Aeronautics and Space Administration (NASA) determined that it needed telerobotic systems to help build and maintain the proposed space station Freedom because of the enormous number of extravehicular activity (EVA) hours required. Consequently, they began development of a telerobotic system called the Flight Telerobotic Servicer (FTS) which was intended to perform assembly and repair EVA tasks under the supervision of an astronaut. The functional requirements for the FTS were fairly similar to those of the proposed flightline robotic system. Unfortunately, cost overruns and budget cuts have essentially halted development of the FTS and the USAF has yet to embrace the concept of telerobotic systems on the flightline. Very recently, however, a feasibility study has begun to determine whether telerobotics technology can improve existing munitions loading equipment and operations. This program, called the next-generation munitions handler (NGMH) project, may field one of the first systems employing telerobotics technology in a combat operations application.

### *1.2 Problem Statement.*

This dissertation explored a method of increasing the level of local autonomy available for robots that perform parts mating tasks under teleoperator control. The principle of the

method was to have a person physically demonstrate the task or skill to be learned and have the robotic system monitor and analyze the resulting data to extract the ability to perform similar kinds of tasks autonomously. The acquired skill would then be a part of the local autonomy toolbox for the telerobotic system.

### *1.3 Objective.*

The specific objective of this dissertation was to determine the feasibility of using a limited set of demonstrations by a human operator to *teach* a robot how to insert a chamferless peg in a chamferless hole by feel alone. The implication of doing the peg-in-hole (PIH) task by *feel* alone is that the robot has no prior knowledge of the table's precise location or orientation and no camera vision data are available to extract estimates of the table's location. Thus, only limited sensory data are available for the controller. Our objective to use a limited set of demonstrations implies that only a small portion of the total reachable workspace will be included in the demonstration data. Thus, the controller must be able to somehow generalize from the demonstration data so it can perform the task anywhere in the reachable workspace of the manipulator regardless of the joint configuration. This trait is commonly referred to as a configuration-independent solution.

### *1.4 Assumptions.*

The geometry of the grasped peg is assumed to be known and the table is stationary relative to the robot during the operation. All parts are taken to be rigid bodies, though the methods presented are robust to small deformations. It is also assumed that the collected demonstration data have been filtered to prevent aliasing. We assume that the kinematic structure of the manipulator is known and it's Jacobian transformation matrix is available. For the edge-mating task, which is described in Section 1.5, another assumption is that the peg initially contacts the table such that the total of the normal and tangential contact forces produces a vector pointing so as to cause a moment that tends to align the peg normal to the surface. In practice, this restriction is a function of the coefficient of friction, the radius of the peg, and the direction of the nominal free-space velocity relative to the table surface.

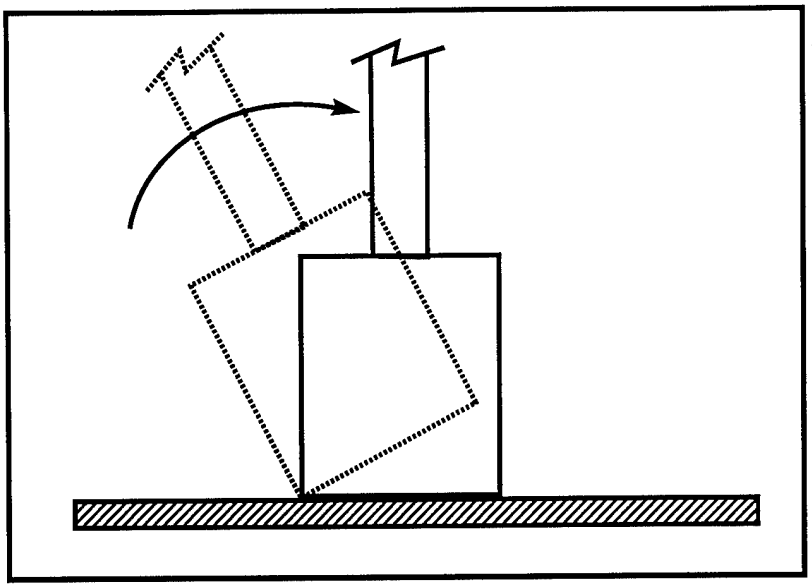


Figure 1.2 Motion associated with the edge-mating task

For the edge-mating task, if the free-space velocity is outward along the axis of the peg, this restriction is always met.

A final assumption is that the only sensor data available during operation are:

- the net 6-axis contact force between the robot and the environment, and
- the robot's internally-measured joint angles.

*1.5 Approach.*

In pursuit of the final objective mentioned above, we began with simpler tasks that would validate the technique, equipment, and software. The progressively-more-difficult tasks identified were:

- to align the flat end of a rectangular peg with the flat surface of a table by *feel* alone. This alignment task has recently been referred to as an *edge-mating* task [31] and is shown in Figure 1.2.
- to insert a rectangular peg into a chamfered slot; this is a planar projection of the general cylindrical peg insertion problem.
- to insert a rectangular peg into a chamferless slot.
- to insert a chamferless cylindrical peg into a round, chamferless hole.

All of these tasks were to be performed by artificial neural network (ANN) controllers trained on data collected from observing a human perform the same task. Beginning with the edge-mating task, alternative sources of training data were to be used to test our understanding of the task. Since the edge-mating task can be accomplished by an accommodation matrix controller, the matrix controller was used as a baseline for performance measurement and a source for gaining insights into the functions and characteristics of the trained ANN controllers. In addition, the accommodation control law commands velocities which, if expressed relative to a coordinate frame attached to the peg, will provide the configuration-independent capability that is a critical part of our objective. Another positive feature of the accommodation control law is that it requires no sensing of the environment; only the direct interaction forces between the peg and the environment need to be measured.

In the interim testing, training data were synthesized from observing the matrix controller in action. The results of those tests were then used in our attempts to train an ANN controller to perform the edge-mating task using human demonstration data. Because of the complexities and risks inherently involved in implementing a new control strategy on a robot manipulator, a simplified computer simulation of the manipulator, the controller, and the task environment was used initially. However, to collect the human demonstration data, the inevitable transition to experimental hardware was made early in the effort. At that early phase of testing, the human demonstration data was collected and used to verify the function of the matrix controller on the manipulator. In addition, the ability of some ANN controllers to complete the edge-mating task after being trained on data collected from observing the matrix controllers was verified. An extensive phase of investigations based on the simulation model was then entered which lasted to the conclusion of the effort. These investigations explored the sensitivity of the success of the ANN controllers to various characteristics of the data presented during the off-line supervisory training of the ANN. The simulations were also used to evaluate several techniques that were proposed to enhance the quality of the training data.

*1.6 Contributions.*

Although this effort never attempted to complete the PIH task with an ANN controller, most of the understandings gained in the investigation of the edge mating task are fundamental to the proposed technique, regardless of the task. As such, those insights are as applicable to the PIH task as they are to the edge-mating task examined. The tests conducted proved that an ANN controller could be trained to perform a simple accommodation task if the off-line, supervisory training data were carefully formulated from a specific control strategy. In the case of the edge-mating task, an accommodation matrix control law was used to prove that the ANN could learn to control a common manipulator to perform the task. Thus, the feasibility of the concept was established. In the process of this investigation, the critical features of the control architecture and training data were identified. Human demonstration of the simple edge-mating task was found to contain several degrading characteristics which were analyzed in detail. The analysis revealed that human demonstration will inherently lead to problematic training data for the controller architecture used. Corrective measures were proposed for several of the detrimental characteristics and their efficacies were demonstrated. In addition, the desirable characteristics of demonstration data collection systems were identified.

# *II. Related Research*

The topic of this dissertation spans several fields which, until recently, were unrelated, specifically the fields of artificial neural networks, robotic part mating, and human skill transfer to robots. Since this dissertation uses only well established ANN techniques and there is a plethora of literature on ANNs, this chapter will not attempt to detail works in that field. The reader can refer to [35], [44], [54] or [28] for general introductions to a variety of ANN methodologies and techniques. In particular, [28] gives a good comprehensive overview of the latest techniques and insights about parameter interactions. References to specific works of interest will be included as required in the remaining text, and Appendix A provides a brief introduction to the ANN methodology used in this dissertation.

The field of robotic part mating is enormous, and a multitude of researchers have tried many different techniques. We quickly draw our attention to only those works that are applicable to the classic part mating tasks of inserting a peg into a hole and edge-mating[1]. Numerous researchers have attacked the PIH task directly, and the most recent and related works are summarized in Section 2.1. Because of its classic simplicity, there are very few works in the literature that discuss the edge-mating task. Therefore, we will draw heavily upon the literature for the PIH task and apply the insights to our simpler edge-mating task.

The final important related field of research has to do with transferring human skills to robots. Section 2.2 identifies previous research that is relevant to the work presented in this dissertation.

## *2.1 Peg Insertion Research.*

The work done directly on the PIH task can be categorized into classical analysis, logic-branching control, and learning control.

*2.1.1 Classical Analysis Works.* Whitney has been one of the trail blazers in increasing our analytic understanding of part mating for assembly. He has done several classic analyses of the peg insertion task. In [56] he examined general quasi-static interactions dur-

[1] See Section 1.5 for a description of the edge-mating task.

ing part mating during the assembly of compliantly-supported rigid parts. In [60] he did a detailed analysis of peg interaction with chamfered holes, and detailed the geometric relationships that designers should use to select chamfer shapes that reduce the peak insertion forces required. It was in this work that the terms *jamming* [2] and *wedging* [3] were first defined, and geometric relationships were derived that predict the occurrence of jamming and wedging. In addition to developing methodologies for designing chamfer shapes that minimize insertion forces, Whitney was involved at Draper Lab in the design of the remote center of compliance (RCC) device which allowed chamfered insertions to be accomplished passively. With this device, chamfered parts could be aligned and mated without exact fixturing of the pieces or explicit force control of the assembly robot. He also was involved in development of an instrumented RCC [16] which expanded the capability of insertion by robots. Probably the best technical overviews of part mating and assembly can be found in [59] and [57]. As a point of interest, in [59] Whitney mentions that Gustavson patented in 1982 a *passive device* that could accomplish chamferless insertions within a limited range of angular offsets.

Caine [11, 12] utilized insights gained from Whitney's analyses to treat the chamferless insertion problem. He showed that when the peg encounters two-point contact without both corners entering the hole, a strategy based on applying an arbitrary force independent of the direction of tilt will not succeed. To solve the problem, he proposed constraining the allowable set of contacts to eliminate any ambiguity about the direction that the peg may be tilted. Given foreknowledge about the direction of tilt, he showed that if you simply rotate the peg while maintaining contact, the critical angle will be reached and the peg will slide into the hole. Although Caine gives expressions for the force and moment required to rotate the peg, they have little practical value for an unstructured task in an unstructured environment, since you cannot ensure the direction of tilt without additional sensory information such as a vision system.

[2] Jamming is defined by Whitney as the condition that exists when the insertion force points too far off the axis of the hole for the peg to advance into the hole.

[3] Wedging is defined by Whitney as an event in which the contact forces between the peg and hole fall within their respective friction cones, thereby preventing the peg from advancing into the hole.

Shahinpoor and Zohoor [46] present an elaborate dynamic analysis and arrive at some constraints to prevent jamming and wedging while performing dynamic PIH insertions. They only present equations for the two-dimensional problem, but they derive dynamic equations for six distinct cases and develop three inequality constraints that must be held to avoid jamming under all possible geometrical situations as well as one inequality constraint to avoid wedging. They assume that all the peg and hole descriptions are available a priori and that precise, noise-free sensor data are also available. Both of these assumptions make it difficult to implement their results in a real system.

Besides the simple mechanical RCC devices used commonly in industrial robots to insert pegs [16, 58, 61], there are analytic methods which utilize either programmable compliant motion control [42] or accommodation control. In Peshkin's work [42] the goal is to develop a method which specifies a single compliance matrix that will guarantee that an insertion task will succeed. The matrix must be "error-corrective" and must be consistent with **every** contact configuration that may possibly occur. For a structured task, the technique can work, although Peshkin does not say anything about how one enumerates all the possible contact configurations nor how one can sense what contact configuration exists with real sensors. These difficulties make it virtually impractical for general applications.

*2.1.2 Logic-Branching Works.* Handelman et al. have tried combining a knowledge-based system with a Neural Network-based Reflex Modulator to have a manipulator learn a skill [25]. As an example, Handelman et al. taught a two-link manipulator how to make a "tennis-like" swing. They came up with the rules through trial-and-error in LISP and then translated them automatically from LISP to Pascal. For the network, they used Cerebellar Model Articulation Controller (CMAC) network modules as described by Albus [1]. Their rule-based Execution Monitor decides how to make a successful swing using rules only. Then it teaches the CMAC by presenting examples. Following the training, the Execution Monitor continuously watches the CMAC performance and re-engages rule-based control if the CMAC performance is below par.

Asada has used Petri nets to control a robot for assembly [36, 37]. He argues that a discrete event approach to assembly is required because, unlike most force-feedback appli-

cations, the state of contact changes during an assembly process. The changes in contact result in different constraint equations, which, in turn, result in varying numbers of motion degrees of freedom and equations of motion. His implementation is similar to the controller configuration of the present work. His discrete event controller (DEC) and process monitor together can be thought of as a real-time trajectory generator which is equivalent to the ANN in the controller of this dissertation. The similarities of the systems include off-line training and the use of a continuous controller to track the commands of the real-time trajectory generator. The off-line training for his system consists of the analysis required to construct the Petri net model for the task.

Buckley proposed an iterative learning method of teaching compliant strategies to a robot which is based on a search tree and a lookup table [9]. The system graphically displays the start and goal regions; the user inputs a commanded position that is used to compute a set of robot configurations that keep the robot in contact with the environment and from which the goal region can be reached via the commanded position. These "solved" robot configurations are then stored in an appropriate lookup table and subtracted out of the start region. If there is any remaining unsolved area of the start region, the process repeats until the start region is consumed by the solved robot configurations. The iterative nature of this method resembles the iteration required by the method proposed in this dissertation. However, Buckley's method uses explicit geometric models of the robot and its environment which are not generally available for an unstructured task.

Another logic-branching or rule-based method of automated assembly was proposed by Vaaler and Seering in [51]. Their machine learning algorithm is a "production system" that automatically generates "production rules." They attack a planar peg-in-hole problem using a gantry-type Cartesian robot called the MIT Precision Assembly Robot (MITPAR). They gather six different state variables as data inputs and discretize them into six levels each. They chose only six levels of discretization on the basis that people do assembly well and do not appear to be capable of resolving forces to more than 5-10 distinct levels in the range of forces used during assembly. The ranges of each of Vaaler and Seering's state variables are determined experimentally. One of the problems with the approach is that the time required to visit all of the states during the learning phase of the algorithm is by far

the largest part of the learning process. Even with the coarse discretization into six levels, there are nearly 280,000 possible states. Since only about 300 of them are really feasible for the particular system, they reduced the search space considerably. The algorithm moves the peg a set distance for correction, and the if-then production rules decide which way to move. Vaaler and Seering had a problem with the dynamics of the robot because they say that essentially all of the time required to assemble a peg and hole was spent waiting for the robot system to settle. This is unexpected because they used a relatively stiff Cartesian gantry robot. Convergence of the learning algorithm took about 20-50 assembly trials starting from random corrections. The algorithm took about 10-20 seconds to complete assemblies after it had learned. They used a soda can as a peg (2.6-inch diameter) and a 1-inch diameter steel peg in a 1.010-inch diameter aluminum hole. Clearance ratios of 3% to 5% were used on the soda can experiments.

*2.1.3 Learning Control Works.* Perhaps the work most similar to this dissertation is by Benady et al. in [8]. They design and implement a learning control scheme for an impedance-controlled robot performing a contact task. They assume that a high-gain position control inner loop will exactly follow the commanded *positions* generated by an outer impedance control loop. This form of impedance control actually imitates true impedance control by modifying the desired reference trajectory of the robot to achieve the compliance effect of the impedance control law. The learning part of this work is an Associative Search Network (ASN) that learns the optimal impedance parameters on-line. The ASN was originally developed by Barto et al. [7]. The ASN is similar to the network of perceptrons used in this dissertation[4], but uses different output equations and has a different training algorithm. Its structure is equivalent to a MLP with no hidden layer and linear outputs from the output layer. In addition, the ASN has a zero-mean random noise superimposed on the output of each node rather than an adapting offset or bias as the MLP does, and the weight update rule is somewhat different. A scalar reinforcement or reward signal is fed back from the controlled system and is used in a nonlinear weight update rule. The reward signal is a

[4]The reader is encouraged to refer to Appendix A at this time for an explanation of the structure and terminology of the network of perceptrons used in this dissertation. This review will be helpful for those readers not already familiar with ANN s and is essential for the comparison between the Multi-layered Perceptron (MLP) and the ASN to be understood.

weighted sum of the relative deviations of the actual manipulator tip velocity and interaction force from their desired values. The weight update rule is similar to the back-propagation training rule of the MLP, but instead of a simple adaptation coefficient, $\alpha$, it uses a decaying exponential function of the scalar payoff signal. The weight update rule is:

$$w(t+1) = w(t) + g\left(z(t) - z(t-1)\right)\left[u(t-1) - u(t-2)\right]x(t-1) \tag{2.1}$$

where $w(t)$ is the weight at time $t$, $z$ is the scalar reward signal, $u$ is the output of the ASN, $x$ is the input to the ASN, and $g(x)$ is given by:

$$g(x) = \begin{cases} 1 - \exp(-x/\lambda) & \text{if} x \geq 0 \\ -\left(1 - \exp(-|x|/\lambda)\right) & \text{if} x < 0 \end{cases} \tag{2.2}$$

where $\lambda$ is a learning parameter. This function is similar in shape to a sigmoid activation function which is biased to run from -1 to +1 and is symmetric about the origin.

Barto et al. implemented the system on an Adept One® robot with a Lord® force/torque sensor to follow both a flat and a curved surface. They had to manually select the stiffness, viscosity, and mass parameters to give stable tracking; then they let the system adapt. There were also many other parameters that required specification such as the contact stiffness, learning parameter ($\lambda$), initial variance of the random noise function, etc.

The Barto et al. system does not attempt to learn the motion strategy of any given task, but instead learns how to formulate its own impedance parameters and then maintains system performance by continuing on-line adaptation of the parameters. As such, it cannot be categorized as a skill-acquisition method.

Gullapalli et al. [23, 22, 24] approached the PIH problem using a special form of an ANN to learn an admittance mapping. The controller consisted of two nonlinear hidden layers and a stochastic real-valued (SRV) output layer which is described more completely in [21]. All of the nodes were fully connected. The admittance mapping performed by the controller took positions and forces as input features and mapped those to commanded output velocities. Similar to the controller proposed in the present work, the Gullapalli et al. controller was a reactive controller which modified the end-effector trajectory in real

time based on the sensor measurements. The nonlinear admittance mapping was learned through repeated attempts at the PIH task. The controller was demonstrated with a Zebra Zero robot inserting a chamferless peg into a chamferless hole. The strength of the system is the SRV output layer which allows the system to train in an un-supervised mode of iterative operation. In addition, the SRV allows the controller to handle some measurement uncertainties which often occur in the joint sensor readings and can additionally be caused by unknown manipulator dynamics and motion of the peg relative to the robot gripper.

The Gullapalli et al. system demonstrated that the controller could be trained to insert the peg in less than 100 time steps after training for about 150 runs. In addition, the amount of excessive contact force generated during the insertions continued to decline for up to 500 training runs. The decreased contact forces are indicative of increased task skill.

Although these results are impressive, we seek a controller and training method that do not have to use the manipulator during a lengthy training session where the learning transients might cause damage to the manipulator or its sensors. In addition, we seek a method of utilizing any available a priori information about the controller's mapping rather than starting the controller from an initial random state. Since the Gullapalli network uses position measurements as part of the input feature vector, the mapping it learns is inherently position-dependent. This means that the trained controller can only be implemented in the part of the workspace in which it was trained. If we want a controller that can operate anywhere within the workspace of the manipulator, then samples across the entire workspace must be included in the training data set. We seek a method that will be able to learn a globally-applicable mapping from data collected in a small sub-space of the entire manipulator workspace. This implies that we want a controller that will operate in tool-frame coordinates as opposed to the Gullapalli controller which operates in world-frame coordinates.

A final observation about the controller proposed in Gullapalli's work is that it requires a priori knowledge of the hole location so that the evaluation criteria can be computed during training. Recognizing that the Gullapalli et al. method is fairly robust to uncertainty, one must still have reasonable estimates of the hole location and orientation in order for the training to take place. When this restriction is combined with the position-dependency of

the learned mapping, it means that we must have prior knowledge of the location of every hole into which we want to insert a peg. We seek a method that does not require prior knowledge of the hole location or orientation.

In [5] Asada proposes using an ANN to perform the pattern recognition task in his deburring system. He also points out that inconsistent data[5] can hinder the learning and performance of a trained ANN. Therefore, he comes up with a criterion to prune out the bad data points. The criterion is Lipschitz's condition for continuity in a data set. By throwing out the pairs of training exemplars that have an output-distance-over-input-distance ratio in excess of a particular cutoff value, the continuity of the data is enhanced. The more continuous the data were, the better the learning and performance of the ANN system he implemented on his robotic deburring task. The concept of using the Lipschitz ratio for clipping out training exemplars is further evaluated in this dissertation.

The piece of research that motivated this dissertation was by Asada [2]. In this paper he proposed using an ANN to recognize the different force patterns associated with a particular condition in a contact task and then provide the proper velocity commands to complete the task all in one network. Since an ANN can perform a nonlinear mapping, the network could handle the problem of nonlinear compliance, i.e. the condition of multiple contacts where the simple combination of the appropriate response for each contact alone is inappropriate. An example of this situation is insertion of a chamferless peg into a chamferless hole. Asada simulated the training and testing of a network to perform a chamferless peg insertion as a two dimensional problem. However, his work suffers from the following two limitations:

- He proposed no general method of obtaining the exemplar vectors used for supervised training. He generated his training exemplars by a detailed theoretical analysis which assumed a simplified contact model (i.e. no friction, rigid bodies, etc.) Obtaining training data is often the hardest part of implementing a neural network solution. Therefore, a simple way of generating training exemplars is needed.

[5] Inconsistent data are defined by Asada to be pairs of training exemplars which are close together in the input space but far from each other in the output space. The distance between the exemplars is computed using the root-mean-square norm.

- The input features for his ANN required measuring the location, magnitude, and direction of the contact forces between the peg and the environment. Such data would only be available if the peg were fully instrumented with an array of contact sensors. Typical implementations will have no more than a single six-axis force/torque sensor which will not be able to discern single-point contact from two-point contact. Therefore, a different set of input features is needed.

### *2.2 Research on Transferring Human Skills to Robots.*

In [62], Yang et al. present a good introduction to the characteristics of a human controller attempting to demonstrate a skill and the challenges of skill learning. A skill is described as a mapping from *stimuli* onto *responses*. The characteristics of the human controller are presented in [62] as:

- In general, it is nonlinear, that is, there is no linear relationship between the stimuli and responses.
- It is time-variant, that is, the skill depends upon the environmental conditions from time to time.
- It is non-deterministic, that is, the skill is of inherently stochastic property, and thus it can only be measured in the statistical sense. For example, even the most skillful artist cannot draw identical lines without the aid of a ruler.
- It is generalizable, that is, it can be generalized through a learning process.
- It is decomposable, that is, it can be decomposed into a number of low-level subsystems.

We pay particular attention to the first two of these characteristics, since they are significant to the method proposed and the results presented in the present work. The nonlinear nature of the stimuli-to-responses mapping requires a system that can capture nonlinearities. For the work of Yang et al., a hidden Markov model (HMM) is used to represent the human skill as a parametric model. The skill learned was to remove a mockup of an Orbit Replaceable Unit (ORU) from its dock using the 7-degree-of-freedom (DoF) self-mobile space manipulator ($SM^2$) designed and built at Carnegie Mellon University. No discussion of the tolerances is presented, but Yang et al. do mention using 100 demonstrations by the operator as the basis for their training set. [62] states that "skill learning is, to a certain

extent, the problem of uncovering the characteristics from recording data which represents the nature of the skill." Thus, modeling human skill and transferring the skill to robots are two separate issues. In both our ANN approach and their HMM approach, however, we attempt to treat both issues simultaneously using a single controller. [62] further asserts that "HMM is a doubly stochastic model which is appropriate for the two stochastic processes that skill learning must deal with, i.e., the mental state (or intention) which is hidden, and the resultant action which can be measured." In the present work, we will rely on the ANN to provide access to the hidden mental state through the inherent averaging which occurs with repeated looks at the same data during training. In addition, the nonlinearity of the ANN will be our tool to properly model the nonlinear stimuli-to-responses mapping of the human controller.

As far back as 1979, Asada has been interested in work which has evolved into skill transfer. His early work amounted to recording and playback of force trajectories [3, 4]. Eventually Asada advocated using pattern recognition techniques to select the proper tool commands from a set of sensor measurements [6]. Asada applied the technique to a robotic deburring task which included a direct-drive selective compliant articulated robot for assembly (SCARA) robot with the electric grinding tool. This technique involved interviewing a human expert to identify some global rules of operation for the process to be learned. Then the operator and task were monitored as the operator demonstrated the task and various data were gathered. For the task of deburring, collected data were the position trajectory of the tool along and normal to the workpiece surface, the force normal and tangential to the workpiece, the angular velocity of the grinding wheel, and the armature current of the grinding tool as an indicator of the load torque. Some set of features were then extracted from the data set and used to form a pattern vector which was normalized by the standard deviation. For the case of deburring, Asada used the mean value, the peak value, and the root mean square as the feature set. Finally, discriminate functions were identified to classify the sensor patterns into the control actions that the human expert identified during his/her interrogation. The control actions for deburring were things like increase/decrease force, how many times the motion should be repeated, etc. The selection of the proper control

action based on the extracted sensor features (pattern recognition) is a perfect application for an ANN, and is probably what prompted Asada to propose just such a scheme in [5].

### *2.3 Summary.*

Several benchmark articles have been used to provide a background on the PIH and edge-mating tasks which we seek to address in this dissertation. Although very little literature was found for the edge-mating task, the PIH research will serve well to prepare one for the edge-mating task because the PIH task is more difficult and edge-mating can be considered as a possible phase of the PIH task. Although many researchers have analyzed and attacked the PIH problem with numerous techniques, none have presented general solutions for the nonlinear chamferless PIH task which are invariant to the hole's location in the manipulator workspace and can be implemented with the limited amount of sensor data we have assumed will be available. We will attempt to extend the ideas of several of these works as we strive for our stated objective.

## *III. Overview of Concept and Nomenclature.*

This chapter will present a quick overall view of how the proposed controller is intended to work. It begins with a block diagram that describes the entire process at a very high level followed by a snapshot of how an ANN controller would be used in an operational system after training had already been completed. Then the specific concepts and nomenclature associated with each segment of the process from raw data collection through ANN training and finally to controller operation are presented.

Figure 3.1 depicts the entire proposed process for learning a new accommodation (ACC) task. Initially the raw data are collected by observing a human demonstrating the task. Then the data are processed in some way to transform them into a suitable set of ANN training exemplars and the off-line training is conducted. Finally, the new set of learned ANN weights are implemented into the operational configuration on a robot and tested. The complete system concept consists of an operational configuration, raw data collection, training data preparation, the off-line training of the ANN, implementation of the controller, and the evaluation of the controller's performance. Each of these facets will be discussed in turn.

### *3.1 Operational Configuration.*

The proposed operational control system consists of an inner feedback control loop and an outer reactive control loop as shown in Figure 3.2. The inner loop is a stiff position-integral-derivative (PID) control servo that would run at a much higher rate than the outer

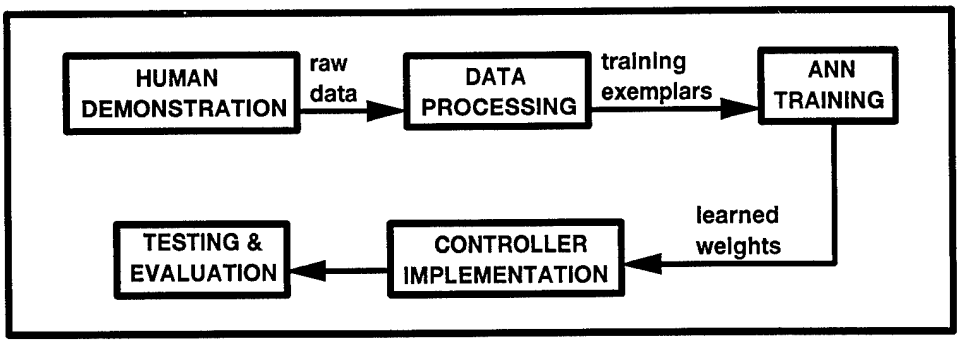



Figure 3.1 Proposed process used to learn an ACC task

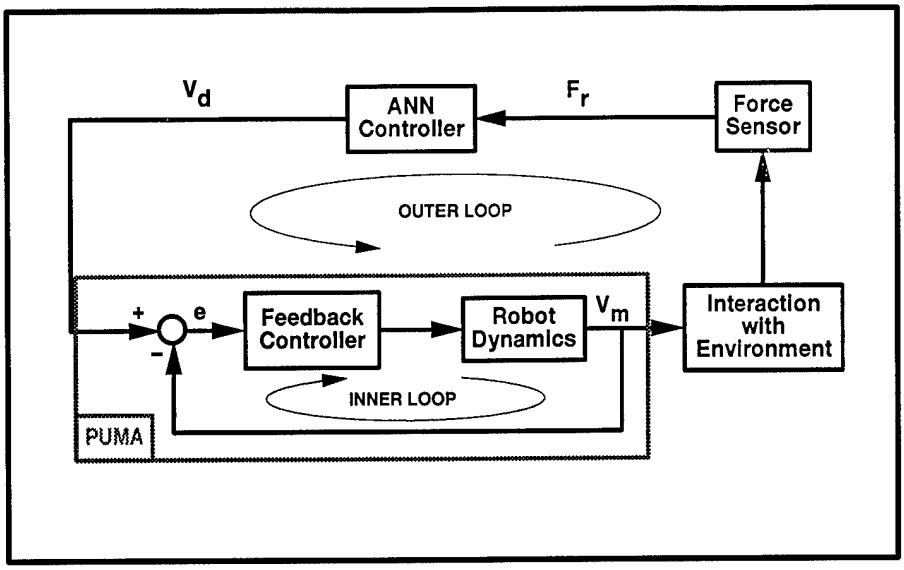



Figure 3.2 Simplified block diagram of operational system configuration

loop so that the inner loop performance can be idealized. The outer loop has an ANN controller which approximates an *accommodation* control law. Accommodation is the inverse relationship to mechanical impedance[48] and is generically given by

$$\vec{V} = \mathcal{A}\,\vec{F} \tag{3.1}$$

where $\vec{F}$ is a vector of forces, $\vec{V}$ is a vector of velocities, and $\mathcal{A}$ is the accommodation matrix. An accommodation control law was selected because it commands manipulator trajectory changes that are relative to the current position. When the commanded velocities are expressed in a coordinate frame that is attached to the peg, the accommodation controller provides commands that are independent of the joint configuration or location of the peg in the workspace of the manipulator. This is a characteristic called configuration-independence and is a major goal of this dissertation effort. As the robot interacts with the environment, it generates interface forces between the end-effector and the environment. A universal force-moment sensor (UFS) measures the interface forces of the environment *on* the peg and presents them to the ANN controller. The ANN maps those measured interface forces into desired velocities which the inner servo loop follows. The ANN can thus be described as a real-time trajectory generator for the low-level cartesian velocity servo loop.

The force vector presented to the ANN and the velocity it produces are both cartesian vectors expressed in the tool-frame. Thus, the ANN has no knowledge of where the end-effector is in the workspace or any knowledge of how the arm is configured. The effect of the peg's weight is removed from the measured force vector prior to its presentation to the ANN. The resulting controller is insensitive to both the location of the workpiece and its orientation with respect to gravity.

### *3.2 Raw Data Collection.*

Before one can implement a multi-layered perceptron ANN, it must be trained. The simple back-propagation training algorithm was used for this dissertation. Back-propagation training is described in Appendix A. It is a *supervised* training method, which means that the network must be presented with training data representing the desired input-output relationship in order to adjust itself to achieve that relationship as the repeated 'looks' are made.

The mapping performed by the ANN controller is

$$ {}^{T}\vec{V}_d(t) = g\left(\vec{F}(t)\right) \tag{3.2} $$

where ${}^{T}\vec{V}_d$ is the desired velocity vector, $\vec{F}$ is a feature vector based on measured forces, and $g(\cdot)$ is, in general, some nonlinear vector function. For the simple edge-mating task which is to align a square-tipped peg normal to a rigid, flat surface, Peshkin [42] shows that the function $g(\cdot)$ can be a linear operator representing a generalized damper. In this case, Eq (3.2) becomes

$$ {}^{T}\vec{V}_d(t) = {}^{T}\vec{V}_n + \mathcal{A}_a\,\vec{F}(t) \tag{3.3} $$

where ${}^{T}\vec{V}_n$ is a nominal free-space trajectory velocity that dictates the motion of the peg when it is not in contact with the environment and $\mathcal{A}_a$ is an accommodation matrix which is the inverse of a damping matrix, and characterizes a generalized damper. For later reference, we find it convenient to define a commanded velocity, ${}^{T}\vec{V}_c$, as

$$ {}^{T}\vec{V}_c(t) \equiv \mathcal{A}_a\,\vec{F}(t) \tag{3.4} $$

which, of course implies that

$$ {}^{T}\vec{V}_d(t) = {}^{T}\vec{V}_n + {}^{T}\vec{V}_c(t) \tag{3.5} $$

In Peshkin's work, ${}^{T}\vec{V}_n$ was taken as a simple vector which is superimposed with the commanded velocity. For the present work, ${}^{T}\vec{V}_n$ will be modulated according to whether the peg is in contact with the environment or not. The function used for modulation is given in Eq (5.47) in Section 5.5.1. An additional difference between the simple accommodation controller and the controller of the present work is that ${}^{T}\vec{V}_c$ is also modulated out of phase to that of ${}^{T}\vec{V}_n$ for the present controller. Thus, when ${}^{T}\vec{V}_n$ is turned on, ${}^{T}\vec{V}_c$ is turned off, and vice versa. The modulating function (referred to as the blending function in later sections) is a continuous function that gradually switches ${}^{T}\vec{V}_n$ off as the contact force magnitude increases from zero while simultaneously switching ${}^{T}\vec{V}_c$ on. The prime motives for introducing the modulating function are to prevent noisy force data encountered during free-space motion from influencing the nominal free-space commanded velocites and, more importantly, to allow for the fact that ${}^{T}\vec{V}_n$ is already implicitly contained in the human demonstration data on which the ANN is trained.

The choice of coordinate frames can greatly simplify the form of $\mathcal{A}_a$ in Eq (3.3). Peshkin [42:pp. 480] shows that if the origin of the coordinates is at the center of the peg tip and the task is planar, then $\mathcal{A}_a$ is a sparsely populated accommodation matrix having non-zero values only in the $(2,2)$ and $(3,3)$ positions. The precise values for these two non-zero elements can be found either analytically or by empirically tuning the matrix while running the controller. The simple linear mapping of Eq (3.3) can readily be learned by an ANN, so it serves as a good starting point for controller development. To teach an ANN the mapping of Eq (3.2) using the back-propagation training method, one must create input-output training pairs that are consistent with the relationship. The starting point for creating those training pairs is to collect raw data.

*3.2.1 Sources of Raw Data.* Three types of raw data were collected: *synthetic*, *real*, and *demonstration*. The synthetic raw data were generated by a computer algorithm that systematically populated the entire range of the input space with input vectors. The distribution of the vector samples within the range of the input space could be algorithmi-

cally controlled. For the planar problem, the input space is 3-dimensional so that evenly distributed vectors can be depicted as an evenly-spaced, 3-dimensional volume grid of coordinate points.

To collect 'real' data, an ACC matrix controller was implemented either on the robot or in simulation to perform the task. The explicit accommodation control law, given by the ACC matrix, provided the trajectory generation commands in response to the interaction forces measured by the UFS. As the accommodation controller performed the task, the measured forces and commanded velocities were recorded as 'real' raw data. Thus, the 'real' raw data represent the behavior of the robot under the command of the explicit accommodation control law.

Demonstration raw data were collected from observation of a human operator performing the desired task. In this case, the operator controlled the motion of the peg which was attached to some mechanism for recording the interaction forces and peg motion. Two different mechanisms were used to collect demonstration raw data during the course of this research: the PUMA robot and the PLIMMS. These mechanisms are described in detail in Section IV.

### *3.3 Configurations of Raw Data.*

Once the raw data are collected, they must be configured to create raw input-output training pairs. Figure 3.3 depicts the four different configurations used.

*3.3.1 SISO Data.* The first configuration is termed SISO because it amounts to configuring synthetic input data with synthetic output data. Creating SISO data involves systematically fabricating input vectors, $\vec{F}^*$, that span the entire input space of $\vec{F}$ as depicted in Figure 3.4(a).

The actual spacing of the $\vec{F}^*$ can be varied as desired, but for this discussion assume the $\vec{F}^*$ are evenly spaced along all input feature axes. Each of the $\vec{F}^*$ are then multiplied by an accommodation matrix, $\mathcal{A}_a$, to produce a desired output, $\vec{V}^*$. SISO data can be generated completely off-line, and they reflect a noise-free, consistent mapping from $\vec{F}^*$ to $\vec{V}^*$. By

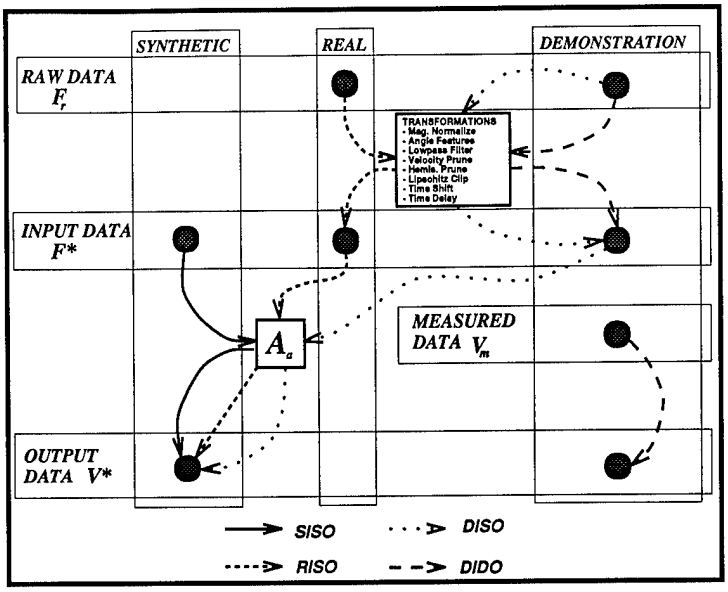



Figure 3.3 Diagram depicting the four configurations used to generate input-output training pairs for the ANN

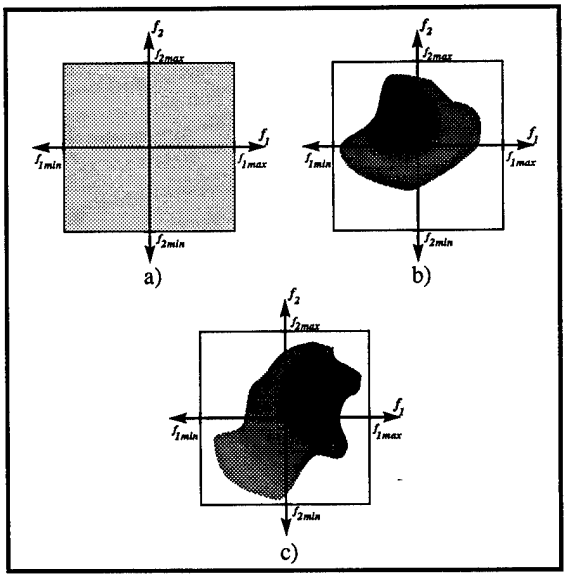



Figure 3.4 Illustration of how input feature vectors might be distributed across the input feature space of a two-input ANN for: a) SISO, b) RISO, and c) DISO training data

consistent, we mean that all the $\vec{F}^*$ are all related to all of their corresponding $\vec{V}^*$ by the exact same mapping matrix.

*3.3.2 RISO Data.* A second method, called RISO, requires implementing the linear feedback accommodation control law of Eq (3.4) on a robot or in simulation and letting it perform the task. With the control law implemented and running, the measured force data, $\vec{F}_r$, are captured while the robot interacts with the environment to complete the task. The captured $\vec{F}_r$ data are then propagated through Eq (3.4) to obtain desired $\vec{V}^*$. The RISO data are noise-free and consistent like the SISO data, but they are unevenly distributed across the input space as illustrated in Figure 3.4(b). An ANN trained from RISO data should learn the same linear accommodation relationship of Eq (3.4) that the SISO data produced. By virtue of how the input vectors were generated, they are clustered in a region of operation that is a subset of the entire input space. One might expect that the concentration of input vectors in the region of operation would provide higher resolution in the training data where the controller is likely to operate.

*3.3.3 DISO Data.* A third method of configuring the input-output training pairs is called DISO because it pairs demonstration input data with synthetically generated output data. The main difference between DISO data and RISO data is that a human is controlling the robot or other similar demonstration device when DISO data are collected. Gravity compensation of the device can relieve the human operator from carrying its weight as the task is demonstrated, but the operator may still have to overcome the friction and inertia of the device, especially if the joints are actuated and high gear ratios are used. The captured $\vec{F}_r$ data are then propagated through Eq (3.4) to obtain the desired $\vec{V}^*$. As a result, the mapping from $\vec{F}^*$ to $\vec{V}^*$ is still noise-free and consistent for DISO data. However, the distribution of the input vectors is naturally different from that of the RISO data because the feedback controller has changed from an explicit accommodation control law to a human controller. Since the characteristics of the controller are very different, one would expect a markedly different clustering of the $\vec{F}^*$ in the input space as Figure 3.4(c) attempts to illustrate. As in the case of SISO and RISO data, an ANN trained from DISO data should learn the linear accommodation relationship of Eq (3.4).

*3.3.4 DIDO Data.* The final method of configuring input-output training pairs is called DIDO because it pairs demonstration input and output data together. The demonstration device is controlled the same as it is for DISO data collection, but the output vector for each training pair is the measured velocity, ${}^{T}\vec{V}_m$, rather than the velocity produced by Eq (3.4). This small change in procedure represents a vast change in principle. A fundamental difference between DIDO data and any of the previous three types described is that an ANN trained from DIDO data may learn some nonlinear relationship as shown in Eq (3.2) rather than necessarily learning the linear relationship in Eq (3.3). Although the clustering of the input feature data for DIDO training data is the same as for DISO, one would expect the output training vectors to differ significantly. Besides the possibility that the human may use a different accommodation matrix mapping, DIDO data can contain several other corrupting factors such as:

- Noise in the output vector data due to measurement noise on the joint angle sensors, differentiation noise, and numerical errors generated while transforming joint velocities to cartesian velocities.
- Inconsistencies in the output vector data due to poor task performance by the human operator.
- The mapping relationship observed by the human operator may vary as a function of time.
- Possible mismatching of input and output vectors due to causal time delay imposed by the human operator.
- DIDO data depict a mapping from $\vec{F}$ to ${}^{T}\vec{V}_d$, which means they include the operator's rendition of ${}^{T}\vec{V}_n$ in accordance with Eq (3.3).

All of these factors substantiate our expectation that DIDO data will not exhibit the same function mapping as the sparsely populated, decoupled accommodation matrix which is derived by Peshkin in [42]. It is very likely that the DIDO data will exemplify a mapping matrix that is fully populated which introduces coupling influences between the axes of the task. However, the simple accommodation matrix form presented by Peshkin for the edge-mating task is not the only matrix form which can perform the task, so there is adequate reason to believe that with careful controller design and the nonlinear mapping capability of the ANN, there is a good possibility that the proposed ANN controller can learn to accomplish the desired task in the presence of all the complications mentioned above.

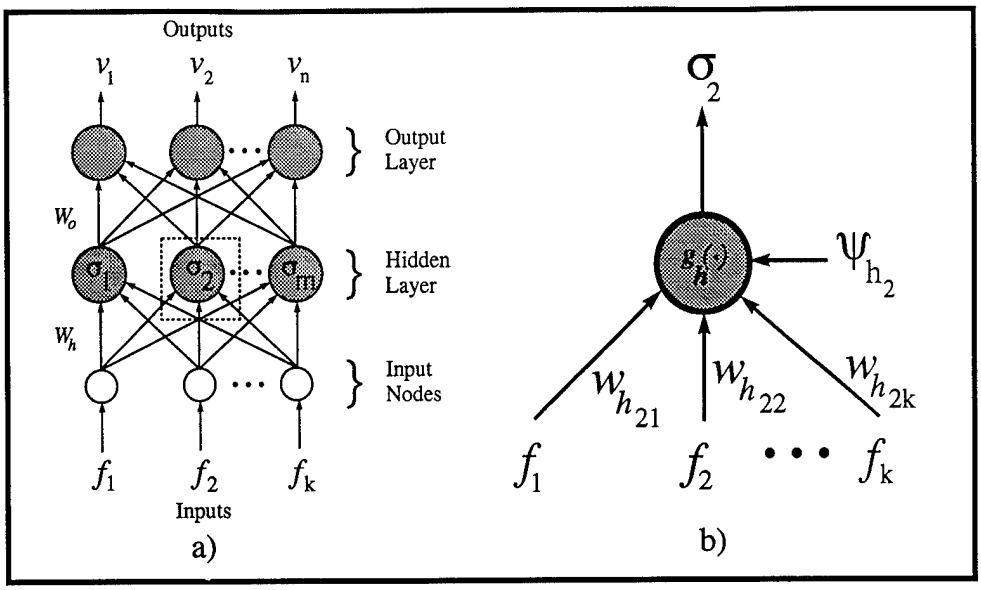



Figure 3.5 Schematic of MLP ANN showing: a) overall architecture, and b) details of second node in hidden layer

### *3.4 Training Data Preparation.*

The raw input-output pairs of data might undergo one or more processing steps in preparation for training. Examples of the data processing steps include low-pass filtering, velocity pruning, collision pruning, etc. These steps are described in detail in Section 5.2.

### *3.5 ANN Architecture and Training.*

Figure 3.5 shows a simple schematic of the MLP ANN used for this dissertation. It is a fully-connected, two-layer ANN. *Fully-connected* means that all of the input nodes are connected to each of the hidden nodes, and all of the hidden nodes are connected to each of the output nodes. Although at first glance it may appear to have three layers, it is called a *two-layer* ANN because the input nodes are simply connections for the input features to be applied and do not perform any transformation on the data. So, the hidden layer and the output layer are the only layers that count.

The algorithm used to train the ANN s is *backward error propagation* with momentum. Appendix A presents a brief summary of the basic equations relating to ANN s. It includes

the equations for computing the ANN output vector for a given input vector as well as the equations used to implement the back error propagation algorithm.

### *3.6 ANN Training Evaluation.*

When a particular ANN structure had been trained, it was evaluated to determine the success of the training session. In its simplest form, the evaluation consisted of examining the shape of the total squared error curve during training, the tracking error between the computed ANN output with the trained weights, $\vec{V}^*$, and the desired output identified by the training data set, $\vec{V}$. Where appropriate, additional evaluations were made by using various techniques to extract the mapping matrix that the ANN appeared to be emulating. These techniques are described in detail in Section 5.4.

### *3.7 Controller Implementation.*

Once training is complete, the weights and biases are implemented in the ANN controller. This amounts to simply capturing the saved weights and biases from the training and plugging them into the ANN feed-forward calculations. The resulting controller is then incorporated into the robot control structure as depicted in Figure 3.2. In the cases where the controller was implemented via simulation, the software was configured to support the block diagram described in Figure 5.8 of Section 5.5.1.

### *3.8 Controller Performance Evaluation.*

Since the final total squared error documented during training cannot be related directly to how well the controller will work, a method had to be developed to evaluate the controller's performance when the weights and biases were implemented. The goal was to have a performance metric that could be used to identify improvements or degradations in the effectiveness of the controller as different sets of weight and bias data were run on the PUMA. With this metric in hand, one could then argue about the effects of varying different ANN architecture and training parameters. The metric is computed *after* a run is complete to assess the results.

A scalar performance metric, $\zeta$, was developed which is a function of the path length traversed during the task, the stability of the motion, and the ease of the motion. The metric is a dimensionless parameter which reflects the performance of the controller of interest relative to the performance of the "best" accommodation matrix controller run under the same conditions. Poor performance results in higher $\zeta$ values, so the goal is to minimize $\zeta$. Thus, the value of $\zeta$ reflects the economy of motion or *directness* of the controller's solution. The details of how $\zeta$ is derived and computed are given in Section 5.7.

### *3.9 Summary*

This section has presented the general scheme of how the proposed ANN accommodation controller is to be trained, implemented, and tested. The three types of raw data (synthetic, real, and demonstration) which were used in this research were described, and the reader was introduced to the four acronyms used to describe how the raw data can be configured into input/output (I/O) data pairs. These four configurations (SISO, RISO, DISO and DIDO) will be referred to extensively in the remainder of this document, so the reader should be familiar with them before reading ahead. After the data are configured into I/O pairs, they can then be processed in one or more ways to produce the actual training data feature vectors which are presented to the ANN during its off-line, supervised training session. Following training, the ANN is implemented as a controller and its performance is evaluated. This is the overall scheme of the proposed controller approach. The following chapters will detail the mechanics of the method and present the results of the investigation into its feasibility.

# *IV. Hardware Description.*

There were four types of custom-built hardware for this dissertation: the end-effector that was mounted on the last link of the robot and the DIDO data collection mechanisms, the training handles that offered the human operator a place to grasp the robot while demonstrating tasks, the PLIMMS used to collect raw DIDO data, and the PUMA robot. Each of these will be discussed in turn.

## *4.1 End-Effector Design.*

The end-effector was simply a rectangular peg fabricated from solid T6016 aluminum stock. The peg had a cross section 75 millimeter (mm) by 35 mm with an overall length of 317.5 mm, as shown in Figure 4.1. The mass of the peg was 0.6947 kilogram (Kg) while its center of gravity (CG) was 176 mm from the face of the mounting flange not including the alignment button. The stiffness of the peg was far greater than that of the robot or either of the data collection mechanisms.

## *4.2 Training Handle Design.*

A training handle was designed to provide a handgrip for a person to backdrive the PUMA robot during collection of DISO and DIDO training data. The handle was designed to mount on the flange of the PUMA's sixth link as shown in Figure 4.2. The screws of the UFS pass through clearance holes in the training handle mounting plate and thread into the flange of link six, thereby sandwiching the training handle in place. The handle grips are made from thin-wall, stainless steel tubing, while the mounting plate is made from stainless steel plate. The curvature of the handles is such that lines extending from the open ends of the handle grips will pass through a point at the tip of the peg. This design feature was intended to enhance the user's ability to make rotation motions about the tip of the peg.

## *4.3 PLIMMS Design.*

A three-link planar mechanism was designed as a light-weight, easily backdriveable device to collect demonstration data. Figure 4.3 shows the basic structure of the device.

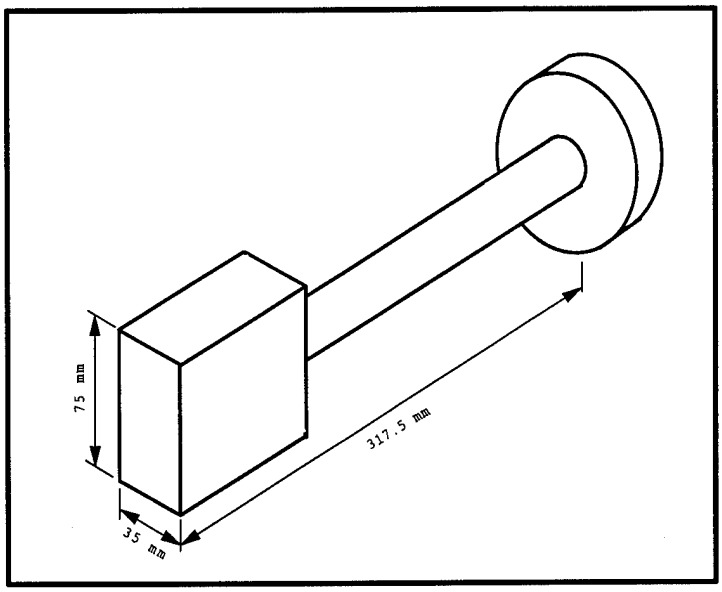



Figure 4.1 Illustration of the end-effector peg showing its dimensions

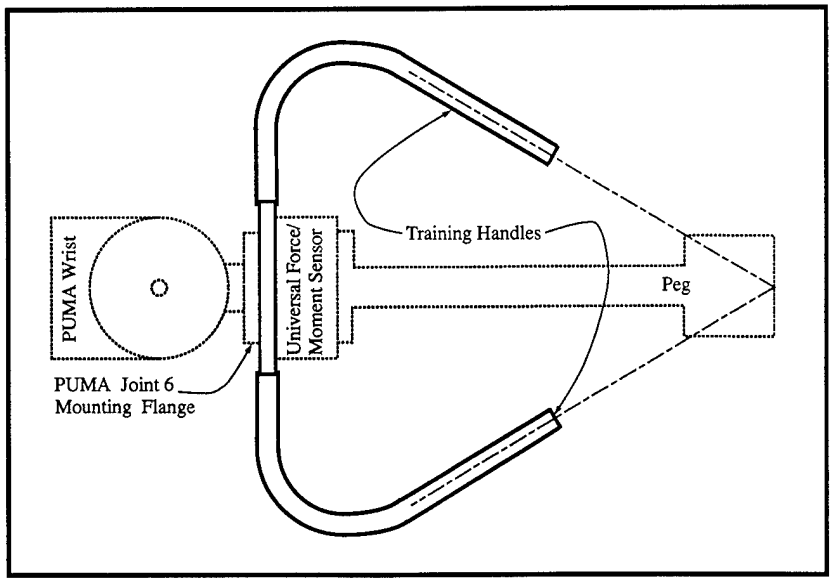



Figure 4.2 Training handles used to backdrive the PUMA robot during demonstration training data collection

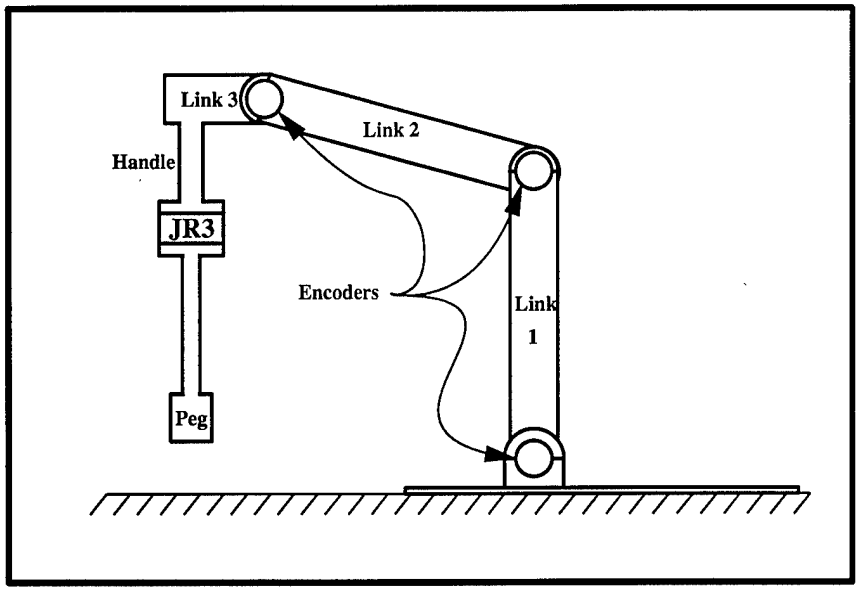



Figure 4.3 Basic structure of the PLIMMS showing the encoder locations and end-effector configuration

Optical encoders were mounted on each of the three revolute joints to measure the linkage joint angles. The 12-bit encoders provided for high accuracy in the position measurement. The links were made of 0.125-inch thick wall aluminum box channel while the dog-ears of the joints were machined from solid aluminum blocks. Ball bearings were used in the joints to minimize the friction and make it easily backdriveable. The forward kinematics for the linkage were used to transform the measured joint positions/velocities to peg tip cartesian positions/velocities. The UFS was mounted on the end-effector flange of link three. The peg was then mounted to the UFS. To operate the device, the user grasped the tubular part of link 3 just above the UFS with one hand and moved the peg as desired. The output of the UFS was recorded on an IBM®-compatible personal computer (PC) along with the joint encoder outputs.

The software used to control the PLIMMS data collection system was written mostly in C with only a few low-level communication routines coded in assembly language. The software for the PLIMMS ran on a PC 386/33 computer. It was compiled using the Borland Corporation TurboC® compiler. The UFS was interfaced to the PC via a parallel I/O card.

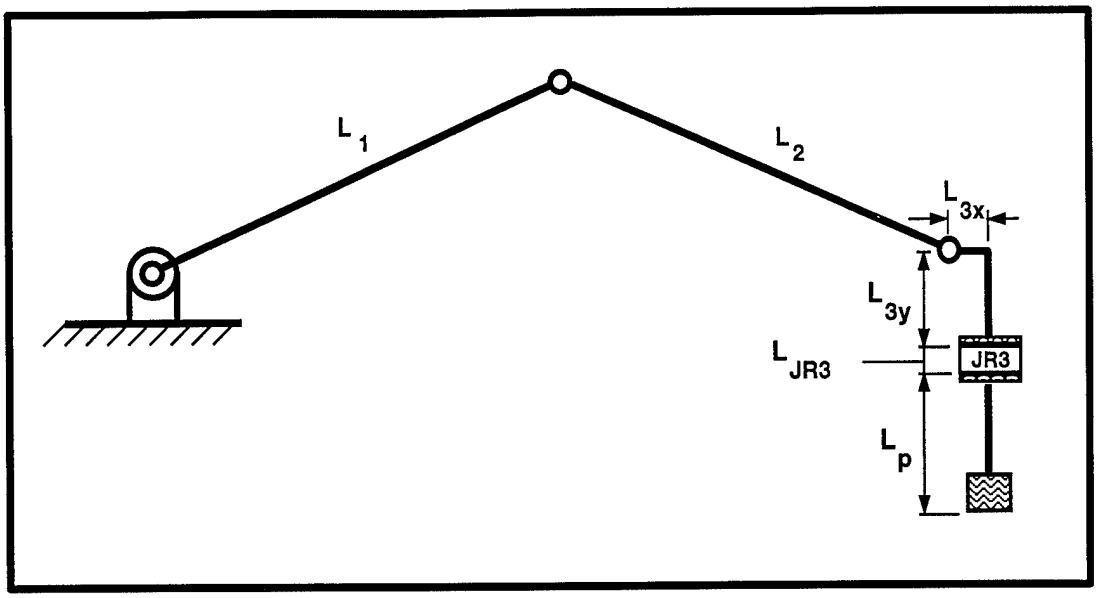



Figure 4.4 Kinematic description of the PLIMMS

Figure 4.4 shows the kinematic layout of the PLIMMS. The as-built dimensions of the PLIMMS were:

$$
\begin{aligned}
L_1 &= 417.5\,\text{mm} \\
L_2 &= 417.5\,\text{mm} \\
L_{3x} &= 40.0\,\text{mm} \\
L_{3y} &= 145.7\,\text{mm}
\end{aligned}
$$

The end plate of link 3 was machined to receive the same UFS that was used on the PUMA robot.

### *4.4 Robot Testbed Description.*

The Unimate PUMA 562 robot served as the testbed for all of the experimental work. The PUMA is a 6-DoF, vertically-articulated, industrial (V-AI) manipulator. The six revolute joints of the PUMA are driven by DC servo motors via geared transmissions. Figure 4.5 shows the basic layout of the PUMA robot.

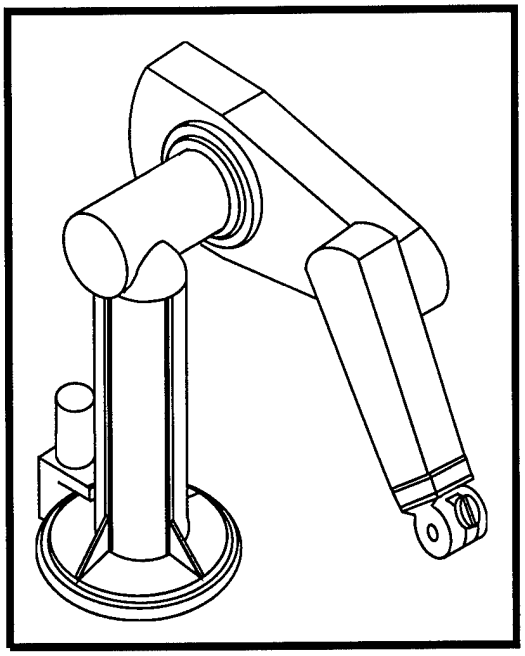


Figure 4.5 The PUMA 562 robot structure

The PUMA robot is a common university research tool, and there are many references that describe its characteristics in detail [14, 20, 34, 49, 50]. The experiments were done using the Air Force Institute of Technology (AFIT) Robotic Control Algorithm Development and Evaluation (ARCADE) operating system which provides the capability to experimentally evaluate a wide range of digital control algorithms [33]. The ARCADE system bypasses the standard Unimate controller and VAL control language so that torques can be commanded directly to each of the joint motors. The commanded torques, $\vec{\tau}_c$, were computed by software running on a Digital Equipment Corporation® (DEC) VAXStationIII host computer (VMS host computer) which is interfaced to the PDP 11/73 that provided low-level control of the PUMA. When the controller was operating, the desired joint velocity vectors, $\dot{\vec{q}}$, were integrated and fed as desired joint angles, $\vec{q}_d$, to the ALTER mode of the PUMA's VAL controller. The PUMA ALTER mode used a low-level, high-speed PID control loop to servo to the $\vec{q}_d$.

A six-axis (3 forces, 3 moments) UFS (model number UFS-3012A25-U560) manufactured by $JR^3$, Incorporated was attached between the last link of the PUMA and the peg to provide force feedback measurements. The UFS comes with its own local power supply, pro-

cessor electronics and operating system [30]. The UFS samples the forces and moments at a rate controlled by its own operating system and makes that data available asynchronously via a parallel connection. This digital input-output (DIO) interface was connected to a DEC DRV-11J digital input-output (DIO) card in the PDP's QBus when ARCADE controlled the robot. The UFS sample rate was set at 300 Hz for the experiments conducted in this research.

Two footswitches were interfaced into the system to control operation. The first footswitch (colored red) was wired in series with the kill switch on the PUMA controller as a safety feature. This footswitch was a normally-closed, momentary contact switch; it gave the operator ready access to disable the PUMA during data collection or operation without having to use his hands or reach for the switch. The second footswitch (colored green) was interfaced into the analog-to-digital converter (ADC) to provide a begin or end signal for the control software. This footswitch was a normally-open, momentary contact switch; it enabled the operator to signal that he was ready to proceed and indicate when a data run was over without having to use his hands or a keyboard interface. An example of how the green footswitch was used occurred during the positioning of the PUMA for the start of a test run. Upon pressing the green footswitch, the PUMA was gravity compensated and backdriveable so the operator could maneuver it to the desired starting position manually.

As was mentioned above, the computing hardware for operating the robot was a VMS host computer with a PDP 11/73 providing the low-level control interface to the PUMA robot under the ARCADE environment. Because the ARCADE environment was evolving, the control software was written in a mix of FORTRAN, C, and assembly languages. Since the original ARCADE system was developed in FORTRAN, all of the communication protocols were based on calls to FORTRAN and VAXLab subroutines. In addition, the VMS host computer was running the VMS operating system which was written in FORTRAN, so all of the routines that interfaced with the operating system or communicated with the robot via the PDP 11/73 were written in FORTRAN. This included routines to read and write data across the serial and parallel connections, data plotting routines, etc. The PDP 11/73 was running the assembly code to provide the low-level serial and parallel communications between the VMS host computer and the robot. That code essentially wrote the torques out

to the PUMA as motor currents and read in the joint positions from the PUMA's joint angle encoders.

### *4.5 Summary.*

This chapter has presented the two main pieces of equipment that were used to collect demonstration training data. The PLIMMS was designed to be a light-weight, low-friction motion measurement system that was easily back-driveable for collecting demonstration data. The PUMA manipulator was used for both data collection and controller implementation and testing. The special training handles were attached to the PUMA manipulator to provide a comfortable handhold when demonstration data were collected. The peg used for all the testing was also described. These tools were critical to the success of this research effort.

# V. Methodology

This chapter describes the methods and algorithms that were used to conduct this research. It is organized as follows: Section 5.1 details the raw data collection procedure, Section 5.2 describes the data processing algorithms available, Section 5.3 explains the ANN architecture and training technique, Section 5.5 describes the implementation of the trained ANN controller, and finally, Section 5.7 identifies how the controller was evaluated.

## 5.1 Raw Data Collection.

*5.1.1 SISO Data Collection.* To collect SISO data, a custom program was written which allowed the user to select the spacing function, the range between the maximum and minimum value on each axis, and how many subdivisions of those ranges to use. The ranges were specified by a single parameter, $\beta$, for each axis which identified the minimum as $-\beta$ and the maximum as $+\beta$ for that axis. The parameter, $\alpha$, specifying the number of subdivisions to use actually specified how many subdivisions to use between the minimum value and zero. If $\alpha$ was given as $n$, then the number of discrete values for that axis was $2n + 1$. The program then systematically generated the input vectors, $\vec{F}$, by starting with all the components at their minimum values and proceeding to increment each component, in turn, according to the prescribed spacing function until the maximum values for all three components were reached. These $\vec{F}$ were then propagated through the desired $\mathcal{A}_a$ to get the $\vec{V}$ as per Eq (3.1).

*5.1.2 RISO Data Collection.* RISO data were collected from two different sources during this research: from the simulation and from the robot manipulator. When the simulation was used to collect RISO data, a desired $\mathcal{A}_a$ was implemented in a controller simulation to produce a recorded stream of $\vec{F}$ and $\vec{V}$ data. To ensure proper time sequence alignment of the data, the recorded $\vec{F}$ from the simulation were subsequently propagated through the desired $\mathcal{A}_a$ to produce the synthetic $\vec{V}$ vectors. The matching set of recorded $\vec{F}$ and $\vec{V}$ constituted the desired RISO training data set.

To collect RISO data from the PUMA manipulator, the control block diagram shown in Figure 5.1 was implemented. The explicit accommodation control law, given by an accommo-

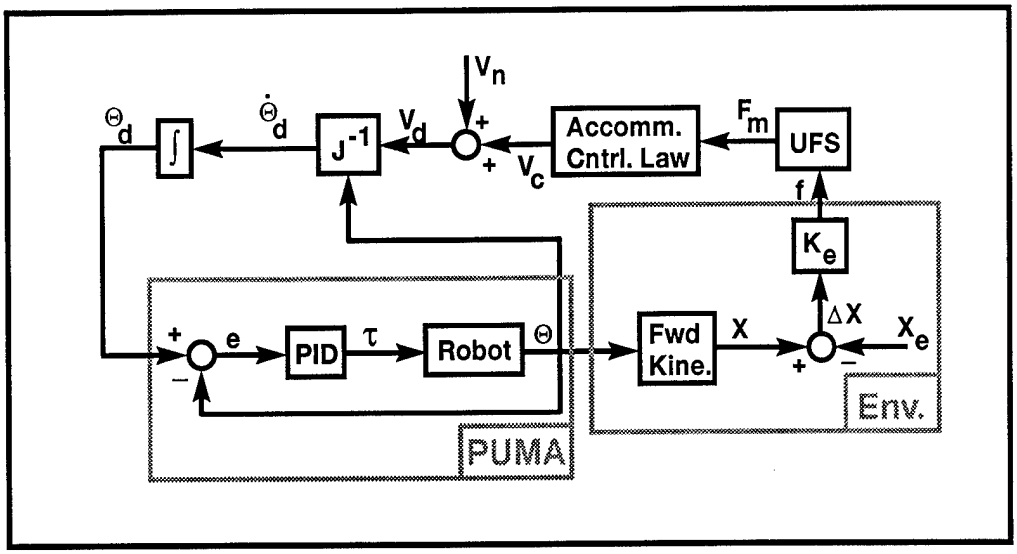



Figure 5.1 PUMA control system block diagram with accommodation controller running during collection of RISO data

dation matrix, provided the trajectory generation commands in response to the interaction forces measured by the UFS. As the accommodation controller performed the task, the measured forces, $F_m$, and commanded velocities, ${}^T\vec{V}_c$, were recorded. The nominal velocity, ${}^T\vec{V}_n$, was maintained at a fixed value throughout the task execution. The desired planar components of the $F_m$ were extracted to form the $\vec{F}$ which was then propagated through the desired $\mathcal{A}_a$ to yield the desired outputs, $\vec{V}$. Thus, the "real" inputs were used to determine the distribution of the data set while Eq (3.1) ensured that a perfectly consistent mapping was synthetically generated. The result, of course, was RISO training data.

*5.1.3 DISO Data Collection.* As mentioned in Section 3.3, DISO data are a hybrid form of the DIDO and SISO data types. The procedure for collecting DISO data was simply to use the demonstration inputs from a DIDO data set and propagate them through a desired $\mathcal{A}_a$ to compute matching synthetic output vectors. There was no special equipment or algorithms unique to the DISO data collection.

*5.1.4 DIDO Data Collection.* DIDO data were collected from two different experimental platforms during the course of this research: the PUMA robot and the PLIMMS.

Each device will be discussed in turn, followed by the method of differentiation which was common to both.

*5.1.4.1 PUMA DIDO Data Collection.* When the PUMA was used to collect data, the controller was configured to provide gravity compensation of the arm's weight while the operator manually moved the peg using the training handles described in Section 4.2 as their interface. In this scenario, the human is providing the trajectory generation commands in response to the interaction forces he/she feels. As compared to the "real" data collection system, the accommodation feedback control loop that controls the robot during "real" data collection disappears and the inner PID feedback loop is replaced with a model-based feed-forward gravity compensation loop as shown in Figure 5.2. For the planar edge-mating task studied herein, the PUMA joints that did not contribute to motion in the vertical plane (joints 1, 4, and 6) were servoed to their zero positions with a PID feedback control law. Thus, the motion of the robot was restricted to lie in the vertical plane. The coordinate system describing the PUMA as a planar manipulator is presented in Figure 5.3. Note that, in the PUMA coordinates, the vertical plane is the X-Z plane which contrasts with the X-Y plane description used for the simulation work described in Section 5.5.1.

To use the DIDO data for training required the velocities to be expressed as cartesian vectors in the tool-frame. The vector of measured robot joint angles, $\vec{q}$, was differentiated to produce a vector of joint velocities, $\dot{\vec{q}}$. The manipulator Jacobian, ${}^{T}\mathcal{J}(\vec{q})$, converted $\dot{\vec{q}}$ to ${}^{T}\vec{V}_p$.

$$ {}^{T}\vec{V}_p = {}^{T}\mathcal{J}(\vec{q})\ \dot{\vec{q}} \tag{5.1} $$

It is important to realize that ${}^{T}\mathcal{J}(\vec{q})$ must be derived specifically for transforming joint velocities to a *tool-frame* cartesian velocity of the point at the origin of the peg coordinate frame in the center of the peg tip. We now derive the ${}^{T}\mathcal{J}(\vec{q})$ for this purpose by differentiating the kinematic equations in the world-frame coordinates and then transforming the result to the tool-frame. There are other methods [20] for deriving the Jacobian, but the differentiation method was deemed easiest for the planar configuration of the PUMA.

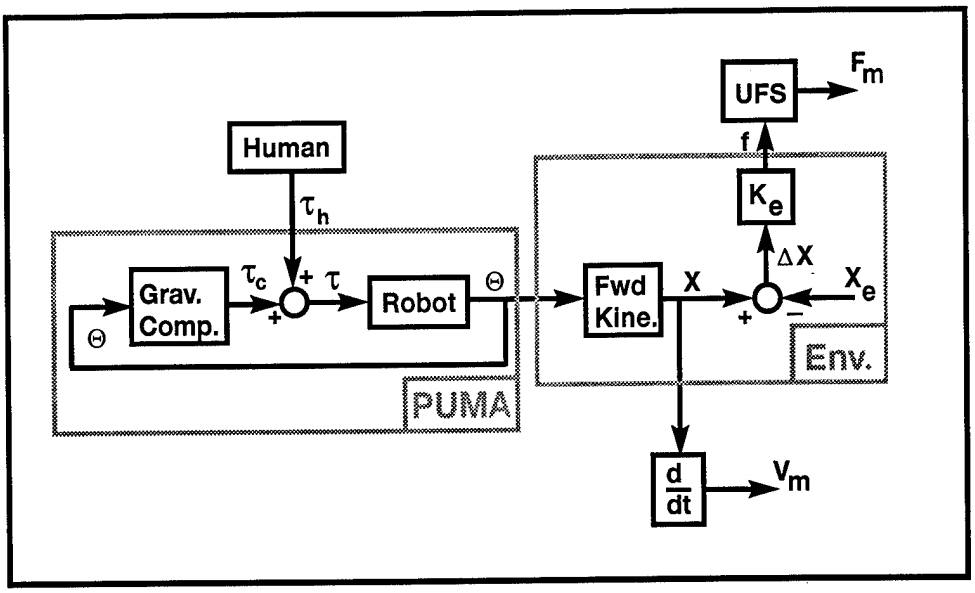



Figure 5.2 PUMA control system block diagram during DIDO data collection

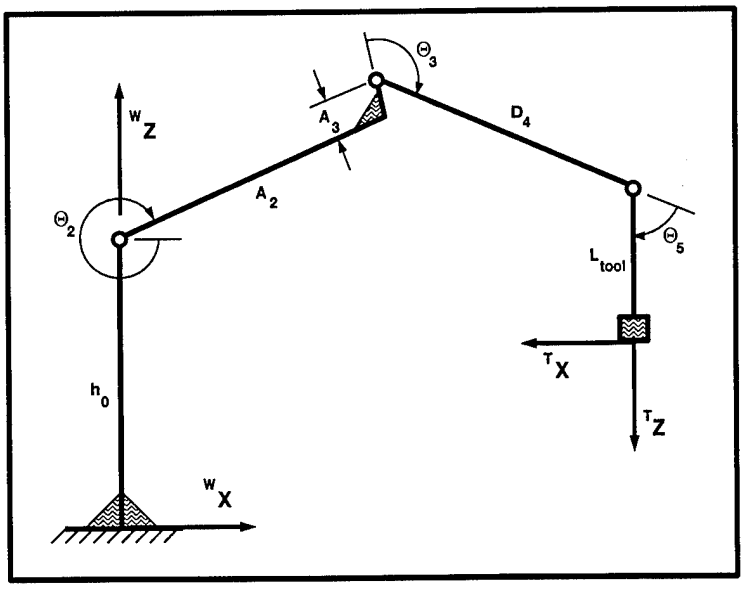



Figure 5.3 Coordinates used to describe the PUMA robot as a planar manipulator operating in the vertical plane

Using Figure 5.3 as a guide, the forward kinematic equations for the position of the peg, ${}^{w}\vec{X} = \left\{ {}^{w}x_p, {}^{w}z_p, {}^{w}\theta_p \right\}^T$ are

$$
\begin{aligned}
{}^{w}x_p &= A_2 \cos\theta_2 + A_3 \sin\theta_2 + D_4 \sin\theta_{23} + L_T \sin\theta_{235} && (5.2)\\
{}^{w}z_p &= h_0 - A_2 \sin\theta_2 + A_3 \cos\theta_2 + D_4 \cos\theta_{23} + L_T \cos\theta_{235} && (5.3)\\
{}^{w}\theta_p &= \theta_{235} && (5.4)
\end{aligned}
$$

where $A_2$ and $D_4$ are the Denavit-Hartenburg parameters [20] that represent the lengths of links 2 and 3, respectively, of the PUMA, $A_3$ is the offset in joint 2, $h_0$ is the height of the joint 2 axis above the ground, $L_T$ is the total distance from the axis of joint 5 rotation to the tip of the peg, $\theta_{23} = (\theta_2 + \theta_3)$, and $\theta_{235} = (\theta_{23} + \theta_5)$. Note that $\theta_p$ is measured about the positive $Y_o$ axis from the positive $X_o$ axis. Taking the time derivative of these equations, we get

$$
\begin{aligned}
{}^{w}\dot{x}_p &= -A_2 \sin\theta_2 \left(\dot{\theta}_2\right) + A_3 \cos\theta_2 \left(\dot{\theta}_2\right) + D_4 \cos\theta_{23} \left(\dot{\theta}_2 + \dot{\theta}_3\right) + L_T \cos\theta_{235} \left(\dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_5\right)\\
{}^{w}\dot{z}_p &= -A_2 \cos\theta_2 \left(\dot{\theta}_2\right) - A_3 \sin\theta_2 \left(\dot{\theta}_2\right) - D_4 \sin\theta_{23} \left(\dot{\theta}_2 + \dot{\theta}_3\right) - L_T \sin\theta_{235} \left(\dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_5\right)\\
{}^{w}\dot{\theta}_p &= \dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_5 \qquad (5.5)
\end{aligned}
$$

which can be written in the form of

$$
{}^{w}\vec{V}_p = {}^{w}\mathcal{J}(\vec{q})\,\dot{\vec{q}} \qquad (5.6)
$$

where the cartesian velocity vector is defined as ${}^{w}\vec{V}_p = \left\{ {}^{w}\dot{x}_p, {}^{w}\dot{z}_p, {}^{w}\dot{\theta}_p \right\}^T$. The manipulator Jacobian matrix expressed in **world-frame** coordinates is given by

$$
{}^{w}\mathcal{J}(\vec{q}) = \begin{bmatrix}
(-A_2 \sin\theta_2 + A_3 \cos\theta_2 + D_4 \cos\theta_{23} + L_T \cos\theta_{235}) & (D_4 \cos\theta_{23} + L_T \cos\theta_{235}) & (L_T \cos\theta_{235}) \\
(-A_2 \cos\theta_2 - A_3 \sin\theta_2 - D_4 \sin\theta_{23} - L_T \sin\theta_{235}) & (-D_4 \sin\theta_{23} - L_T \sin\theta_{235}) & (-L_T \sin\theta_{235}) \\
1 & 1 & 1
\end{bmatrix} \qquad (5.7)
$$

and the vector of joint velocities, $\dot{\vec{q}}$, is defined as $\dot{\vec{q}} = \left\{\dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_5\right\}^T$ .

Eq (5.7) expresses the velocity of the point at the center of the peg tip as a function of the manipulator joint velocities and expressed in the *world-frame* cartesian coordinates. For the ANN controller, we require the velocity to be expressed in the *tool-frame* coordinates that are attached to the peg. To change the frame of reference of the world-frame Jacobian to the tool-frame, the following relationship out of [14:p.172] is used for the full 6-DoF problem:

$$ {}^T\mathcal{J}(\vec{q}) = \begin{bmatrix} {}^TR_w & \mathbf{0} \\ \mathbf{0} & {}^TR_w \end{bmatrix} {}^W\mathcal{J}(\vec{q}) \tag{5.8} $$

where ${}^TR_w$ is the rotation matrix that transforms a velocity in the world-frame to one in the tool-frame, and **0** is a 3×3 zero matrix. For the *planar* problem at hand,

$$ {}^TR_w = \begin{bmatrix} \cos(\theta_2+\theta_3+\theta_5) & -\sin(\theta_2+\theta_3+\theta_5) & 0 \\ \sin(\theta_2+\theta_3+\theta_5) & \cos(\theta_2+\theta_3+\theta_5) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.9} $$

and Eq (5.8) reduces to just the upper three rows of Eq (5.8) and can, therefore, be written as:

$$ {}^T\mathcal{J}(\vec{q}) = {}^TR_w \; {}^W\mathcal{J}(\vec{q}) \tag{5.10} $$

Applying Eq (5.10) to the expressions for the world-frame Jacobian given in Eq (5.7), the manipulator Jacobian matrix expressed in **tool-frame** coordinates is found to be

$$ {}^T\mathcal{J}(\vec{q}) = \begin{bmatrix} (L_T + A_2\sin\theta_{35} + A_3\cos\theta_{35} + D_4\cos\theta_5) & (L_T + D_4\cos\theta_5) & (L_T) \\ (-A_2\cos\theta_{35} + A_3\sin\theta_{35} + D_4\sin\theta_5) & (D_4\sin\theta_5) & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.11} $$

In summary, to collect the DIDO output data on the PUMA robot, the recorded time history of $\vec{q}$ is differentiated to yield $\dot{\vec{q}}$ which is then premultiplied by ${}^T\mathcal{J}(\vec{q})$ given in Eq (5.11) to yield the desired ${}^T\vec{V}_p$ as desired output data. The planar components of the recorded time history of force data, $F_m$, are used as the input data without transformation.

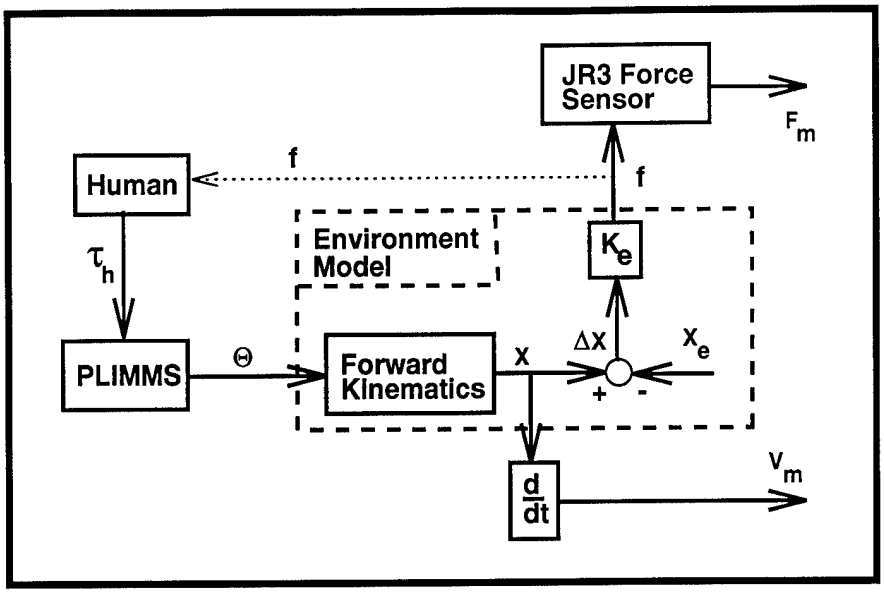



Figure 5.4 PLIMMS controller block diagram during demonstration data collection

*5.1.4.2 PLIMMS DIDO Data Collection.* Since the PLIMMS was not an actuated system, it did not have any feedback control embedded in the block diagram as shown in Figure 5.4. The PLIMMS was designed to be lightweight, low-friction, and easily backdriveable. Section 4.3 gives a more detailed description of the PLIMMS hardware and control software. The joint encoders and the UFS were calibrated prior to each session of DIDO data collection with the PLIMMS.

In a fashion equivalent to that used for the PUMA when collecting DIDO data, the PLIMMS provides measurements of joint positions, $\vec{q}$, rather than joint velocities, $\dot{\vec{q}}$. Therefore, we must use the same procedure to differentiate the $\vec{q}$ and transform the resulting $\dot{\vec{q}}$ into ${}^{T}\vec{V}_p$ using the Jacobian matrix, ${}^{T}\mathcal{J}(\vec{q})$, describing the PLIMMS structure. We now derive the desired ${}^{T}\mathcal{J}(\vec{q})$ in the same way it was done in Section 5.1.4.1 for the PUMA.

Figure 5.5 shows the coordinates used to describe the position of the peg expressed in world-frame coordinates. The forward kinematics of the PLIMMS express the position of the point at the center of the peg tip in world-frame coordinates, ${}^{w}\vec{X}$, as a function of the PLIMMS joint angles. They are given by

$$ {}^{w}x_p = L_1 \cos\theta_1 + L_2 \cos\theta_{12} + L_{3x} \cos\theta_{123} + (L_{3y} + L_{\mathrm{JR3}} + L_p) \sin\theta_{123} \qquad (5.12) $$

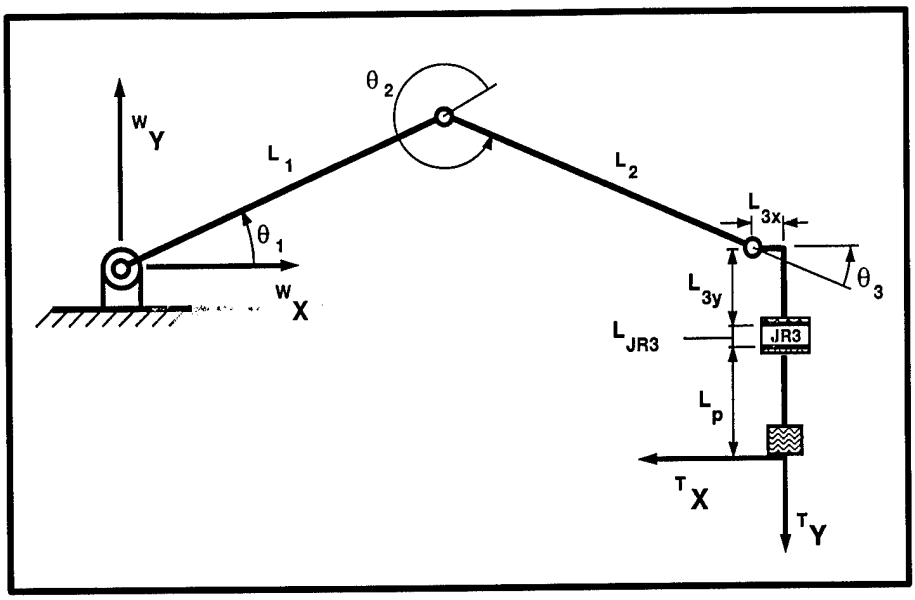



Figure 5.5 Coordinates used to describe the position of the peg on the PLIMMS during demonstration data collection

$$
\begin{aligned}
{}^{w}y_p &= L_1 \sin\theta_1 + L_2 \sin\theta_{12} + L_{3x} \sin\theta_{123} - (L_{3y} + L_{\mathrm{JR3}} + L_p) \cos\theta_{123} && (5.13)\\
{}^{w}\theta_p &= \theta_{123} - 90^\circ && (5.14)
\end{aligned}
$$

where $\theta_{12} = \theta_1 + \theta_2$ and $\theta_{123} = \theta_{12} + \theta_3$.

The simple time derivative of the kinematics yields

$$
\begin{aligned}
{}^{w}\dot{x}_p &= -L_1 \sin\theta_1 \left(\dot{\theta}_1\right) - L_2 \sin\theta_{12} \left(\dot{\theta}_1 + \dot{\theta}_2\right) \\
&\quad - \left[L_{3x} \sin\theta_{123} - (L_{3y} + L_{\mathrm{JR3}} + L_p) \cos\theta_{123}\right] \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right) && (5.15)\\
{}^{w}\dot{y}_p &= L_1 \cos\theta_1 \left(\dot{\theta}_1\right) + L_2 \cos\theta_{12} \left(\dot{\theta}_1 + \dot{\theta}_2\right) \\
&\quad + \left[L_{3x} \cos\theta_{123} + (L_{3y} + L_{\mathrm{JR3}} + L_p) \sin\theta_{123}\right] \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right) && (5.16)\\
\dot{\theta}_p &= \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 && (5.17)
\end{aligned}
$$

Rearranging this into the matrix form of Eq (5.6) produces the following *world-frame* Jacobian matrix

$$
{}^{w}\mathcal{J}(\vec{q}) = \left[\begin{array}{ccc}
-L_1\sin\theta_1 - L_2\sin\theta_{12} - [L_{3x}\sin\theta_{123} - (L_{3y} + L_{\mathrm{JR3}} + L_p)\cos\theta_{123}] \\
L_1\cos\theta_1 + L_2\cos\theta_{12} + [L_{3x}\cos\theta_{123} + (L_{3y} + L_{\mathrm{JR3}} + L_p)\sin\theta_{123}] \\
1
\end{array}\right.
$$

$$
\left.\begin{array}{ccc}
-L_2\sin\theta_{12} - [L_{3x}\sin\theta_{123} - (L_{3y} + L_{\mathrm{JR3}} + L_p)\cos\theta_{123}] \\
L_2\cos\theta_{12} + [L_{3x}\cos\theta_{123} + (L_{3y} + L_{\mathrm{JR3}} + L_p)\sin\theta_{123}] \\
1
\end{array}\right.
$$

$$
\left.\begin{array}{ccc}
[L_{3x}\sin\theta_{123} - (L_{3y} + L_{\mathrm{JR3}} + L_p)\cos\theta_{123}] \\
[L_{3x}\cos\theta_{123} + (L_{3y} + L_{\mathrm{JR3}} + L_p)\sin\theta_{123}] \\
1
\end{array}\right] \tag{5.18}
$$

To transform Eq (5.18) from the world-frame to the tool-frame, we use Eq (5.10) where ${}^{T}R_{w}$ for the PLIMMS coordinates is

$$
{}^{T}R_{w} = \left[\begin{array}{ccc}
\cos\theta_{123} & \sin\theta_{123} & 0 \\
-\sin\theta_{123} & \cos\theta_{123} & 0 \\
0 & 0 & 1
\end{array}\right] \tag{5.19}
$$

For brevity, the final expression for the tool-frame Jacobian matrix is not given here explicitly, though it is simply the product of ${}^{T}R_{w}$ in Eq (5.19) premultiplying ${}^{w}\mathcal{J}(\vec{q})$ given in Eq (5.18).

*5.1.5 Differentiation.* When collecting the DIDO data, both the PLIMMS and the PUMA robot measured the joint positions rather than joint velocities. Consequently, the positions were differentiated to produce the measured velocities. To perform the differentiation, the 3-point backwards difference formula [10:p.140] was used:

$$
\frac{df}{dx} \approx \frac{1}{2T}[3f(x) - 4f(x-T) + f(x-2T)] \tag{5.20}
$$

where $T$ is the time, in seconds, between samples.

*5.2 Training Data Generation Procedures.*

Once the raw data were collected and configured as a suitable I/O pair, one or more steps of data processing transformed it into suitable training vectors for presentation to the ANN. There were seven possible data processing options that *could* be applied to the data. These seven options are explained in the following sections followed by a discussion of when and in what combinations they might be applied.

*5.2.1 Magnitude Normalization.* All of the training data are expressed in the tool-frame cartesian coordinate system because that is the task coordinate system in which the accommodation or ANN controller operates. For the planar problem, the input force vector of the controller, $\vec{F}$, has three components ($f_x$, $f_y$, $m_z$) and the output velocity vector, $\vec{V}$, has three components ($v_x$, $v_y$, $\omega_z$). These vectors contain two DoF of direction information and one DoF of magnitude information. By normalizing the magnitude of the vectors, the magnitude DoF is lost. The resulting unit magnitude vector simply identifies the direction of the original vector. The normalized force vector, $\tilde{F}$, is given by

$$\tilde{F} = \left( \frac{f_x}{M_f}, \frac{f_y}{M_f}, \frac{M_z}{M_f r} \right) \tag{5.21}$$

where $M_f$ is given by

$$M_f = \sqrt{(f_x)^2 + (f_y)^2 + \left(\frac{m_z}{r}\right)^2}$$

and $r$ is a characteristic radius taken as the radius (or half width) of the peg. The normalized velocity vector, $\tilde{V}$, is given by

$$\tilde{V} = \left( \frac{v_x}{M_v}, \frac{v_y}{M_v}, \frac{\omega_z r}{M_v} \right) \tag{5.22}$$

where $M_v$ is given by

$$M_v = \sqrt{(v_x)^2 + (v_y)^2 + (\omega_z r)^2}$$

It is important to note that after normalization, the number of independent parameters contained in the normalized vectors is reduced by one.

When magnitude normalization is applied to the I/O features of a training data set, the result is a set of unit-length vectors which identify the vector directions of the commanded

velocities in response to the measured force directions. Since the magnitude information is lost, the number of independent DoF in the mapping is reduced by one. This has both good and bad implications on the success of the ANN controller trained from these data. Having one less DoF to learn can theoretically mitigate the burden of learning for the ANN. However, since the stability of the robot controller is inherently sensitive to the gains of the system, the I/O gain of the mapping from force to velocity is important. One must reconstruct a proper gain to apply to the controller output after it has mapped the direction for the commanded velocity. The question then becomes, "what gain should be applied to the controller output?"

In an attempt to answer this question, we first look at the gains contained in the original training data before they were magnitude normalized. One quickly finds that, even for SISO data generated using a constant $\mathcal{A}_a$, the ratio of the velocity magnitude over the force magnitude, i.e. the I/O gain, varies considerably. This can be explained by considering the results of taking any two vectors, multiplying them by the same matrix, and then computing the magnitude of the resulting vector divided by the input vector. Even if the matrix is diagonal with equal elements along the diagonal, the only way to get the same I/O magnitude ratio is to have the two input vectors linearly dependent (i.e. $\vec{f}_1 = k\vec{f}_2$.) The point is, we have no single I/O gain that we can capture from the original training data and use to reconstruct the sensitivity of the mapping when implemented as a controller. Unsuccessful attempts were made to use the average I/O magnitude ratio of the training data in the implemented controller. Although this is a serious problem when one considers that it destroys the original integrity of SISO data, it is possible that it could be used to treat DIDO data which may originally contain an undesirable I/O gain. This possibility was not investigated.

*5.2.2 Low-Pass Filtering.* The low-pass filtering of the raw force and velocity data was done using custom-written software which made calls to C-language routines from "Numerical Recipes in C" [43]. The algorithmic kernel of `lpfilter.c` was the "Recipe's" subroutine called `smooft()` which first removed any linear trend, applied a Fast Fourier Transform (FFT) to low-pass filter the data, and then reinserted the linear trend. In `smooft()` the amount of smoothing is specified as the number of points in the series over which the

smoothing should occur. The number is not constrained to be an integer, and specifying zero results in no filtering at all. Since `smooft()` does not accommodate directly specifying a cutoff frequency or roll-off rate, the resulting filtered data were examined to determine the effective cutoff frequency. To approximate the cutoff frequency, the power spectral density (PSD) of the filtered data was examined using DADiSP®, developed by DSP Development Corporation [18, 19].

The difficulty with determining the corresponding cutoff frequency using `smooft()` eventually led to the use of a separate computer program that would do the low-pass filtering in the time domain. The program was written in C-language by the author and a colleague. Using that new program, a clear relationship between signal bandwidth and the program arguments was possible. The argument to the program reflected the fraction of bandwidth of the sample frequency to pass so it was much simpler to use.

*5.2.3 Velocity Pruning.* When a person is demonstrating an accommodation task, it is possible that they will generate bad examples of behavior as well as good. A particularly troublesome instance of a bad behavior is when the demonstrated velocity is zero while the contact force is not zero and the alignment task is incomplete. In such a case, the contact force vector can vary widely in both magnitude and direction within the bounds of friction[1]. Consequently, raw data which reflect mapping from a non-zero force vector to a zero velocity vector is deemed counter-productive. The process of excluding these counter-productive vectors is termed *velocity pruning*.

To perform velocity pruning, the raw velocity vectors are each examined to determine if the magnitude of the velocity vector exceeds a given threshold. If a velocity is encountered which does not exceed the threshold, both the velocity and force samples are discarded for that sample point. If the threshold, $V_t$, is exceeded, then the force/velocity sample pair is

[1] It is important to note that the contact force vector may vary widely for **any** velocity state when the human is in charge. This is indicative that the mapping may not be one-to-one. The goal of the post-collection data processing is to make the data reflect a *subjective* mapping where every velocity vector is the image of at least one force vector.

passed without modification. The algorithm to perform velocity pruning is described by:

$$\text{Keep:} \qquad \text{if } \left\|\vec{V}\right\| > V_t \tag{5.23}$$

$$\text{Discard:} \qquad \text{if } \left\|\vec{V}\right\| \leq V_t \tag{5.24}$$

For an accommodation matrix controller, the mapping from non-zero forces to a zero velocity would only occur in two situations. The first case is when the peg is stuck in a friction cone and it cannot move. However, because we have chosen the tool-frame coordinate system origin to be a the center of the peg tip and contact must occur at the peg corners, there is almost always going to be a moment component to the force measurement which will have a corresponding non-zero desired angular velocity component. Therefore, we should rarely, if ever, see $\vec{V}$=0 due to the first case.

The second possibility for observing $\vec{V}$=0 is upon completion of the edge mating task when the peg has been aligned and motion has stopped. In the latter case, it is a desired behavior and we want to include it in the training data set or the ANN may never learn to stop motion upon alignment. We will demonstrate this behavior in the results to follow by experimenting with two different $V_t$ on some sets of training data. In the end, we will find that we must use caution with velocity pruning so as to only prune exemplar vectors *prior* to the peg being aligned.

*5.2.4 Hemisphere Pruning.* This method of pruning the data excludes all the measured force/velocity vector pairs which have an included angle, $\Psi$, greater than a specified threshold, $\Psi_t$. The idea behind hemisphere pruning is that if $\vec{F}$ and $\vec{V}$ point in roughly the same hemisphere of space, then it is likely that $\vec{V}$ will be moving the peg so as to comply with $\vec{F}$, and, therefore, reduce the contact force. If $\vec{V}$ points in the opposite direction to $\vec{F}$, then the contact force will increase. The included angle is computed using

$$\Psi = \arccos\left(\frac{\vec{F} \bullet \vec{V}}{\left\|\vec{F}\right\| \left\|\vec{V}\right\|}\right) \tag{5.25}$$

where • indicates the dot product operator and $||*||$ indicates the root-mean-squared (RMS) magnitude of the vector argument. Using Eq (5.25) to compute $\Psi$, the rule for hemisphere pruning can be written as:

$$\text{Keep:} \qquad \text{if } \Psi \leq \Psi_t \tag{5.26}$$

$$\text{Discard:} \qquad \text{if } \Psi > \Psi_t \tag{5.27}$$

If we choose to simply prune out the vector pairs that indicate $\vec{F}$ and $\vec{V}$ are separated by more than 90-degrees, then we can reduce Eq (5.25) and the criterion for pruning to simply

$$\text{Keep:} \qquad \text{if } \vec{F} \bullet \vec{V} > 0 \tag{5.28}$$

$$\text{Discard:} \qquad \text{if } \vec{F} \bullet \vec{V} \leq 0 \tag{5.29}$$

When the two vectors have a positive dot product the velocity vector points within the same hemisphere as the force vector which is a prerequisite for a controller to reduce contact force magnitudes.

*5.2.5 Lipschitz Clipping.* Asada [5] has suggested that Lipschitz's condition for continuity of a function can be used to ensure that the training data are consistent before they are presented to an ANN for training. Such a screening process is said to improve the efficiency of training and is also said to allow the training to reach a global minimum[2]. The Lipschitz condition ensures that two points which are close together in the input space will map to two points in the output space which are also close together. Mathematically, the Lipschitz ratio, $\mathcal{L}_{ij}$, is expressed as:

$$\mathcal{L}_{ij} = \frac{||\vec{v}_i - \vec{v}_j||}{||\vec{f}_i - \vec{f}_j||} \quad \forall\, i,j \quad \in (0 < i < N), (0 < j < N), i \neq j \tag{5.30}$$

[2]This author does not accept Asada's statement that a neural network converges to a global minimum when the teaching data satisfying the Lipschitz condition are used as training samples.

where $N$ is the total number of force/velocity vectors. Note that the Lipschitz ratio, $\mathcal{L}_{ij}$, must be computed for *all* combinations of $i$ and $j$ in the data set except for the cases when $i = j$. This means Eq (5.30) must be computed $N^2 - N$ times. For the present work, the root-mean-squared norm was used to compute the distances in Eq (5.30).

The Lipschitz clipping criterion is given by:

$$\begin{aligned} \text{Keep:} \quad & \text{if } \mathcal{L}_{ij} \leq \mathcal{L}' \\ \text{Discard:} \quad & \text{if } \mathcal{L}_{ij} > \mathcal{L}' \end{aligned} \tag{5.31}$$

When a pair of vectors is encountered that fails to meet the criterion given in Eq (5.31), **both** of the potential training vectors are discarded since the blame for failure cannot be associated with either of them individually. It is the relationship between the two vectors that is the target of this examination.

There are two significant shortcomings of using Lipschitz clipping to process the training data. First, there are no sound criteria for choosing the value for $\mathcal{L}'$ as a function of any system parameters. In practice, it might be chosen rather arbitrarily based on a qualitative examination of a histogram plot of $\mathcal{L}_{ij}$ population for all potential training vectors. This examination would consider the percent of the vectors that a given $\mathcal{L}'$ would discard and the $\mathcal{L}'$ might be chosen so as not to discard more than about 20 percent of the potential training vectors. Because we have no method for relating $\mathcal{L}_{ij}$ to the resulting ANN controller's performance, we would be forced to rely on an iterative approach to selecting a "proper" value for $\mathcal{L}'$.

The second problem stems from the expectation that SISO data should pass unchanged through all the processing steps, since it is perfectly consistent and accurate data. With the Lipschitz ratio clipping, however, we find that, even though there may be a constant linear analytic mapping from the input space to the output space, the Lipschitz ratio will not necessarily be a constant. The implication is that if the ratio is not a constant for a perfect mapping, then selecting a cutoff ratio, $\mathcal{L}'$, which excludes any of the vectors, may exclude 'good' vectors as well as 'bad' vectors. The only time that the Lipschitz ratio will be a

constant is if the mapping matrix is diagonal and all the elements are identical. Otherwise, the mapping matrix warps the input space as it maps it onto the output space.

To understand how the Lipschitz ratio can vary for a constant mapping matrix, consider a two-dimensional example. The Lipschitz ratio takes the ratio of the distance between two points in the output space over the distance between their corresponding two points in the input space. If we have two input points in the two-dimensional input space which are separated by a distance of one in the x-axis direction and one in the y-axis direction (i.e. their difference is $(1,1)$), we can picture a right triangle whose hypotenuse connects the two points in the input space. If we have the following mapping matrix:

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \tag{5.32}$$

then the two corresponding output points will be likewise connected by the hypotenuse of a right triangle whose sides measure $(1,2)$. The Lipschitz ratio between the two input/output pairs is $\sqrt{5}/\sqrt{2} = 1.58$. If, on the other hand, we had two input points separated by coordinates $(1,2)$, the input triangle hypotenuse would be $\sqrt{5}$ in length while that of the output triangle would be $\sqrt{17}$, so the Lipschitz ratio would be $1.84$. Thus, we have shown that for a constant linear mapping matrix, the Lipschitz ratio is a function of the distance between the input vectors, and is NOT a constant when the mapping matrix has unequal diagonal elements. If the mapping matrix has equal diagonal elements, such as:

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \tag{5.33}$$

then the Lipschitz ratio is preserved as a constant all across the input space for all distances between input points. In the example given above, the ratio for the $(1,1)$ input vector would be 2, as it would also be for the $(1,2)$ input vector.

It is important to note that, although using the Lipschitz ratio as a criterion to exclude training vectors is not compatible with our objective of excluding only 'bad' vectors, the vectors which are not excluded will be passed unchanged by the examination. This means

that Lipschitz clipping will not distort the I/O mapping of any vectors which meet the criterion expressed in Eq (5.31). Despite the fact that it will not distort the mapping, however, we discard Lipschitz clipping as a useful technique for data processing and will not investigate it further in the present work.

*5.2.6 Collision Pruning.* In free-space motion, the force measurement, $\vec{F}_r$, is identically zero, which generates a zero output from either the ACC or the ANN controller. This essentially turns off the controller until the peg contacts the table surface. Consequently, we defined a nominal behavior for the controller that would command the peg motion when there were no sensed forces. That behavior consisted of a constant nominal velocity, ${}^T\vec{V}_n$, which was chosen as a pure translational velocity in the direction the axis of the peg pointed (y-axis in the planar tool-frame coordinates.) This nominal commanded velocity was assumed to be capable of bringing the peg into contact with the table. Upon contact, the $\vec{F}_r$ becomes non-zero in magnitude and the nominal velocity is turned off according to Eq (5.47) that follows. Since neither the accommodation matrix controller nor the ANN controller had any authority in free-space motion, provisions were made to exclude that data from the training data sets. This process of exclusion was called *collision pruning*. The intent of collision pruning was to remove all of the $\vec{F}$ and $\vec{V}$ pairs which represented data taken prior to the initial contact of the peg with the table surface. To accomplish this, the original training data force vector, $\vec{F}$, was scanned until a consecutive series of $N$ points was found which all exceeded the desired threshold RMS force magnitude, $F_t$. The value of $N$ was specified by the user. Upon finding $N$ points that exceeded $F_t$, the index of the first point in the $N$-length window was identified as $k$. Having thus identified the index of the first data point believed to be taken after contact between the peg and the table, the algorithm to perform collision pruning is described by:

$$\text{Keep:} \qquad \vec{F}_i, \vec{V}_i \quad \forall\, i \geq k \tag{5.34}$$

$$\text{Discard:} \qquad \vec{F}_i, \vec{V}_i \quad \forall\, i < k \tag{5.35}$$

where $i$ is the counting index of data vectors.

*5.2.7 Subsampling.* In some cases, it was desirable to see if fewer data points could be used to train the ANN. In these cases, one approach was to simply extract every $n$th vector from the original training data file. The algorithm to extract the desired subset of vectors simply used the modulo operator to determine whether to keep a vector with a particular index. Any leftover vectors after the last vector matching the modulo criterion were also discarded.

*5.2.8 Allowable Data Processing Combinations.* Although one may apply more than one processing option to a particular raw training data file in succession, not all combinations of the possible processing options make sense. Table 5.1 shows which processing options can be combined together in sequence and which cannot. Note that the table is interpreted by assuming that the option listed across the top of the table is executed before the option listed along the left side of the table. For instance, one cannot velocity prune vectors **after** they have been magnitude normalized because all of the vectors will already have the same magnitude (unity) and would be pruned or kept as a whole set. On the other hand, velocity pruning can be applied *before* magnitude normalization. The primary reason for the disallowed combinations is that the first operation destroys the integrity of the time sequence for the data set and the second operation requires that sequence to be intact. This is the case for the velocity pruning, hemisphere pruning, and Lipschitz clipping which all destroy the time base, thereby preventing one from subsequently low-pass filtering or collision pruning[3] the data.

## *5.3 ANN Training.*

To completely describe the ANN training requires mention of the structure of the ANN and the training algorithm. Since development of the ANN was not the central theme of this research, a specific structure and a specific training algorithm were selected early and held constant throughout. Both the structure and the training algorithm are simple.

[3]The reason we cannot collision prune data after the time base is corrupted is that our collision pruning criterion looks for a consecutive sequence of points that exceed a given value. If the sequence is corrupted by pruning out data points, then it is no longer correct to choose the collision point with this criterion. However, if we use a window size of one, the time base is no longer a prerequisite for the collision pruning to be successful.

Table 5.1 Table depicting allowable combinations (•) and disallowed combinations (o) of processing options for raw data.

| 1st Operation → / 2nd Operation ↓ | Mag. Normalize | Low-pass Filter | Velocity Prune | Hemisph. Prune | Lipschitz Clip | Collision Prune | Sub-Sample |
|---|---|---|---|---|---|---|---|
| Mag. Normalize | N/A | ● | ● | ● | ● | ● | ● |
| Low-pass Filter | ● | N/A | ○ | ○ | ○ | ● | ○ |
| Velocity Prune | ○ | ● | N/A | ● | ● | ● | ● |
| Hemisph. Prune | ● | ● | ● | N/A | ● | ● | ● |
| Lipschitz Clip | ● | ● | ● | ● | N/A | ● | ● |
| Collision Prune | ○ | ● | ○ | ○ | ○ | N/A | ● |
| Sub-sample | ● | ● | ● | ● | ● | ● | N/A |

*5.3.1 ANN Structure.* The architecture consisted of a fully-connected, two-layer MLP using the sigmoid nonlinear squashing function (shown in Figure A.1 of Appendix A) on the output of the hidden layer nodes. Because the ANN was being used to learn a functional mapping, output layer nodes did not have a nonlinear squashing function applied to their values. Some of the early work maintained five nodes in the hidden layer. However, virtually all of the later simulation work used 10 nodes in the hidden layer.

Every node in the MLP had a bias input as described in Appendix A. The bias nodes allow the ANN to adjust the location of the central region of the sigmoid nonlinearity to suite the magnitude of the data set. This is important for data sets that may have large magnitudes, because it prevents the data from being summarily "squashed" by the asymptotic tails of the sigmoid function. The bias also means that the sigmoid function does not have to be explicitly offset to provide both positive and negative outputs.

*5.3.2 ANN Training Algorithm.* The algorithm used for training was a form of back-propagation for a MLP. Recently many approaches to MLP back-propagation training have been developed to improve convergence. [28] provides a good summary of some of those approaches. The specific algorithm used in this research to train the ANN is described in

detail in Appendix A. Although the algorithm is simple, it has many parameters that can be varied to affect the learning performance. For this research effort, only a few of the possible parameters were varied because the ANN was not the central theme of this research.

The training rate, $\eta$, and the training momentum, $\alpha$, were held fixed for large groups of runs that were to be compared. Over the course of the research, a few different values were used, but comparisons between the results from using different values were not made.

Some ANN training programs reserve part of the training exemplars for testing after training is finished. The programs used for the present work did not reserve any training exemplars from the training set; all of the data were used in training.

Based on the information in [29], the initial random weights were all set to values between -0.1 and 0.1 to enhance the convergence of the back-propagation algorithm. This technique is usual practice to start the training in a relatively "safe" place on a multi-dimensional error surface that is less likely to contain a local minima.

For each set of training data, at least four attempts to train were made. The procedures were identical except for the random seed used to initialize the random weights. The same random seed was never used twice in any of the training. This is particularly significant because the UNIX® implementation of the "random number" generating subroutine will generate the exact same sequence of pseudo-random numbers every time it is given the same random seed.

As previously mentioned, the large number of training data configurations precluded extensive manual tuning of the training parameters to achieve success in all cases. The paradigm developed in this research is that the ANN is given an equal opportunity to learn from each set of training data, and its ability to learn each set is used to determine the suitability of that data set. In an effort to provide an even footing to all the training data set configurations, the total number of training vectors presented to the ANN was held constant. This consistent exposure of the ANN to training was used as the basis for comparisons concerning the ease of learning. Under these criteria, training was terminated based on the number of exposures rather than the minimum of the total squared-error curve or any of the

other popular termination criteria that are mentioned in [28]. The total number of training vectors presented was maintained at 3,000,000 for all the simulation results presented.

### *5.4 ANN Training Evaluation.*

Once the ANN was adequately trained using the criteria of the total RMS error for the entire data set, it was tested using one or more of three techniques. One technique simply confirmed that the computed output errors of the ANN were small, while the other two techniques interrogated the ANN to determine what kind of ACC matrix it emulated. Each of these is discussed below.

*5.4.1 ANN Error Tracking.* The first technique was to simply apply the training input feature vectors, $\vec{F}^*$, one at a time and plot the computed ANN output, $\vec{V}$, against the training output, $\vec{V}^*$. This test simply determined how well the training data set had been learned. Viewing it as a tracking error for the training data trajectory was a more relevant means of evaluating whether training was complete enough. It is important to note that the test data were identical to the training data for this test. Therefore, the data did not evaluate the ability of the ANN to generalize or interpolate, they merely indicated how well the ANN had learned the training data set and served to validate the training software.

*5.4.2 ANN Interrogation by Unit-Vector Probing.* The second technique for testing was called *Unit-Vector Probing (UVP)* the ANN to determine the $\mathcal{A}_a$ it had learned to emulate. Using the UVP method, the simple linear relationship of Eq (3.1) was used as a model. If one assumes that the ANN mapping is a direct replacement for the $\mathcal{A}_a$ of Eq (3.1), then one can extract the columns of the pseudo-accommodation matrix, ${\mathcal{A}_a}^*$, by sequentially applying unit vectors along each of the coordinate axes to the ANN. The computed outputs are then taken as the columns of ${\mathcal{A}_a}^*$. For example, to extract the first column of ${\mathcal{A}_a}^*$, the vector $\vec{F} = (1,0,0)^T$ is presented to the ANN. The second column is produced by applying $\vec{F} = (0,1,0)^T$, and the third column is extracted using $\vec{F} = (0,0,1)^T$. Due to the nonlinearity in the hidden layer of the ANN, for a given set of trained weights it is possible to derive significantly different ${\mathcal{A}_a}^*$ for relatively small variations of the input vectors. This means

that the sensitivity of the $\mathcal{A}_a{}^*$ to variations from the unit vectors may be large for some mappings.

*5.4.3 ANN Interrogation by LSMF.* The final testing technique was called the *LSMF* method of extracting the $\mathcal{A}_a$ learned by the ANN. This technique takes advantage of using the right pseudo inverse to determine the best fit matrix, $\mathcal{A}_a{}'$. In this case, $\mathcal{A}_a{}'$ is a best fit to the data set in a total least-squares sense. For the relationship of Eq (3.1), one can gather $N$ samples of the input vector, $\vec{F}$, and the output vector, $\vec{V}$. If all $N$ samples of each are concatenated into new matrix variables, $\mathcal{F}$ having size $(k \times N)$ and $\mathcal{V}$ having size $(n \times N)$, the relationship is:

$$\mathcal{V} = \mathcal{A}_a{}' \, \mathcal{F} \tag{5.36}$$

Since the $\mathcal{F}$ and $\mathcal{V}$ are given, we want to find $\mathcal{A}_a{}'$ which represents the best fit of the data set. In the least-squares error sense, $\mathcal{A}_a{}'$ is given by

$$\mathcal{A}_a{}' = \mathcal{V} \, \mathcal{F}^{\#} \tag{5.37}$$

where $\mathcal{F}^{\#}$ is the right pseudo inverse of $\mathcal{F}$ and is given by [38] as

$$\mathcal{F}^{\#} = \mathcal{F}^T \left( \mathcal{F} \, \mathcal{F}^T \right)^{-1} \tag{5.38}$$

For the remainder of the present work, references to $\mathcal{A}_a{}'$ or the LSMF technique are directed towards the method of using the relationship of Eq (5.37).

In practice, the $\mathcal{F}$ and $\mathcal{V}$ can be produced from either the original training data set, or from presenting a series of $\vec{F}$ to the trained ANN and computing the series of $\vec{V}$. Using training data does not provide any insight into the mapping that an ANN has learned; however, it is useful for validating the method and for checking the consistency and content of the training data. When using the LSMF to evaluate a trained set of ANN weights, the I/O mapping was created using the original training data inputs for $\mathcal{F}$ and computing the ANN outputs as the $\mathcal{V}$.

*5.4.4 Matrix Similarity Indexes.* Once the ANN had been interrogated using the UVP technique described in Section 5.4.2 or the LSMF method as described in Section 5.4.3, the resulting matrices were compared with $\mathcal{A}_a$. Because both the form and the magnitude of $\mathcal{A}_a$ were important to its function, the comparison for similarity was difficult to do manually. Therefore, indexes of similarity were developed which captured the important features of similarity desired. Given a desired $\mathcal{A}_a$ depicted as

$$\mathcal{A}_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} \tag{5.39}$$

and a matrix to be compared for similarity as

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \tag{5.40}$$

we define the structural similarity index, $\Upsilon_s$, as:

$$\Upsilon_s \equiv \left(k_{11}^2 + k_{12}^2 + k_{13}^2 + k_{21}^2 + k_{23}^2 + k_{31}^2 + k_{32}^2\right) \tag{5.41}$$

$\Upsilon_s$ is a measure of how closely the off-diagonal terms and the $k_{11}$ term are to their desired values of zero. $\Upsilon_s$ increases as the errors get larger. A similar, but separate, measure of the error in the (2,2) and (3,3) elements of $K$ is the gain similarity index, $\Upsilon_g$, which is defined as:

$$\Upsilon_g \equiv \left(\left(k_{22} - a_{22}\right)^2 + \left(k_{33} - a_{33}\right)^2\right) \tag{5.42}$$

$\Upsilon_g$makes a direct magnitude comparison between the only two non-zero elements of $\mathcal{A}_a$ and their corresponding elements in $K$. As an indicator of correct sign in the (2,2) and (3,3) elements, a sign similarity index, $\Upsilon_\pm$, is defined as:

$$\Upsilon_\pm \equiv \left(\left|\text{sign}\left(a_{22}\right) - \text{sign}\left(k_{22}\right)\right| + 2\ \left|\text{sign}\left(a_{33}\right) - \text{sign}\left(k_{33}\right)\right|\right) \tag{5.43}$$

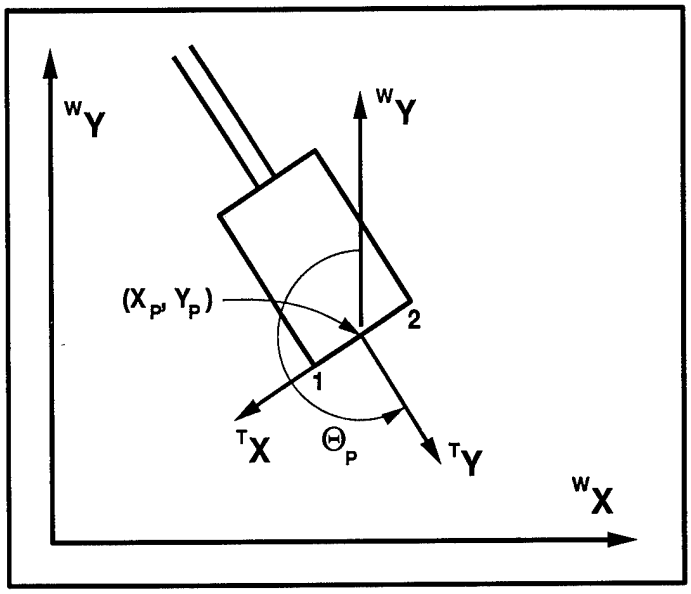



Figure 5.6 Illustration of the coordinate system used to describe the position of the peg.

where $|*|$ indicates the absolute value operator and sign(*) is the sign operator which returns +1 if its argument is positive or zero and -1 if its argument is negative. $\Upsilon_{\pm}$ checks the signs of the individual elements against what is desired for them. If both of the signs are the same, $\Upsilon_{\pm}$=0. If the $k_{22}$-element has the wrong sign, 2 is added to $\Upsilon_{\pm}$, while 4 is added if the $k_{33}$-element has the wrong sign. Thus there are discrete increments of $\Upsilon_{\pm}$ which reveal the pattern of sign matching.

The final similarity index is called the ratio similarity index, $\Upsilon_r$. It is defined as:

$$\Upsilon_r \equiv \left( \left| \frac{k_{33}}{k_{22}} \right| - \left| \frac{a_{33}}{a_{22}} \right| \right)^2 \tag{5.44}$$

$\Upsilon_r$ provides a measure of the error in the magnitude of the ratio between the two non-zero diagonal elements of $K$ as compared to those of $\mathcal{A}_a$. The ratio is crucial to achieving the proper system sensitivity to moments as compared to forces.

### *5.5 Controller Implementations.*

The controllers were implemented in both simulation and hardware. In both cases, a common set of cartesian coordinates was used to describe the position of the peg. Figure 5.6

illustrates the coordinate system used to describe the position of the peg. The specifics of each implementation are now presented in detail. They are followed by a brief discussion about the integration algorithm which was used in both implementations.

*5.5.1 Implementation via Simulation.* The simulation software enabled rapid and safe evaluation of the many different ANN controllers that were trained in the course of this research. It was intended to use the same ANN controller software subroutines for the simulation as were used on the software controlling the robot. Because of the revised focus of this research effort, this capability was never fully realized. However, it is entirely possible to do this in a follow-on effort.

Figure 5.7 shows a complete kinematic block diagram modeling the controller, the robot, and the interaction between the robot and the environment. The interaction between the robot and the environment is embodied in the propagation of the cartesian position error through the manipulator Jacobian to get joint position errors that, in turn, are input to a PID feedback controller that attempts to eliminate them. The torques produced by the PID controller can then be conceptually transformed into cartesian forces via the same manipulator Jacobian matrix used to transform the position errors. An assumption that must be true for this model to be valid is that the manipulator motion between servo samples must be very small so as to allow the approximation of a fixed Jacobian during the servo loop period. In addition, using the Jacobian to transform errors assumes that those errors are differentially small or can be approximated as such. For most robotic control systems, both of these assumptions are valid when manipulator speeds are in the low range typically used during parts mating.

It is worth noting that the model depicted in Figure 5.7 does not include any dynamic properties such as inertia, damping, joint friction, etc. The effects were ignored for this research because they typically have little influence when the manipulator is moving slowly. The real PUMA manipulator obviously has all of these dynamic properties, but they are unimportant during the slow motion of parts mating.

In an effort to distill the problem down to its essential components, the simulation model was simplified to the form shown in Figure 5.8. In the simplified form, the robot is

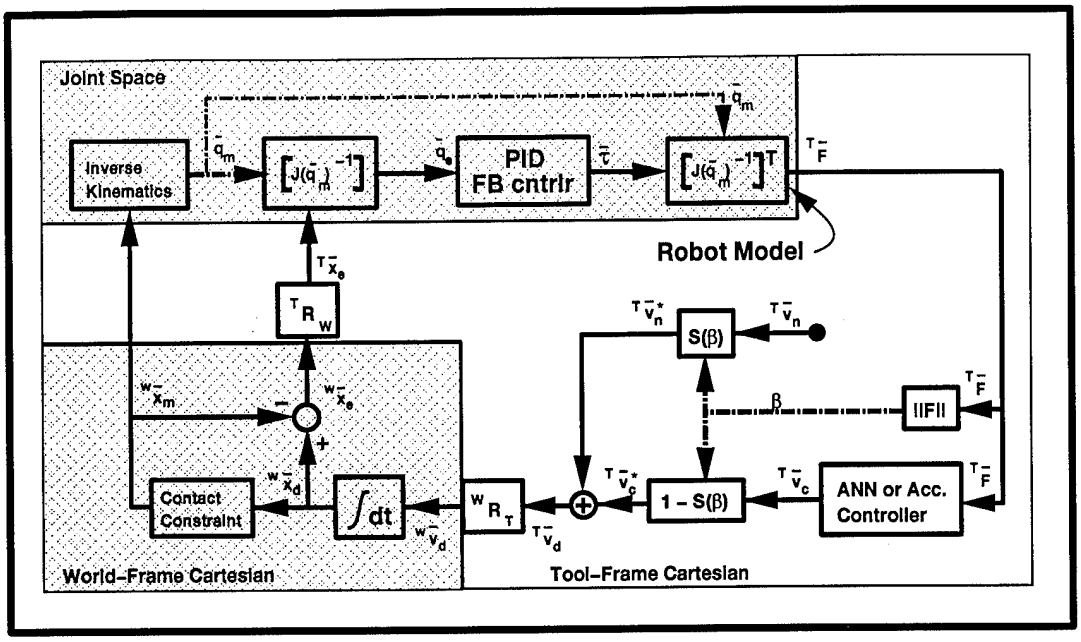



Figure 5.7 Controller block diagram for complete simulation model.

modeled as a perfect plant which has no dynamics and introduces no errors into the system. Therefore, it can be represented by a unity transfer function and does not appear in Figure 5.8. Although one can argue that no robot is perfect, a typical state-of-the-art, high-performance, direct-drive manipulator is close enough to perfect to validate this assumption at slow velocities. The interaction forces between the robot and the environment are generated by the PID controller's stiffness in Figure 5.7 while for the simplified model of Figure 5.8, an environmental stiffness matrix, ${}^{W}K_e$, determines the contact forces. There is a subtle, but possibly significant, difference between these approaches. In the complete model, the cartesian stiffness realized at the tip of the peg is a configuration-dependent relationship. This is because the controller stiffness is a joint-space entity and it is transformed into cartesian space by the configuration-dependent Jacobian matrix. In the simplified model, the ${}^{W}K_e$ is invariant for all configurations.

The world-frame superblock on the left of Figure 5.8 reflects all the simulation components that model the interaction of the peg with the environment. All of the vectors in that superblock are expressed in world-frame cartesian coordinates as indicated by the 'W'-superscript preceding each variable. The ${}^{T}R_{w}$ and the ${}^{W}R_{T}$ are matrices that transform

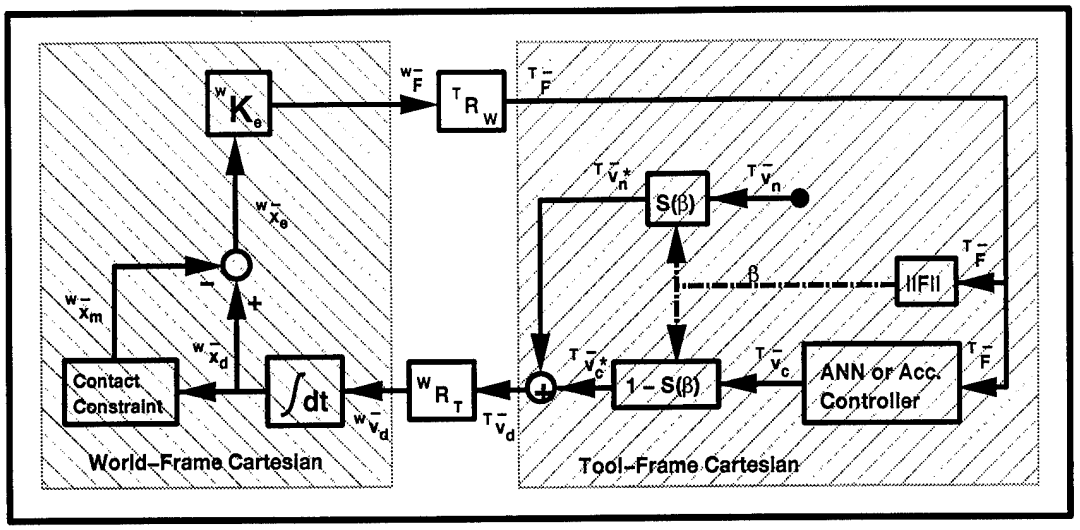



Figure 5.8 Controller block diagram for simplified simulation model.

vectors from world-frame to tool-frame and from tool-frame to world-frame, respectively. The world-frame desired velocity vector, ${}^{T}\vec{V}_d$, is integrated to yield the desired position, ${}^{w}\vec{X}_d$. The contact constraint block in Figure 5.8 models the position-based constraint between the peg and the table surface. It examines ${}^{w}\vec{X}_d$ and computes the constrained position of the peg tip as a "measured" position vector, ${}^{w}\vec{X}_m$. The first step of that computation is to determine which corner of the peg is closer to the table. Referring to Figure 5.6, the coordinates for corner 1, ${}^{w}\vec{X}_1$, and for corner 2, ${}^{w}\vec{X}_2$, are given by:

$$ {}^{w}\vec{X}_1 = \left({}^{w}x_p + r_p \cos\theta_p\right)\hat{i} + \left({}^{w}y_p + r_p \sin\theta_p\right)\hat{j} \tag{5.45} $$

$$ {}^{w}\vec{X}_2 = \left({}^{w}x_p - r_p \cos\theta_p\right)\hat{i} + \left({}^{w}y_p - r_p \sin\theta_p\right)\hat{j} \tag{5.46} $$

where $\hat{i}$ and $\hat{j}$ are unit vectors along the $X$ and $Y$ axes in the world-frame coordinates and $r_p$ is the radius[4] of the peg.

The algorithm used to compute ${}^{w}\vec{X}_m$ depends on the friction model or coefficient of friction, $\mu$, used. For the frictionless model (i.e. $\mu$=0), if ${}^{w}\vec{X}_d = ({}^{w}x_d, {}^{w}y_d, \theta_d)$, ${}^{w}\vec{X}_m = ({}^{w}x_m, {}^{w}y_m, \theta_m)$ and the height of the table is $y_t$, the contact constraint model can be described by the following pseudo-code which is illustrated in Figure 5.9:

[4]For the rectangular peg used, the "radius" is equal to one-half the peg width.

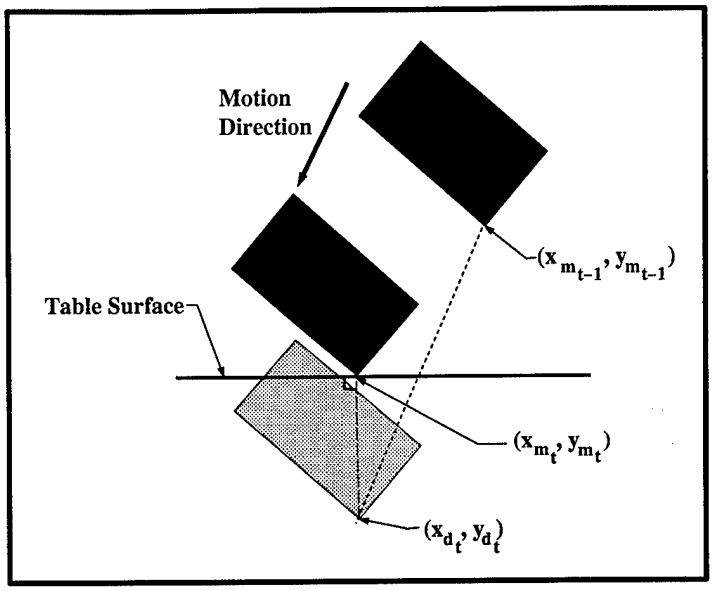



Figure 5.9 Illustration of frictionless constraint model used in the simulation.

```
if (ʷy_d < ʷy_t)
then
    ʷy_m = y_t
else
    ʷy_m = ʷy_d
endif
ʷx_m = ʷx_d
θ_m = θ_d
```

For the case of $\mu >> 1$ (i.e. no sliding), Figure 5.10 illustrates the no-sliding algorithm which is described by the following pseudo-code:

```
if (ʷy_d < ʷy_t)
then
    ʷy_m = y_t
else
    ʷy_m = ʷy_d
endif
ʷx_m = (ʷy_m − ʷy_{m(t−1)}) * (ʷx_{d_t} − ʷx_{m(t−1)})/(ʷy_{d_t} − ʷy_{m(t−1)}) + ʷx_{m(t−1)}
θ_m = θ_d
```

which is a simple linear interpolation of the x-coordinates to find ${}^{w}x_m$ where the path crosses ${}^{w}y_t$. Note that the evaluations expressed in the pseudo-code must be applied to the corner of the peg that is closest to the table surface. Therefore, the coordinates for both corners must

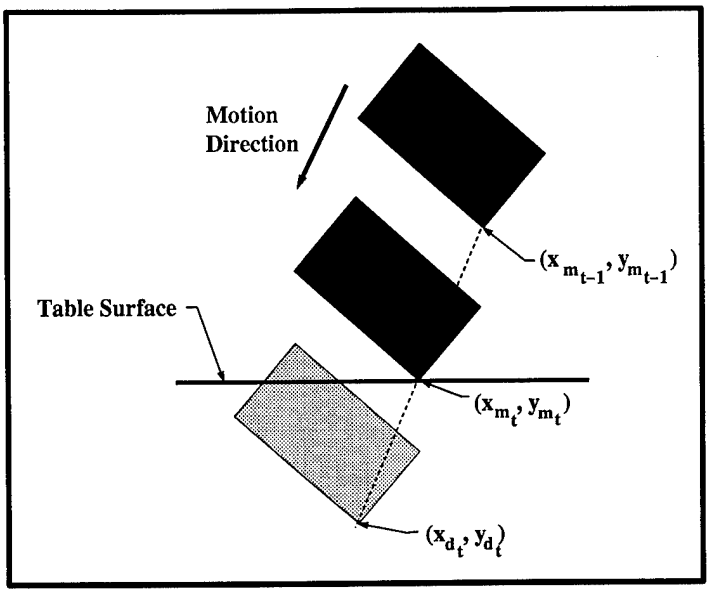



Figure 5.10 Illustration of no-sliding constraint model used in the simulation.

be computed to determine which one, if any, is to be constrained by the table surface. In this model, ${}^{w}\vec{X}_m$ is the interpolated value between the two ${}^{w}\vec{X}_d$ which bracket the collision with the table. Note that only the height of the peg is constrained by the frictionless model.

Referring back to Figure 5.8, the difference of ${}^{w}\vec{X}_d$ and ${}^{w}\vec{X}_m$ is computed as a virtual surface deflection, ${}^{w}\vec{X}_e$, which is multiplied by the environmental stiffness matrix, ${}^{w}K_e$ to produce a "measured" contact force, ${}^{w}\vec{F}_r$ expressed in the world-frame coordinates.

The superblock on the right of Figure 5.8 contains the description of the controller. The tool-frame vector of measure forces, ${}^{T}\vec{F}$ is fed into either the ANN or the ACC controller, whichever is present. For the controller to generate a non-zero commanded velocity, it must have a non-zero measured force vector. When the peg is in free-space motion, the force measurements differ from zero only by some magnitude of noise caused by the inertial forces of the peg end-effector and the electronics noise of the sensor itself. If the free-space motion is slow, so as to make it easier to deal with a rigid environment, then the force sensor readings will also be small. Thus, if the peg starts off in free-space, it may never move because the controller output may generate joint torque commands that are below the threshold of stiction in the joints. To ensure that a fixed free-space command is generated, a nominal

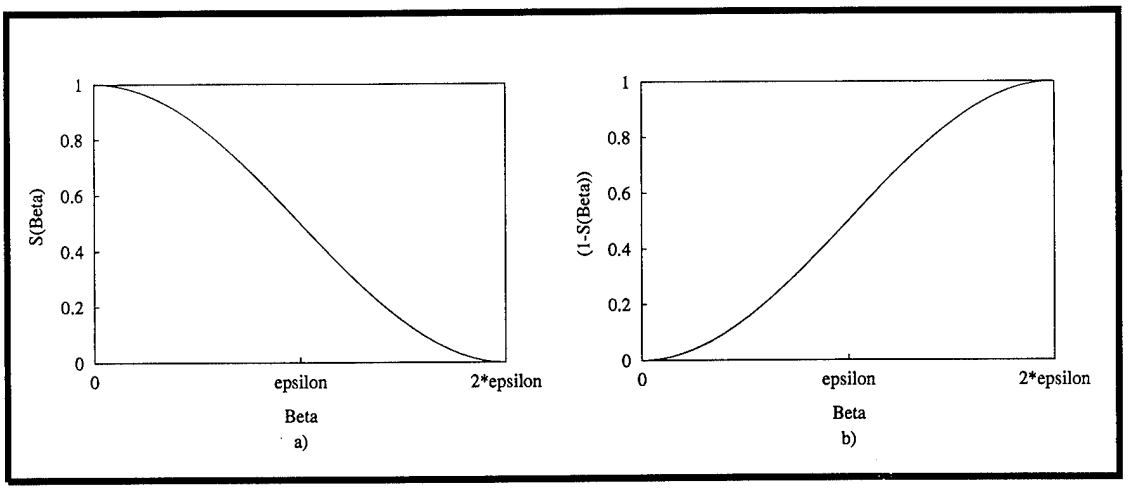



Figure 5.11 Plot of the blending function showing: a) $S(\beta)$ which turns off the ${}^{T}\vec{V}_n$ as $\beta$ increases, and b) $(1 - S(\beta))$ which turns on the ${}^{T}\vec{V}_c$.

velocity, ${}^{T}\vec{V}_n$, is superimposed on the controller command, ${}^{T}\vec{V}_c$. After the peg contacts the surface, however, it is undesirable to superimpose ${}^{T}\vec{V}_n$ because it may prevent the controller from recovering from jamming or wedging during execution of a peg-in-hole task. In addition, the DIDO data exemplifies a mapping from $\vec{F}$ to ${}^{T}\vec{V}_d$, so it already implicitly includes ${}^{T}\vec{V}_n$. Thus, continually superimposing ${}^{T}\vec{V}_n$ with ${}^{T}\vec{V}_c$ to get ${}^{T}\vec{V}_d$, would only serve to increase the contact force between the peg and the table.

A nonlinear blending function, $S(\beta)$, was used to "turn off" the nominal velocity as the peg contacted the surface. As $S(\beta)$ turns off ${}^{T}\vec{V}_n$, $(1 - S(\beta))$ gradually turns on ${}^{T}\vec{V}_c$. The function $S(\beta)$ is shown in Figure 5.11 and defined as:

$$S(\beta) = \begin{cases} \frac{1}{2}\left[\cos\left(\frac{\beta\pi}{2\epsilon}\right) + 1\right] & \forall 0 \leq \beta \leq 2\epsilon \\ 0 & \forall \beta > 2\epsilon \end{cases} \tag{5.47}$$

Returning to Figure 5.8, when the measured force magnitude is computed, it is assigned to $\beta$ which is used as an argument to the blending function, S($\beta$), described above which controls the blending of the nominal velocity, ${}^{T}\vec{V}_n$, and the commanded velocity from the controller, ${}^{T}\vec{V}_c$. After the blending function has operated on ${}^{T}\vec{V}_n$ and ${}^{T}\vec{V}_c$ to produce ${}^{T}\vec{V}_n{}^{*}$ and ${}^{T}\vec{V}_c{}^{*}$ respectively, they are superimposed to give the desired velocity, ${}^{T}\vec{V}_d$.

Table 5.2 Summary of parameters used for the simulations.

| Parameter | Nominal Values | |
|---|---|---|
| | SISO RISO & DISO | DIDO |
| ${}^{w}K_e$ / 1,000 | $\begin{bmatrix} 10\text{N/m} & 0 & 0 \\ 0 & 10\text{N/m} & 0 \\ 0 & 0 & 10\text{N-m/rad} \end{bmatrix}$ | $\begin{bmatrix} 50\text{N/m} & 0 & 0 \\ 0 & 200\text{N/m} & 0 \\ 0 & 0 & 10\text{N-m/rad} \end{bmatrix}$ |
| $\vec{V}_n$ | 0.05 m/s | 0.0125 m/s |
| $T$ | 1 ms | 28.8 ms |
| $\epsilon$ | 0.2 | 0.2 |

Because the parameters controlling the servo rate and the contact model were so important to the overall performance and stability of the simulated controller, they were carefully selected. Table 5.2 summarizes the parameters used for the SISO, RISO, DISO, and DIDO simulations. The differences between the parameters for the SISO/RISO/DISO simulations as compared to those for the DIDO simulations are largely due to the mismatch in the servo rate which occurred due to a lack of foresight about its importance. Section 6.1.3 gives a more detailed explanation of the parameter selection.

*5.5.2 Implementation on Robot Testbed.* Figure 5.12 depicts the block diagram of the implementation of the ANN controller on the robot. The superblock labeled "PUMA" contains all the components that were internal to the PUMA and its low-level controller. As indicated in Figure 5.12, a low-level PID controller servoed the manipulator to the desired joint positions, $\vec{q}_d$. The superblock identified as "Env." is a model of the interaction between the robot and the environment which created the contact forces measured by the UFS. Note that the forward kinematics of the PUMA and the ${}^{w}K_e$ transformation shown in the "Env." block were not explicitly computed, but rather implemented as hardware. When the measured forces were presented to the ANN controller, it computed the commanded velocity which was superimposed with the nominal velocity to create the desired velocity vector. The inverse Jacobian matrix transformed the desired cartesian velocity into a vector of desired joint velocities which were then integrated to yield the desired joint angles that are commanded to the low-level servo of the PUMA robot.

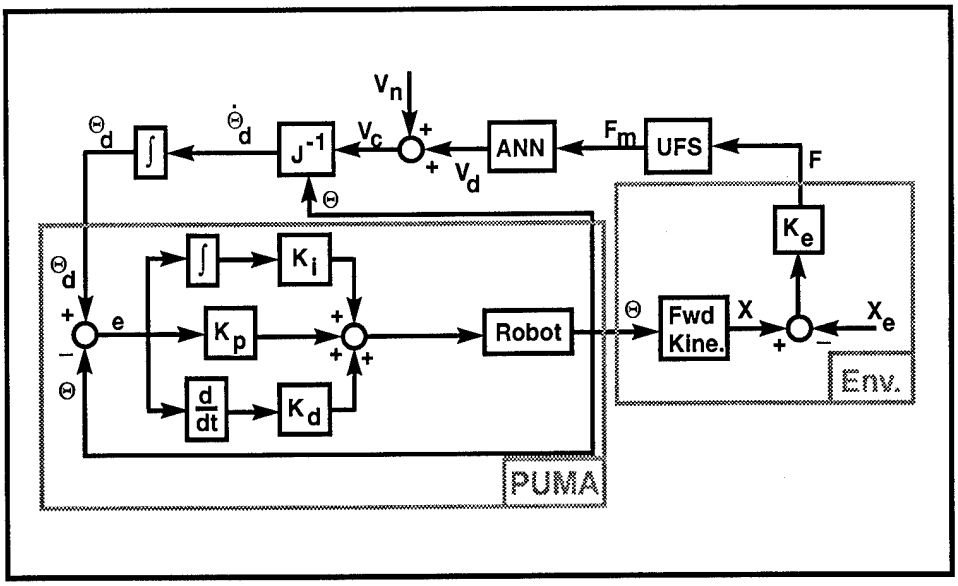



Figure 5.12 Detailed block diagram of the scheme used to implement controllers on the PUMA manipulator.

The expression for the tool-frame inverse Jacobian matrix, ${}^{T}\mathcal{J}(\vec{q})^{-1}$, for the PUMA robot as a planar manipulator is:

$$ {}^{T}\mathcal{J}(\vec{q})^{-1} = \begin{bmatrix} ij_{11} & ij_{12} & ij_{13} \\ ij_{21} & ij_{22} & ij_{23} \\ ij_{31} & ij_{32} & ij_{33} \end{bmatrix} \tag{5.48} $$

where the individual components are given by:

$$ ij_{11} = \frac{\sin\theta_5}{A_2\cos\theta_3 - A_3\sin\theta_3} \tag{5.49} $$

$$ ij_{12} = \frac{-\cos\theta_5}{A_2\cos\theta_3 - A_3\sin\theta_3} \tag{5.50} $$

$$ ij_{13} = \frac{-L_T\sin\theta_5}{A_2\cos\theta_3 - A_3\sin\theta_3} \tag{5.51} $$

$$ ij_{21} = \frac{A_2\cos\theta_{35} - A_3\sin\theta_{35} - D_4\sin\theta_5}{D_4(A_2\cos\theta_3 - A_3\sin\theta_3)} \tag{5.52} $$

$$ ij_{22} = \frac{A_3\cos\theta_{35} + D_4\cos\theta_5 + A_2\sin\theta_{35}}{D_4(A_2\cos\theta_3 - A_3\sin\theta_3)} \tag{5.53} $$

$$ ij_{23} = \frac{L_T(-A_2\cos\theta_{35} + A_3\sin\theta_{35} + D_4\sin\theta_5)}{D_4(A_2\cos\theta_3 - A_3\sin\theta_3)} \tag{5.54} $$

$$ij_{31} = \frac{-A_2 \cos\theta_{35} + A_3 \sin\theta_{35}}{D_4(A_2 \cos\theta_3 - A_3 \sin\theta_3)} \tag{5.55}$$

$$ij_{32} = \frac{A_3 \cos\theta_{35} + A_2 \sin\theta_{35}}{D_4(-A_2 \cos\theta_3 + A_3 \sin\theta_3)} \tag{5.56}$$

$$ij_{33} = \frac{A_2 L_T \cos\theta_{35} + A_2 D_4 \cos\theta_3 - A_3 L_T \sin\theta_{35} - A_3 D_4 \sin\theta_3}{D_4(A_2 \cos\theta_3 - A_3 \sin\theta_3)} \tag{5.57}$$

The low-level servo loop on the PUMA updated 32 times as fast as the trajectory updates were sent down to it. The trajectory updates were computed every 28.8 milliseconds (ms). The low-level servo loop was accessed via the ALTER mode of the PUMA controller. The PID gains of the low-level servo loop were the factory preset gains.

*5.5.3 Velocity Integration.* This section describes how the velocity signals generated by the controllers were integrated to generate position trajectories. The method of integration is important because small differences in the algorithm can have significant effects on the results. The method of integration was the simple trapezoidal approximation described by:

$$\int_{t=0}^{t=NT} f(t)dt \approx \sum_{i=1}^{i=N} \frac{T}{2}\left[f(iT) + f((i-1)T)\right] \tag{5.58}$$

For the implementation in simulation, the integration was not complicated by the presence of the inverse of the Jacobian matrix in the controller. Since all the simulation computations were done in cartesian space, Eq (5.58) was applied to the ${}^{w}\vec{V}_d$ to produce ${}^{w}\vec{X}_d$. For the PUMA implementation, however, the desired cartesian velocity output of the ANN controller is multiplied by the inverse Jacobian matrix, ${}^{T}\mathcal{J}(\vec{q})^{-1}$, and integrated to generate a desired joint position trajectory. Since ${}^{T}\mathcal{J}(\vec{q})^{-1}$ is a function of joint position, *instantaneous* velocities are produced when it is multiplied with a desired cartesian velocity. If the robot's joint angles change very much during the integration interval, error will be introduced by the variation of ${}^{T}\mathcal{J}(\vec{q})^{-1}$. Consequently, it is important to keep the period of integration sufficiently small so that the change in ${}^{T}\mathcal{J}(\vec{q})^{-1}$ will likewise be small. This is accomplished by subdividing the position update period into smaller integration intervals over which ${}^{T}\mathcal{J}(\vec{q})^{-1}$ does not change significantly. For the workspace region of interest and the magnitude of commanded velocities, subdividing the position update interval into two subintervals has proven sufficient.

Another important consideration in the velocity integration was whether to use a *reference* trajectory or not. The difference is that, for a reference trajectory, the previous *commanded position*, $\vec{q}_{d(t-1)}$, is used as the starting point for each integration period, whereas one would otherwise use the current *measured position*, $\vec{q}_m$. The significance of using a reference trajectory is that position errors are accumulated so that the PID servo controller can continue to work on them during subsequent servo intervals. In contrast, if $\vec{q}_m$ is used as the starting point for each integration period, any error in position left by the PUMA's PID servo controller at the beginning of the next servo interval is forever forgotten. This lack of memory can lead to end-point drift due to measurement noise and quantization error on a manipulator. Therefore, the reference trajectory was used for the PUMA implementation of the various controllers. For the simulation, however, the measured position, ${}^{w}\vec{X}_m$, was used in the integration. This is because for the simulation, we are assuming that the robot is perfect so the measured and desired positions are identical.

### *5.6 Timing Considerations.*

The task at hand is to train an ANN controller to mimic the performance of an existing controller system. This task has been referred to as a system identification [39] because we are using the ANN to identify the characteristics of the controller system. For this discussion, we make the following assumptions about the controller:

- it is a "black-box" for which we can only observe the I/O-mapping
- the controller exhibits desireable system performance[5] when implemented with the plant of interest,
- the processing time delay is unknown and may vary

Because it is a "black-box" we do not have access to the inner workings of the controller so we must examine samples of its inputs and outputs to ascertain its behavior. Thus, we seek to learn the input-output mapping that the controller performs as it controls a plant of interest.

[5]Desireable performance characteristics include stability, quick settling time, minimal overshoot, and good tracking.

*5.6.1 Processing Delay Time.* Since the controlled system is stable, we infer that the "black-box" controller has an acceptable processing delay. For digital control systems running on a single microprocessor, the processing delay of the controller is necessarily less than the time between servo updates. As long as the control algorithm can be computed in time to send the resulting controller command down at each servo period, it doesn't matter how short the actual processing delay of the control law is. The actual processing delay of the ANN control law is likewise unimportant as long as its output can be computed in time to be available for the controller updates at the servo rate. However, even with the apparent room for error, there are several possible problems that can occur if timing is not carefully considered when training the ANN.

The first problem has to do with the computational burden of the ANN. If the servo rate has to be reduced to allow time for the ANN to compute its output, then performance equivalent to that of the "black-box" controller cannot be expected. Although the difference may be insignificant, increasing the sampling period always carries a cost penalty in system performance [27]. The time it takes to compute the ANN output is a function of the microprocessor speed and the architecture of the network itself. As the number of inputs nodes, output nodes, or hidden layer nodes increases, the number of multiplies and adds required for the feedforward computations goes up rapidly due to the massively interconnected nature of the network. Consequently, smaller networks can be run at a higher servo rate on a given microprocessor platform.

A second timing problem is that the processing delay of the "black-box" controller may actually be a multiple of the apparent servo rate of the controller. This can occur when there are nested control loops running on separate microprocessors in the system. For example, Figure 5.13 shows the block diagram of a system designed to track a moving target with the end-effector of a robot using a vision system for feedback. The inner loop of such a system might be a low-level position servo loop that is running at a very high servo rate, $T_s$, with a PID feedback control law to provide accurate tracking of position commands. Surrounding this control loop may be a feedback controller commanding positions to the inner loop based on signals from the camera. The imaging system may take a much longer time, $kT_s$, to process the camera data even though it runs on a faster microprocessor. As a result, the

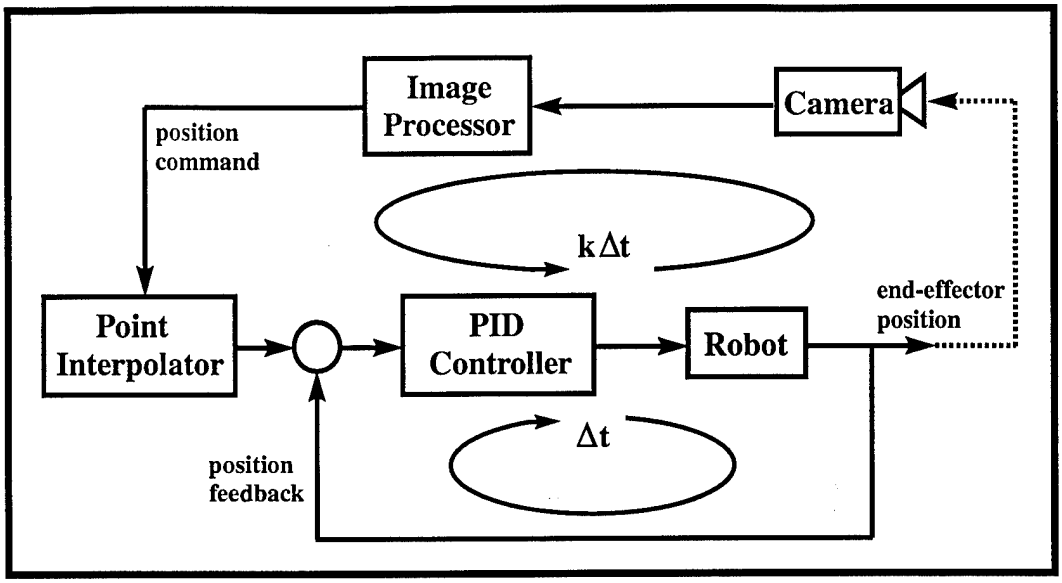



Figure 5.13 Block diagram of a system designed to track a moving target with the end-effector of a robot using a vision system for feedback.

inner servo loop may be interpolating between the camera-generated position commands in order to provide a continuous trajectory to track.

In a system such as this, the problem occurs when we want to identify the outer control system but we do not know that it is operating with a longer delay. In our ignorance, we may sample the input to the camera and the output of the image processor at the inner loop servo rate. As a result, the output data we capture will appear discretized since it changes only every k$T_s$ while the input data changes every $T_s$. The problem is not severe, however, since examination of the data will reveal its true nature and it can be solved by sampling the data every k$T_s$ and associating the $n+1$ output sample with the $n$th input sample to account for the one cycle processing delay.

A third possible problem is that the outer controller may be able to process data at the same rate as the inner servo loop but it has a startup delay. This problem can occur even when both control loops are being computed on a single microprocessor. Multistep integrations and differentiations in discrete controller require some startup delay until the proper number of input samples have been accumulated. Once the requisite number of samples are available, the algorithm can output a result every time step. In this case, there

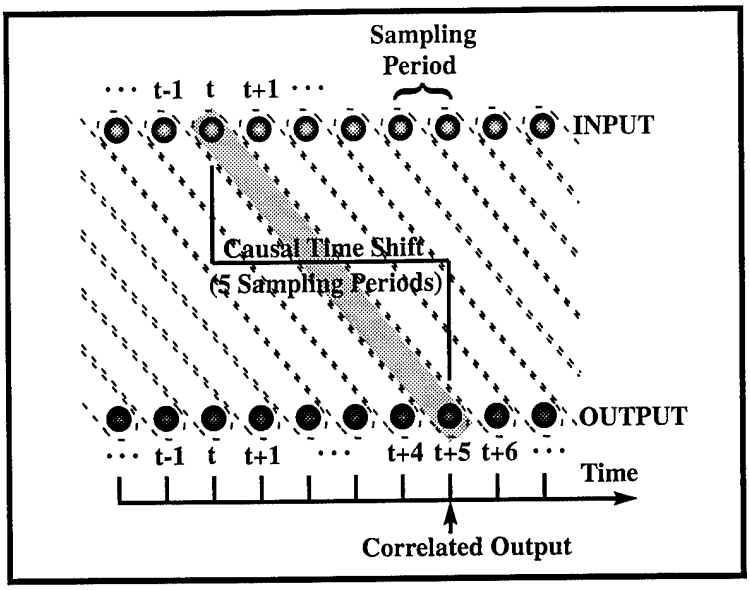



Figure 5.14 Illustration of causal time shift between captured input and output data for an example case of a controller having a processing delay of five sampling periods.

is simply a time shift in the data that must be accounted for when examining the input-output data captured at each servo period. If the startup delay was five servo periods, then the sampled output data stream must be shifted back in time five servo periods when it is correlated with the sampled input data stream.

Finally, the fourth and most dreaded possible problem is that the processing delay of the controller may not be constant. This is difficult because it defeats any efforts to simply time shift the data to account for the delay. Researchers in speech recognition have developed a method called *dynamic time warping* to account for the effect of variable duration in speech signals [40:p297]. It is possible that such a method could be adapted to solve the variable time delay problem.

*5.6.2 Causality.* According to Payandeh [41], *causality* is defined as "the property of a linear system (operator) when its output at time $t$ is dependent on its input up to the time $t$." This gives us a basis to discuss the *Causal Time Shift* below.

Figure 5.14 illustrates sampled input and output data streams as two series of parallel dots running horizontally with increasing time to the right. These sampled data are to

be used to construct input-output data pairs for training an ANN controller to imitate the controller. At each sampling interval, an instantaneous snapshot is taken of the input data and output data. If a controller having a processing time delay equal to 5 sampling periods has generated the output data stream in response to the input data stream, it is clear that the output at time t is not a function of the input at time t. Rather, the output at time t+5 corresponds to the input at time t and, for the purpose of this discussion, that output (at t+5) will be referred to as the *correlated output*. The processing time delay can alternatively be called a *Causal Time Shift* because it is the time shift required for there to be a causal relationship between the input-output data pairs. Since the controller in Figure 5.14 has a 5 sample period processing delay, it could not have possibly produced the output at time t+4 in response to the input at time t because the input did not have time to propagate through the controller. Similarly, the output at time t+6 was generated by the input at time t+1. If one knows the processing time delay and knows that it is a constant, then the captured data can simply be time shifted by the causal time shift to restore the causal relationship between the input-output pairs.

Output data that occur earlier than, or later than, the correlated output may or may not be strongly related to the input in terms of cause-effect. For example, in Figure 5.14 the output at times t+4 and t+6 may or may not be strongly related to the input at time t. If the data in the neighborhood of the correlated output are very similar, then the observed input-output mapping is relatively insensitive to accurately identifying the correct causal time shift. On the other hand, under some conditions the observed mapping may be very sensitive to the causal time shift.

Two factors affect how sensitive the observed input-output mapping is to the causal time shift; the dynamic bandwidth of the controller and the frequency content of the input signal. Figure 5.15 attempts to depict the general relationships between the input signal frequency content, the controller bandwidth, and the sensitivity of the input-output mapping to variations in the causal time shift.

Higher input-output mapping sensitivity is indicated by a darker shading in Figure 5.15. If the controller has a high bandwidth and the frequency content of the input is low, then output data in the region of the correlated output will relate fairly strongly to the

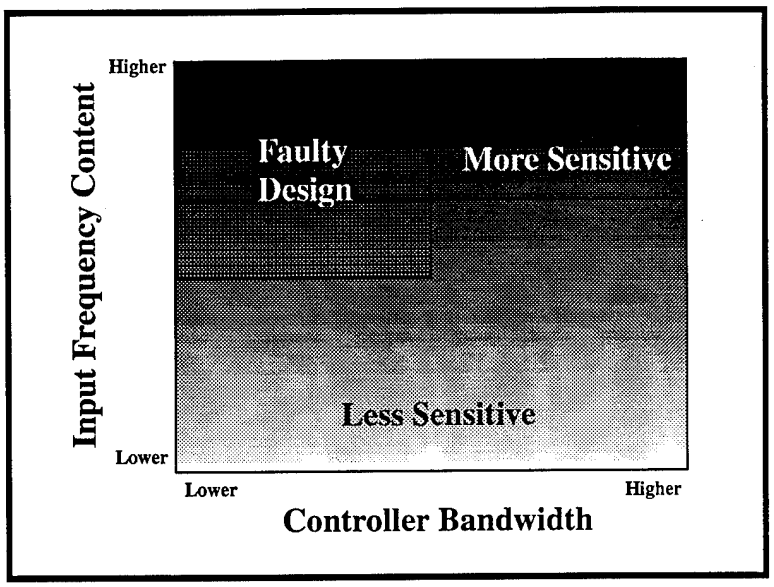



Figure 5.15 Illustration of the general relationships between the input signal frequency content, the controller bandwidth, and the sensitivity of the input-output mapping to variations in the causal time shift.

input sample because the output will vary slowly and there will be little difference between neighboring output samples. If the neighboring output samples are strongly related to the input, then only a small error will be introduced by associating the input with the wrong output sample. Consequently, the mapping will have a low sensitivity to incorrectly specifying the causal time shift as shown in the lower right quadrant of Figure 5.15.

If, on the other hand, the output is rapidly changing in response to a rapidly changing input, the output data in the region of the correlated output will not relate strongly to the input sample. Since the neighboring output samples vary considerably, a larger error will be introduced by associating the input with the wrong output sample. Consequently, the mapping will have a high sensitivity to incorrectly specifying the causal time shift as shown in the upper right quadrant of Figure 5.15. This discussion thus encourages one to sample quickly so as to have slowly varying data from one output sample to the next. Shannon's sampling theorem also encourages one to sample as quickly as possible to prevent aliasing in the data. However, reality has a way of constraining the upper limits of sampling speed.

The upper left region in the figure is labeled "faulty design" because one would not want to use a low bandwidth controller to control a high bandwidth system. In addition, if the physical characteristics of the system cause it to respond slowly to input changes, then the output data will be a weak function of the input even when properly time shifted. A large inertia is one possible cause of slow response which essentially low-pass filters the input. This could lead to problems during ANN training because a weak functional mapping implies that there could be inconsistent input-output pairs contained in the data. Inconsistent data would, in turn, slow down the progress of training.

*5.6.3 The Human Factor.* When a human operator is introduced into a control system, there are always inconsistencies in the exact control response that will be measured for a given stimulus input. Even if the person responds in the same way every time he receives a given stimulus, his reaction time may be substantially different for two different instances. A person's reaction time is a direct function of his alertness, which, in turn, is a function of many things such as his level of concentration and how rested he is. [62] also presents these concerns for transferring human skills to robots via analysis of demonstration data. Owing to these factors, it is clear that a person's reaction time can vary significantly during the course of even a brief task. In terms of control system analysis, reaction time represents a processing time delay so we are faced with a variable processing time delay.

## *5.7 Controller Performance Evaluation.*

To make an objective comparison of the performance quality for the many trained ANN controllers, a performance metric was derived. The purpose of the metric is to provide a single scalar index of performance that reflects the quality of how well the ANN performs the edge-mating task. The desired metric has the following features:

- penalizes for spurious motion during the alignment task
- penalizes for slow execution of the alignment task
- penalizes for instabilities or divergence away from the aligned position once alignment is achieved
- penalizes for excessive contact force magnitudes between the peg and the table surface

- grades the performance relative to the performance of the "best" accommodation matrix controller's performance

The expression for the performance metric, $\zeta$, is given by:

$$\zeta = \frac{\chi}{\tilde{\chi}} \tag{5.59}$$

where $\chi$ is the composite performance index for a given run and $\tilde{\chi}$ is the composite performance index for the "best" accommodation controller. The values for these composite performance indices are computed from:

$$\chi = \rho\mathcal{D} + \beta\Delta t + \gamma|\Delta P|_{\max} + \xi|F|_{\text{avg}} + \phi|F|_{\max} \tag{5.60}$$

where $\rho, \beta, \gamma, \xi$ and $\phi$ are positive-valued weighting factors, $\mathcal{D}$ is the root-mean-squared path length traversed from impact to first crossing of the aligned position, $\Delta t$ is the amount of time elapsed between impact and first crossing of 180-degrees, $|\Delta P|_{\max}$ is the maximum position amplitude away from the aligned position after first alignment, $|F|_{\text{avg}}$ is the average of the absolute values of the forces measured after the first alignment, and $|F|_{\max}$ is the maximum (peak) RMS force amplitude after alignment. The weighting factors were heuristically chosen as:

$$\rho = 2.0 \tag{5.61}$$

$$\beta = 1.0 \tag{5.62}$$

$$\gamma = 5.0 \tag{5.63}$$

$$\xi = 2.0 \tag{5.64}$$

$$\phi = 5.0 \tag{5.65}$$

One can see that the first term in Eq (5.60) includes the performance cost for spurious motion while the second term penalizes for lethargic motion. The third term includes a measure of post-alignment stability by penalizing for position deviations, either positive or negative, from the aligned position. The fourth term provides a penalty based on the average force magnitude after the peg has been aligned. This term captures a sense of how hard the

peg is being pressed against the table once the task is completed. Its presence in the metric assures that controllers which jam the peg hard will not go unpunished. The final term of Eq (5.60) captures any transient force spikes that may occur after the peg has been aligned. During early testing, several controllers which would unexpectedly pulse the peg against the table were noted. The fifth term is intended to capture the behavior of those controllers.

## *5.8 Summary.*

This chapter has presented the mechanics of the algorithms which were applied during this research effort. It is a toolbox from which techniques were drawn as needed to unfold the characteristics of the proposed control method. Our toolbox contains the procedures used to collect the various raw data types (SISO, RISO, DISO, and DIDO) as well as the seven data processing options that were available to convert the raw data to training data. Two data processing options, the magnitude normalization and the Lipschitz clipping, were immediately discarded because of revealed weaknesses.

The structure of the MLP ANN and the back-propagation training algorithm were also presented, along with descriptions of how the efficacy of the training could be evaluated. Among the methods used to evaluate the success of training were the simple error tracking and network interrogation by way of both the UVP and the LSMF techniques. The interrogation techniques were designed to extract the matrix that the ANN is most closely emulating in operation. Four different matrix similarity indexes were also presented which are used to determine how the structure, gain, sign, and ratio characteristics of the extracted matrices compared with a given objective matrix.

The particular details of the simulation code and PUMA robot control code were also derived. The details presented included derivations of the required coordinate system transformations and Jacobian matrices. In addition, the frictionless and unity friction contact models used in simulation were presented in detail.

In the last sections of this chapter, timing considerations that may have affected the success of the ANN controllers were discussed and the metric used to evaluate the performance of the controllers was presented. The metric is designed to reward the controller

for directness of motion, conservative use of contact force, and stability after the peg has reached alignment. These tools will provide a solid foundation from which to conduct our investigation.

# *VI. Results*

This chapter presents all of the data and analysis conducted during this research effort. It is organized to present a developing story from baseline information, through intermediate investigations to the final tests of the overall conceptualized control paradigm. We start by describing the system-verifying baseline experiments using the accommodation matrix controller. This is followed by the various observations and investigations of SISO, RISO and DISO training data. Finally, the efforts to utilize DIDO training data are presented with a short summary at the end.

## *6.1 Baseline Accommodation Matrix Controller.*

To validate the software and to establish a baseline of performance, a "best" ACC matrix, $\mathcal{A}_a$, was empirically derived. For tasks in planar coordinates, the $\mathcal{A}_a$ is a 3x3 matrix. Peshkin [42] shows that if the origin of the coordinates is at the center of the mating surface of the peg tip and the task is planar, then $\mathcal{A}_a$ is a sparsely populated accommodation matrix having non-zero values only in the $(2,2)$ and $(3,3)$ position. This configuration of $\mathcal{A}_a$ provides for accommodation in the tool-frame $Y$-axis as well as angular accommodation.

*6.1.1 Nominal Task Execution.* It is useful to examine plots representing the nominal (typical) execution of the edge-mating task as a reference for our future discussions. Figure 6.1 shows the positions ((a)-(c)), measured forces ((d), (f), and (h)), and commanded velocities ((e) and (g)) for an idealized case of task execution. The position data shown in Figure 6.1 ((a)-(c)) are measured in the world-frame coordinate system while the other data are shown in the tool-frame coordinates. We see that upon initial contact, the angular alignment error is corrected linearly with respect to time, as shown in Figure 6.1(c), and the curvature in the $X$-axis position shown in Figure 6.1(a) is caused by the translation of the point at the center of the peg tip due to constant angular rotation about the corner of the peg in contact with the table and some sliding allowed by the frictionless contact model used for this example. The $Y$-axis position plot is nearly linear due to the constant peg rotation rate, but is slightly curved because of the peg bouncing slightly against the table top. Once the peg reaches its first alignment with the table, as indicated by $\theta_p = 180°$,

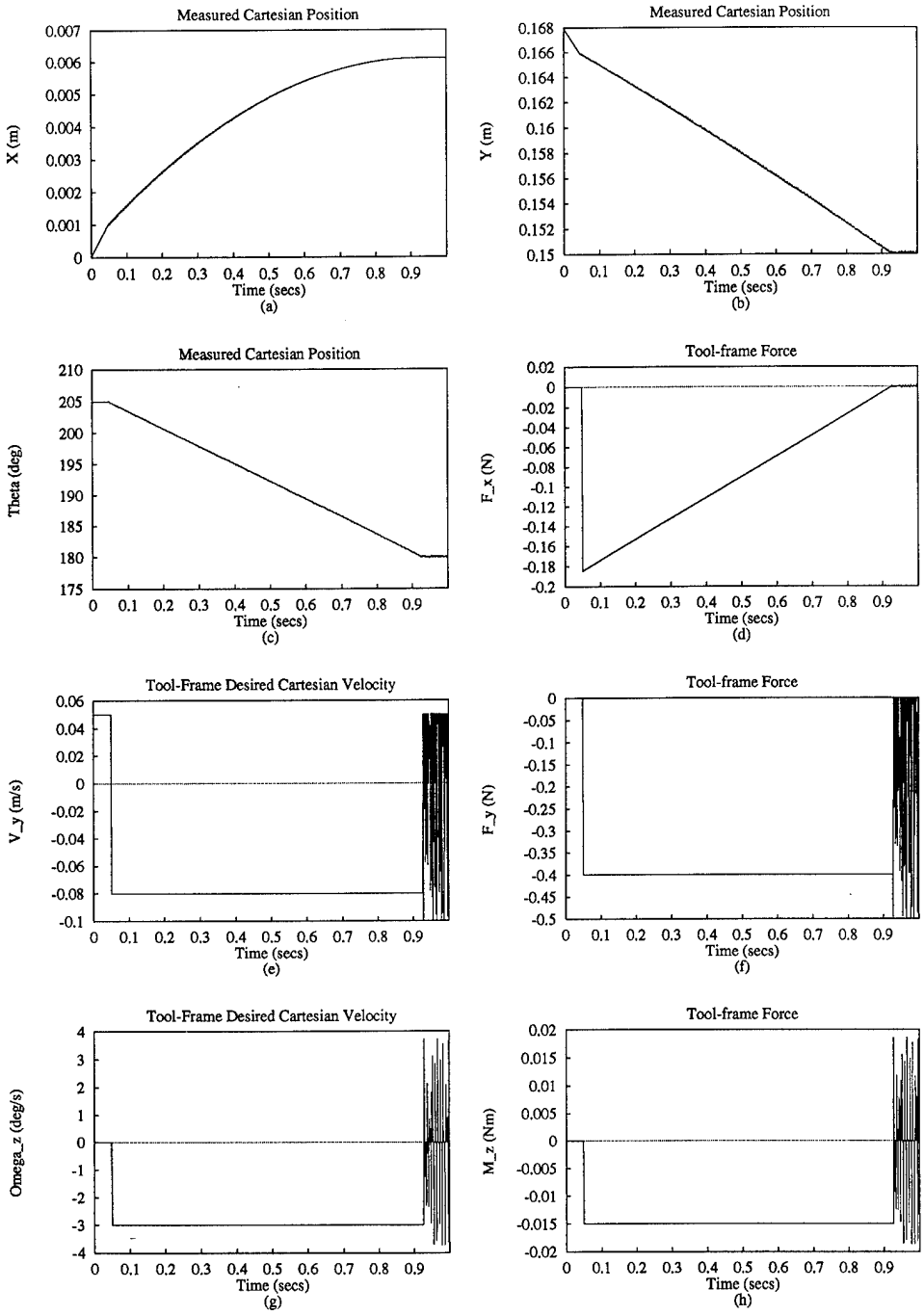



Figure 6.1 Idealized results of a nominal ACC controller performing the edge-mating task from a CCW initial misalignment angle. Time histories of position ((a)-(c)), commanded velocities ((e),(g)), and measured forces ((d),(f),(h)) are shown. Note that $V_x = 0$.

it begins a small-magnitude rocking or oscillation about the aligned position, as the trace of $M_z$ shown in Figure 6.1(h) best indicates. The rocking is caused by the stiffness of the environment interaction model interplaying with the controller servo rate and the fact that there are no dynamics in our simulation model to damp out the oscillations. The tool-frame $F_x$ shows a linear decrease to zero as the table contact force, which is always normal to the table for the frictionless contact model shown, points more and more along the tool-frame $Y$-axis as the peg is stood up into alignment. It should be clearly understood that the data presented in Figure 6.1 were fabricated for the purpose of this explanation to orient the reader. Under close scrutiny, the data shown for the position history and the commanded velocity history will certainly be found to lack consistency in the figure. For example, if the peg was constantly moved in the -$Y$-axis direction as shown by the constant negative region of Figure 6.1(e) with the angles shown in Figure 6.1(c), then the measured $Y$-axis position shown in Figure 6.1(b) would not have been manifested since the peg would have been rising from the table rather than falling. Despite these subtle details, the author hopes that the description of the nominal task will serve well to orient the reader to the data presented in subsequent simulation plots.

*6.1.2 Experimental Tests.* Through experimental trial-and-error, the $\mathcal{A}_a$ yielding the best qualitative performance on the robot was found to be:

$$\mathcal{A}_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.0007 & 0 \\ 0 & 0 & 0.2 \end{bmatrix} \tag{6.1}$$

The criteria used to make the qualitative judgement of best performance were the same criteria that are included in the expression for the performance metric, $\zeta$, presented in Section 5.7. Naturally, these $\mathcal{A}_a$ values are affected by the controller gains, the hardness of the peg and table, the inertia of the robot, and the magnitude of the nominal velocity, ${}^T\vec{V}_n$, used to move into contact with the table. Figures 6.2 and 6.3 show sample runs of data with the $\mathcal{A}_a$ controlling the robot. Notice that the system behaves significantly differently when the initial misalignment angle is clockwise (CW) (ref. Figure 6.2) as compared to when it

is counter-clockwise (CCW) (ref. Figure 6.3). This is due primarily to the configuration-dependent nature of the manipulability for the robot. When the peg is initially misaligned CCW with the wrist bent inward (pointing towards the base of the robot), the peg tends to stay in contact with the table and displays a higher inertia. When the peg is CW, the wrist is pointing outward and the peg comes out of contact with the table more readily in response to the commanded velocities. This behavior was universal in the experiments on the robot.

*6.1.3 Simulation Tests.* For the simulation, the $\mathcal{A}_a$ was significantly different than for the experimental work on the robot. The $\mathcal{A}_a$ selected for the simulation was:

$$\mathcal{A}_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 200.0 \end{bmatrix} \tag{6.2}$$

Figure 6.4 shows the simulation results of the ACC controller performing the edge-mating task from a CCW initial misalignment angle. The position vector components of the peg, given in the world-frame cartesian coordinates, are shown in Figures 6.4(a)-(c). The commanded velocity vector components of the ACC controller expressed in the tool-frame cartesian coordinates are depicted in Figures 6.4(e) and (g). Note that the $V_x$ component of the commanded velocity is identically zero for all time and is not shown in Figure 6.4 or any of the subsequent simulation result plots. Figures 6.4(d), (f), and (h) show the contact force history between the peg and the table expressed in the tool-frame. Because the simulation is based purely on kinematics and contains no dynamic properties such as manipulator inertia, the limit-cycle motion of the peg in contact with the table is exaggerated. The contact force history shows the peg bouncing out of contact nearly 50% of the time when it is "touching" the table. No attempt was made to introduce dynamic properties into the simulation model. The kinematic model is considered satisfactory for the purposes of this research.

Although the form of $\mathcal{A}_a$ for the simulation was identical to that of $\mathcal{A}_a$ used in the experimental work, the values for the $(2,2)$ and $(3,3)$ elements were quite different. This is due to several factors including:

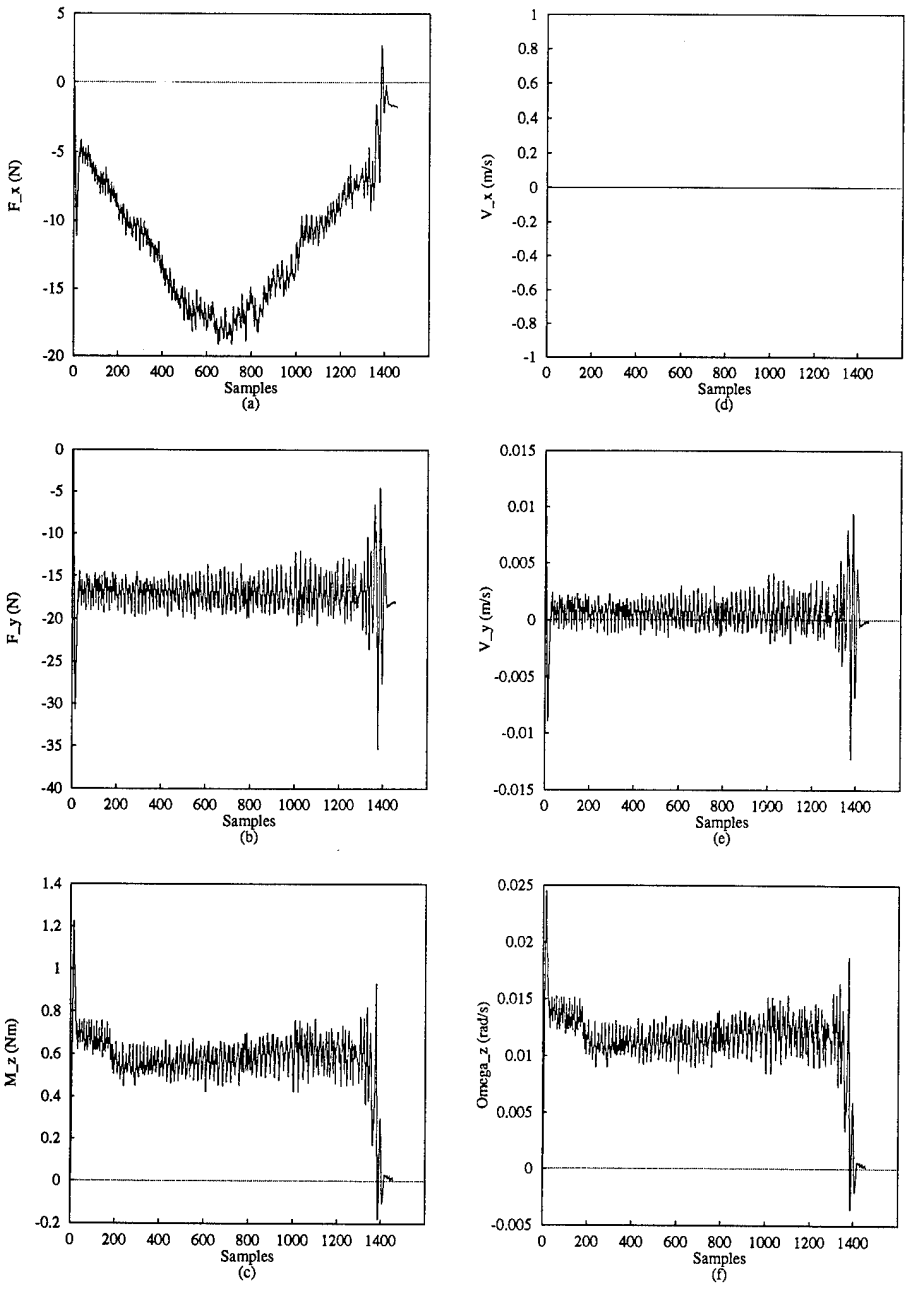



Figure 6.2 Force and commanded velocity time histories of an ACC controller performing the edge-mating task from a CW initial misalignment angle on the PUMA robot.

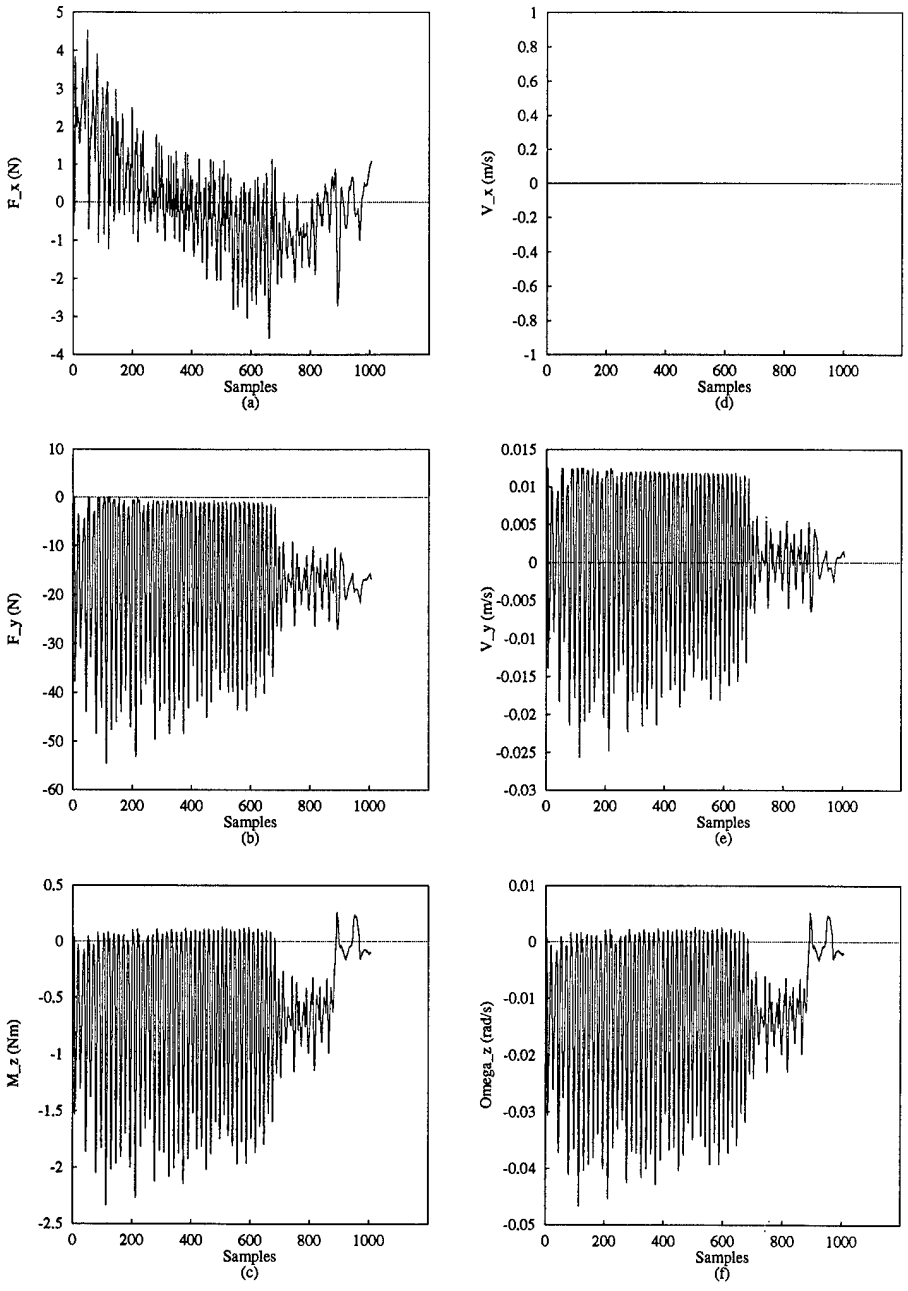



Figure 6.3 Force and commanded velocity time histories of an ACC controller performing the edge-mating task from a CCW initial misalignment angle on the PUMA robot.

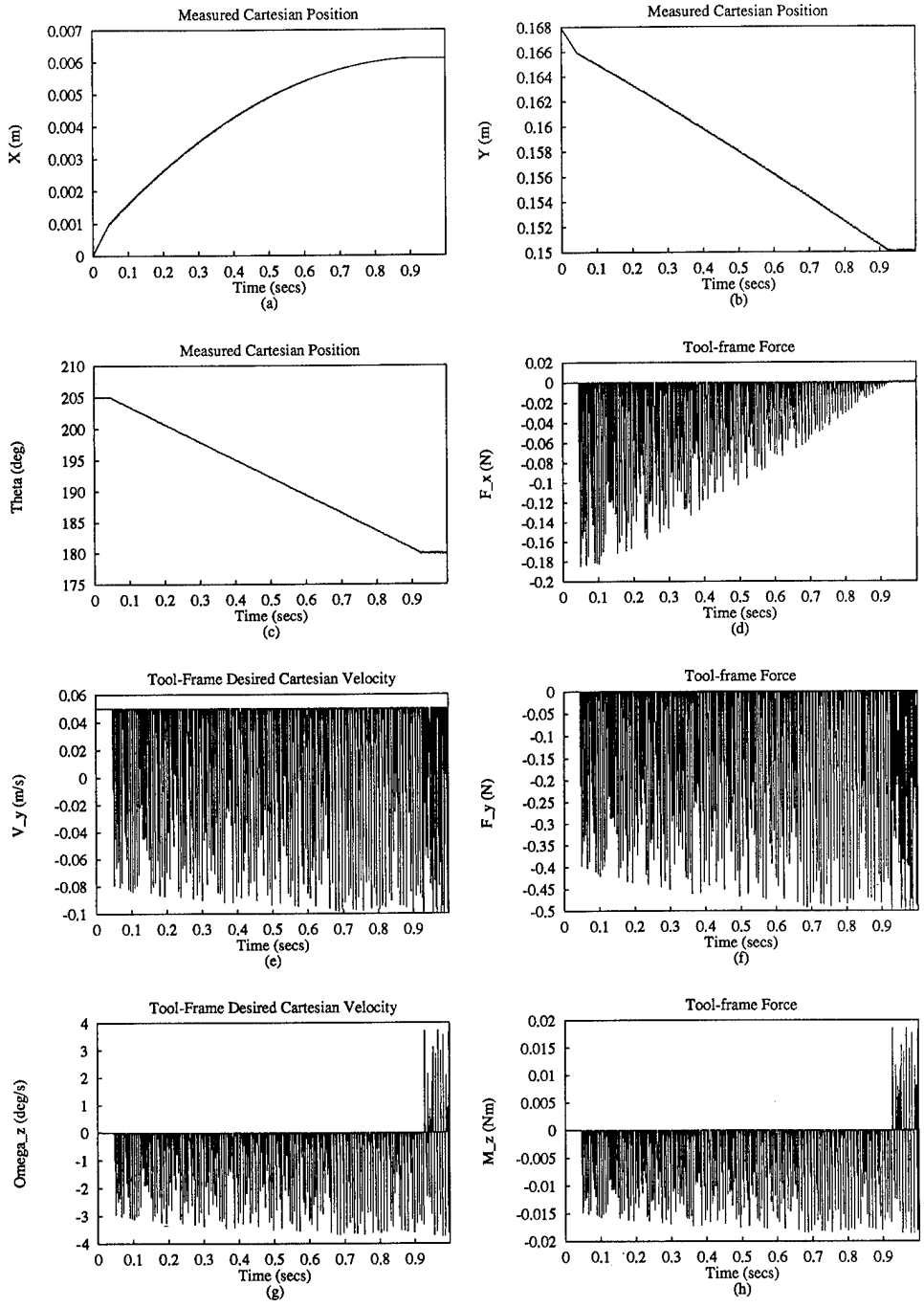



Figure 6.4 Simulation results of the ACC controller performing the edge-mating task from a CCW initial misalignment angle. Time histories of position ((a)-(c)), commanded velocities ((e),(g)), and measured forces ((d),(f),(h)) are shown when the $\mathcal{A}_a$, given in Eq (6.2), is implemented. Note that $V_x = 0$.

- The difference between the sampling rate used for the robot (28.8 ms) and that used for the simulation (1.0 ms).
- The difference between the environmental interaction stiffness of the robot and that used in the simulation. For the robot, the stiffness stems from the hardness of the peg, the hardness of the table, and the stiffness of the PID controller gains. For the simulation, the environmental stiffness, ${}^{w}K_e$, was entirely based on a cartesian spring model.
- The difference between the nominal velocity, ${}^{T}\vec{V}_n$, used to approach the table when the peg is in freespace motion: for the robot, ${}^{T}\vec{V}_n$=0.0125 meter/second as compared to 0.05 meter/second for the simulation.

Although these parameters are easily controllable in the simulation, matching them to the actual values of the robot experiments had only limited success in replicating the output of the experimental system. The result of tuning the simulation to match the experiments is reflected under the DIDO heading in Table 5.2 in Section 5.5.1. The ${}^{T}\vec{V}_n$ and $T$ were selected to exactly match those used on the PUMA hardware. The ${}^{w}K_e$ was tuned to give roughly the same performance in simulation as the robot demonstrated in hardware. The simulation results of the $\mathcal{A}_a$ given in Eq (6.1) using the parameters shown in Table 5.2 are presented in Figure 6.5.

The difference between the interaction stiffnesses was the key obstacle since the robot stiffness was a position-dependent, nonlinear relationship based on the manipulator Jacobian matrix, and the simulation was a simple, constant linear relationship.

A final detail worth noting in Figure 6.4 is that it reflects the case with $\mu = 0$ between the peg and the table. When friction is modeled such that no sliding is allowed ($\mu = 1$), the results look identical except for the $X$-axis position trace. Figure 6.6 shows a detailed comparison between examples of $\mu = 0$ and $\mu = 1$ at the contact interface when starting from the same initial conditions. Note that the sliding of about 0.6 mm adds to the $X$-axis displacement that is always present due to the rotation of the peg about the corner in contact with the table.

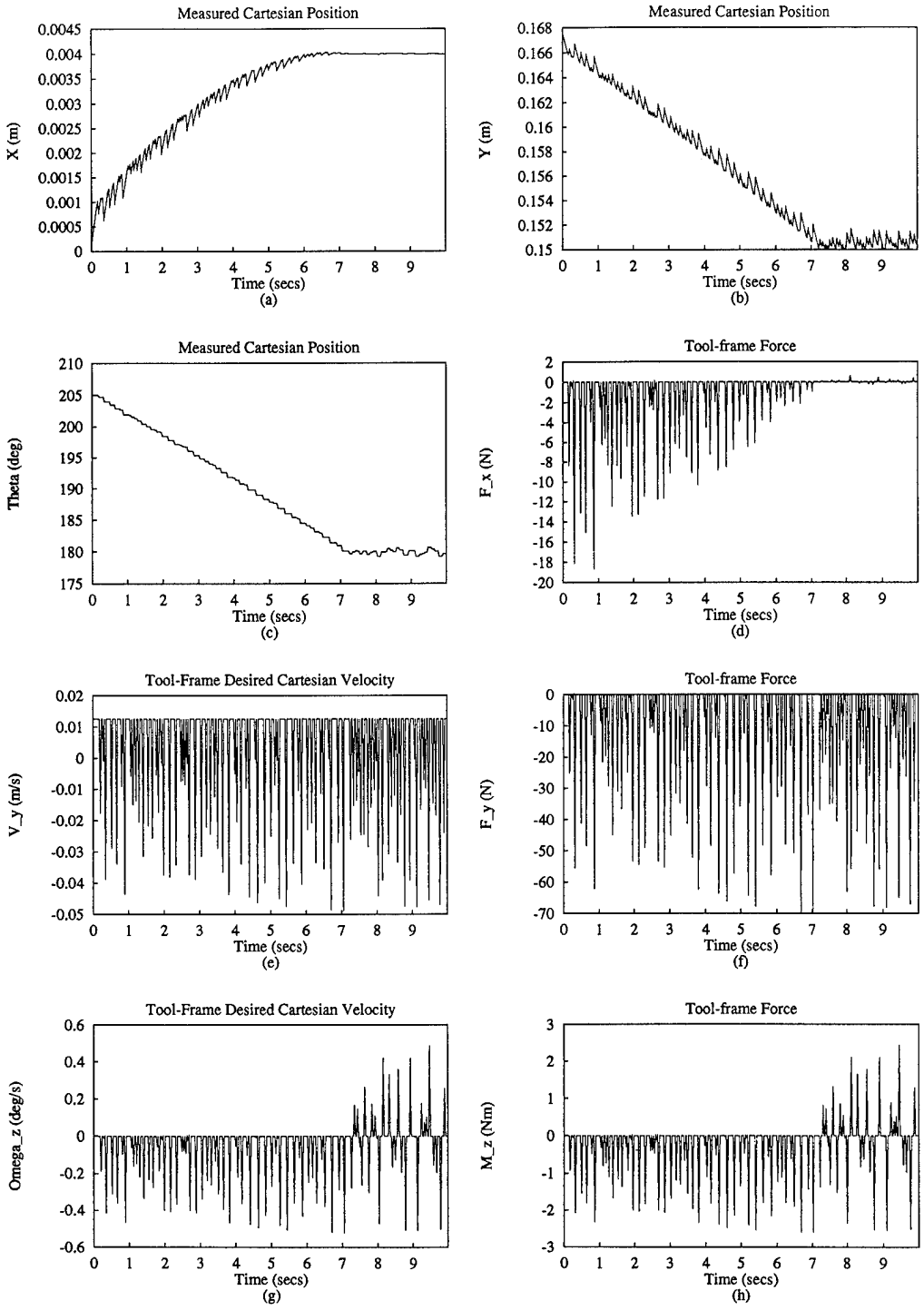



Figure 6.5 Simulation results of ACC controller modeling the PUMA robot environment.

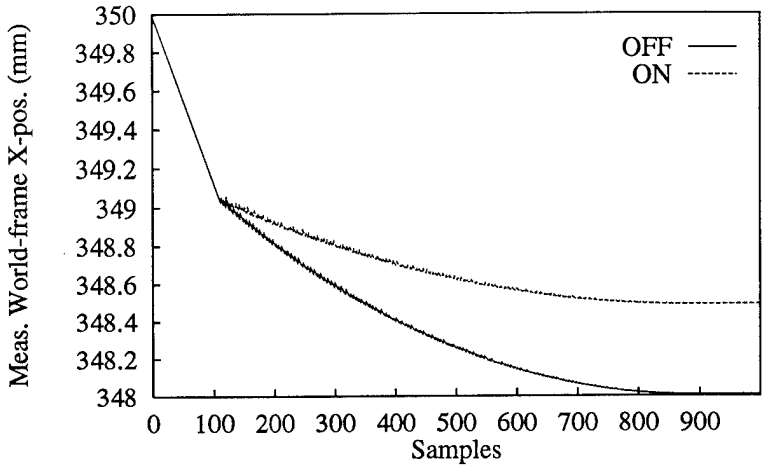



Figure 6.6 Effect of turning the modeled friction on and off in the simulation software on the X-axis position trace.

## 6.2 *SISO Observations.*

This section will describe the results of investigations into the characteristics of the proposed controller that were conducted using SISO training data. The SISO training data provided a well-conditioned, fully-tractable set of data from which the analysis could begin. The analysis focuses on how the training data were distributed across the input vector space since that was found to be an important factor. The SISO data were also the ideal reference from which the reliability and usefulness of the proposed matrix interrogation techniques could be determined. The points of interest are now presented.

*6.2.1 Factors Related to Training Data Distribution.* When obtaining training data for an ANN, a fundamental consideration is that the data must properly represent the relationship to be learned. There are several factors which affect how well the training data represent the relationship. ANN s are known to be poor at extrapolating outside of the range of data with which they were trained. To ensure the ANN will only perform interpolation when implemented, the training data must cover the proper *range* of the input and output spaces. Another important characteristic of similarity that should be maintained between

the training data and the measurement stream of realtime data when the ANN controller is implemented is the distribution of the data. This essentially means that the training data should be statistically similar to the stream of data expected in the application of the controller. To gain an understanding of the effect of different distributions of data on the performance of the ANN controller, a series of tests were done with SISO data. The statistics were indirectly controlled by varying the *range*, the *number*, and the *spacing* of training vectors.

The range of data is simply described by the maximum and minimum values of each component of the input and output vectors. The number of training vectors is a function of the number of distinctly different vectors within the range as well as the *frequency* of repeated vectors. Repeating vectors gives the ANN multiple looks at the same data, which tends to emphasize it as the I/O mapping is created during training. Therefore, one can consider artificially emphasizing a particular training vector or region by simply repeating it in the training data set. Although this technique was considered, it's investigation was outside of the scope of this research project.

The spacing of training vectors refers to how they are distributed in the input feature space between the maximum and minimum values. For example, evenly distributed data would have equal spacing between each of the training vectors. The spacing function, $\mathcal{S}(x)$, is the analytic expression chosen to map an evenly incremented distribution within a given range into an arbitrary distribution within that same range. The spacing function is applied individually to each component of the input feature vector of the SISO data. Each of the spacing functions was chosen to have two particular characteristics:

- The output must be continuously defined over the range of $[-1, 1]$
- The output must cover the full range of $[-1, 1]$ for the range of the input

The transformation process is best described by example. Suppose we choose $\mathcal{S}(x) = \sin(x)$ and want the input to cover the range $x \in [-5, 5]$ with 10 increments between points for a total of 11 points. Here are the steps taken to obtain the desired spacing:

- The original input range, $[-5, 5]$, is subdivided evenly to create the desired number of input vectors, in this case 11 points located at (-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5).

- The input points are transformed into the input range of the desired spacing function. In our example, we desire the input to range from $-\pi/2$ to $\pi/2$, so we multiply each point by $\pi/10$.
- $\mathcal{S}(x)$ is applied to each point to yield a new set of points that are spaced according to $\mathcal{S}(x)$ and range from -1 to 1. In the example, we get (-1, -0.951, -0.809, -0.588, -0.309, 0, 0.309, 0.588, 0.809, 0.951, 1).
- These outputs are rescaled to fill the original desired input range. In our example they are each multiplied by 5 and we get (-5, -4.76, -4.04, -2.94, -1.55, 0, 1.55, 2.94, 4.04, 4.76, 5).

Thus, we have transformed an evenly-spaced set of data ranging from -5 to +5 into a sine-spaced set having the same range.

To investigate the effect of spacing on the training and performance of the ANN controller, four different spacing functions were used on SISO data. In particular, the chosen functions were:

- **even:** $y = x$
- **sine:** $y = \sin(x)$
- **cubic:** $y = x^3$
- **complex:** $y = 0.88\left[(\sin x + 0.3)\cos 10x \tan x + 0.46\right]$

Figures 6.7 through 6.10 depict data in two variables based on the four spacing functions described. Note that the complex spacing function is the only one that generates data sets having a mean value, $\varpi$, which is non-zero.

If the mean, $\varpi$, and standard deviation, $\sigma$, of the measurement stream are appreciably different than that of the training data set, then the ANN controller may be operating in an input region with which it is only sparsely, or not at all, familiar. This can be especially true if one introduces a large non-zero offset (bias) to the training data set and then places the trained ANN into a small-valued, zero-mean measurement stream. If the training data still bound the range of the measurement stream, the ANN will properly interpolate rather than extrapolate the data. However, the mapping functional approximation may be

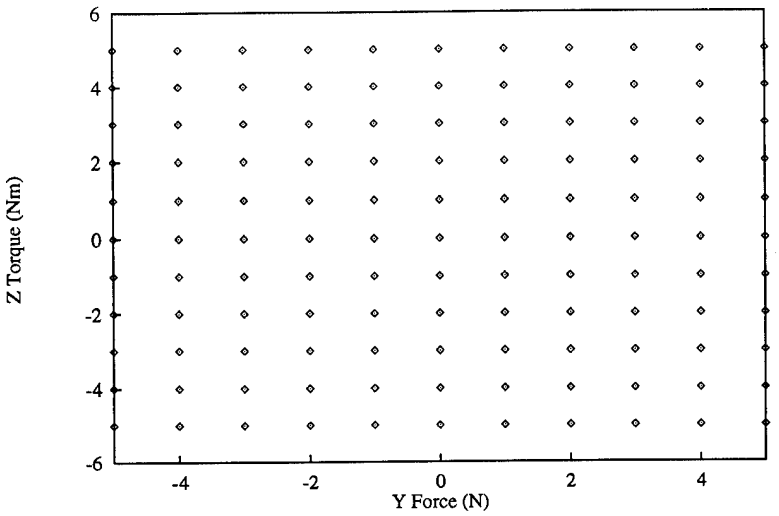



Figure 6.7 Two-dimensional projection of data distributed according to the *even* spacing function ($y = x$).

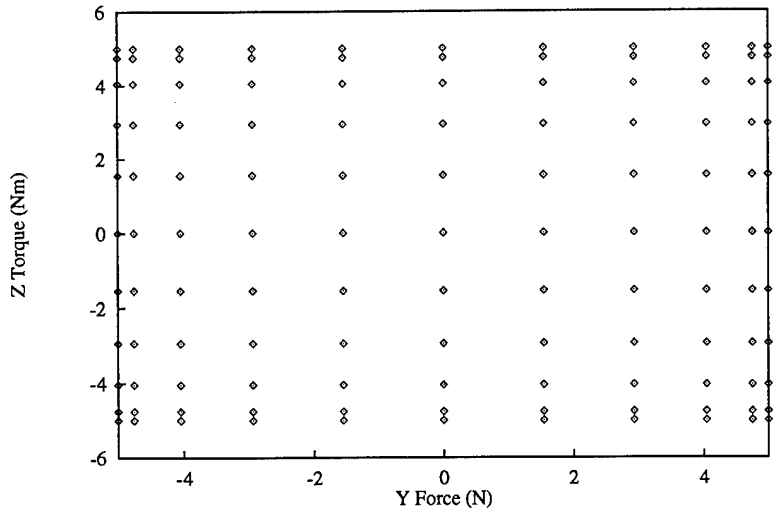



Figure 6.8 Two-dimensional projection of data distributed according to the *sine* spacing function ($y = \sin(x)$).

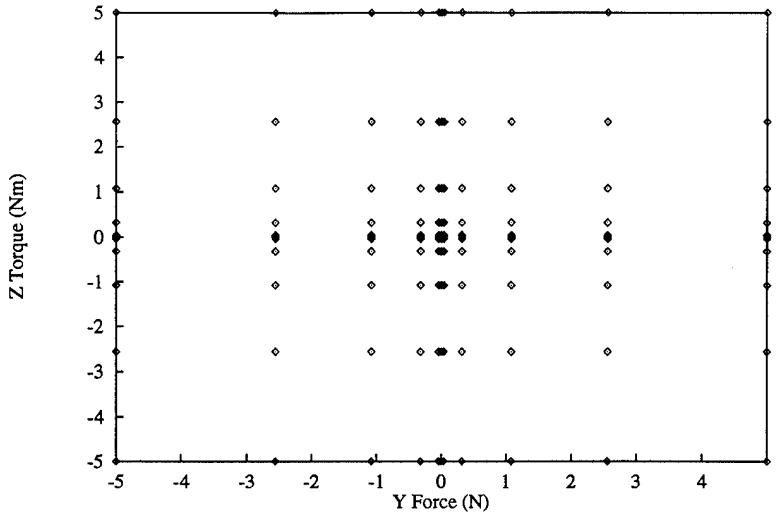



Figure 6.9 Two-dimensional projection of data distributed according to the *cubic* spacing function ($y = x^3$).

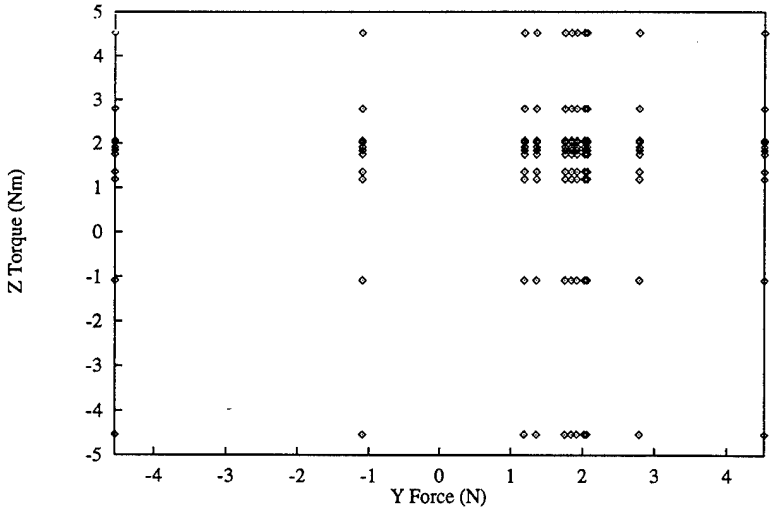



Figure 6.10 Two-dimensional projection of data distributed according to the *complex* spacing function ($y = 0.88\,[(\sin x + 0.3)\cos 10x \tan x + 0.46]$ ).

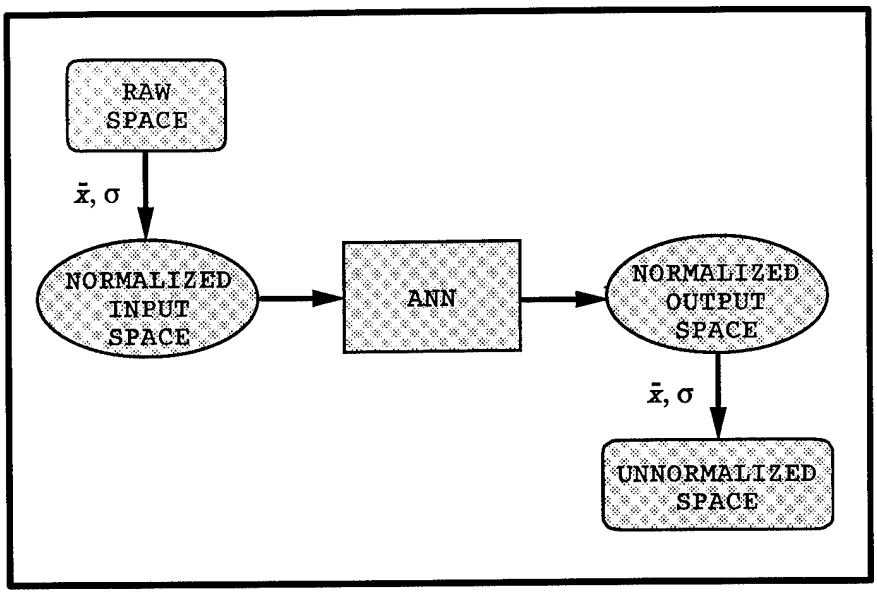



Figure 6.11 Illustration of modified input and output spaces resulting from using gaussian normalization on the training data.

underdeveloped in the important region near zero, which might yield poor results. In the worst case, the bias might cause the training data to exclude the region of operation and the ANN will attempt extrapolation when implemented as a controller. Note that this is a concern that is particularly evident with training data generated with the 'complex' spacing function as shown in Figure 6.10.

Another possible problem that can occur when the $\bar{x}$ and $\sigma$ of the measurement stream are appreciably different than that of the training data set is exacerbated when gaussian normalization is used during training. The convergence of the back-propagation training algorithm is significantly enhanced when each component of the input and output feature vectors is gaussian normalized over the entire training data set. The effect of the normalization is to group the data near zero where the sigmoid function is most linear. This expedites the error convergence and improves the robustness of training. However, if training is conducted with normalized data, then the resulting ANN has learned a modified mapping from a normalized input space to a normalized output space, as depicted in Figure 6.11.

This requires the measurement stream of data to be transformed into the normalized input space before presentation to the ANN, and the output must be un-normalized to

bring it out of the normalized output space. These transformations are done using the same $\varpi$s and $\sigma$s that were derived for the training data. The problem arises when the normalization statistics of the training data are incorrect for the measurement stream that the ANN controller encounters when implemented. A particular problem occurs for the edge-mating task when the training data have a large non-zero $\varpi$. For the typical successful controller performing the edge-mating task, the measured forces are small-valued, which can be approximated as a zero $\varpi$. When the large non-zero $\varpi$ of a biased training data set is subtracted from the force measurements to normalize the data for presentation to the ANN, it can throw the input value out of the range occupied by the training data and thereby cause the ANN to attempt extrapolation instead of interpolation. Since the statistics of the measurement stream cannot be known with certainty prior to the implementation, it is difficult to ensure that the training data will match the measurement stream in terms of those statistics. This is a serious limitation of applying an ANN controller in the architecture used for this research and it remains a difficult problem with which to cope.

*6.2.2 SISO Training Data Distribution Investigation.* The results of exploring the effects of the various distribution parameters on the best obtained performance metric are summarized in Figures 6.12 through 6.16. These surface plots depict the best metric obtained from four or more attempts at training on the data. The horizontal axes describe the combination of parameters used to generate the training data, and the vertical axis reflects the best performance metric value obtained from among all the ANN controllers trained on a given combination of parameters. Since all the metrics are measures of the controller's performance relative to the accommodation matrix controller, exact reproduction of the accommodation matrix controller's performance would yield a $\zeta$=1. Thus, if all the controller configurations were able to learn the task as well as the original matrix controller from which they were trained, the surface depicted in these figures would be flat sheets at $\zeta$=1. In some cases that the reader will encounter later, none of the controllers were able to successfully complete the task when trained on a particular set of training data. For these cases, a value of $\zeta$=0 was artificially assigned to the controller since that is the only unique value which can be assigned as a flag for failure. As a consequence, however, these failed cases show up as large dips in the shape of the surface plot which go well

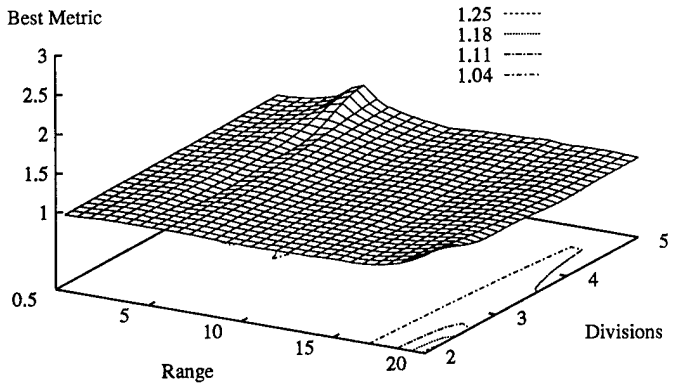



Figure 6.12 Performance metrics of controllers trained on *evenly-spaced* SISO data as a function of the range and number of divisions.

below $\zeta$=1. Although there were some controllers that succeeded in bettering the baseline matrix controller by some margin, no controller ever achieved $\zeta$<0.7. Therefore, small dips below $\zeta$=1 are representative of very successful controllers while valleys in the surface which approach $\zeta$=0 are indicative of failed controllers. Failed controllers will be emphasized in the text when they appear in the forthcoming figures.

Figure 6.12 shows that controllers trained on evenly spaced SISO data had little trouble learning the task, even when there were only a small set of training vectors used. In the minimum case of two divisions, only 125 training vectors were in the training data set. The small 'bump' in the surface plot where the range is five and the number of divisions is five is not considered indicative of a significant trend.

The results of training the ANN controller using a sparse data population in the region of controller operation (near zero) are depicted in Figure 6.13. When the sine spacing function is used, the training data population is pushed to the outer edges of the given range for the data, as shown in Figure 6.8. This causes the ANN to interpolate across larger point spacings in the region of small magnitude vectors, which happens to be its typical operating region. Figure 6.13 indicates that the ANN was fully capable of learning the mapping from these

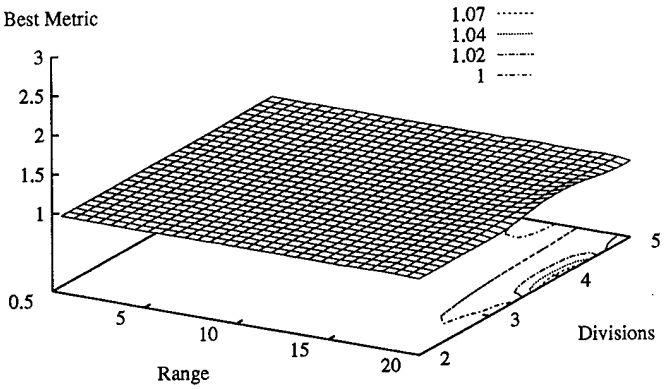



Figure 6.13 Performance metrics of controllers trained on *sine-spaced* SISO data as a function of the range and number of divisions.

data. Not shown in the figure is the fact that it took more and more attempts to properly train the ANN as the range of the training data got larger. In fact, even the intermediate metric results from the intermediate attempts at training the controller had generally increasing values in correlation with the range. This reflects the fact that the interpolation becomes more and more approximate in the region near zero as the points are spread out to cover a larger and larger range. When the training data are clustered with the range of $[-0.5, 0.5]$, even a small number of training vectors were sufficient for the ANN to learn the mapping with only a few tries.

The cubic spacing results shown in Figure 6.14 further reinforce our understanding of how a coarse interpolation grid can affect controller performance. In contrast to the sine spacing function, the cubic spacing function concentrates the training vectors near zero, which provides a higher density interpolating grid in the region of operation. Consequently, we do not experience the trend of increased training difficulty with increasing range as was observed in the sine spacing results. Instead, the results were essentially the same as those for the evenly-spaced data.

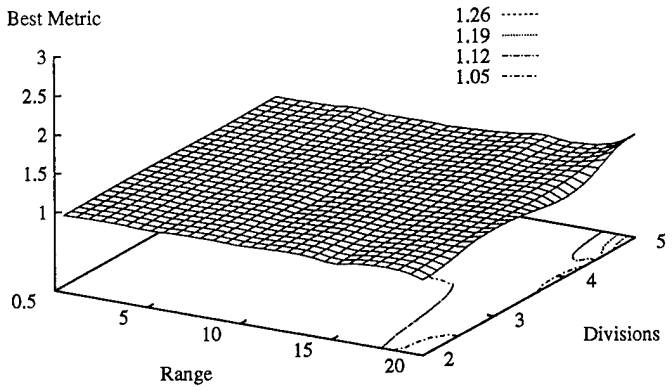



Figure 6.14 Performance metrics of controllers trained on *cubic-spaced* SISO data as a function of the range and number of divisions.

Perhaps the most difficult SISO data for the ANN to learn the ACC mapping matrix from was the training data generated using the complex spacing function. In Figure 6.15 we observe that the trained controller consistently failed to complete the task properly when it was trained on complex data having a small number of training vectors. Failures are indicated by performance metrics of zero which appear as large dips below $\zeta$=1 on the surface plot in Figure 6.15. The "humps" along the divisions=2 edge of Figure 6.15 are artifacts of the surface fitting routine used to plot the data. In fact, there were no controllers trained on data having a range other than at 0.5, 5, 10, 15, and 20. Therefore, we determine that only in the case where the range of the data was very small (i.e. $\pm$0.5) did the controller properly learn the task from a limited number of training vectors. This again reinforces the conclusion that training data must not only reflect the proper I/O mapping (i.e. consistent matrix), but it must also properly cover the input space (i.e. correct distribution).

In the case of the complex-spaced data, they are not only unevenly spaced, as indicated in Figure 6.10, but they also have a non-zero mean, $\varpi$, value for each vector component. The significance of non-zero $\varpi$ was mentioned in Section 6.2.1. The non-zero $\varpi$ becomes a problem because the measured forces for the edge-mating task are inherently small and

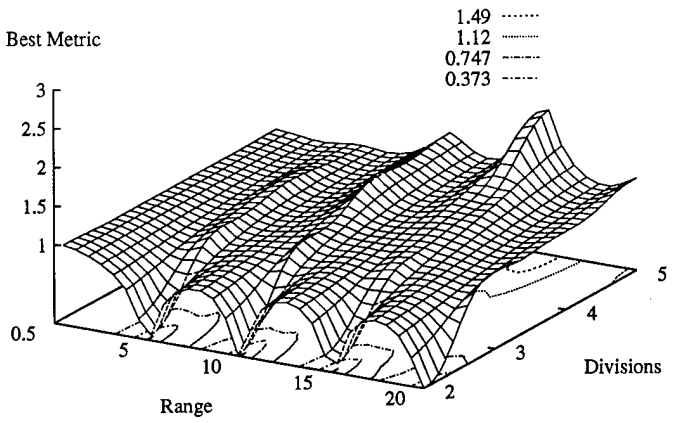



Figure 6.15 Performance metrics of controllers trained on *complex-spaced* SISO data as a function of the range and number of divisions.

centered near zero. Normalizing those small readings with an improper $\varpi$ that is relatively large causes the readings to be skewed off into the less developed part of the I/O mapping learned by the ANN. As Figure 6.15 shows, a large number of training vectors can be used to overcome this problem by making sure that even the less-developed parts of the I/O mapping have sufficient samples to construct the relationship.

Another approach for utilizing training data that have a non-zero $\varpi$ is to augment the data set so as to force a zero $\varpi$. A simple way of augmenting the data set is to mirror the original training vectors about all the axes. Specifically, each component of each original training input vector is negated, one at a time, along with its corresponding output vector component. This causes the number of training vectors to increase by a factor of eight, but the range of maximum to minimum values on each axis is unchanged. Several of the SISO complex-spacing training data sets were mirrored, trained and tested. Figure 6.16 shows the resulting improvement as compared with Figure 6.15. Note that there were no mirrored data tested for any combinations with a range of 0.5, and those combinations were assigned zero performance metrics to complete the surface plot of Figure 6.16 in a manner consistent with Figures 6.12 through 6.15. It is also important to remember that, although the number

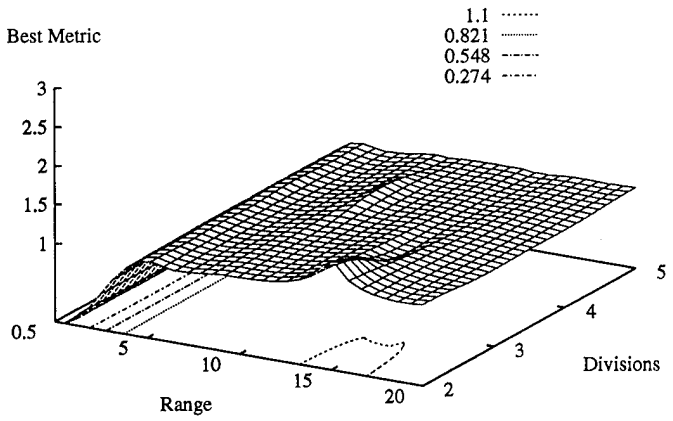



Figure 6.16 Performance metrics of controllers trained on *complex-spaced* SISO data after being *mirrored* about all axes as a function of the range and number of divisions. Note that no controllers were trained on data with a range of 0.5, so they are artificially assigned zero performance metrics.

of training vectors in the data set was larger by a factor of eight for the mirrored data, the number of training iterations through the data set was reduced by a factor of eight so as to maintain a constant number of total training vector exposures for the ANN.

To be sure that the additional number of training vectors was not the reason for the good performance for controllers trained on mirrored data, the mirrored data sets were subsampled to reduce their sizes by a factor of eight. These subsampled, mirrored training data files were the same size as the original complex training data files before they were mirrored. Figure 6.17 shows the results for the ANN controllers trained on those smaller data files. The results are essentially identical to those for data prior to subsampling, which shows that it is not the added number of exemplar vectors that improved the performance of the controllers. The improved performance in Figures 6.16 and 6.17 confirms the mirroring technique as a valid way of modifying the training data statistics without collecting any additional data. The success of mirroring on the SISO data is a prelude of the significant success it will demonstrate when used with the DIDO data, as will be discussed in Section 6.5.2 later.

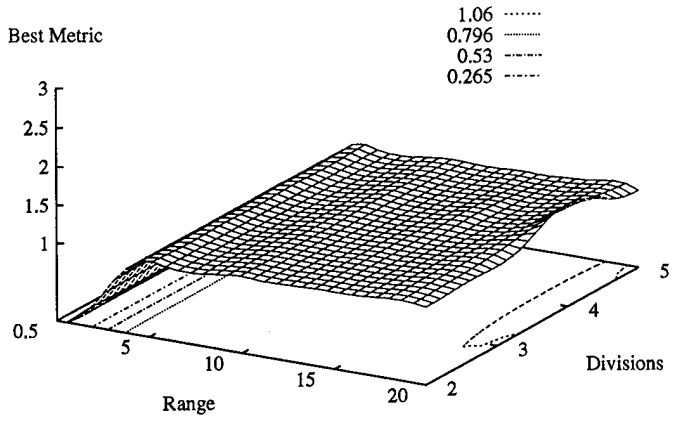



Figure 6.17 Performance metrics of controllers trained on *complex-spaced* SISO data after being *mirrored* about all axes and *subsampled* back to their original size. The results are plotted as a function of the range and number of divisions.

*6.2.2.1 LSMF Technique.* As was described in Section 5.4.3, the LSMF technique is intended to extract the best-fit matrix that a set of trained ANN weights has learned to emulate. To check the technique's validity, it was first applied to I/O pairs of training data. Since the training data were perfect examples of the $\mathcal{A}_a$ mapping, there should be little, if any, difference between the fitted matrix, $\mathcal{A}_a'$, and the original $\mathcal{A}_a$. This was, in fact, found to be universally true for SISO training data. All four similarity indexes described in Section 5.4.4 were identically zero when fitted to the SISO training data.

Having used the SISO training data to validate the method, the LSMF technique was then applied to the weights from several ANN controllers trained on evenly-spaced SISO data sets. An example of the results is shown in Figure 6.18 for an LSMF fitting window size of 200 samples. Although Figure 6.18(a) and (c-d) shows that the ANN learned the structure of the $\mathcal{A}_a$ including the ratio between the non-zero elements, it did not properly learn their overall magnitudes. This is indicated by Figure 6.18(b) and was a common result when interrogating SISO-trained ANN controllers.

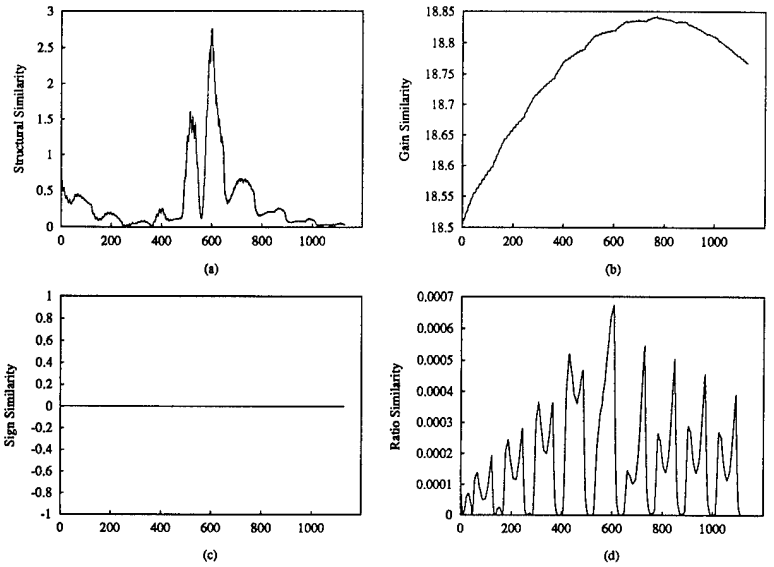



Figure 6.18 Matrix similarity indexes for a series of $\mathcal{A}_a{}'$ fitted to the mapping from the weights trained on evenly-spaced SISO data. Shown is (a) structural similarity, (b) gain similarity, (c) sign similarity, and (d) ratio similarity. Window size of fitting was 200 samples.

It is important to note that the results in Figure 6.18 can be a function of several parameters including the window size for the LSMF routine. Figure 6.19 shows how the similarity indexes differed when using a fitting window of 600 as compared to Figure 6.18 which used 200. Note that the shape of the structural similarity index, $\Upsilon_s$, curve is significantly different in Figure 6.18(a) and that the peak value of the $\Upsilon_s$ is slightly less. Figure 6.18(d) also shows considerable variability in the ratio similarity index, $\Upsilon_r$. The sign similarity index, $\Upsilon_\pm$, and the gain similarity index, $\Upsilon_g$, were mostly unaffected by the window size used for the LSMF technique, as shown by comparing Figures 6.18(b) and (c) with Figures 6.19(b) and (c), respectively. This was found to be true for all of the ANN controllers trained on SISO data which were evaluated.

It is worth noting that when fitting the original training data, the window size dependency does not exist because all of the data reflect a perfect mapping; constant similarity indexes result regardless of the window size used. When fitting I/O data taken via the trained ANN controller, however, the mapping may not be perfectly linear, and it affects the LSMF results as shown. Although one might be concerned about the task of choosing

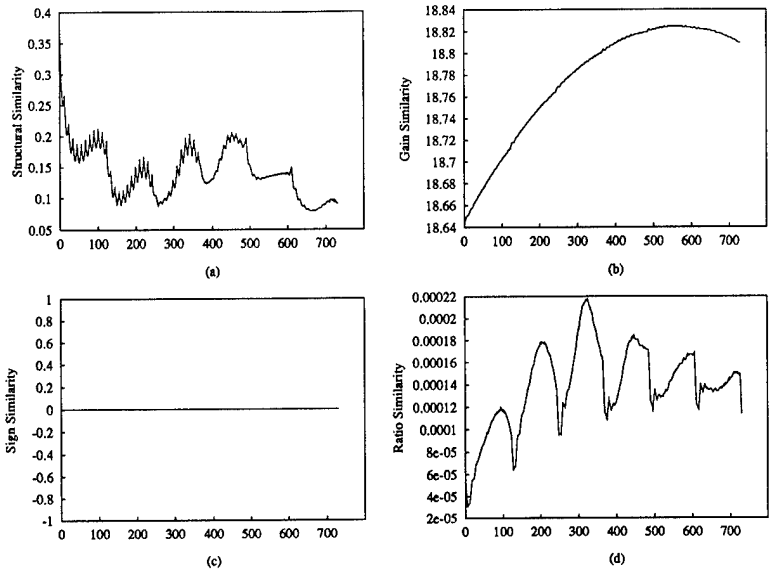



Figure 6.19 Matrix similarity indexes for the same data as in Figure 6.18, but fit with a window size of 600 samples.

a proper window size, in the context of SISO data it is difficult to imagine a criterion for choosing anything less than the full data set available. There are at least two reasons for this: 1) since the ANN was trained from a static matrix relationship, there is no reason to look for trends indicating changes of strategy; and 2) the more samples included in $\mathcal{F}$ and $\mathcal{V}$, the better the fit[1].

The reason for highlighting this window-size dependency is to illuminate the fact that, if the mapping is linearly inconsistent, the results of the LSMF technique are affected by the size of the window. Window-size dependency is a common feature of many signal processing techniques such as a short-time Fourier transform. It is not considered to be a problem, but, rather, just another parameter of control when evaluating the trained ANN s. In fact, we hope to use it to identify a shift in the accommodation strategy during a particular execution of a task. This will require the data to have a valid time base (or causality) to them, which is the case only for the RISO, DISO and DIDO data. Therefore, we will investigate its use

[1]This assumes that all the samples represent a consistent mapping with only small errors. If the data are not consistent, then increasing the window size may simply add to the confusion, thereby yielding no improvement.

as a means to evaluate DIDO training data sets for consistency prior to ANN training in a later section.

As was mentioned in Section 6.2.1, matching the normalization statistics of the measurement data when the controller is implemented to the statistics of the data used for training is important. For the same reasons, matching statistics is also important for getting a good $\mathcal{A}_a{}'$ when interrogating a trained ANN because the $\vec{V}$ contained in $\mathcal{V}$ of Eq (5.37) are, in fact, computed outputs. All the considerations presented in Section 6.2.1 therefore apply to finding a good $\mathcal{A}_a{}'$. For the ANN to interpolate well requires a well-developed mapping, and that results from exposing it to a sufficiently rich set of training data. If the ANN is well-trained with a sufficiently rich set of data, then the LSMF technique will yield a good $\mathcal{A}_a{}'$ for any representative set of $\mathcal{F}$ and $\mathcal{V}$. Fortunately, this limitation does not reflect badly on the LSMF technique, but, instead, reiterates a known limitation of the ANN due to training data normalization.

The next investigative step in exercising the LSMF technique was to apply it to the whole SISO data set to get an overall $\mathcal{A}_a{}'$ which could then be implemented as an ACC matrix controller. For evaluating the success of this approach, the performance of the trained ANN controller was used as a comparative benchmark. The hope was that the extracted ACC matrix controller would behave similarly to the trained ANN controller. For the example shown in Figure 6.18, the ANN controller achieved a performance metric, $\zeta$, of 0.985 and the simulation plots looked almost indistinguishable from those of the original $\mathcal{A}_a$ shown in Figure 6.4. When the LSMF technique was applied to the whole data set, the overall $\mathcal{A}_a{}'$ extracted was:

$$\mathcal{A}_a{}' = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.6327 & -0.0003 \\ -0.2656 & 0.0415 & 632.83 \end{bmatrix} \tag{6.3}$$

which has a structural similarity index of $\Upsilon_s$= 0.0723, a gain similarity index of $\Upsilon_g$= 187,342.2, a sign similarity index of $\Upsilon_\pm$= 0.0, and a ratio similarity index of $\Upsilon_r$= 0.0422.

We observe that the magnitudes of the $(2,2)$ and $(3,3)$ elements of this $\mathcal{A}_a{}'$ were 3.16 times larger than in the original $\mathcal{A}_a$. That higher overall gain caused a noisier position trace, force spikes as shown in Figure 6.20, and a performance metric, $\zeta$, of 3.584. Thus, we

see the direct influence of having too large a controller gain on the performance. This was further validated by setting all the off-diagonal elements of $\mathcal{A}_a{}'$ to zero and confirming that the controller performance was essentially unchanged. The results shown in Figure 6.20 indicate that the extracted $\mathcal{A}_a{}'$ is not necessarily equivalent to the ANN as a controller. Additional tests indicated that $\mathcal{A}_a{}'$ was always similar in performance, but never quite as good as the ANN controller. However, it is important to note that this $\mathcal{A}_a{}'$ was able to successfully substitute as an ACC controller.

Given that $\mathcal{A}_a{}'$ seems to be an approximate representation of the strategy learned by the ANN, it seems reasonable to expect that accommodation matrix controllers using the $\mathcal{A}_a{}'$ might have the same trends in $\zeta$ as the ANN controllers from which they are extracted. If this were true, it would imply that the nonlinearities contained in the ANN are not utilized when the ANN is controlling the peg. To investigate this, the $\mathcal{A}_a{}'$ extracted from the various ANN controllers were implemented and their performances were compared to those of their respective controller using $\zeta$.

Owing to the fact that the performance metrics for the trained ANN controllers were quite good, the original performance metric plots shown in Figures 6.12 through 6.14 were essentially featureless. In comparison, Figures 6.21 through 6.23 show that the performance of the $\mathcal{A}_a{}'$ degrades tragically as the range increases among the values tested. Even the distinct performance trend shown for controllers trained on the complex-spaced data in Figure 6.15 is not recreated by their corresponding $\mathcal{A}_a{}'$ ACC controllers, as shown in Figure 6.24. We further observe from comparing Figure 6.25 to Figure 6.24 that mirroring the complex data only makes the robustness of the extracted $\mathcal{A}_a{}'$ worse.

Overall, we conclude that the $\zeta$ for the $\mathcal{A}_a{}'$ do not have the same trends as the original ANN controllers when examined in terms of data range and the number of subdivisions within the range. If there is any consistency, it is that the $\zeta$ for the $\mathcal{A}_a{}'$ are affected oppositely by the range when compared to the $\zeta$ for the ANN controllers. The most robust $\mathcal{A}_a{}'$ controllers were those extracted from ANN controllers trained on complex-spaced data. Therefore, we conclude that we cannot use the performance of the extracted $\mathcal{A}_a{}'$ as a prediction of the performance of a given ANN controller.

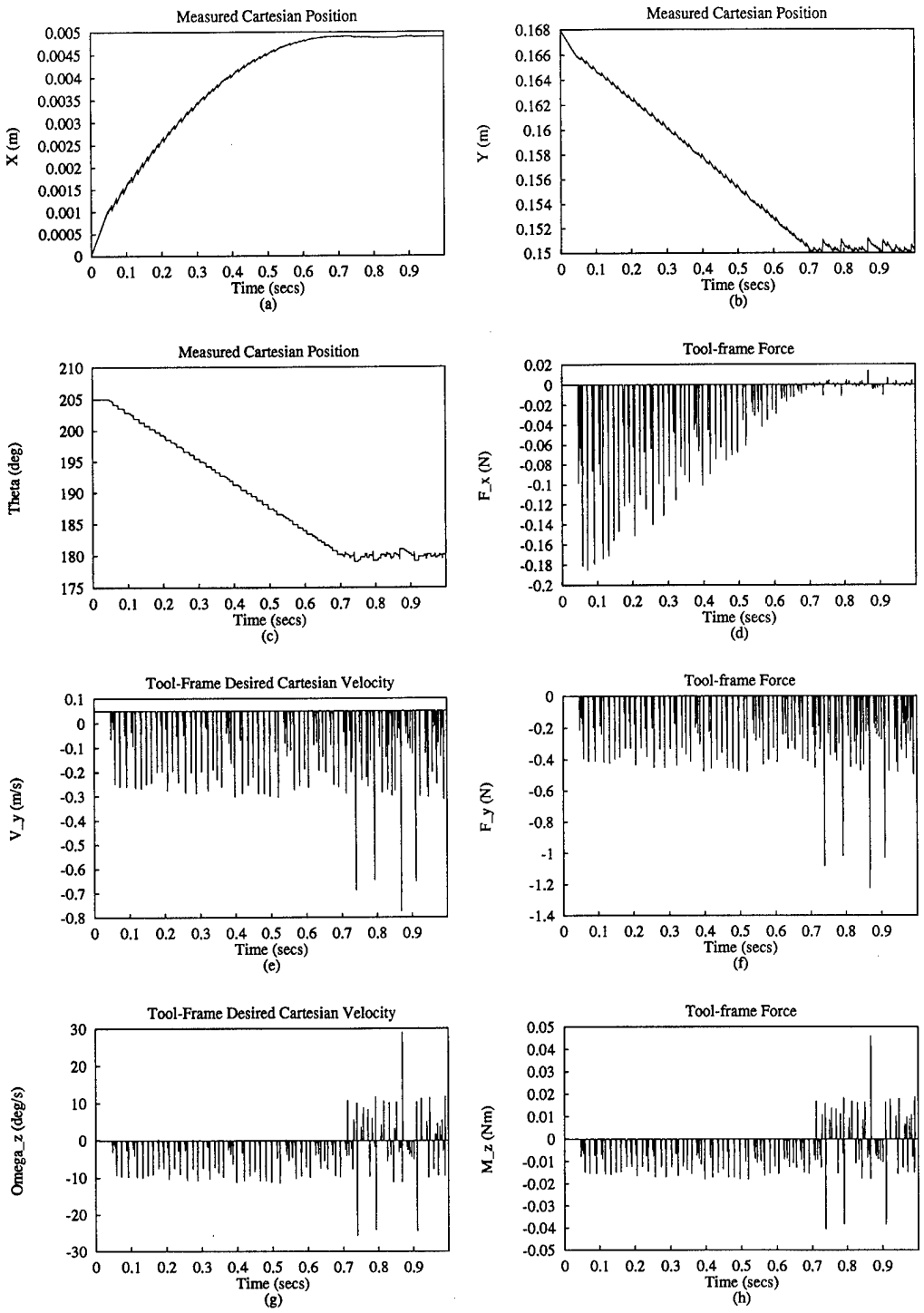



Figure 6.20 Simulation results from implementing the ACC matrix controller derived from the same weights interrogated using the LSMF technique in Figure 6.18. The matrix was derived using the LSMF technique on the entire data set.

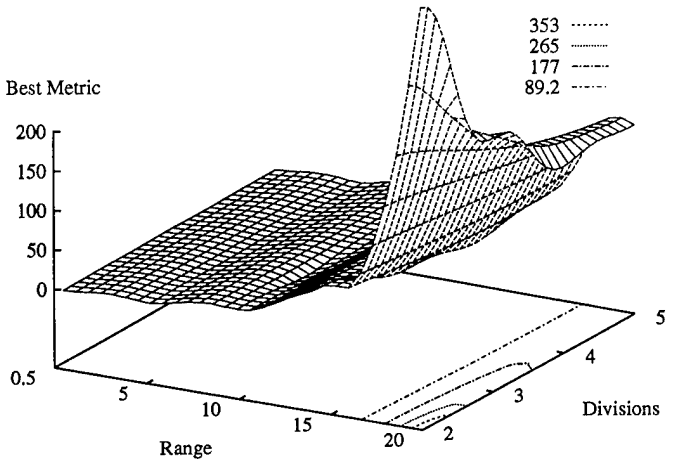



Figure 6.21 Performance metrics achieved by $\mathcal{A}_a'$ extracted from ANN controllers trained on *even*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.12. Note, however, the difference in the scale of the Best Metric axis.

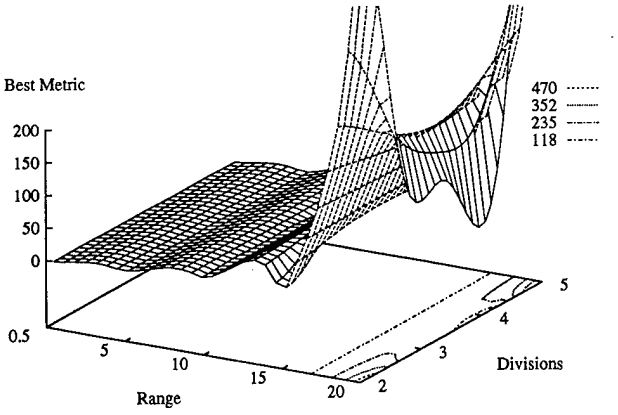



Figure 6.22 Performance metrics achieved by $\mathcal{A}_a'$ extracted from ANN controllers trained on *sine*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.13. Note, however, the difference in the scale of the Best Metric axis.

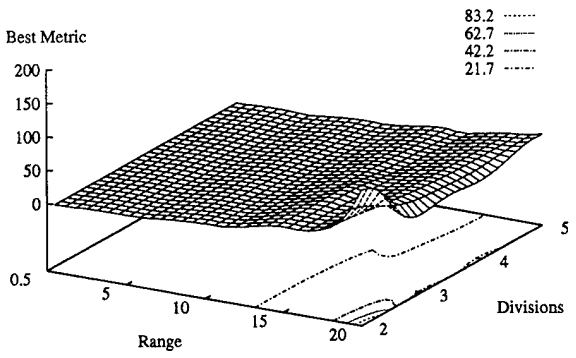



Figure 6.23 Performance metrics achieved by $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *cubic*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.14. Note, however, the difference in the scale of the Best Metric axis.

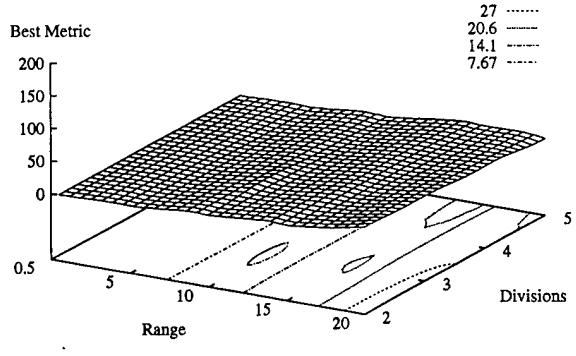



Figure 6.24 Performance metrics achieved by $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *complex*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.15. Note, however, the difference in the scale of the Best Metric axis.

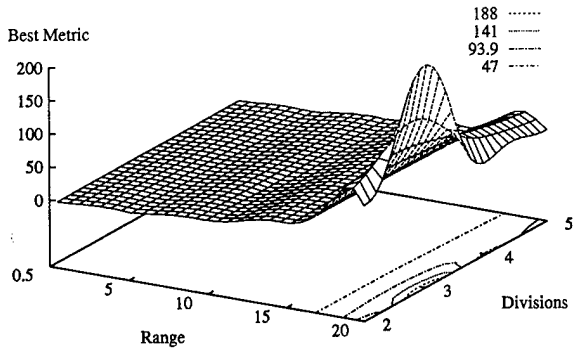



Figure 6.25 Performance metrics achieved by $\mathcal{A}_a'$ extracted using the LSMF technique from ANN controllers trained on *complex*-distributed SISO data after they were *mirrored* about all axes. Data are plotted as a function of range and divisions for comparison to Figure 6.16. Note, however, the difference in the scale of the Best Metric axis.

We are also interested in whether any of the matrix similarity indexes are well correlated to the performance metric, $\zeta$, for SISO data. If so, they could be used to predict the controller performance without actually implementing the controller. It would also provide insight about the sensitivity of $\zeta$ to variations in $\mathcal{A}_a$. To investigate this, we extract $\mathcal{A}_a'$ from some of the ANN controllers whose performances have been presented, compute their similarity indexes, and plot them. For the evenly distributed training data, whose $\zeta$ trends are shown in Figure 6.12, the surface plots of the similarity indexes are shown in Figures 6.26 through 6.28. The sign similarity index, $\Upsilon_{\pm}$, was identically zero for all combinations, so it is not plotted as a figure. $\Upsilon_s$ in Figure 6.26 is roughly similar to $\zeta$ in Figure 6.12, but the location of the only significant feature is at the wrong combination of range and number of divisions. In Figure 6.27, $\Upsilon_g$ is unaffected by subsampling and varies inversely to the range of data covered. In comparison, the best metric plotted in Figure 6.12 does not vary significantly with either subsampling or range, so no good correlation is revealed. However, comparing Figure 6.28 to Figure 6.12 reveals that $\Upsilon_r$ is highly correlated to the performance metric for evenly-spaced, SISO-trained ANN controllers.

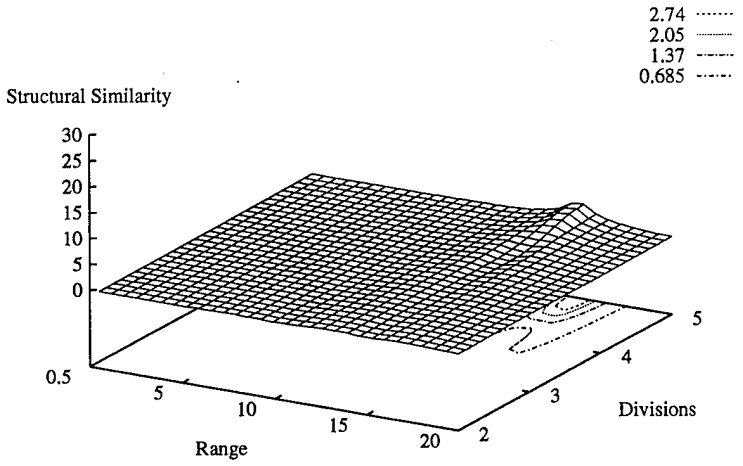



Figure 6.26 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *even*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.12.

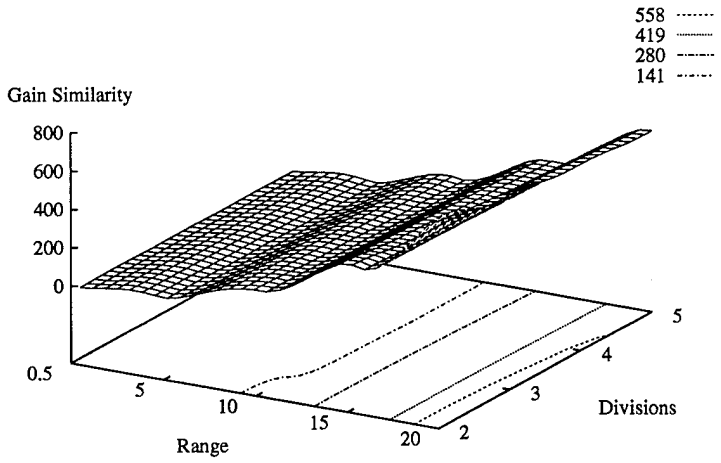



Figure 6.27 Matrix gain similarity index, $\Upsilon_g$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *even*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.12.

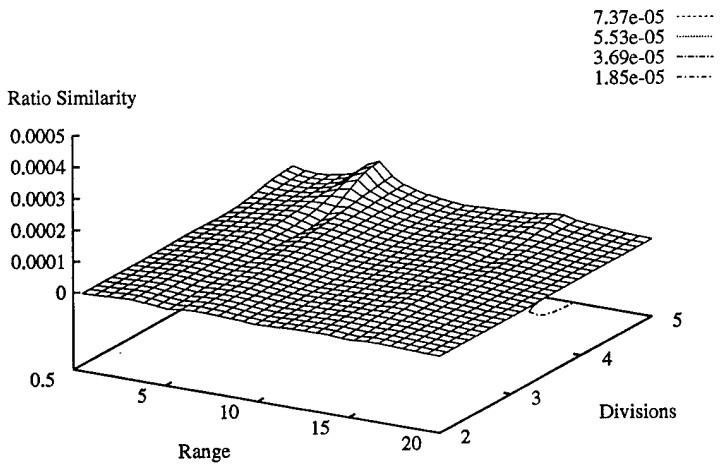



Figure 6.28 Matrix ratio similarity index, $\Upsilon_r$, of $\mathcal{A}_a'$ extracted from ANN controllers trained on *even*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.12.

The $\Upsilon_r$ results for ANN controllers trained on the sine spacing function are shown in Figure 6.31, which should be compared to Figure 6.13. The high correlation between $\Upsilon_r$ and $\zeta$ that was evident in the controller trained on evenly-spaced data is not as strong for the case of sine-spaced data. None of the matrix similarity indexes are well-correlated with $\zeta$ for sine-spaced data. Further, when we examine the remaining matrix similarity plots presented in Figures 6.32 through 6.40, we find no consistent relationship between any of the matrix similarity indexes and $\zeta$ for the controllers trained on SISO data.

There are at least two possible explanations for the lack of correlation between the matrix similarity indexes and $\zeta$ for the controllers trained on SISO data. First, if there is a measure of similarity that is consistently correlated to the metric, it may be a combination of the separate indexes presented. There may be some appropriately weighted sum of $\Upsilon_s$, $\Upsilon_g$, $\Upsilon_\pm$, and $\Upsilon_r$ which would yield a consistent correlation when plotted beside the metric. Secondly, it is very possible that the $\mathcal{A}_a'$ approximation of the ANN is too rough to yield a consistent correlation. We certainly know that the ANN has nonlinear mapping capability at its disposal, and using the linear mapping of $\mathcal{A}_a'$ to represent the ANN controller cannot

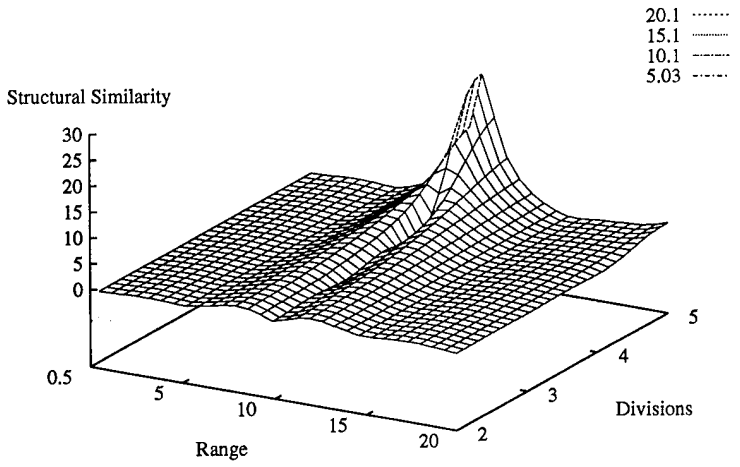



Figure 6.29 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *sine*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.13.

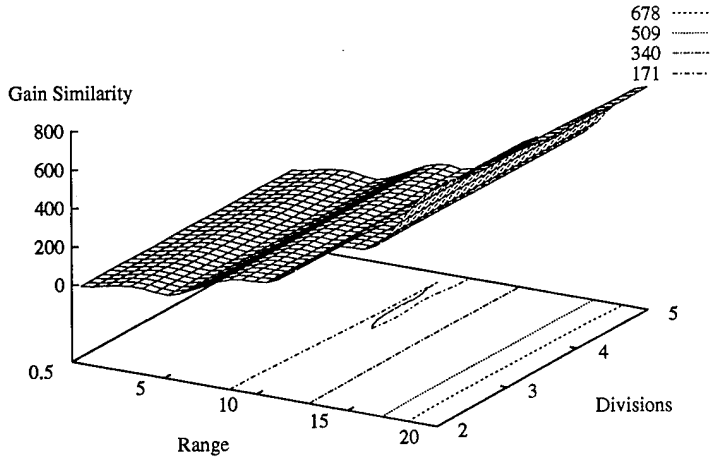



Figure 6.30 Matrix gain similarity index, $\Upsilon_g$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *sine*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.13.

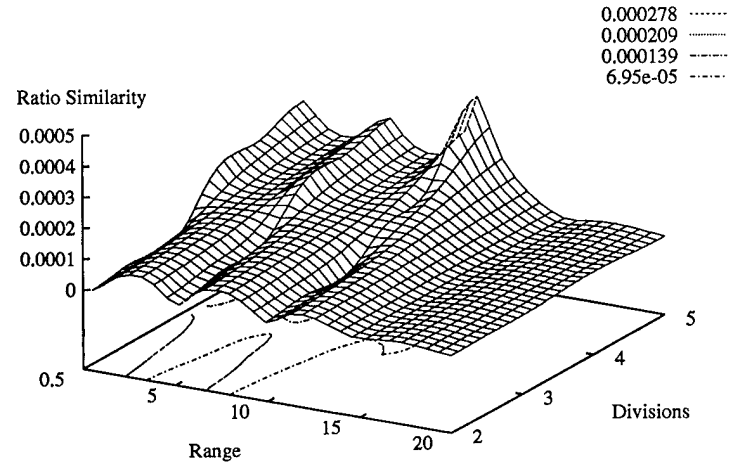



Figure 6.31 Matrix ratio similarity index, $\Upsilon_r$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *sine*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.13.

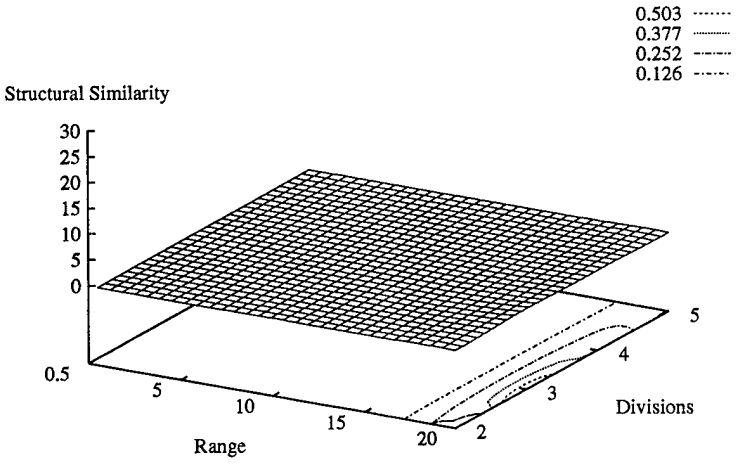



Figure 6.32 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *cubic*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.14.

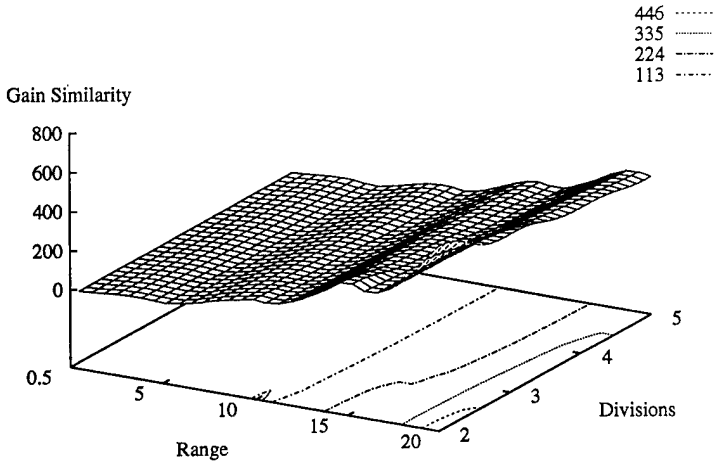



Figure 6.33 Matrix gain similarity index, $\Upsilon_g$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *cubic*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.14.

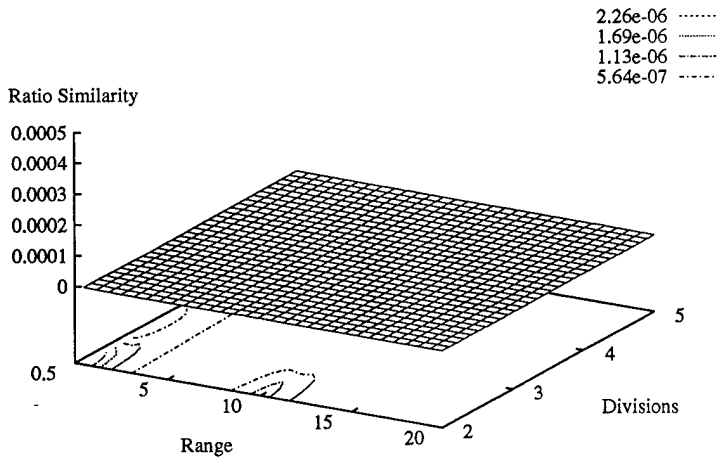



Figure 6.34 Matrix ratio similarity index, $\Upsilon_r$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *cubic*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.14.

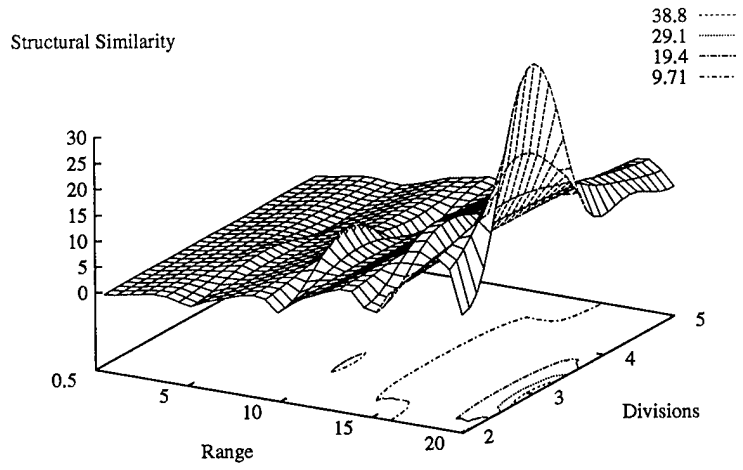



Figure 6.35 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a'$ extracted from ANN controllers trained on *complex*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.15.

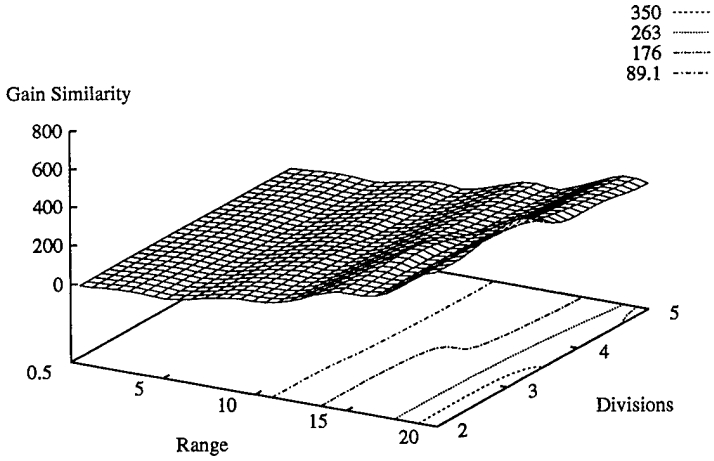



Figure 6.36 Matrix gain similarity index, $\Upsilon_g$, of $\mathcal{A}_a'$ extracted from ANN controllers trained on *complex*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.15.

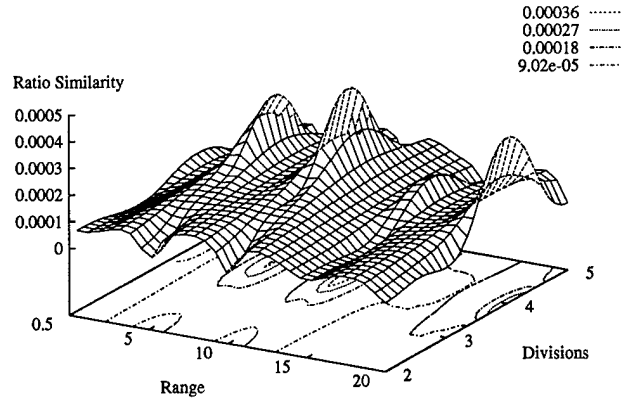



Figure 6.37 Matrix ratio similarity index, $\Upsilon_r$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on *complex*-distributed SISO data using the LSMF technique. Data are plotted as a function of range and divisions for comparison to Figure 6.15.

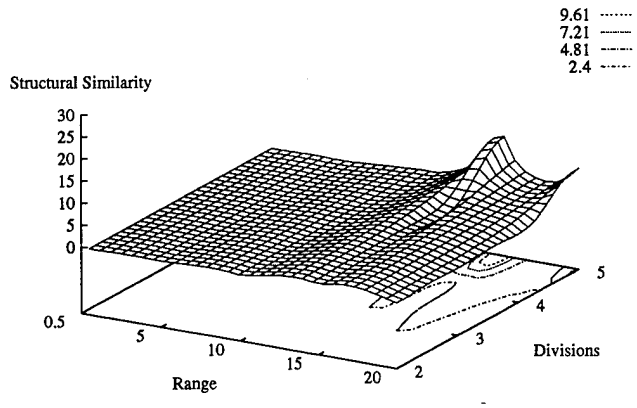



Figure 6.38 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a{}'$ extracted using the LSMF technique from ANN controllers trained on *complex*-distributed SISO data after they were *mirrored* about all axes. Data are plotted as a function of range and divisions for comparison to Figure 6.16.

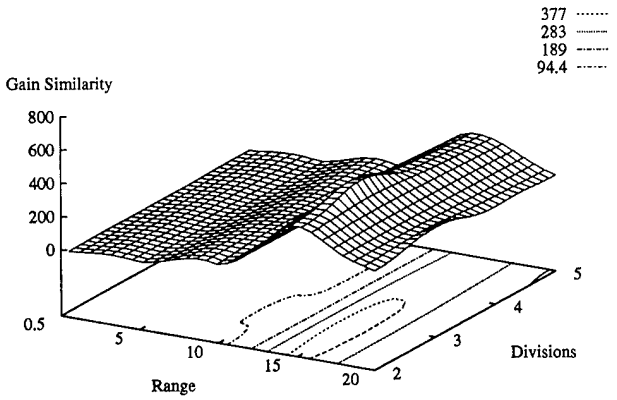



Figure 6.39 Matrix gain similarity index, $\Upsilon_g$, of $\mathcal{A}_a{}'$ extracted using the LSMF technique from ANN controllers trained on *complex*-distributed SISO data after they were *mirrored* about all axes. Data are plotted as a function of range and divisions for comparison to Figure 6.16.

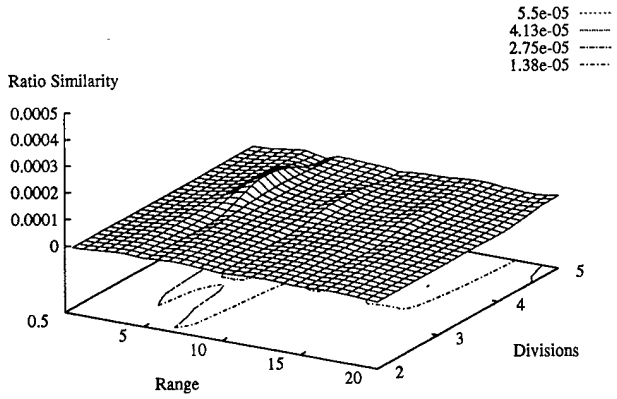



Figure 6.40 Matrix ratio similarity index, $\Upsilon_r$, of $\mathcal{A}_a{}'$ extracted using the LSMF technique from ANN controllers trained on *complex*-distributed SISO data after they were *mirrored* about all axes. Data are plotted as a function of range and divisions for comparison to Figure 6.16.

capture the nonlinear behavior. If the ANN is utilizing its nonlinear capability to perform the task, then its performance may well exceed that of a constant, linear, mapping matrix and the two will never be in full agreement.

Overall, these LSMF results confirm the value of using the LSMF technique to understand how well the ANN has learned to approximate the ACC matrix controller. We pointed out that the testing data used must have normalization statistics which match those of the training data, but recognize this as a universal requirement for ANN s. We have shown that a single LSMF matrix, $\mathcal{A}_a'$, taken from the entire testing data set is only an approximation of what the ANN controller has learned, since the ANN controller is often capable of better performance than the $\mathcal{A}_a'$. We were unsuccessful in our efforts to correlate indexes of similarity between the $\mathcal{A}_a'$ and the original $\mathcal{A}_a$ with the performance metrics of the ANN controllers from which the $\mathcal{A}_a'$ were extracted. However, we demonstrated that the LSMF technique may be used on subsets of a data set. The similarity of the extracted $\mathcal{A}_a'$ might reveal inconsistency in the mapping of the data set before training is attempted. This idea cannot be fully tested using SISO data because there is not time base to these data, so it will be investigated further in a later section.

*6.2.2.2 UVP Technique.* After an ANN completed training, the UVP technique described in Section 5.4.2 was used to extract an accommodation matrix approximation, $\mathcal{A}_a^*$, to the mapping learned by the ANN. The matrix similarity indexes were then used to compare $\mathcal{A}_a^*$to the original $\mathcal{A}_a$. As it turns out, the UVP technique was found to be a very poor method to determine the learned mapping. When applied to the same trained ANN weights that were interrogated using the LSMF technique to yield the $\mathcal{A}_a'$ shown in Eq (6.3), the UVP technique yielded an $\mathcal{A}_a^*$ of:

$$\mathcal{A}_a^* = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 \\ -0.3643 & 0.7317 & -0.3654 \\ -365.36 & -365.09 & 731.05 \end{bmatrix} \tag{6.4}$$

which has a structural similarity index of $\Upsilon_s$= 266,779, a gain similarity index of $\Upsilon_g$= 282,014, a sign similarity index of $\Upsilon_\pm$= 0, and a ratio similarity index of $\Upsilon_r$= 0.7892.

This $\mathcal{A}_a{}^*$ fails miserably as an ACC controller when implemented. Given the success of the ANN as a controller and the $\mathcal{A}_a{}'$ of Eq (6.3) as an ACC controller, this indicates that the UVP technique is not well-suited to interrogating a network to determine its ACC strategy. Some insight into the failure of the UVP method can be gained by comparing the results of two ANN s trained on the same set of training data. In one case, the training data are normalized prior to presentation to the ANN, whereas, with the other case, no normalization is performed. Upon normalizing for training, the normalization statistics become part of the ANN controller, as was discussed in Section 6.2.1. As a feature vector is presented to such an ANN, it is first pre-normalized using the saved statistics from training. This transforms the vector into a normalized input space as illustrated in Figure 6.11. This transformation highlights the inappropriateness of using UVP to extract the emulated matrix from an ANN controller because the original unit vectors are no longer unit length or pointing along the feature axes. Thus, the premise of the UVP method is compromised.

When the training data are not normalized prior to training, the UVP method may yield a better match between the $\mathcal{A}_a{}^*$ and the original $\mathcal{A}_a$, but it is very difficult to successfully train an ANN on un-normalized data. Although they usually converged to some set of weights, they were rarely ever successful as controllers for the edge-mating task.

The poor performance of the UVP method of interrogation is also credited to the fact that it is only a three-point estimate of a high-dimensional, nonlinear mapping function. It is probably unrealistic to expect such a simple, sparsely-sampled approach to yield a meaningful insight to the complicated mapping functional used by the ANN. Thus, we conclude that the UVP method is unsatisfactory for interrogating an ANN, whether it was trained using normalized training data or not.

*6.2.3 SISO Summary.* The SISO data has been used to explore the effects of varying the distribution of the training data on the potential peformance of the ANN controller. The mean value of the input feature vector components was found to have a significant bearing on the success of the trained controllers. This was found to be caused by the mismatch between the normalization statistics which were computed for the training data and those for the stream of measured forces encountered when the ANN controller was implemented,

especially the mean of the data. Because the same normalization statistics are used to pre-normalize the measurement stream as were used with the training data, it is extremely important that the actual statistics are similar. Otherwise, the ANN may be forced to attempt to map from an underdeveloped area of the input space, which can lead to poor results. The mirroring technique was found to significantly mitigate the non-zero mean problem which most affected the controller performance. This was determined to be feasible because the input force magnitude for the measurement stream is characteristically low for the edge mating task.

The LSMF technique for interrogating the ANN did not yield a very reliable estimate of the performance potential of the controller. The validity of the technique for extracting an equivalent accommodation matrix was proven by applying it to SISO training data. However, when interrogating I/O pairs generated by testing an ANN controller, the results were significantly different. The UVP interrogation technique was found to be less informative that the LSMF technique and it was summarily abandoned as a result.

### *6.3 RISO Observations.*

As mentioned in Section 5.1.2, there were two sources for collecting the RISO data, the PUMA manipulator and the simulation. Figure 6.41 shows a typical-looking RISO raw data file collected via simulation. ANN controllers trained on either simulated or PUMA-based RISO data had little difficulty in completing the edge-mating task. Figures 6.42 and 6.43 show data captured on the PUMA manipulator while it was under the control of a RISO-trained ANN controller. The performance is very similar to that of the accommodation matrix controller on the PUMA, as was depicted in Figures 6.3 and 6.2. These results decisively prove the capability of the ANN controller to operate on real hardware when trained on RISO data, but it does not give us much insight about the sensitivity of the controller performance to the various data processing options that are under consideration. To gain further insight, the RISO results presented in the following sections were based on raw training data collected via simulation. In the sections that follow, we will examine the influence of various data processing methods on the performance of the RISO-trained ANN controllers and continue our evaluation of the merits and liabilities of the LSMF interrogation technique.

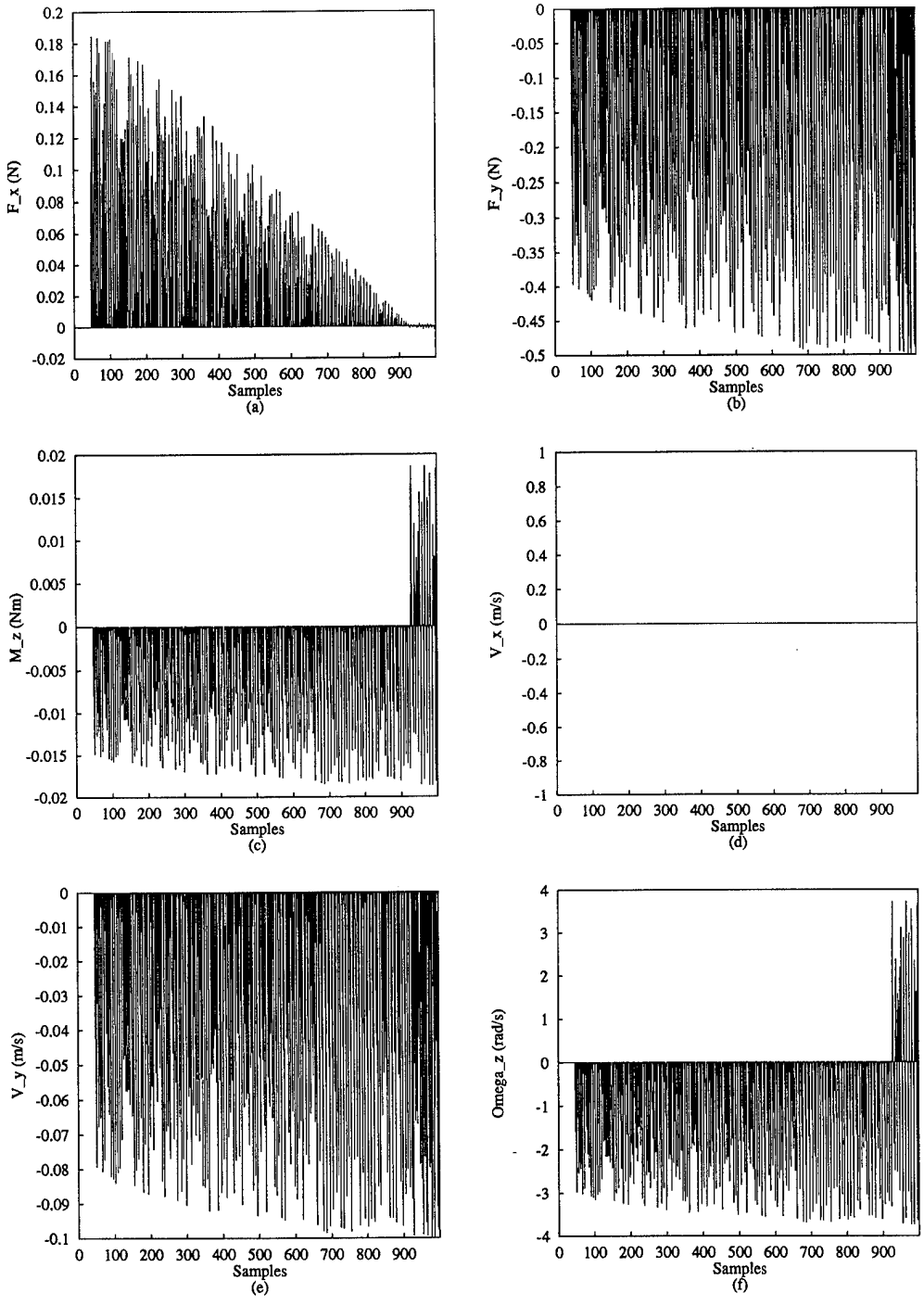



Figure 6.41 Sample of raw RISO force and velocity training data collected via simulation.

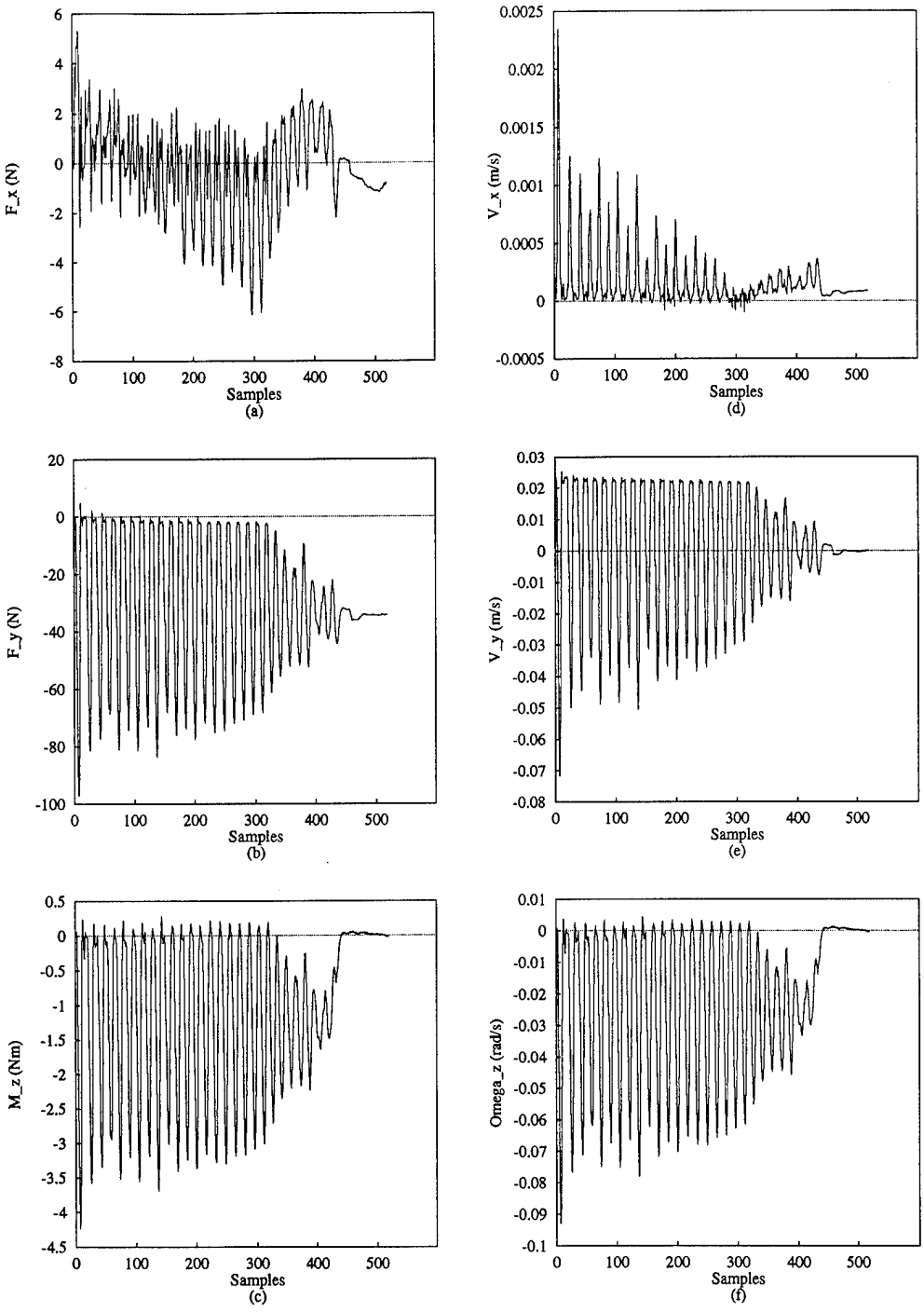



Figure 6.42 PUMA manipulator completing the edge-mating task under the control of an ANN controller trained on RISO data starting from a CCW misalignment angle.

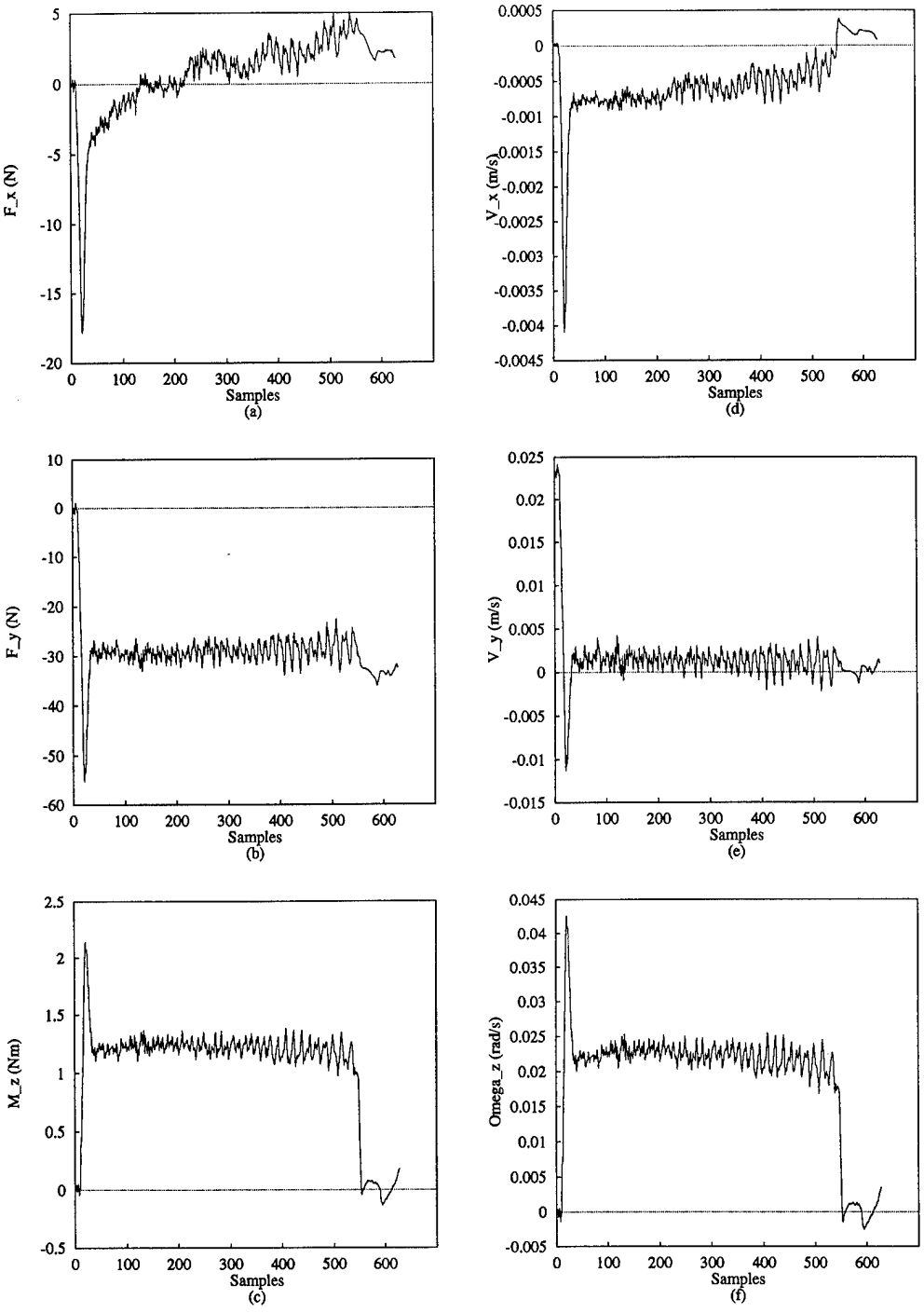



Figure 6.43 PUMA manipulator completing the edge-mating task under the control of an ANN controller trained on RISO data starting from a CW misalignment angle.

*6.3.1 Effects of Data Processing on RISO controllers.* Investigation of the RISO training gave additional insights about how the distribution of the training data affected performance and revealed some of the characteristics of several training data generation procedures described in Section 5.2. After low-pass filtering the data, they look as shown in Figure 6.44. The input clustering of the RISO data was predictably different than that of the SISO data sets. Figure 6.45 shows how a raw set of RISO input data looked as a phase-volume plot. Note that it is remarkably different than the SISO distributions shown in Figures 6.7 through 6.10. The primary difference is that it is clustered in a sub-region of the overall input space that was populated in the SISO data sets. We also note that the character of the RISO data changes when it is low-pass filtered as shown in Figure 6.46, which, of course, affects the distribution statistics. The distribution statistics, in turn, affect the performance of the controller in that they should resemble the statistics of the measured data stream.

*6.3.1.1 Low-pass Filtering and Subsampling.* In contrast to SISO data vectors, which were arbitrarily ordered as the input space was scanned, the RISO data vectors were sequentially related by causality. Thus, they contained a time base and it made sense to attempt some of the time-based data processing steps outlined in Section 5.2. Figure 6.47 shows how subsampling and low-pass filtering affected the controller performance metric, $\zeta$, for RISO data. One immediately notices that the best performance (lowest $\zeta$) occurs when the data are not filtered at all. Before the data are filtered, all of the I/O pairs represent a desired ACC mapping from $\vec{F}$ to $\vec{V}$. When they are low-pass filtered, the individual vector components are modified separately, and this modification corrupts the original ACC mapping contained in the data. Therefore, it is not surprising that the filtered data generated a controller which did worse than before. On the other hand, subsampling had no effect on controller performance. This makes sense in light of the fact that the mapping is unchanged by processing steps that simply decimate the original data file by dropping vectors rather than modifying them. Presumably there is an upper bound to subsampling which would start to cause deteriorative effects if it were exceeded. The bound may have more to do with the total number of vectors passed through than with the frequency of sampling. Determination of that envelope was outside of the scope of the present work.

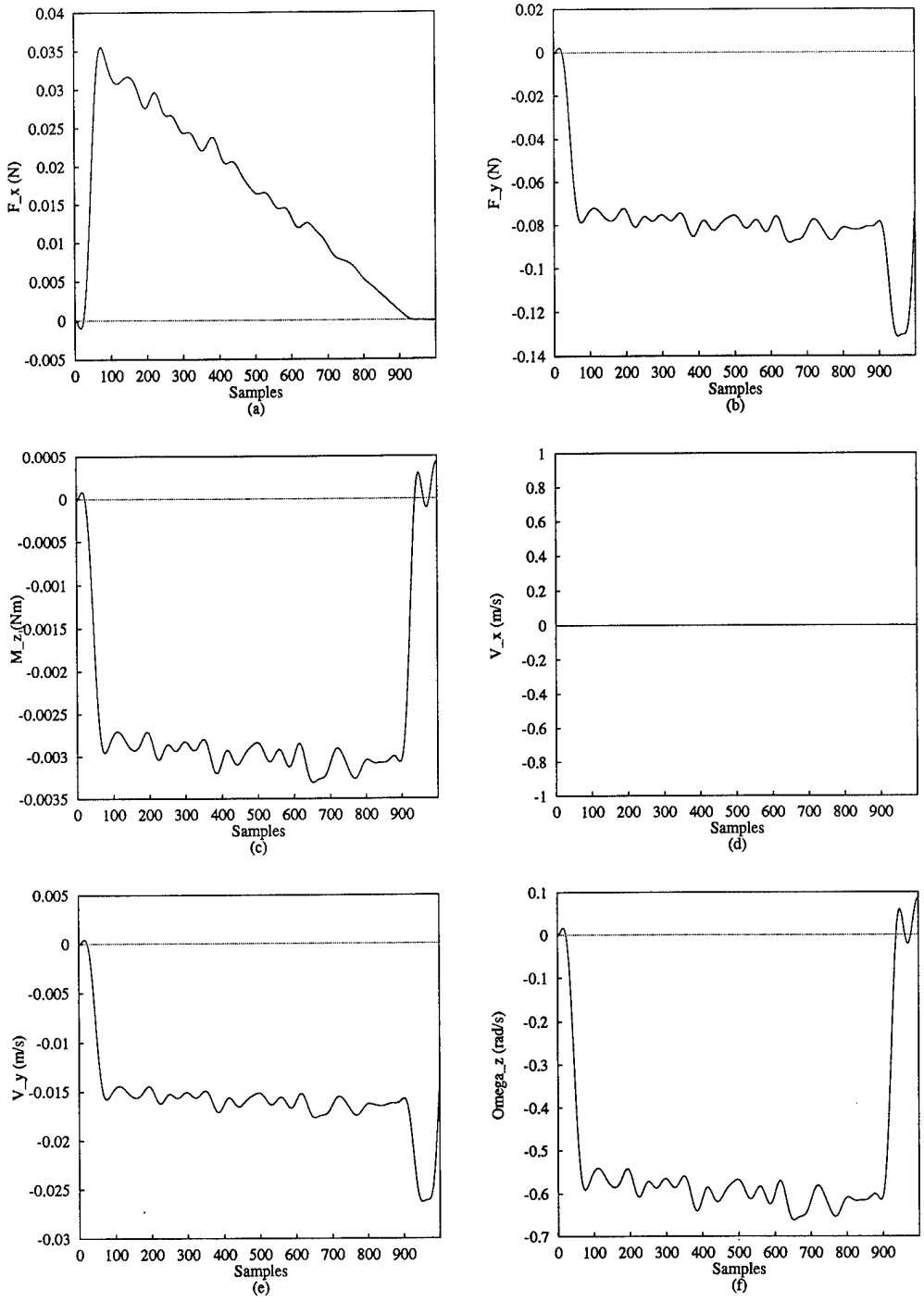



Figure 6.44 RISO data from Figure 6.41 after low-pass filtering it to 1/20th of its original bandwidth.

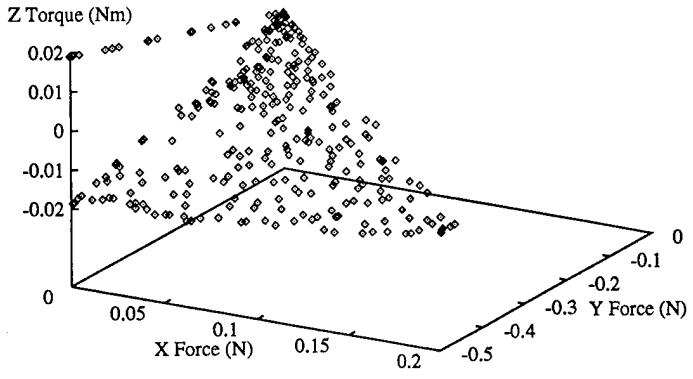



Figure 6.45 Phase-volume diagram showing distribution of the same raw RISO data presented in Figure 6.41.

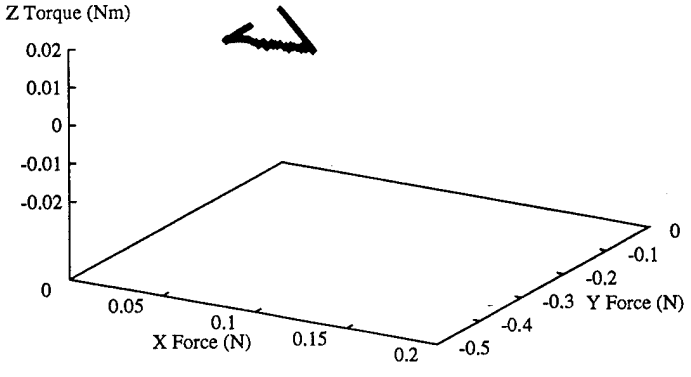



Figure 6.46 Phase-volume diagram showing distribution of the same filtered RISO data presented in Figure 6.44.

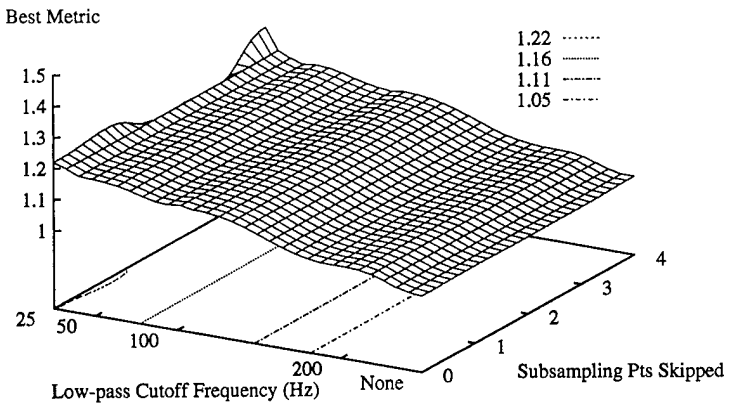



Figure 6.47 Performance metrics of ANN controllers trained on RISO data as a function of the low-pass cutoff frequency and number of points skipped between subsampling intervals.

*6.3.1.2 Velocity Pruning.* As described in Section 5.2.3, velocity pruning was designed to eliminate all I/O pairs in the training data which had $\|\vec{V}\| \leq V_t$. When applied to the RISO data with $V_t$=0.05 meter/second, it had little effect on the performance metric as shown in Figure 6.48, which is consistent with the theory that decimation of a RISO data file does not corrupt the $\mathcal{A}_a$ it presents to the ANN for training. However, if we raise $V_t$ to 0.1 meter/second, the decimation to the RISO data becomes more severe, and it has an impact on the results as shown in Figure 6.49. The increase of $V_t$ introduces the task termination problem mentioned in Section 5.2.3. The higher $V_t$ does this by removing enough samples of low-magnitude output from the RISO data sets that many of the ANN controllers failed to properly learn how to complete the task. The common mode of failure for those cases was controller overshoot of the final aligned position. This was particularly evident when the data had previously been low-pass filtered with a low cutoff frequency. For the data that had been filtered down to 25 hertz (Hz) in Figure 6.49, the controllers failed to properly complete the edge-mating task, and were consequently assigned $\zeta$=0. The failed cases shown in Figure 6.49 were the result of controllers that never learned to halt motion upon alignment and, consequently, they overshot the desired position.

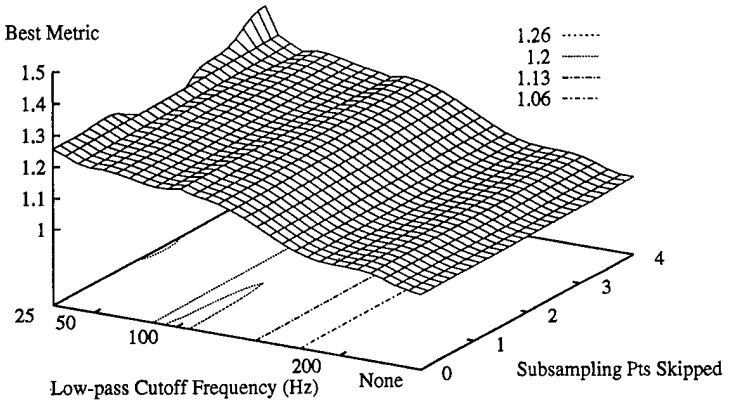



Figure 6.48 Performance metrics of ANN controllers trained on RISO data after velocity pruning with a threshold of 0.05 m/s. Plotted as a function of the low-pass cutoff frequency and number of points skipped between subsampling intervals.

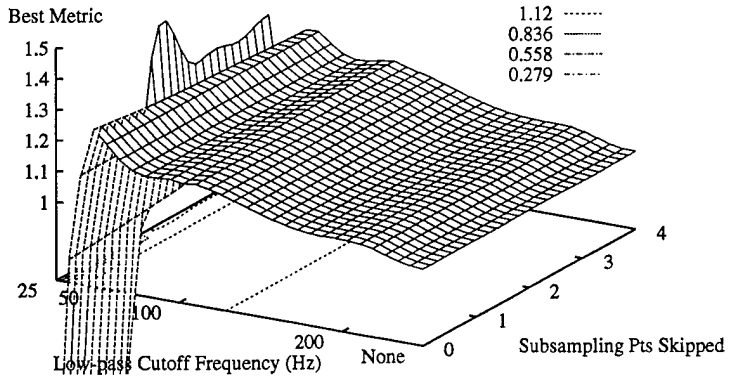



Figure 6.49 Performance metrics of ANN controllers trained on RISO data after velocity pruning with a threshold of 0.1 m/s. Plotted as a function of the low-pass cutoff frequency and number of points skipped between subsampling intervals.

From these data, we conclude that velocity pruning is not very useful if the I/O mapping is of high quality, as is the case for SISO, RISO, and DISO data sets. We will investigate its usefulness for processing DIDO data in Section 6.5.2.

*6.3.1.3 Hemisphere Pruning.* Figure 6.50 shows how the results of Figure 6.47 are changed after each of the training data sets is hemisphere pruned with a cutoff, $\Psi_t$, of 86.5 degrees. There is little change in the best metric obtained for each combination of low-pass filtering and subsampling tested. This indicates that the hemisphere pruning did nothing to enhance or degrade the usefulness of the data sets. Choosing $\Psi_t$=86.5 degrees was determined by iteration to be a "crisp" dividing line between excluding just a few vectors and excluding nearly all of the vectors in the training data. As it turns out, the RISO data have a very consistent included angle, $\Psi$, between the $\vec{F}$ and the $\vec{V}$ of each sample. This phenomenon is discussed further in Section 6.3.2 below. The angle is a function of the ratio between the $(2,2)$ element and the $(3,3)$ element of the $\mathcal{A}_a$ used to compute the $\vec{V}$ from the $\vec{F}$. Therefore, for SISO, RISO, or DISO training data, hemisphere pruning has no beneficial effect because the data are already consistent from the start.

It is interesting to note that, if one uses ${}^T\vec{V}_d$ rather than ${}^T\vec{V}_c$ as shown in Figure 5.8 for the output for RISO training data, the blended superpositioning of ${}^T\vec{V}_n$ with ${}^T\vec{V}_c$ upon contact will severely corrupt the crispness of the $\Psi$ and the consistency of the I/O mapping in the data set. Prior to contact, $\Psi$ is undefined because $\vec{F}$=0. Once $\vec{F}$=2$\epsilon$ is exceeded, the I/O mapping obeys $\mathcal{A}_a$. For the range between $\vec{F}$=0 and $\vec{F}$=2$\epsilon$, however, the superpositioning of ${}^T\vec{V}_n$ and ${}^T\vec{V}_c$ to generate ${}^T\vec{V}_d$ could produce I/O pairs having far different $\Psi$ than those generated by $\mathcal{A}_a$. Thus, it is not by accident that we have defined our RISO and DISO outputs to be ${}^T\vec{V}_c$.

*6.3.1.4 Collision Pruning.* Collision pruning was applied to each of the RISO data sets to determine if it offered any potential to repair the deleterious effects of low-pass filtering. Figure 6.51 shows the results of applying collision pruning with $F_t$=0.05 newtons. There is no significant difference between the performance of the controllers before and after collision pruning. This is not unexpected in light of the fact that collision pruning does not modify the data, but simply removes from the training data file exemplars showing free-

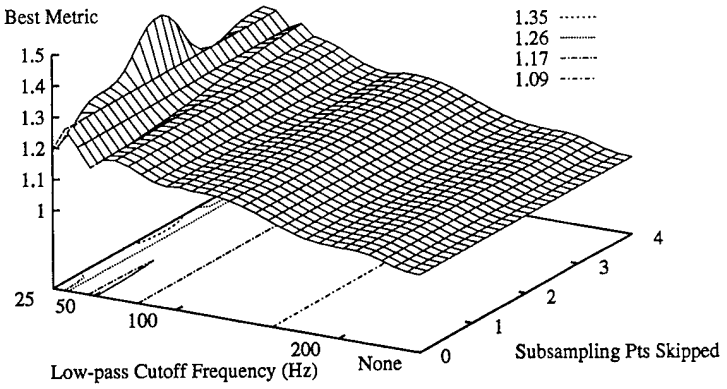



Figure 6.50 Performance metrics of ANN controllers trained on RISO data after hemisphere pruning with a threshold of 86.5 degrees. Plotted as a function of the low-pass cutoff frequency and number of points skipped between subsampling intervals.

space motion which might reflect a different controller strategy. In the case of the original RISO data, the free-space motion is a mapping from $\vec{F}$=0 to $\vec{V}$=0, so the ANN has no problem coping with it. Thus, no benefit is gained from collision pruning RISO data, though it may still prove useful for DIDO data.

*6.3.2 Included Angle Statistics.* The experience with the crisp $\Psi_t$ for hemisphere pruning RISO data described above revealed the fact that most of the raw RISO data had nearly the same $\Psi$. The consistency of the $\Psi$ was a product of the constant $A_a$ and the narrow range of $\vec{F}$ in RISO data used to generate the I/O pairs. Given the tight clustering of the $\Psi$ for the raw RISO data, the effect of low-pass filtering on the clustering of $\Psi$ was investigated. The standard deviation of $\Psi$, $\sigma_\Psi$, for each training data set was used as a measure of how tightly clustered the $\Psi$ were. Figure 6.52 shows how the $\sigma_\Psi$ varied as a function of low-pass filtering cutoff frequency and the number of points skipped between subsampling intervals. Comparing Figure 6.52 to Figure 6.47 reveals a high degree of correlation between the performance metric, $\zeta$, and the $\sigma_\Psi$. In its original raw form (no

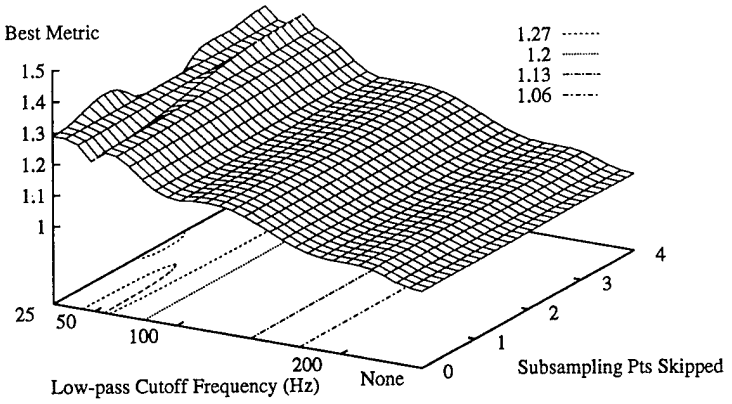



Figure 6.51 Performance metrics of ANN controllers trained on RISO data after collision pruning with a threshold of 0.05 and a window size of 1. Plotted as a function of the low-pass cutoff frequency and number of points skipped between subsampling intervals.

filtering), the consistency of the $\mathcal{A}_a$ mapping is the highest and the $\sigma$ is consequently lowest. As the degree of filtering increases (i.e. lower cutoff frequency), the $\sigma_\Psi$ increases. This suggests that low-pass filtering injects a form of noise into the originally-clean mapping between $\vec{F}$ and $\vec{V}$. This noise, in turn, leads to worse performance for the ANN controller trained on that data. This trend continues to hold true for RISO data which is velocity pruned, hemisphere pruned, or collision pruned as shown in Figures 6.53 through 6.56.

Figures 6.52 through 6.56 confirm our expectation that decimation of a data file by subsampling does not change the $\sigma_\Psi$ significantly. In addition, they show that decimation by velocity pruning, hemisphere pruning, or collision pruning does not influence the $\sigma_\Psi$ either. We determine, then, that these data processing steps have no effect on the consistency of the $\mathcal{A}_a$ mapping between the $\vec{F}$ and the $\vec{V}$.

Although the uniformity of the RISO results shown indicates promise for using $\sigma_\Psi$ as a measure of consistency in the $\mathcal{A}_a$ mapping for RISO data, we realize that the included angle is dependent on the input vector as well as the mapping matrix. If we use a perfectly

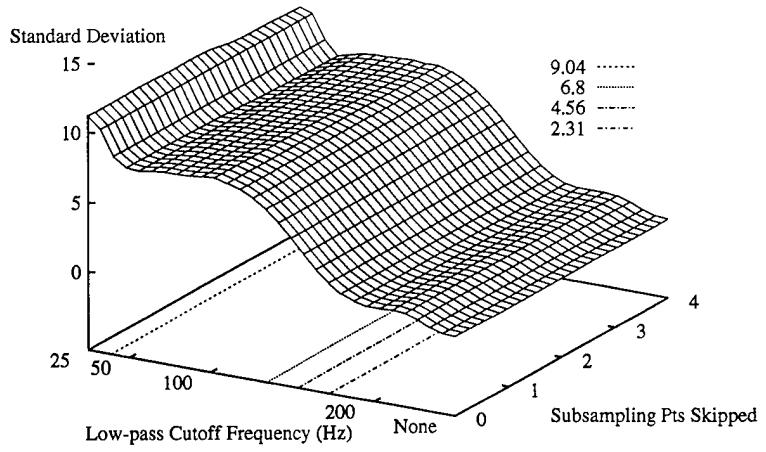



Figure 6.52 Standard deviation of included angle between $\vec{F}$ and $\vec{V}$ in RISO data as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals.

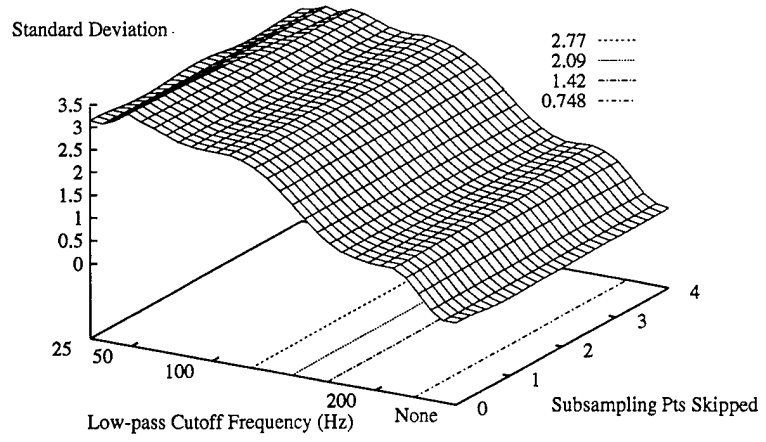



Figure 6.53 Standard deviation of included angle between $\vec{F}$ and $\vec{V}$ in RISO data as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after velocity pruning with a threshold of 0.05 m/s.

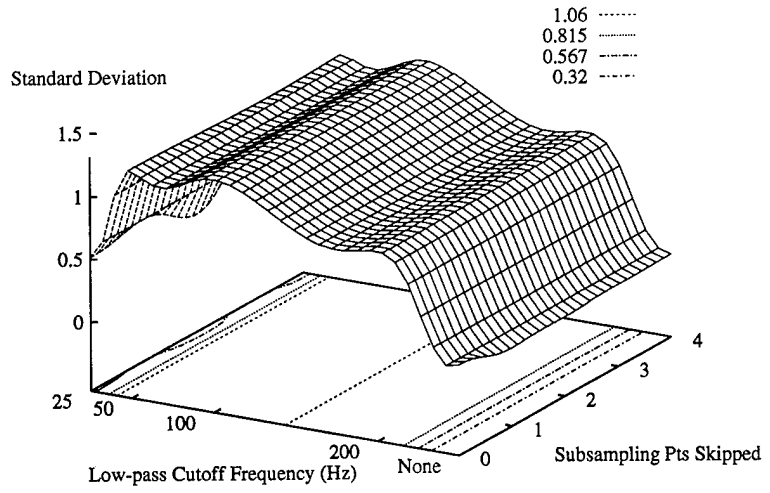



Figure 6.54 Standard deviation of included angle between $\vec{F}$ and $\vec{V}$ in RISO data as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after velocity pruning with a threshold of 0.1 m/s.

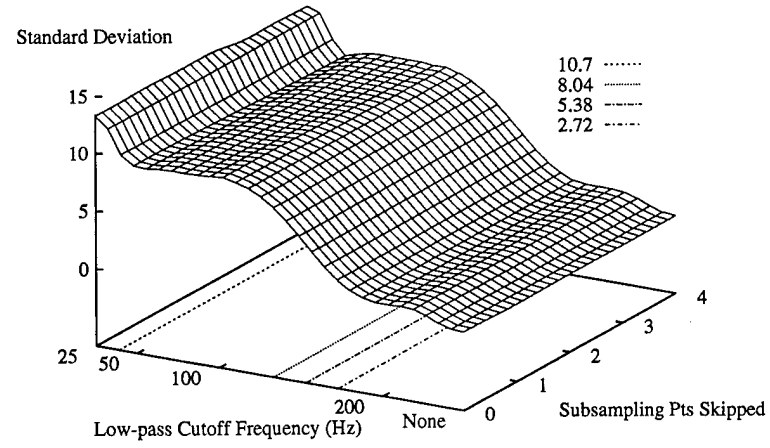



Figure 6.55 Standard deviation of included angle between $\vec{F}$ and $\vec{V}$ in RISO data as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after hemisphere pruning with a threshold of 86.5 degrees.

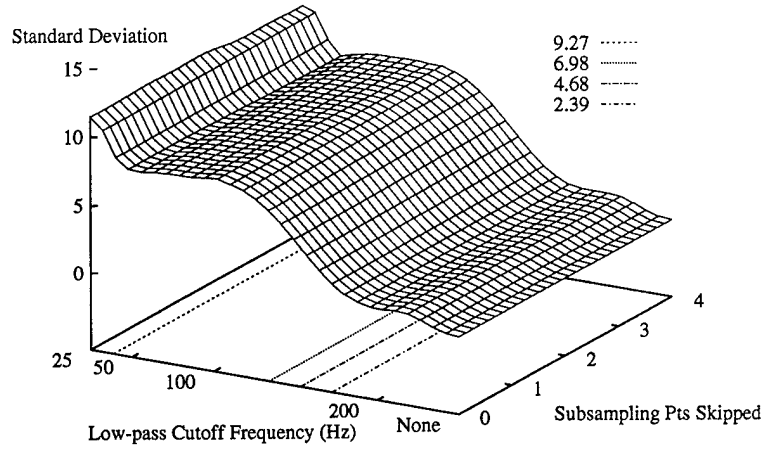



Figure 6.56 Standard deviation of included angle between $\vec{F}$ and $\vec{V}$ in RISO data as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after collision pruning with a threshold of 0.05 using a window size of 1.

consistent mapping matrix (as is the case for SISO, RISO and DISO data) and compute $\Psi$ for a whole range of $\vec{F}$, we will find that $\Psi$ varies, albeit not enormously. The small range of $\vec{F}$ in the RISO data, therefore, enhances the tight clustering of $\Psi$ for RISO data as compared to SISO data. This variability with the applied $\vec{F}$ can mask the perfect consistency of the underlying mapping matrix. Depending on the $\vec{F}$ we choose, we can cause the $\sigma_\Psi$ to reflect a poor consistency when, in fact, the mapping is perfect. Therefore, we discard the standard deviation of the included angle, $\sigma_\Psi$, between $\vec{F}$ and $\vec{V}$ as a measure of mapping consistency.

*6.3.3 Matrix Similarity Indexes.* When applied to RISO training data, the LSMF technique of interrogating reveals nearly perfect matches between $\mathcal{A}_a{}'$ and $\mathcal{A}_a$. This is consistent with our experience with SISO training data, confirming that the LSMF technique is valid for extracting a consistent linear mapping. When LSMF is applied to interrogate the trained weights of an ANN controller, the similarity of the resulting $\mathcal{A}_a{}'$ is not nearly so impressive, which is also consistent with our SISO experience. Although the sign similarity index, $\Upsilon_\pm$, was consistently zero, the other indexes were not. This confirms that the ANN is

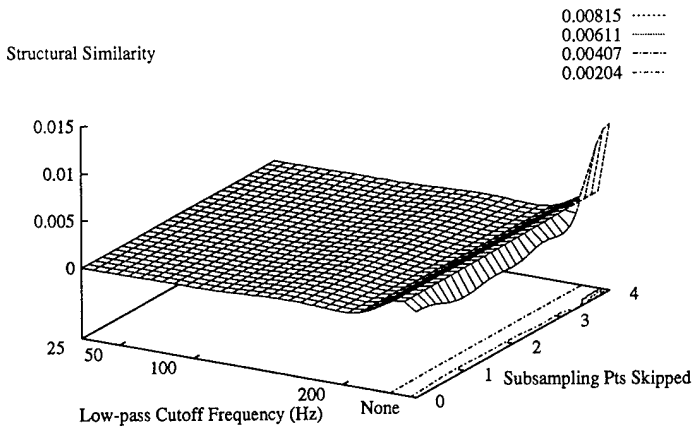



Figure 6.57 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on RISO data using the LSMF technique. Data are plotted as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after *velocity pruning* with a threshold of 0.05 m/s.

not adhering to a single linear mapping from $\vec{F}$ to $\vec{V}$ and the $\mathcal{A}_a{}'$ is only an approximation of the ANN mapping.

To further reinforce the results obtained with the SISO data, we compute the four matrix similarity indexes for the $\mathcal{A}_a{}'$ extracted from each of the ANN controllers trained on RISO data and evaluate their correlation with the performance metric, $\zeta$. After processing nearly all of the previously presented RISO-trained controllers, we find that none of the matrix similarity indexes are well correlated to the performance metric, $\zeta$, for RISO data, as shown by comparing Figures 6.57 through 6.65 with Figures 6.48, 6.50, and 6.51, respectively. This lack of correlation confirms our disillusion for using the similarity indexes of $\mathcal{A}_a{}'$ to predict the performance potential of the ANN controllers without actually implementing them.

*6.3.4 RISO Summary.* The RISO data have reinforced the results obtained with the SISO data. Any of the data processing steps that distorted the originally perfect mapping of $\vec{F}$ to $\vec{V}$ via the $\mathcal{A}_a$ of RISO data was found to corrupt the chances of training successful ANN controllers. Low-pass filtering was determined to be the main culprit in this regard.

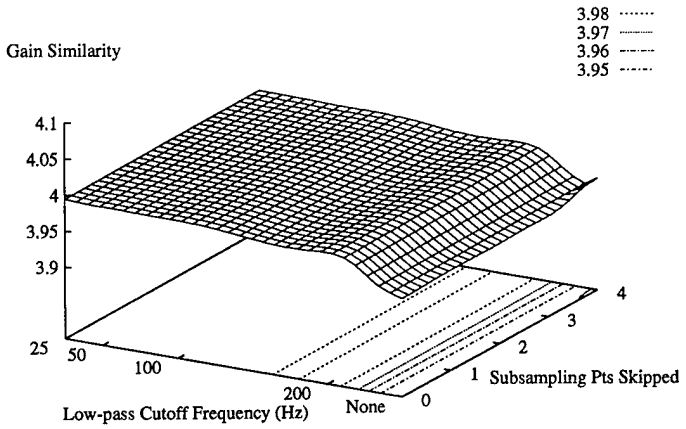



Figure 6.58 Matrix gain similarity index, $\Upsilon_g$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on RISO data using the LSMF technique. Data are plotted as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after *velocity pruning* with a threshold of 0.05 m/s.

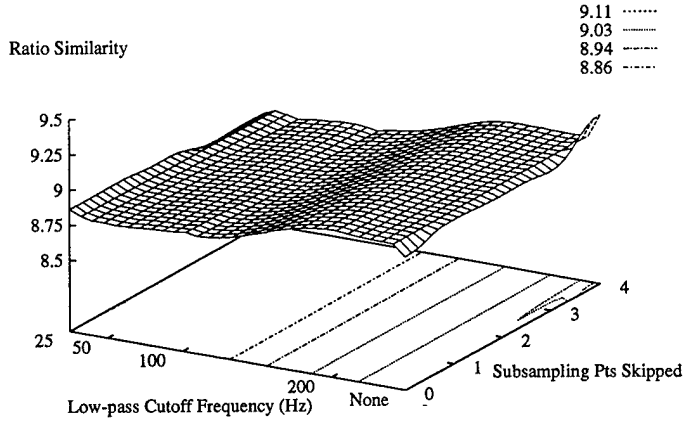



Figure 6.59 Matrix ratio similarity index, $\Upsilon_r$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on RISO data using the LSMF technique. Data are plotted as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after *velocity pruning* with a threshold of 0.05 m/s.

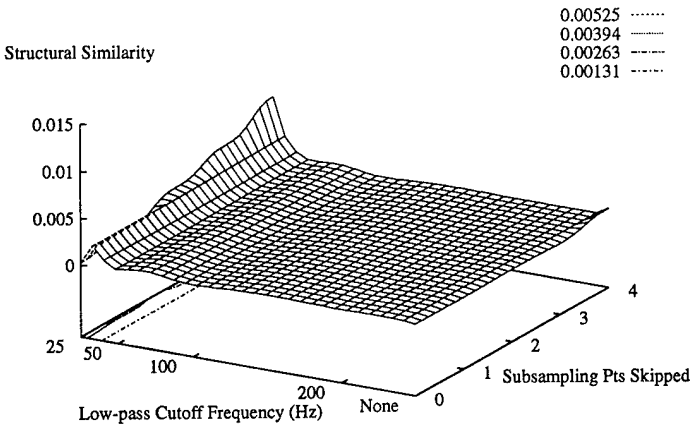



Figure 6.60 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on RISO data using the LSMF technique. Data are plotted as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after *hemisphere pruning* with a threshold of 86.5 degrees.

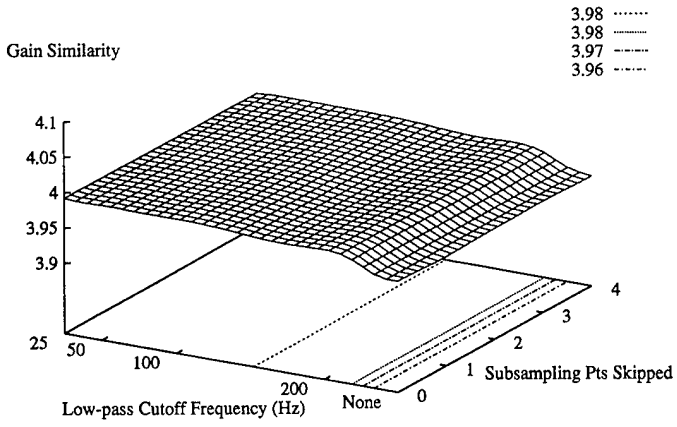



Figure 6.61 Matrix gain similarity index, $\Upsilon_g$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on RISO data using the LSMF technique. Data are plotted as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after *hemisphere pruning* with a threshold of 86.5 degrees.

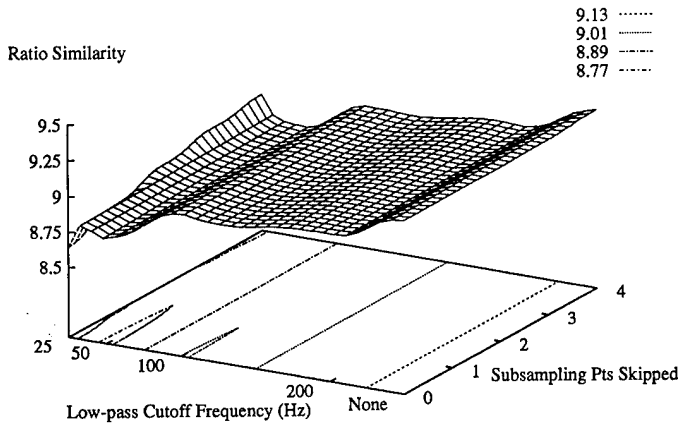



Figure 6.62 Matrix ratio similarity index, $\Upsilon_r$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on RISO data using the LSMF technique. Data are plotted as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after *hemisphere pruning* with a threshold of 86.5 degrees.

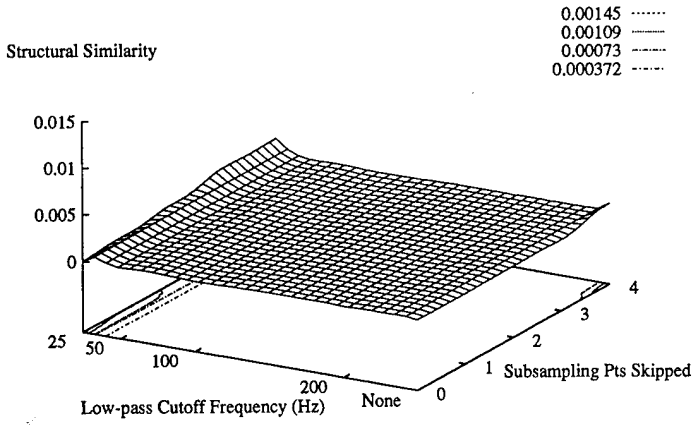



Figure 6.63 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on RISO data using the LSMF technique. Data are plotted as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after *collision pruning* with a threshold of 0.05 N.

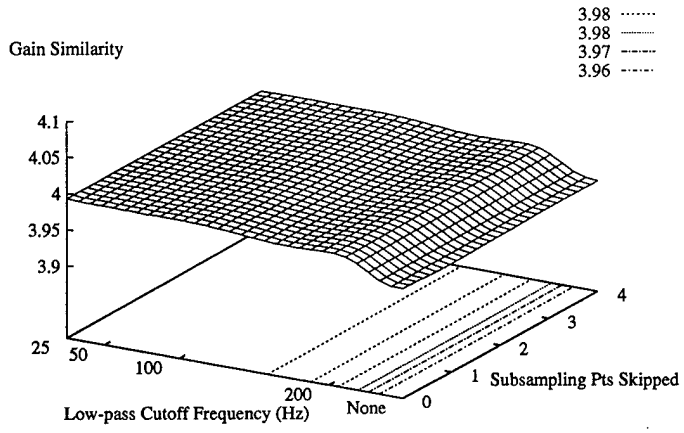



Figure 6.64 Matrix gain similarity index, $\Upsilon_g$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on RISO data using the LSMF technique. Data are plotted as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after *collision pruning* with a threshold of 0.05 N.

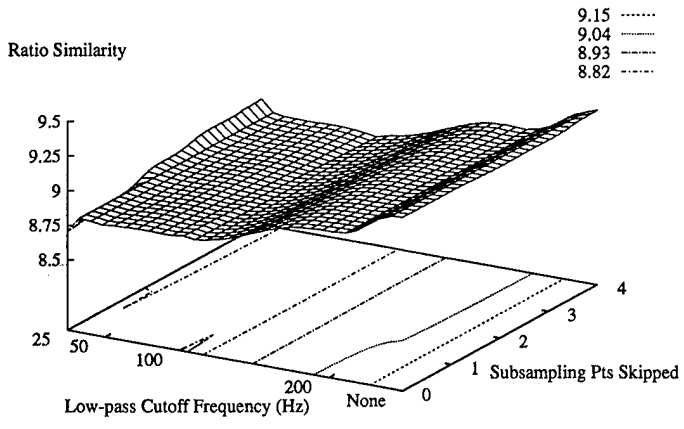



Figure 6.65 Matrix ratio similarity index, $\Upsilon_r$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on RISO data using the LSMF technique. Data are plotted as a function of low-pass cutoff frequency and number of points skipped between subsampling intervals after *collision pruning* with a threshold of 0.05 N.

The limits of velocity pruning were illustrated when $V_t$ was set too large and the trained controllers tended to overshoot the final desired alignment position because they had not been trained on enough data samples to illustrate the terminating condition. During the course of testing the merits of hemisphere pruning, the possibility of using the standard deviation of the included angle, $\sigma_\Psi$, appeared to show promise as a means of measuring the consistency of the I/O mapping in a set of training data, thereby having the potential of predicting the performance of trained controllers. However, in the end, it was shown that $\sigma_\Psi$ varied as a function of the input vectors used to test a mapping matrix, and therefore varied even though the matrix was perfectly consistent. None of the data processing steps that simply removed "bad" data vectors from the data sets had any significant effect on the trainability and performance of ANN controllers exposed to RISO training data. Since the RISO data set is already nearly ideal for training an ANN, no improvements were expected. However, we have also noted that no degradations were introduced, which was not the case when the data were low-pass filtered. This indicates that simple decimation of a data file does not corrupt the I/O mapping and the ANN can adequately learn from the data that remained in the training data sets.

### *6.4 DISO Observations.*

As was mentioned in Section 3.3, the DISO data have the input distribution of DIDO data and the idealized I/O mapping of the SISO data. Therefore, DISO data is a good resource for exploring the effect of distribution on the training and implementation of ANN controllers for the edge-mating task. For the present work, DISO data were important in answering two particular questions. First, at the outset of attempting to train an ANN controller to emulate an accommodation matrix controller, one should first establish whether the DIDO data contain an accommodation matrix control strategy. This question is equivalent to asking, "Does the human operator use an accommodation strategy when demonstrating the edge-mating task?" We will investigate this question by comparing the outputs of DISO and DIDO training data.

The second question to be answered using the DISO data concerns whether the distribution of DIDO training data is suitable for teaching an ANN controller. As we have

discovered in the SISO training and testing activity of Section 6.2, the distribution of the input feature vector, $\vec{F}$, can affect the chance for success in training and implementing an ANN controller even if the I/O mapping is perfect. Thus, by examining the success of DISO training data, we can decouple the concerns of $\vec{F}$ distribution from the integrity of the I/O mapping.

Figure 6.66 shows the difference between the time histories of output from demonstration number one of DIDO data collected on the PLIMMS and its corresponding DISO output. Similar plots for all 10 of the PLIMMS DIDO demonstrations are found starting on page C-22 in Appendix C.

The example shown in Figure 6.66 shows that both the gain and the strategy of the human demonstration in the DIDO training data are different from those of the $\mathcal{A}_a$ explicitly embedded in the DISO training data. Whereas the $\mathcal{A}_a$ identified an increasing need to recoil the peg in the negative ${}^{T}Y$-axis direction in response to the increasing magnitude of ${}^{T}F_y$, the human simply maintained table contact. The result is a many-to-one mapping of a whole range of $F_y$ to a small group of $V_y$. This was also the case, but to a lesser degree, for the $\omega_z$ component, as shown in Figure 6.66(c). We also note that the rocking of the peg at the end of the task (indicated by the sign-changing deviations in the $M_z$ curve of Figure C.11 from which Figure 6.66 was derived) passed by the human without response. This also causes a many-to-one mapping and corrupts the consistency embodied in the DIDO data.

The characteristics described for the example shown in Figure 6.66 are typical of all the other PLIMMS DIDO data as well. This shows that the human is not utilizing a consistent accommodation control strategy when demonstrating the edge-mating task. The author hypothesizes that the human is using additional sensory information, such as vision and hearing, as well as a full mental model of the task to achieve success. In the DIDO demonstrations recorded, the operator had full access to vision and hearing. In addition, he knew by looking where the table surface was located.

Despite the obvious difference in the input distributions of the DISO and RISO training data, ANN controllers trained on DISO data have no trouble learning the edge-mating task. In fact, of the 66 ANN controllers trained, none of them failed to complete the task, and

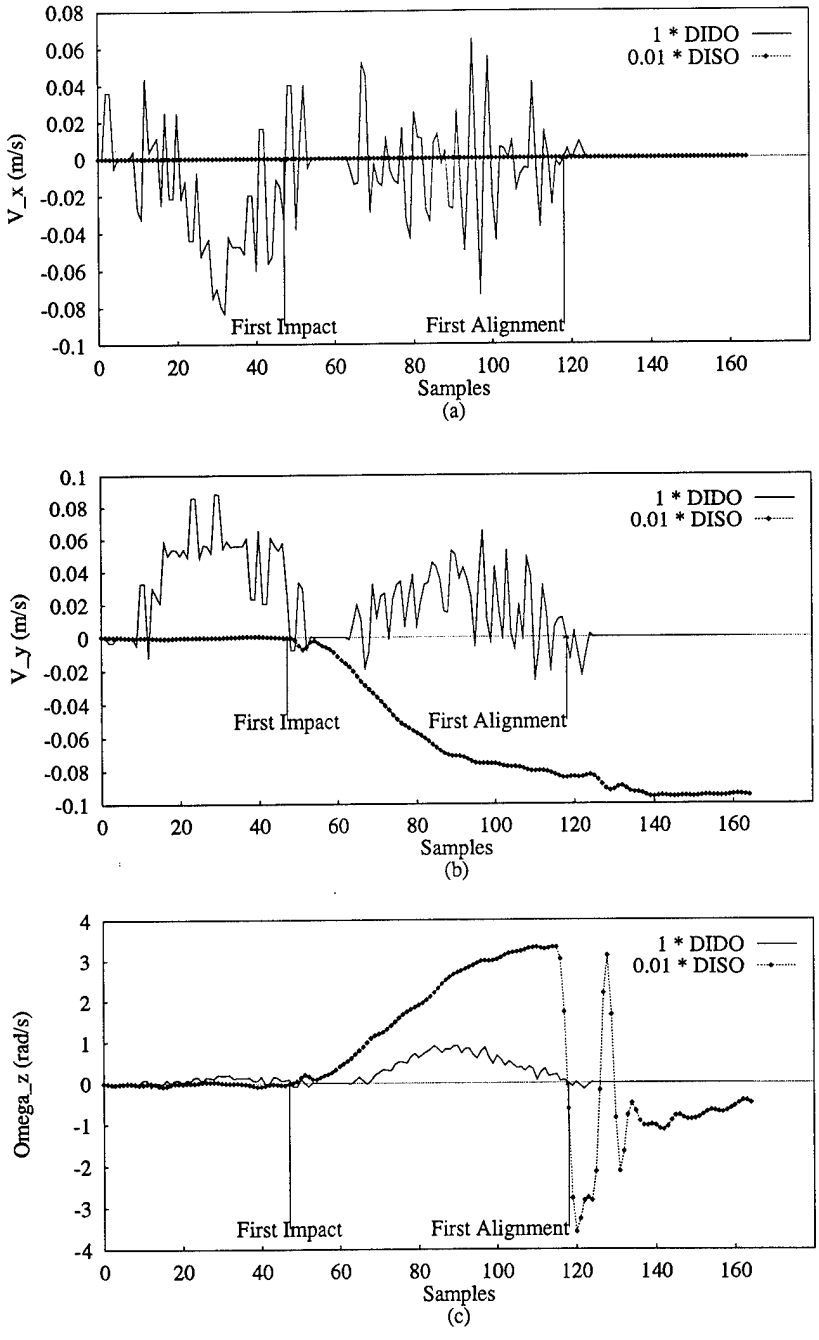



Figure 6.66 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for DISO and DIDO training data originally collected as demonstration number 1 on the PLIMMS. See Figure C.11 for the complete plot of the DIDO data.

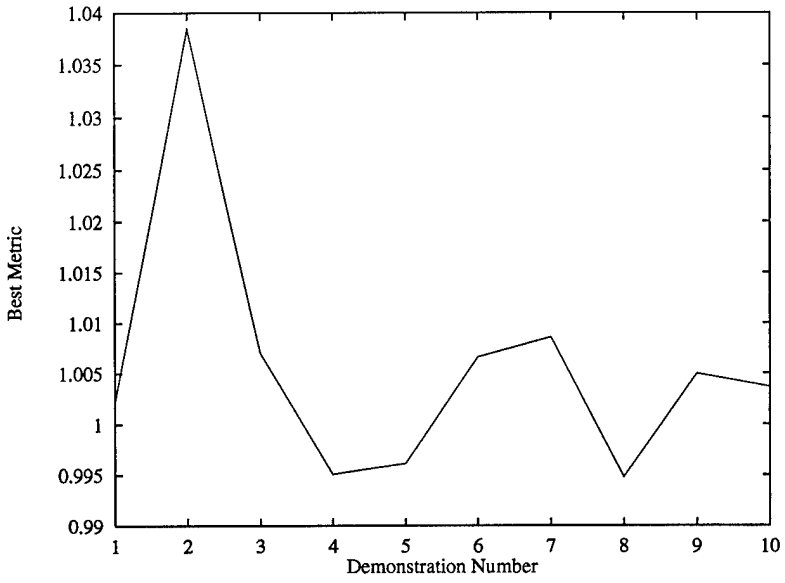



Figure 6.67 Performance metrics of ANN controllers trained on DISO training data.

the worst performance metric was 1.666. Figure 6.67 shows the best performance metric for each of the 10 DISO training data files transformed from PLIMMS DIDO data. The results depicted in Figure 6.67 came from training on the raw DISO data with no processing actions taken.

Figures 6.68 and 6.69 show force and velocity recordings of data taken while an ANN trained on DISO data was controlling the PUMA manipulator. These plots are proof-positive that, if the I/O mapping were exactly correct and consistent, an ANN controller could be trained from DIDO data to perform the edge-mating task.

As another progressive step in understanding the usefulness of the matrix similarity indexes in screening training data files for consistency prior to training, we apply the LSMF technique to DISO training data and compute the indexes for various window sizes and evaluate the results for insights. We start by examining the evolution of the four matrix similarity indexes for $\mathcal{A}_a{}'$ extracted from DISO training data using the LSMF technique, since it has a perfectly consistent mapping between the input and output vectors. As expected from our previous experience with SISO and RISO training data, the similarity indexes are very good. Except for $\Upsilon_s$, all the similarity indexes are identically zero regardless of the

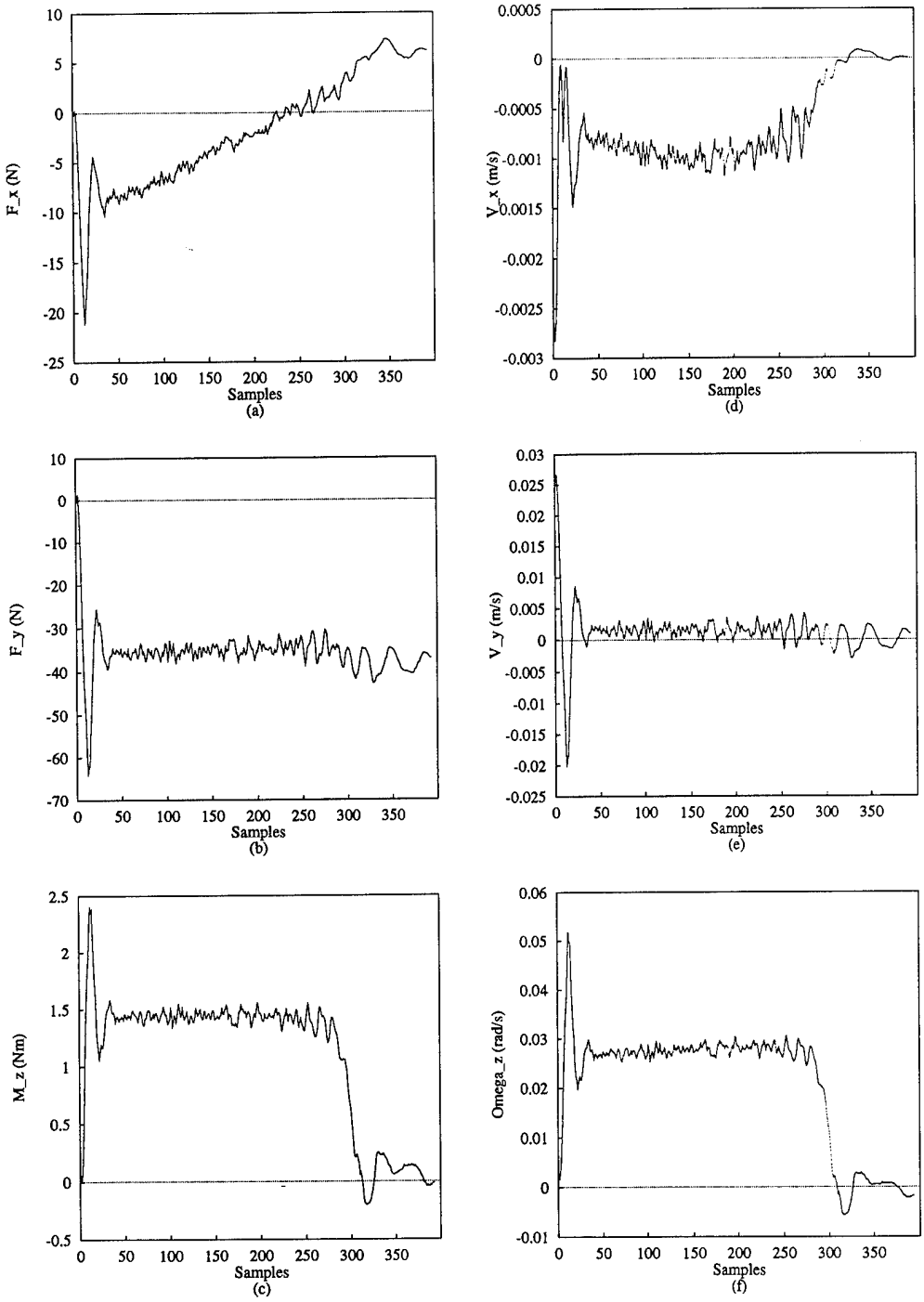



Figure 6.68 Recordings of (a)-(c) forces and (d)-(f) velocities from ANN controllers trained on DISO training data and implemented on the PUMA manipulator. Peg rotation was counter-clockwise.

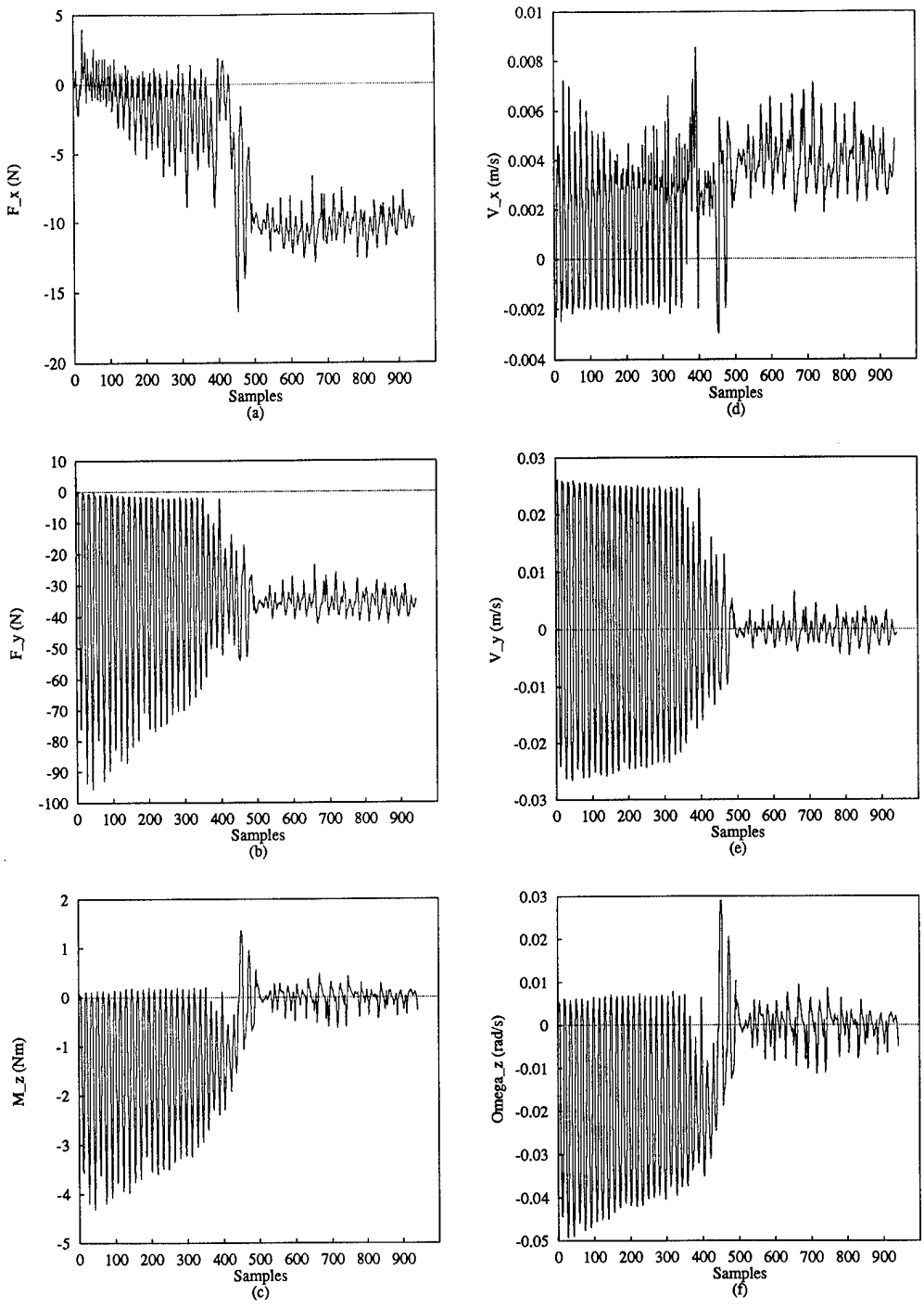



Figure 6.69 Recordings of (a)-(c) forces and (d)-(f) velocities from ANN controllers trained on DISO training data and implemented on the PUMA manipulator. Peg rotation was clockwise.

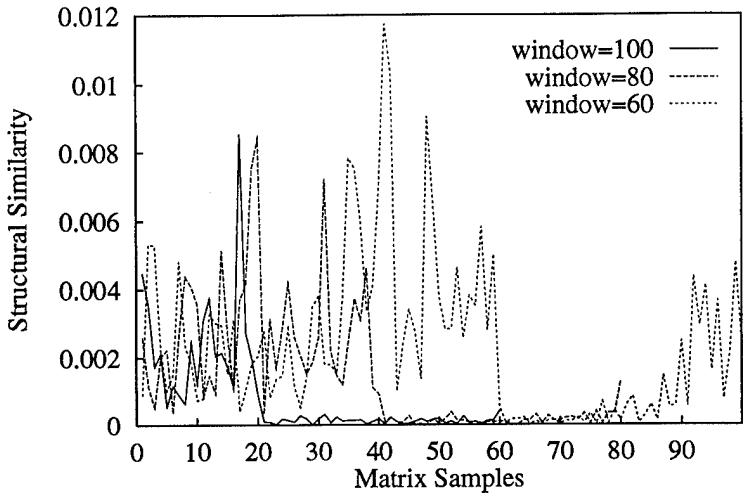



Figure 6.70 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a'$ extracted from DISO training data using the LSMF technique. Data are plotted for 3 different fitting window sizes to show variation.

fitting window size used. Even $\Upsilon_s$ is very small, as shown in Figure 6.70 for a typical set of DISO training data. The rather erratic shapes of the curves for each window size in Figure 6.70 are unfortunate, since they jeopardize the chances that subtle changes in accommodation matrix strategies (as embodied in changes of $\mathcal{A}_a$) can be recognized in the evolution of $\Upsilon_s$. We know that the DISO training data contain no changes in $\mathcal{A}_a$, so any variations from a constant $\Upsilon_s$ evidenced in DISO data are disruptive in our efforts to identify $\mathcal{A}_a$ changes later in DIDO training data. However, we reserve judgment on the applicability of the technique for screening DIDO training data until we investigate it fully.

As expected, the interrogation results are much worse for the $\mathcal{A}_a'$ extracted from the ANN weights trained on DISO data. Figures 6.71 through 6.74 present the evolutions of $\Upsilon_s$, $\Upsilon_g$, $\Upsilon_\pm$, and $\Upsilon_r$, respectively, for three different fitting window sizes ranging from 60 to 100 points for a typical DISO-trained ANN controller. These results are consistent with the poor results obtained when interrogating ANN controllers trained on SISO and RISO data.

*6.4.1 DISO Summary.* The DISO data have shown that both the gain and the strategy of the human demonstration in the DIDO training data are different from those of

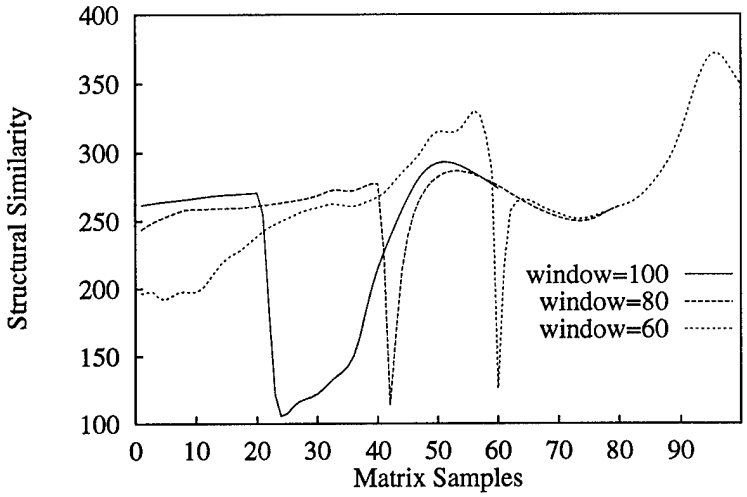



Figure 6.71 Matrix structural similarity index, $\Upsilon_s$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on DISO data using the LSMF technique.

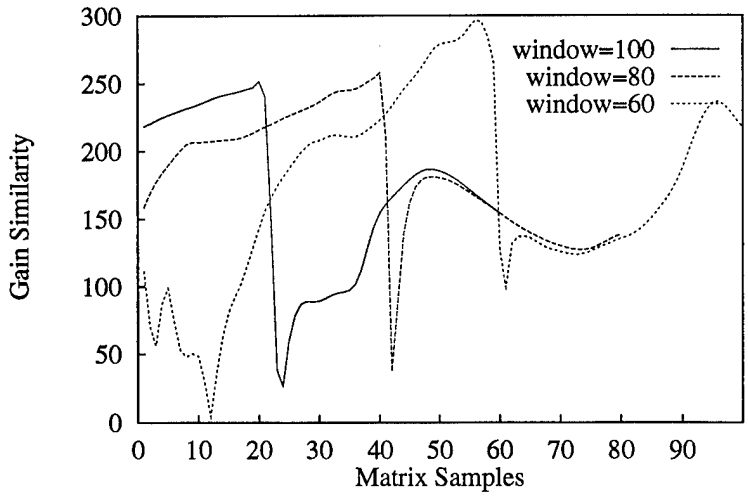



Figure 6.72 Matrix gain similarity index, $\Upsilon_g$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on DISO data using the LSMF technique.

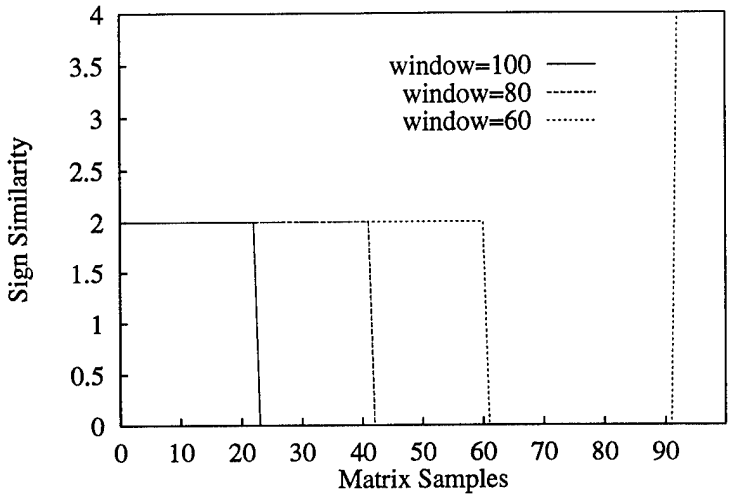



Figure 6.73 Matrix sign similarity index, $\Upsilon_{\pm}$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on DISO data using the LSMF technique.

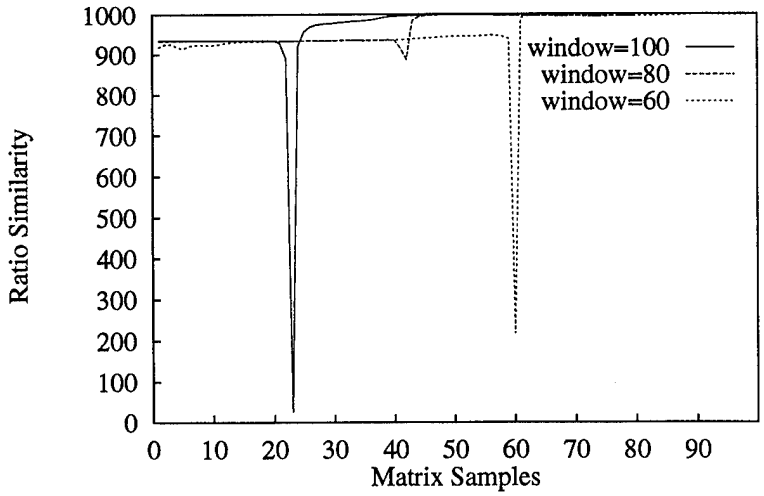



Figure 6.74 Matrix ratio similarity index, $\Upsilon_r$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on DISO data using the LSMF technique.

the $\mathcal{A}_a$ explicitly embedded in the DISO training data. This is proof that the human has not relied on an accommodation controller mapping to complete the task. Instead, we proposed that a mental model and additional sensory information besides the sensed contact forces were used to complete the task. This discovery casts a serious shadow over the feasibility of using DIDO data for training an ANN to control an accommodation task. In addition, the significant variability of the four matrix similarity indexes as a function of the matrix fitting window-size has been found to preclude any chance of using them to evaluate the consistency of an I/O mapping. When applied to DISO training data, the indexes were expected to be constant since the mapping matrix never changed. However, this was not found to be the case.

On the bright side, the DISO results demonstrated that an ANN controller could overcome any undesireable characteristics of the input feature vector distribution and learn to successfully perform the edge-mating task if the I/O mapping was perfectly consistent. This was derived from the successes in both simulation and on the PUMA manipulator of ANN controllers trained with DISO data. In the next section, the DIDO data will be investigated to determine if the predictions we have made based on the SISO, RISO, and DISO data will hold true.

### *6.5 DIDO Observations.*

DIDO data were collected from two different physical systems. The PUMA DIDO data collection system is described in Section 5.1.4.1 while the PLIMMS DIDO data collection system is described in Section 4.3. The need for the PLIMMS system arose from the results obtained using the PUMA system as described below.

*6.5.1 DIDO Data Collected From PUMA Manipulator.* The original DIDO data were collected using the PUMA manipulator with the training handles as described in Section 5.1.4.1. Ten edge-mating demonstrations were performed, and the force and velocity data were collected. The raw form of these data is shown in Figures C.1 through C.10 starting on page C-2 of Appendix C. For convenience, a plot of demonstration number three is shown in Figure 6.75. Several attempts were made to process the PUMA DIDO data and

then train an ANN controller. Groups of six or more of the individual task demonstrations were combined together into training data files after being individually processed by collision pruning using a threshold of 1.0 with a window size of 5 points and velocity pruning with a threshold of 0.20 meters/sec. The resulting controllers were implemented on the PUMA robot for evaluation. Prior to allowing the manipulator to touch the peg to the rigid table, the response of the controller was tested in free-space motion by manually applying forces and torques to the peg. The results of the free-space tests precluded allowing the peg to contact the table because the ANN controller insisted on rotating the peg, regardless of the force/torque pair applied. In addition, the ANN controller did not show any tendency to accommodate the y-axis force at the tip of the peg.

After repeated attempts to retrain and retest the controllers, it was concluded that the controller may have been doing precisely what it had been shown in the collected data. When DIDO data were collected on the PUMA, the robot was gravity-compensated to ease the operator's task. However, despite the compensation, it was still difficult to back-drive the PUMA through its high-ratio, geared transmission and slow controller servo update rate. As a result of the effort required to move the PUMA, virtually no sensation of peg-to-table contact force could be felt. Undeterred, however, the goal-oriented operator still completed the edge-mating task demonstration based on visual and auditory cues. The measured velocities demonstrated by the human controller were an end unto themselves rather than a reaction to a perceived contact force. Regardless of what force existed, the demonstrated velocity was a rotation of the peg about its contacting corner. Thus, virtually all $\vec{F}$ mapped to roughly the same $\vec{V}$, which was a rotation about the contacting corner. This is precisely the behavior of the trained ANN controllers.

To determine whether the matrix similarity indexes can distinguish poor training data, we interrogated all the PUMA DIDO training data sets to extract a single $\mathcal{A}_a{}'$ from each. Then the similarity indexes were computed for each $\mathcal{A}_a{}'$ as compared to the $\mathcal{A}_a$ given in Eq (6.1), which was considered representative of a good accommodation matrix for the PUMA robot controller. The results, shown in Figure 6.76, indicate that demonstration 7 was the worst match and that demonstration 1, 3, or 4 may be the best. No further testing was performed on the PUMA DIDO data because it was thought to have more corruptions than

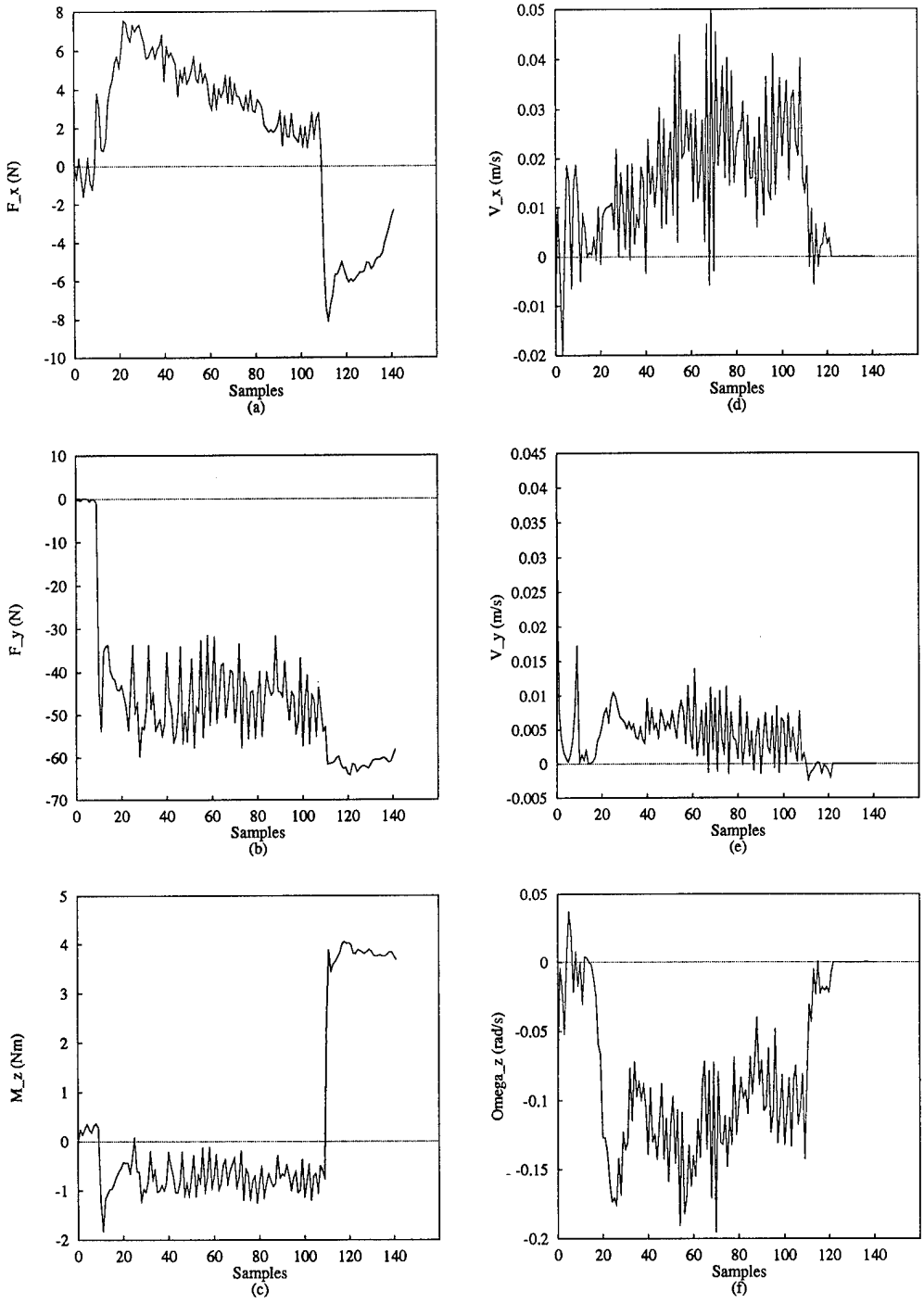



Figure 6.75 Demonstration number 3 of DIDO training data collected on the PUMA manipulator.

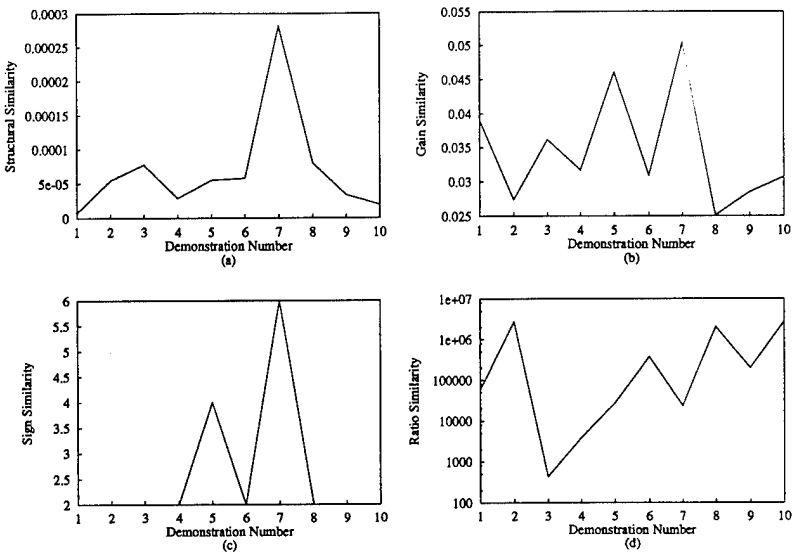
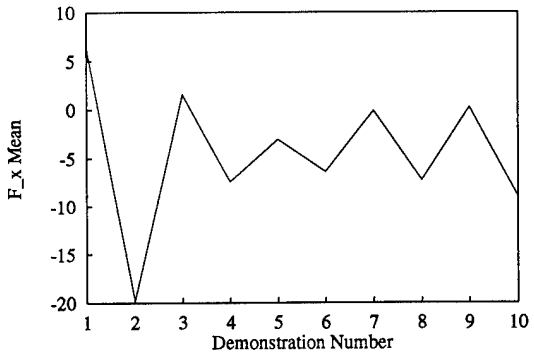



Figure 6.76 Matrix similarity indexes of $\mathcal{A}_a{}'$ extracted using the LSMF technique from DIDO training data collected on the PUMA manipulator showing (a) structural similarity, (b) gain similarity, (c) sign similarity, and (d) ratio similarity.

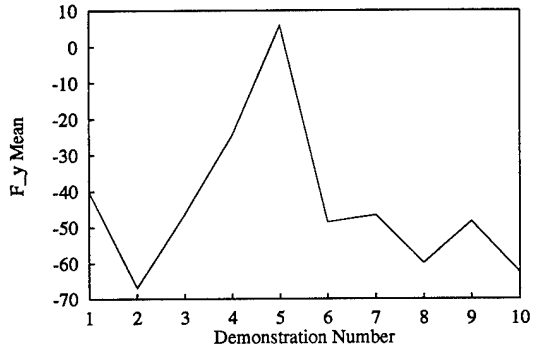
the PLIMMS DIDO data discussed in the next section. The results shown in Figure 6.76 will be compared with similar results for the PLIMMS DIDO training data sets that are presented in the next section to determine which best represented the desired accommodation mapping.

As a point of comparison with the PLIMMS DIDO training data, the mean values for each component of $\vec{F}$ collected on the PUMA were computed and are plotted in Figure 6.77. Notice that there is a significant bias to the mean values, which was shown in the SISO data investigation to cause difficulty when used as training data. We note, however, that these DIDO data could be mirrored to correct the bias in the mean values as we will explore further in the next section.

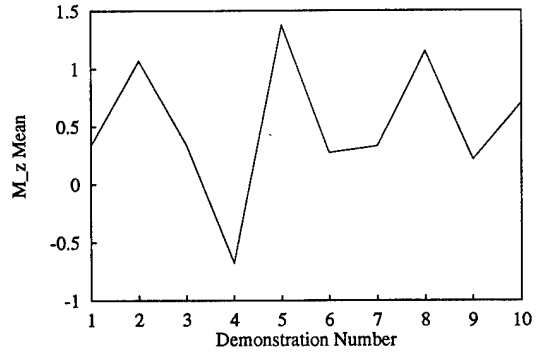
*6.5.2 DIDO Data Collected From PLIMMS..* Based on the hypothesis that the effort to back-drive the PUMA was corrupting the task demonstration, the PLIMMS mechanism was designed and built to be a lighter and more easily back-drivable DIDO data collection system. The system is fully described in Section 4.3. There were 10 sets of DIDO data collected on the PLIMMS which were studied in detail. Appendix C contains a catalog of




(a)




(b)




(c)

Figure 6.77 Mean values for the (a) $F_x$, (b) $F_y$, and (c) $M_z$ components of $\vec{F}$ of the DIDO data collected on the PUMA manipulator.

plots showing all 10 of the demonstrations (Figures C.11 through C.20 starting on page C-12). For convenience, sample one of the PLIMMS DIDO data is shown in Figure 6.78.

Examination of the PLIMMS DIDO data reveals that it does not reflect proper accommodation in the y-axis direction. To display an accommodation relationship, a negative $V_y$ component would be present when a negative $F_y$ was sensed. The data plots in Figures C.11 through C.20 do not contain such a relationship. The operator continued to move the peg into the aligned position using a positive $V_y$ despite the rather large $F_y$ component. The small magnitude of the $V_y$ component identifies it as the y-axis motion in the tool-frame coordinates due primarily to the coupling of the translational motion of the peg tip to the rotation of the peg about its corner in contact with the table. Thus the $V_y$ demonstrated motion, although it does not depict an accommodation mapping, is inevitable for the task and the coordinates. The fact that the $F_y$ was large and had no apparent $V_y$ response is indicative of a controller with a very sluggish response or a large deadband. Either of these characteristics makes the training data very difficult to learn from.

*6.5.2.1 LSMF Investigation.* We begin our investigation of the PLIMMS DIDO data by applying the LSMF matrix interrogation technique to extract $\mathcal{A}_a{}'$ from the training data and examining the similarity indexes of those $\mathcal{A}_a{}'$. This was accomplished for all the DIDO training data configurations described in Table 6.1. The resulting similarity indexes for the original (raw) PLIMMS DIDO data are shown in Figure 6.79 for comparison with Figure 6.76. The similarity indexes were computed relative to the $\mathcal{A}_a$ given in Eq (6.1) for the robot accommodation matrix controller. Comparing the results to those shown for the PUMA DIDO data in Figure 6.76 indicates the PUMA data has lower (better) values in general for all four similarity indexes. This indicates that our effort to enable the human operator to better feel the interaction forces during the task may have been counter-productive in the sense of similarity to the $\mathcal{A}_a$ given in Eq (6.1). It is not clear, however, that better similarity to the accommodation of Eq (6.1) is a guarantee that better controllers can be trained from these data. It is an interesting point of comparison, however, and will be discussed further in Section VII.

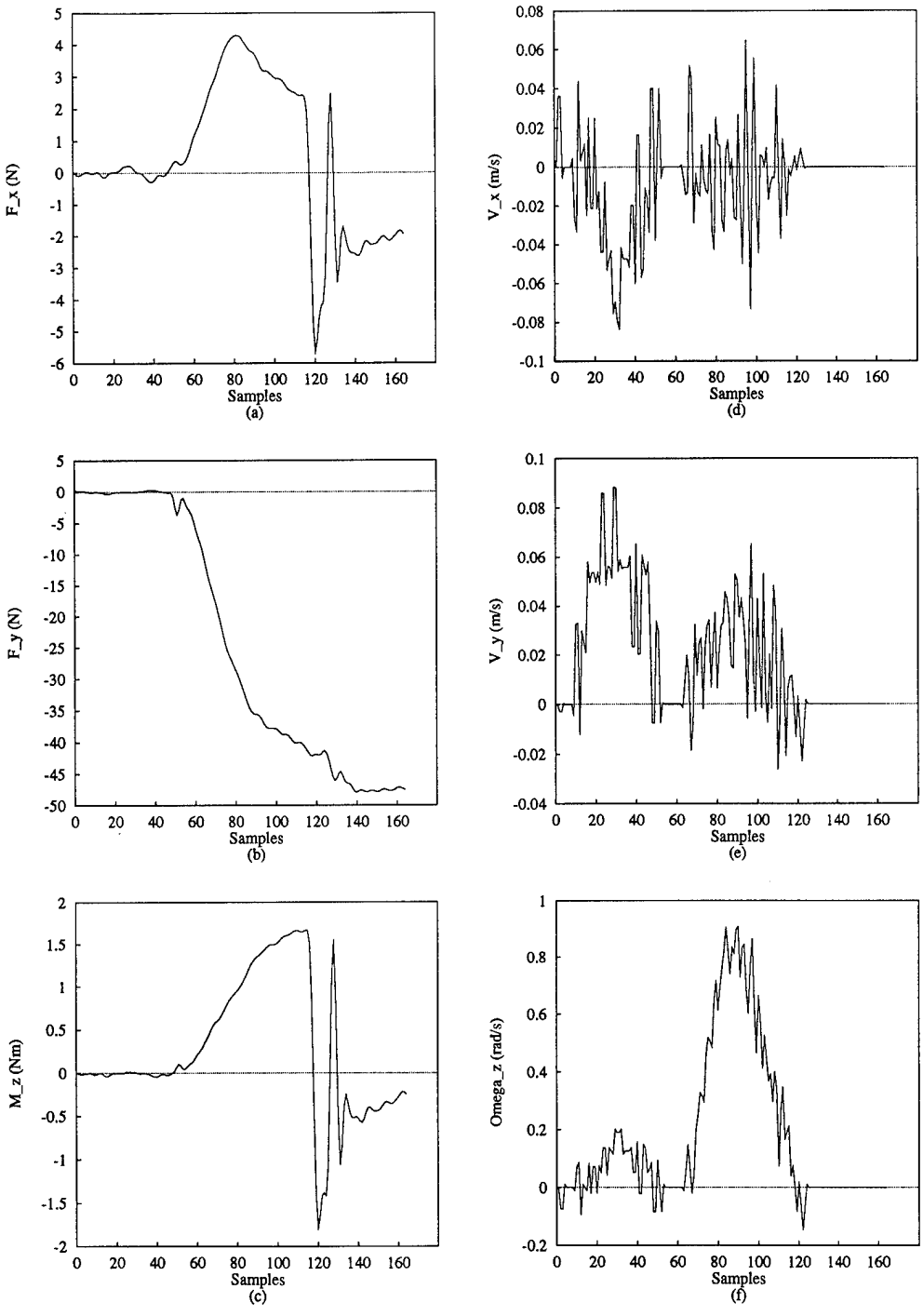



Figure 6.78 Demonstration number 1 of DIDO training data collected on the PLIMMS.

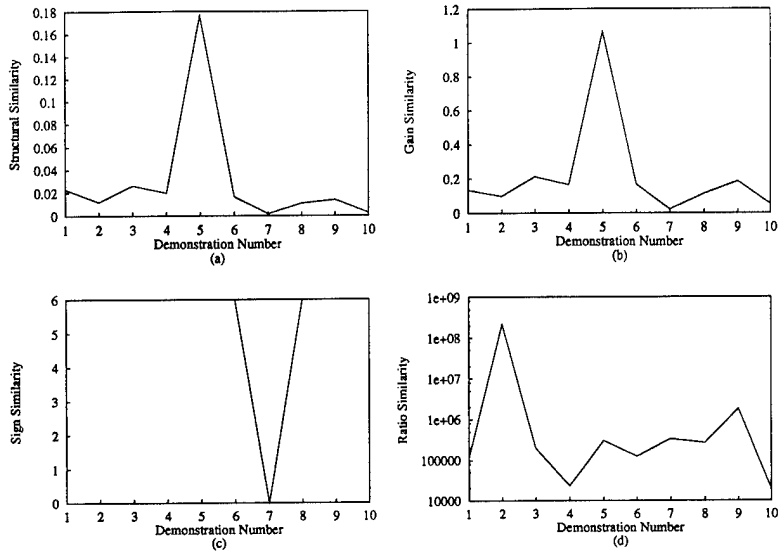



Figure 6.79 Matrix similarity indexes of $\mathcal{A}_a{}'$ extracted using the LSMF technique from DIDO training data collected on the PLIMMS showing (a) structural similarity, (b) gain similarity, (c) sign similarity, and (d) ratio similarity.

The similarity data for all the configurations are plotted in Figures 6.80 through 6.83. These data indicate that demonstration number 5 is the least similar and demonstration number 7 is the most similar for the majority of the configurations examined. Further, we observe that the $\Upsilon_s$, $\Upsilon_g$, and $\Upsilon_\pm$ were essentially unchanged by the 13 different data processing combinations applied to the data sets. In some of the configurations for demonstration numbers 2 and 10, there were sign changes in one element or the other, but we can distinguish no consistent pattern relating to the usefulness of any single processing step.

When we examine Figure 6.83 in conjunction with the tabulated values of $\Upsilon_r$ presented in Table B.16 of Appendix B, we note that the hemisphere pruning at 90-degrees threshold, $\Psi_t$, caused detrimental effects on demonstrations 8 and 9, while pruning with $\Psi_t$= 95 degrees did not. This is an unexpected effect which we will evaluate further when examining the performance metrics resulting from implementing the ANN controllers trained on these data.

The big spike in Figure 6.83 for configuration 6 of demonstration number 2 indicates that low-pass filtering with too low a cut-off frequency can affect the $\mathcal{A}_a{}'$ extracted from some

Table 6.1 Key to configuration codes of DIDO training data listed in Figures 6.80 through 6.83.

| Code | Coll. Pruned varied | Vel. Pruned @0.05 m/s | Low-pass filtered @ 5pts | Low-pass filtered @ 10pts | Hem. Pruned @ 90° | Hem. Pruned @ 95° |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | X | | | | | |
| 3 | | | | | X | |
| 4 | | | | | | X |
| 5 | | | X | | | |
| 6 | | | | X | | |
| 7 | | X | | | | |
| 8 | X | | | | X | |
| 9 | X | | | | | X |
| 10 | X | | X | | | |
| 11 | X | | | X | | |
| 12 | X | X | | | | |
| 13 | | X | X | | | |
| 14 | X | X | X | | | |

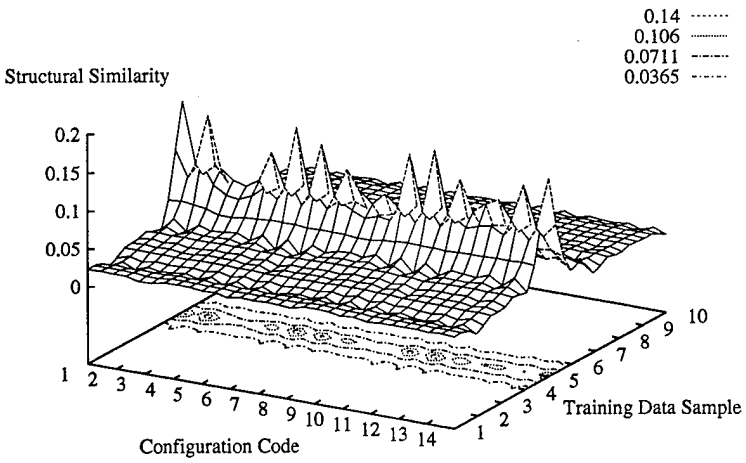



Figure 6.80 Matrix *structural* similarity index, $\Upsilon_s$, for DIDO training data collected on PLIMMS. Ten different demonstration files are examined as a function of the data processing steps applied.

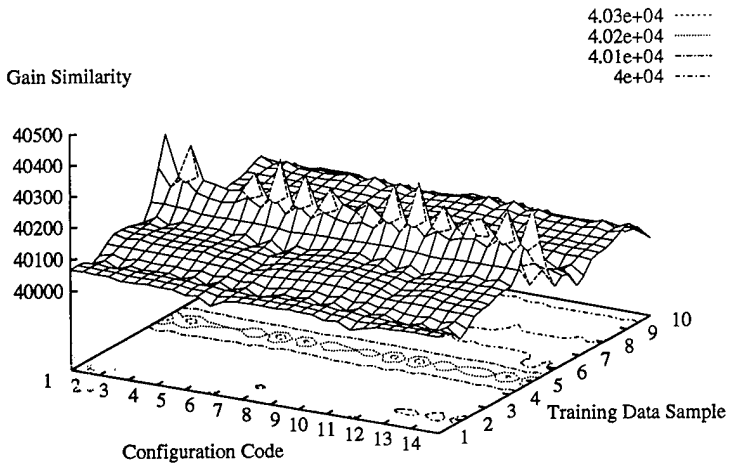



Figure 6.81 Matrix *gain* similarity index, $\Upsilon_g$, for DIDO training data collected on PLIMMS. Ten different demonstration files are examined as a function of the data processing steps applied.

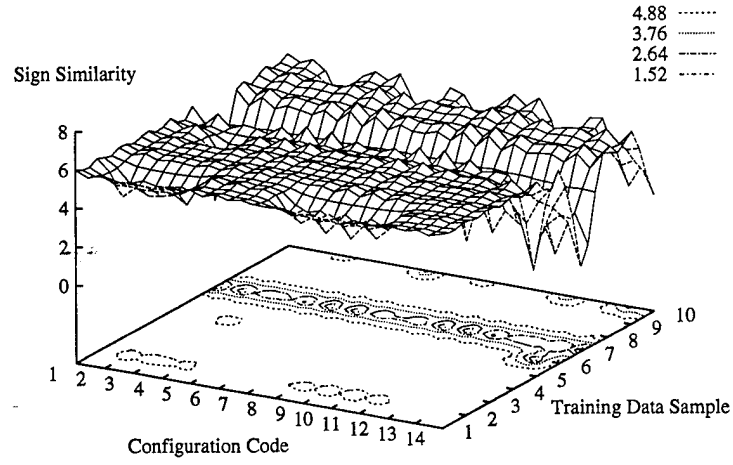



Figure 6.82 Matrix *sign* similarity index, $\Upsilon_{\pm}$, for DIDO training data collected on PLIMMS. Ten different demonstration files are examined as a function of the data processing steps applied.

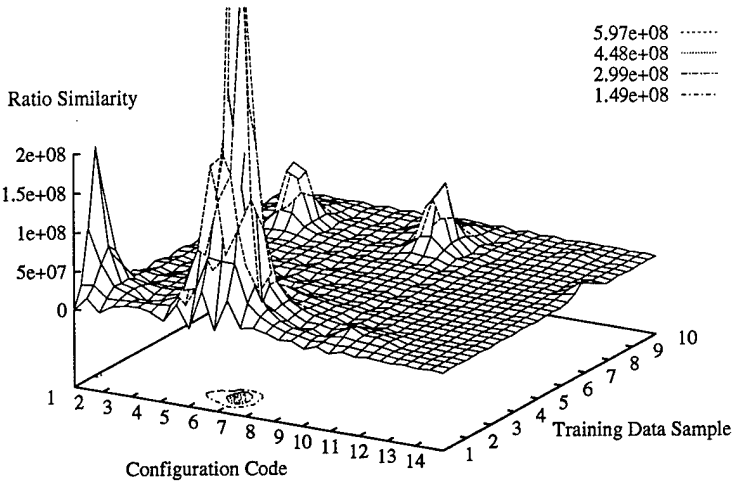



Figure 6.83 Matrix *ratio* similarity index, $\Upsilon_r$, for DIDO training data collected on PLIMMS. Ten different demonstration files are examined as a function of the data processing steps applied.

data files. Examining the original data for demonstration number 2 shown in Figure C.12 reveals nothing unusual to distinguish it from the other DIDO data.

With the matrix similarity indexes of the DIDO training data presented, the results of training and testing the ANN controllers was investigated to determine if the similarity indexes provide any insight about preferred training demonstrations. When the PLIMMS DIDO data sets described by the configuration codes in Table 6.1 were presented for training and the resulting ANN controllers were implemented in simulation using the DIDO parameters listed in Table 5.2, very few of the controllers could complete the task. Even fewer could do it with a reasonably stable termination state. Table 6.2 summarizes the results of these tests.

These performance metric data were too sparse to plot, but Figures 6.84 through 6.87 show the simulation results of the most successful controllers for configurations 3, 10, 11, and 12, respectively. The plots display acceptable goal positions, but generally contain some erratic behavior during the task. Each of the controllers was tested from a variety of initial positions to ensure they had not learned just a single solution for the task. Many of the other

Table 6.2 Summary of the simulation results for testing the ANN controllers trained on PLIMMS DIDO data.

| Config. Code | No. of successes | Best Metric |
|---|---|---|
| 1 | 0 | — |
| 2 | 0 | — |
| 3 | 1 | 18.380 |
| 4 | 0 | — |
| 5 | 0 | — |
| 6 | 0 | — |
| 7 | 0 | — |
| 8 | 0 | — |
| 9 | 0 | — |
| 10 | 1 | 4.175 |
| 11 | 1 | 4.145 |
| 12 | 2 | 9.179 |
| 13 | 0 | — |
| 14 | 0 | — |

controllers that were tested and failed did so because they could only complete the task from an initial position having a CCW angular position error.

A comparison between the trends of success in Table 6.2 and the plots of the four similarity indexes presented in Figures 6.80 through 6.83 reveals no apparent correlation between any of the similarity indexes and the ANN controller successes. Identification of a correlation is precluded by the fact that there were so few successes with which to correlate. Even if we had a significant number of successes to work with, we would have to make the comparisons on a demonstration-by-demonstration basis, i.e. the number of successes achieved by controllers trained on demonstration 3 compared to the similarity index data of demonstration 3. In the present case, however, there are too few successes for it to matter.

The erratic motions depicted in Figures 6.86 and 6.87 illustrate motion histories that would not be possible with a linear accommodation matrix controller. The big jumps can only be the result of nonlinearities in the controller mapping. Therefore, we consider whether linear controllers might do a better overall job. To examine this possibility, we used the LSMF

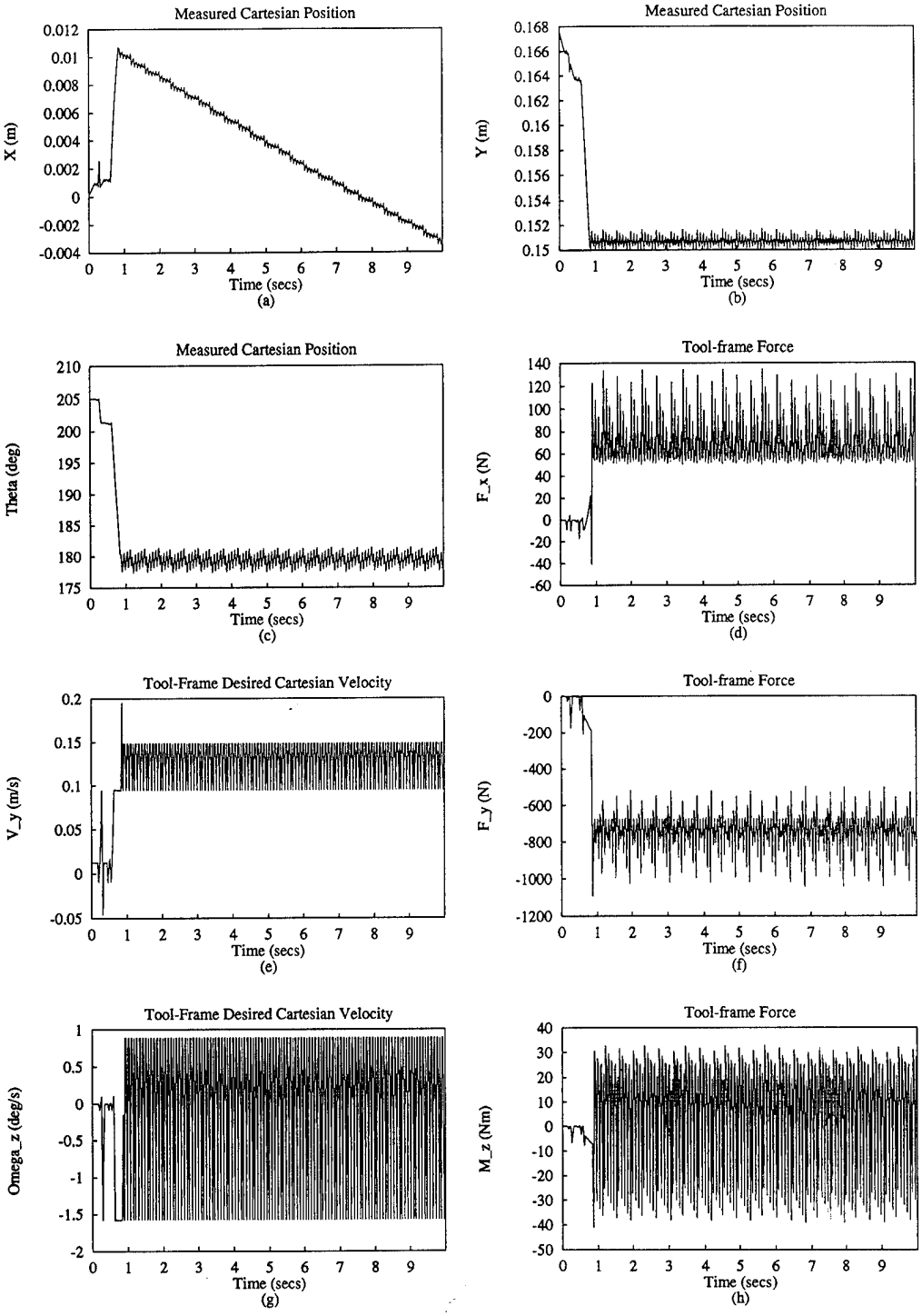



Figure 6.84 Simulation results from the ANN controller trained on PLIMMS DIDO training data after hemisphere pruning with a threshold of 90 degrees (configuration code 3).

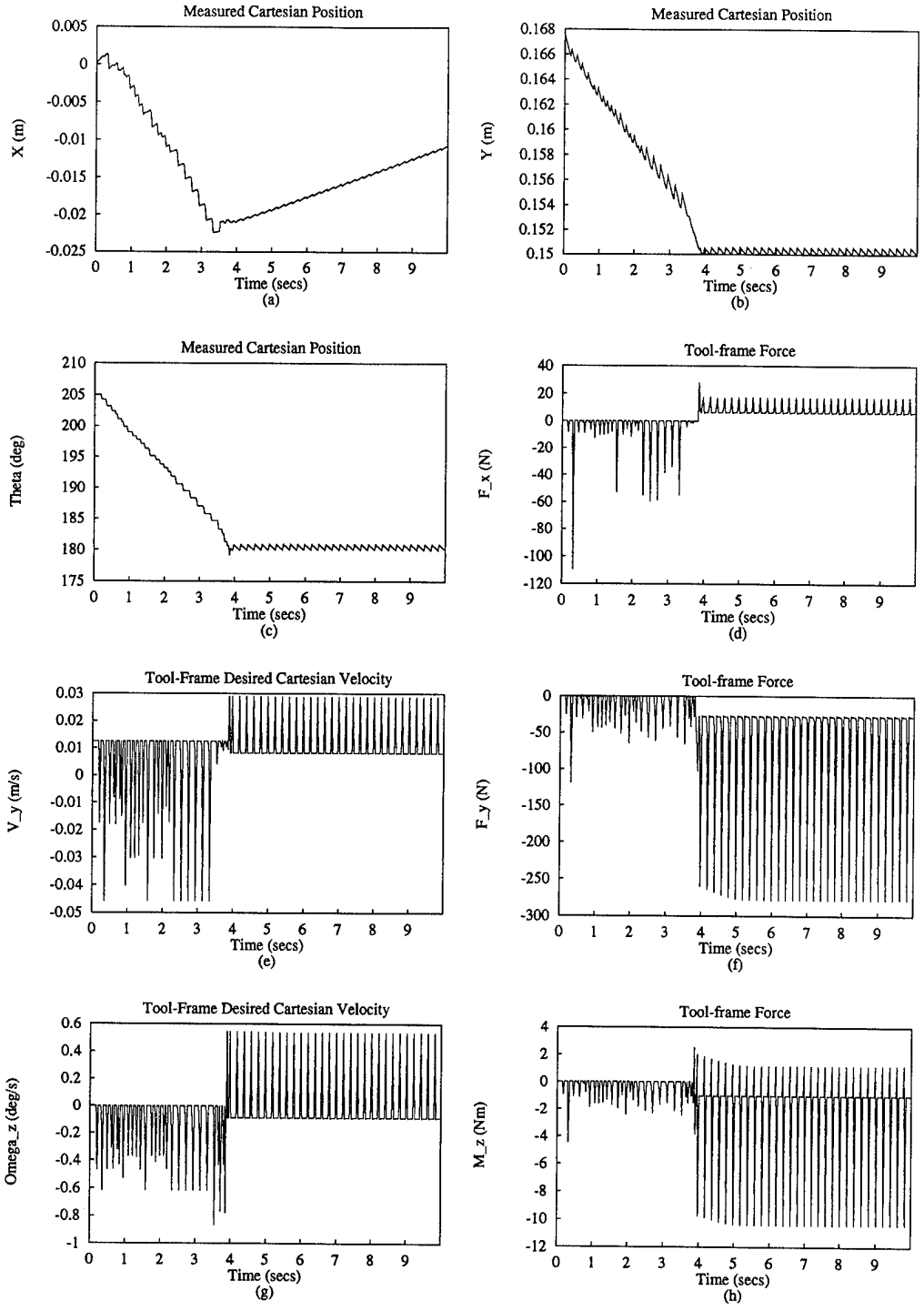



Figure 6.85 Simulation results from the ANN controller trained on PLIMMS DIDO training data after collision pruning with a threshold of 0.05 and low-pass filtering using 5.0 points (configuration code 10).

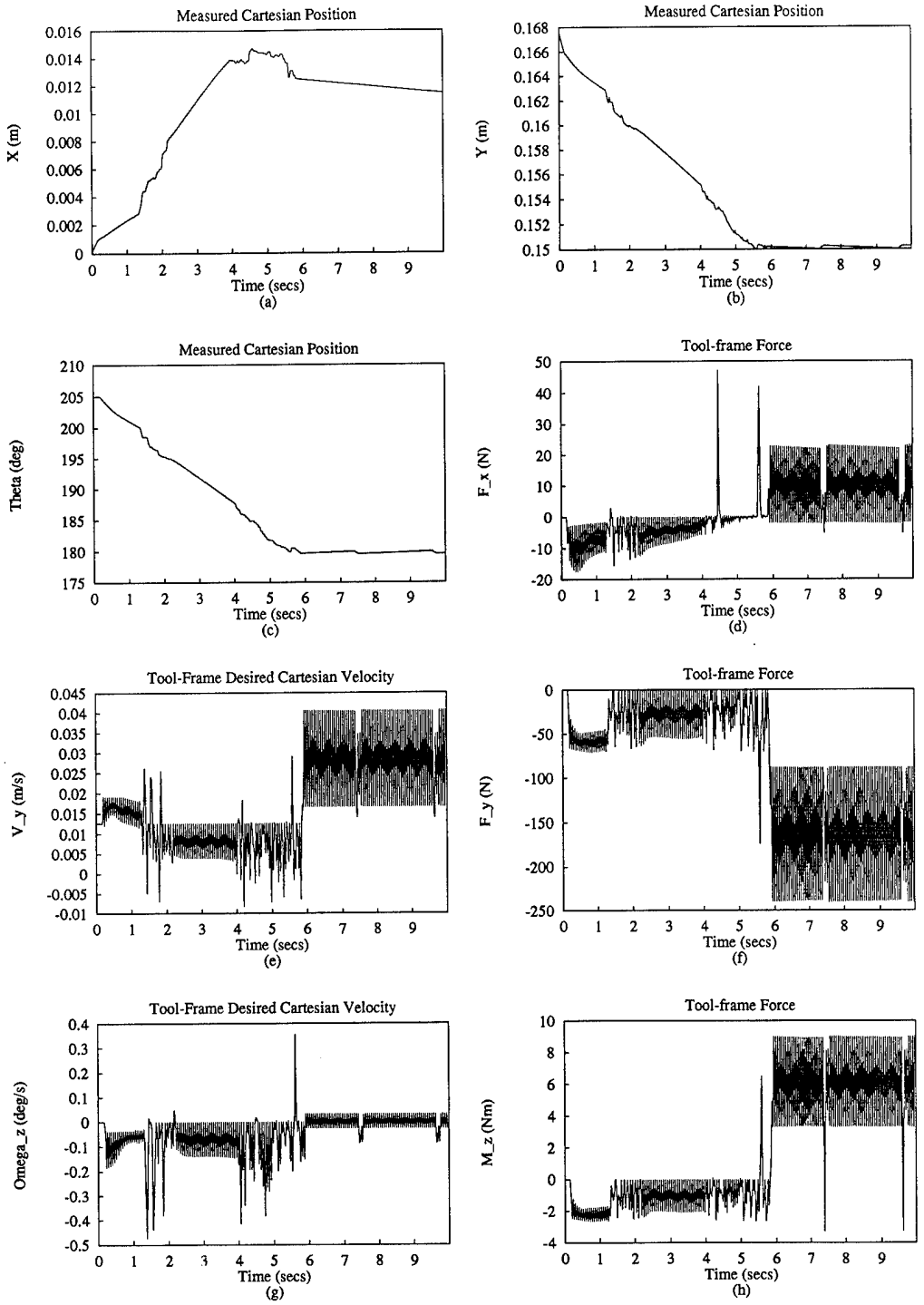



Figure 6.86 Simulation results from the ANN controller trained on PLIMMS DIDO training data after collision pruning with a threshold of 0.05 and low-pass filtering using 10.0 points (configuration code 11).

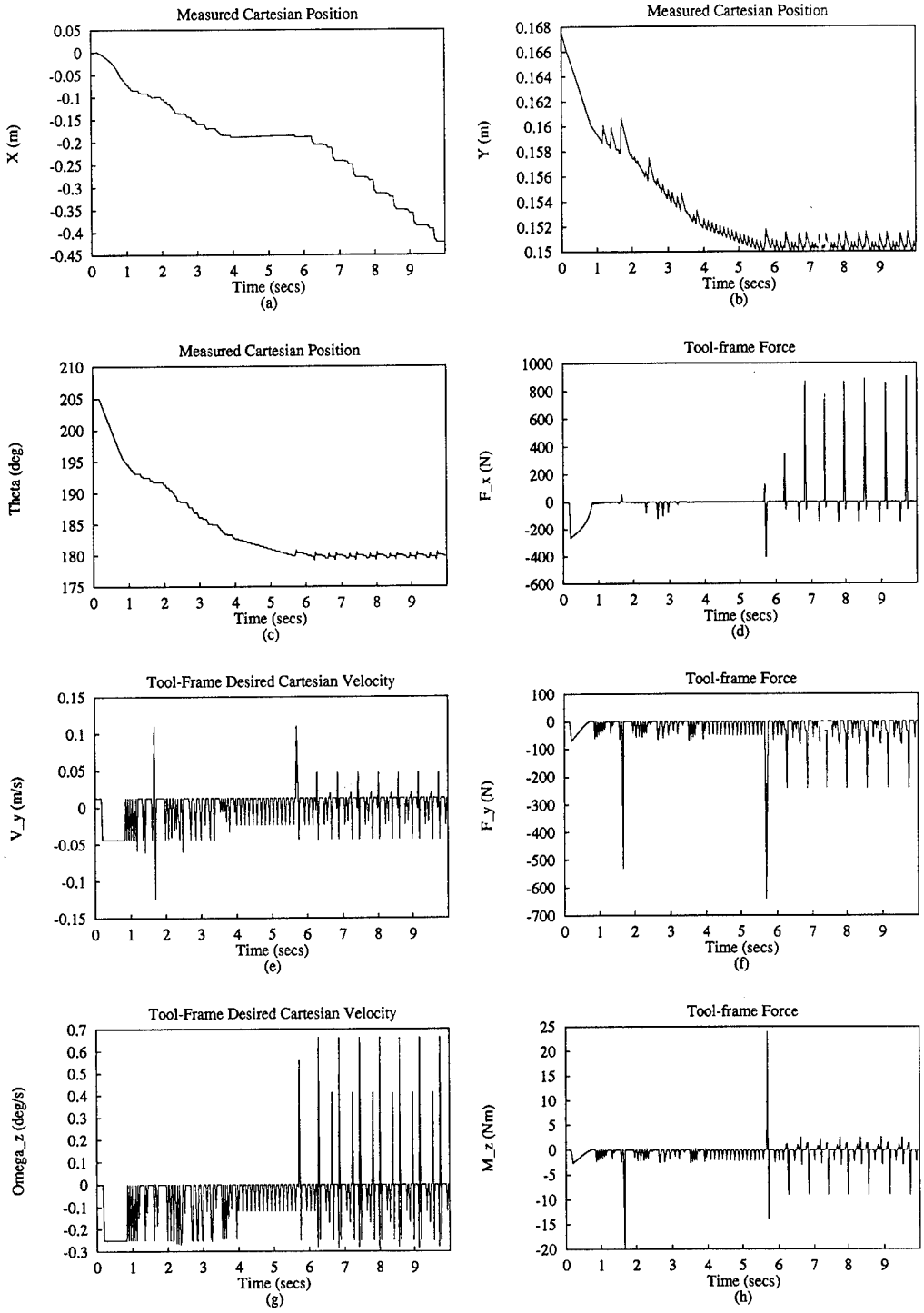



Figure 6.87 Simulation results from the ANN controller trained on PLIMMS DIDO training data after collision pruning with a threshold of 0.05 and velocity pruning at 0.05 m/s (configuration code 12).

Table 6.3 Summary of the simulation results for testing the accommodation matrix controllers using the $\mathcal{A}_a{}'$ extracted from PLIMMS DIDO training data

| Config. Code | No. of successes | Best Metric |
|---|---|---|
| 1 | 3 | 1.059 |
| 2 | 2 | 1.065 |
| 3 | 1 | 0.753 |
| 4 | 1 | 1.041 |
| 5 | 3 | 1.042 |
| 6 | 2 | 3.558 |
| 7 | 1 | 0.748 |
| 8 | 1 | 0.648 |
| 9 | 1 | 1.051 |
| 10 | 1 | 1.052 |
| 11 | 2 | 2.764 |
| 12 | 1 | 0.793 |
| 13 | 2 | 0.740 |
| 14 | 1 | 0.637 |

interrogation technique to extract the $\mathcal{A}_a{}'$ from each ANN controller and implemented them as accommodation matrix controllers. The results are presented in Table 6.3.

The performance data in Table 6.3 for the $\mathcal{A}_a{}'$ controllers represent a significant improvement over that of Table 6.2 for the ANN controller. However, these successes were still difficult to obtain, as evidenced by the small number of controllers which succeeded as compared to the 10 controllers tried for each configuration. The fact that the results in Table 6.3 are overall better than those of Table 6.2 shows that the nonlinear capability of the ANN controllers is more of a liability than an asset if the training data are not carefully composed. The fact that the results of Table 6.3 are poor indicates that the DIDO training data are inadequate representations of the skill needed to complete the task. Based on our experience with $\mathcal{A}_a{}'$ extracted from SISO, RISO, and DISO data, we know that if the DIDO data were better representations, more of the $\mathcal{A}_a{}'$ matrices would have succeeded as accommodation matrix controllers.

The DIDO training and testing results presented also substantiate the conclusion that velocity pruning is not a helpful data processing step. We observe that configuration codes 7

Table 6.4 Means and standard deviations for $\vec{F}$ components of RISO training data.

| Component | Mean value | Std. Dev. |
|---|---|---|
| $F_x$ | 0.01572 | 0.03631 |
| $F_y$ | -0.07964 | 0.14555 |
| $M_z$ | -0.00260 | 0.00565 |

and 12-14 used velocity pruning as one of the processing steps and find the performance of those controllers are not significantly improved over any of the other controllers. Recalling the hazard of decimating the exemplar vectors depicting the proper task termination which was discussed in Section 6.3.1.2, we conclude that velocity pruning is not a viable technique for improving the performance of controllers trained on DIDO data.

The SISO and RISO controllers identified concerns for both the consistency of the mapping in the training data and the input feature vector distribution. Although the DISO results clearly indicated that the input distribution alone could not cause the DIDO-trained controllers to fail, the possibility of improving the DIDO controllers leads us to try correcting for any problems in the input distributions of the DIDO training data.

We use the mean and standard deviation of the raw RISO data $\vec{F}$ as our model with which to match the DIDO data because we know the RISO data represent our best estimate of what the input vectors will look like for a successful controller in operation. We are presently only interested in the means which are presented in Table 6.4 along with the standard deviations for the RISO data.

These means are very nearly zero, especially when compared to the means of the DIDO data which are shown in Figures 6.88 through 6.90 for $F_x$, $F_y$, and $M_z$, respectively. The tabulated values for these data plots are presented in Table B.15 in Appendix B starting on page B-15. We conclude that the means of the DIDO training data do not approximate those expected in the force measurement stream when the controllers are implemented. Especially the $F_y$ mean values for the DIDO data are much higher than for the RISO data. To address this condition, we apply the data mirroring technique to force the DIDO training data to have a zero mean for each component of $\vec{F}$.

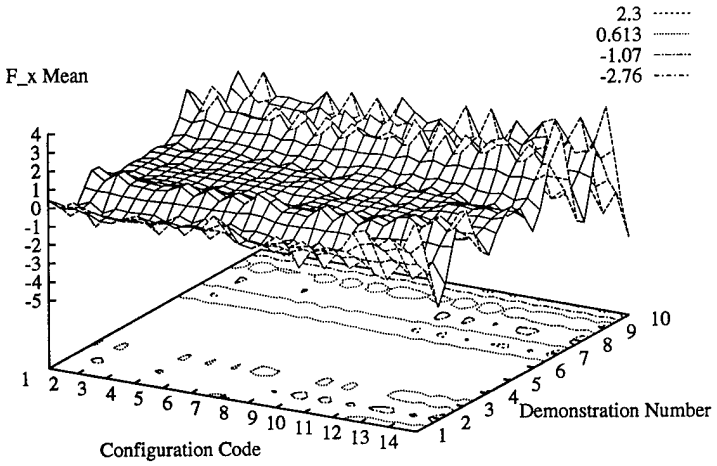



Figure 6.88 Mean values for the $F_x$ components of $\vec{F}$ for DIDO data collected on the PLIMMS manipulator. Ten different demonstration files are examined as a function of the data processing steps applied.

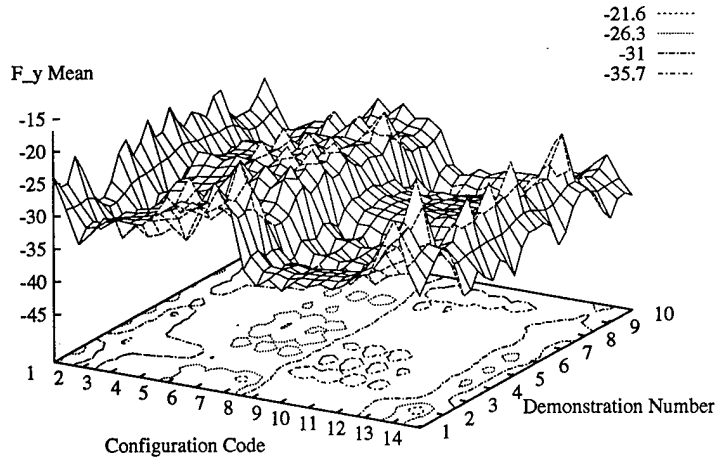



Figure 6.89 Mean values for the $F_y$ components of $\vec{F}$ for DIDO data collected on the PLIMMS manipulator. Ten different demonstration files are examined as a function of the data processing steps applied.

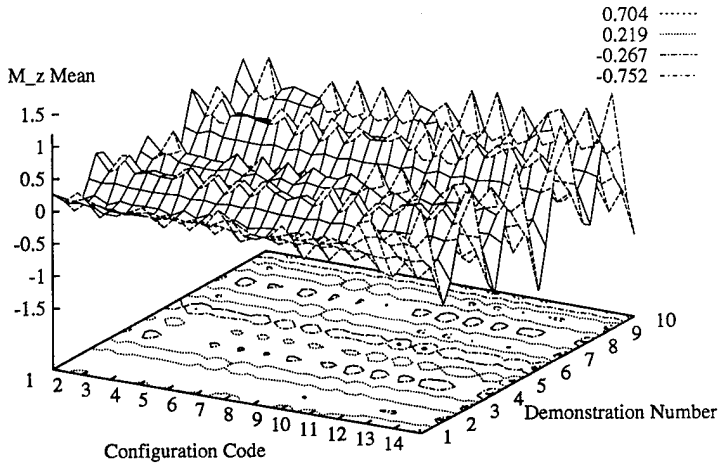



Figure 6.90 Mean values for the $M_z$ components of $\vec{F}$ for DIDO data collected on the PLIMMS manipulator. Ten different demonstration files are examined as a function of the data processing steps applied.

Figure 6.91 shows the resulting performance metrics of the ANN controllers that were training on mirrored DIDO data. The erratic shape of the plot surface conveys no trends between the configuration of the data tested and the performance metrics. Note that only configurations 1-6 and 10 were mirrored and tested. The salient feature to note from the mirroring results is that there were far more successful controllers for each of the seven tested configurations after mirroring the data than there were prior to mirroring it. In fact, whereas there were at most two successful controllers resulting from any given configuration prior to mirroring, there were 10-17 successful controllers for each configuration after mirroring. This indicates that the mirroring technique can indeed have a beneficial effect on the robustness of training even when we know the mapping itself is poor. We conclude that the poor results of the original DIDO-trained controllers were a product of both bad input distribution (large mean values) and the bad I/O-mapping. Data mirroring has resolved at least part of the detrimental effects of a biased DIDO data set.

*6.5.3 DIDO Summary.* The DIDO training and testing was not nearly as successful as SISO, RISO, or DISO. Initial attempts to collect DIDO data using the PUMA manipulator

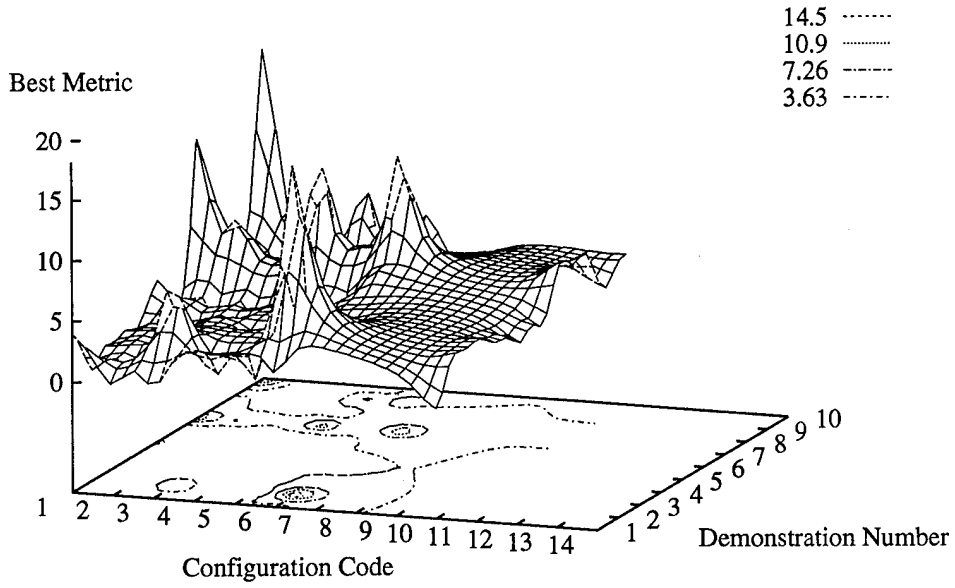



Figure 6.91 Best performance metric, $\zeta$, achieved for ANN controllers trained on DIDO data after they were *mirrored*. Ten different demonstration files are examined as a function of the data processing steps applied. Table 6.1 provides the translations for the configuration codes used. Note that only configuration codes 1-6 and 10 were mirrored and tested. Other codes are set to zero and included to enhance plot labeling uniformity.

resulted in trained ANN controllers which were determined to rotate the peg irregardless of the contact force state. Attributing this behavior to the fact that the operator could not really feel the contact forces during demonstration because of the difficulty in back-driving the PUMA, the PLIMMS was designed and built. The relative ease with which the PLIMMS could be moved allowed the operator to feel the contact forces and perform the demonstrations much better. However, even the PLIMMS data does not correctly depict an accommodation mapping in all the axes required. In particular, the $y$-axis data was conspicuously lacking any indication of a compliant response to measured forces. This is a concern that would most likely be addressed by a change in the controller architecture in future work.

After trying to use the LSMF interrogation technique in concert with the four matrix similarity indexes to predict the performance potential of the ANN controllers or to evaluate the quality of the DIDO training data, no confidence was gained in its feasibility. Though the matrix similarity evaluation of the training data implied that some of the demonstration samples should be better or worse than others, only a few of the ANN controllers trained on any of the DIDO data were able to complete the task well enough to earn a performance metric. The five successful controllers out of the 560 that were trained were not significant enough to draw any correlations, since they were spread among four different training data configurations.

Though the ANN controllers were generally impotent, the $\mathcal{A}_a{}'$ extracted from them were moderately successful. There were 22 successful $\mathcal{A}_a{}'$ controllers for the 140 tested. This implies that the nonlinearity of the ANN may be degrading its performance in this linear edge mating task, since the best-fit linear mapping approximation of the $\mathcal{A}_a{}'$ did better. However, even the performance of the $\mathcal{A}_a{}'$ controllers pale when compared to that of the ANN controllers trained on mirrored DIDO data. For the mirrored DIDO data, 108 out of 280 controllers were able to complete the task. This is an order of magnitude improvement over the results obtained for the original DIDO data. Two corrupting characteristics of the training data have been shown to have a deleterious effect on training: a biased input vector distribution and an inconsistent or incorrect I/O mapping. However, the DISO results proved that a biased input vector distribution can be easily overcome if the I/O mapping is very good. When compared to the very successful DISO results, the moderate improvement obtained by

mirroring the DIDO data indicates that the integrity of the I/O mapping is more important to controller success than the input feature vector distribution. Thus, future effort should concentrate on establishing the integrity of the I/O mapping in preference to correcting input bias.

# *VII. Conclusions*

Although considerable insight has been gained into the difficulties associated with transferring human skills to robots, our proposed scheme of using an ANN to learn accommodation tasks which can be added to a "toolbox" of local autonomy was not found to be robust. The ANN controllers trained on SISO, RISO, and DISO data had little trouble learning to complete the edge-mating task in either the simulation or on the actual PUMA robot manipulator. Despite these successes, in the best case fewer than half of the trained ANN controllers could perform the task successfully when trained on the DIDO data collected. Part of the problem was due to the poor quality of demonstration exhibited by the DIDO data, and part was attributed to characteristics of the ANN architecture used. These deficiencies are detailed below followed by conclusions concerning several of the data processing options and the results of efforts to use matrix interrogation to screen training data and predict controller performance.

## *7.1 Demonstration Data Quality.*

The main problem with training ANN controllers from human demonstration is the poor quality of the required information in the demonstration data. Initial experience with collecting training data using the PUMA manipulator identified the need for the data collection system to be as transparent to the operator as possible because difficulty in back-driving the PUMA completely corrupted the operator's ability to sense the contact forces between the peg and the table...the very forces to which he was supposed to respond. As a result, the controller performed in the same way as the human did by rotating the peg regardless of what force vector was measured. To cope with this problem, the PLIMMS was designed and built to back-drive more easily and to be lighter so the operator could feel the contact forces. Though the collected data from the PLIMMS seemed better, there were still several concerns about the demonstration data which were anticipated in Section 3.3.4. An additional problem, however, is that additional sensory information beyond the sensed force is typically used in determining the commanded velocity. This is partly due to the availability

of visual and auditory information[1]. For this simple edge-mating task, the mental model of the task goal combined with the visual information was enough to completely determine the motion required.

### 7.2 *Effectiveness of Data Processing Options.*

Although clipping training vectors out of a data set using the Lipschitz ratio as a criterion may improve trainability, Lipschitz clipping is not useful as a criterion for pruning out training I/O pairs that depict an inconsistent I/O mapping. Consistency in the I/O mapping was found to be a far more critical feature of the training data than the simple continuity of the mapping, but since the Lipschitz ratio varies over the input range even when the I/O mapping is perfectly consistent, there is no benefit from using the Lipschitz ratio to precondition the training data. In none of the cases tested did Lipschitz clipping significantly change the success of the trained controller.

Velocity pruning is a technique to reduce the number of training exemplar vectors which mapped to small magnitude velocities. This technique can prevent the inclusion of training vectors depicting a many-to-one mapping from $\vec{F}$ to $\vec{V}$. However, choosing the threshold, $V_t$, is a sensitive decision because too large a $V_t$ can cause all the training vectors depicting the desired terminal state for the edge-mating task to be lost. ANN controllers trained on such decimated data files will commonly overshoot the aligned position and fail to properly complete the task. Thus, velocity pruning is not worthwhile in the form used for the present work. An enhancement to the velocity pruning algorithm that may make it more useful would be to prune only the vectors prior to the first alignment of the peg in the training data file. Detection of alignment might require some innovative technique without additional sensor data, but the advantage would be that one would no longer have to be concerned about deleting all the examples of how to terminate the task. Another possible fix would be to simply reinsert some examples of the terminal condition after velocity pruning was applied to a training data file. Neither of these ideas was investigated in the present work.

[1]Visual and auditory information could easily be denied the person performing the task when the DIDO data are collected. See Section VIII for ideas on how this could be accomplished.

Training normalization allows the ANN to train well, but introduces a significant problem of matching the training and implementation statistics for the input feature vectors. This stems from the fact that the same component-wise normalization statistics (mean and standard deviation) are used to prenormalize the input measurement stream of force data when the controller is implemented as were computed for the training data. If those statistics are not proper for the measurement stream, they could force the ANN into an underdeveloped or never-before-seen region of the input space. Consequently, erratic outputs may occur which would defeat the performance of the implemented ANN controller. The goal is to make the training data as similar to (or as representative of) the expected measurement stream of data as possible. Since the magnitude of the forces proves to be relatively small for the accommodation matrix controller when implemented, we can approximate the mean for our measurement stream as zero for the edge-mating task.

Mirroring training data vectors originally having a non-zero mean about all the axes was shown to improve the performance potential of controllers. This was determined by exploring the effects of introducing a bias to SISO training data and then mirroring those data, thereby forcing them to have a zero mean without modifying the mapping. The resulting ANN controllers trained on these mirrored SISO data were shown to have more successes and generally better performance metrics than controllers trained on the original biased data.

The benefits of mirroring was also evaluated for DIDO training data. The original DIDO training data had significant biases for all the axes of input and output. After mirroring the data, the number of ANN controllers which could successfully complete the edge-mating task increased dramatically. This showed that mismatched training and implementation statistics for the input feature vectors was at least partly to blame for the dismal DIDO training results. It was not fully to blame, however, because all of the ANN controllers trained on DISO data sets were able to successfully complete the task. This indicates that if the I/O mapping of the DIDO data were as consistent and correct as that of the DISO, the DIDO data would also work well.

*7.3 Matrix Interrogation Investigations.*

The ideal scenario of operation presented in Section 1.2 depicted training an ANN controller using just a few examples of a human demonstration with some appropriate data preprocessing to ensure the data would result in a successful controller. If we concede that success cannot necessarily be assured, another possible approach is to handle the uncertainty by evaluating each trained controller off-line to determine if it will work. Several alternatives were investigated towards this end. They were based on interrogating the ANN to determine what kind of accommodation matrix it had learned to emulate. Unfortunately, the ANN performance could not be fully predicted with either method.

The first method of interrogation, called UVP, was found to be a poor indicator because it relied on only a 3-point sample of the ANN's I/O mapping to derive the equivalent matrix, $\mathcal{A}_a{}^*$. In addition, it attempted to use a linear relationship to extract information from a nonlinear ANN with just those three data points. The problem was further complicated by the fact that the desired $\mathcal{A}_a$ was known to lack full rank. Overall, the UVP method was useless.

The second matrix interrogation method, called LSMF, was found to work well if the data it was interrogating represented a consistent linear relationship. It was used to verify the perfect linear mapping of training data which were synthesized using a given $\mathcal{A}_a$. However, when applied to I/O pairs generated by testing a trained ANN, the LSMF method had difficulties. Even if the ANN had been trained on SISO data generated with a perfectly consistent $\mathcal{A}_a$ mapping, the extracted $\mathcal{A}_a{}'$ matrices were not found to be consistently similar to the original $\mathcal{A}_a$. The similarity of the matrices was judged using four indexes of similarity we developed which measured the structural similarity, the gain similarity, the sign similarity, and the ratio similarity[2]. The results were found to be an intractable function of the number of I/O pairs taken for each matrix sample fitted as well as a function of the particular input vectors selected to generate the I/O pairs. The latter dependency was found to be equivalent to the problem of matching the training data statistics which was mentioned above.

[2]See Section 5.4.4 for explanations of the four similarity indexes.

Given these two observations, synthetic data were used to investigate the potential of using $\mathcal{A}_a'$ to predict the controller performance of an ANN. This was done by extracting a single $\mathcal{A}_a'$ from an entire set of I/O pairs generated by using the same data used for training and implementing it as an accommodation matrix controller. The resulting $\mathcal{A}_a'$ could sometimes resemble the performance of the ANN controller, but it could never match the ANN controller's performance. In most cases, the overall gain of the $\mathcal{A}_a'$ was wrong, even though it was otherwise similar to $\mathcal{A}_a$. In numerous cases, the ANN controllers trained on synthetic data outperformed even the original $\mathcal{A}_a$ used to generate its training data. This indicated that the ANN controllers were making use of their nonlinear mapping capabilities. Therefore, the performance of the $\mathcal{A}_a'$ is not a reliable prediction of the performance potential of the ANN controller from which the $\mathcal{A}_a'$ was interrogated.

After finding that the $\mathcal{A}_a'$ did not reproduce the behavior of the ANN controllers, the four measures of similarity were examined to see if any of them could predict the performance potential of the ANN controller by revealing how close the ANN mapping resembled $\mathcal{A}_a$. Unfortunately, none of them was shown to have a consistent correlation to the performance metric. This is most likely due to the nonlinearity of the ANN which allows it to maintain a performance edge over the accommodation matrix controllers but complicates the determination of any consistent correlation.

When combined with the similarity indexes, the LSMF matrix interrogation technique could not reliably screen training data for I/O mapping consistency either. It was anticipated that a small matrix fitting window could be used for the interrogation and trends in the indexes could be evaluated over a particular task demonstration to see if the mapping matrix approximating the data was consistent. However, the similarity indexes varied considerably even when applied to DISO data which had a perfectly consistent mapping. Part of the problem was that fitting window size dependency could not be resolved into a rationale for selection of the window size.

### *7.4 ANN Training Difficulties.*

Since the ANN architecture and training algorithm were not the focus of this research, a very simple structure and training method was used. Of course, when using a simple train-

ing algorithm, one may encounter difficulties in getting the desired training accomplished. Ultimately, one would like the ANN off-line training to be as rapid and robust as possible so task skills could be rapidly added to the teleoperator toolbox mentioned in Section 1.2. The present back-propagation training algorithm is relatively slow and could be improved[3]. Part of the difficulty encountered when interrogating trained ANN controllers was that the results were found to be dependent on the input feature vectors applied to the ANN when generation the I/O pairs of data to which the LSMF technique was applied. When the input vectors from the training data were used to generate the I/O pairs, the results were markedly better than when some other input vectors were used. This indicates that the ANN was not doing a very good job of generalizing away from the data set it had been trained on. This is typically a result which stems from either overtraining or too many nodes in the hidden layer (or both). The possibility of overtraining the ANN was significant since the training termination criteria was unsophisticated. Each of the controllers were trained for the same total number of training vector exposures (3,000,000) so for some cases it was probably too few and for some it may have been too many. In many cases, the results of training were evaluated using the techniques mentioned in Section 5.4 to ensure quality. However, there is not doubt room for improvement which could be obtained by using training techniques which are faster and more robust.

### *7.5 Summary.*

Formulating a simple ANN controller which can watch a typical human perform an accommodation task a few times and thereby acquire that skill for a robot remains an elusive goal. This is especially true if we want the trained controller to be able to perform the demonstrated task at any reachable position in the manipulator's workspace. The desire for configuration-invariance and generalizability were major criteria for our selection of the I/O features, the ANN architecture, and the controller configuration. In the end, it appears that the combination is not able to reliably produce a working controller, though it may produce some successes. The analysis showed that if the training data faithfully depicted an accommodation mapping, as is the case for DISO data, then the ANN could be easily

[3] See Section VIII for some ideas on how to improve the training speed and reliability.

trained to successfully control the manipulator, regardless of the input distribution problems identified for the DIDO data. However, the observable quantities which can be measured during a human demonstration of the edge-mating task appeared to lack some of the basic accommodation information that the ANN needs to learn to be successful. Techniques that reinforce or restore a consistent and correct accommodation mapping to DIDO data will significantly benefit the feasibility of using the proposed controller development scheme. Without such a technique, however, the DIDO results showed that simply correcting the input distribution problem will improve the training and testing performance moderately. The mirroring technique is a simple way to obtain improved performance without collecting any additional data.

The analysis presented in this dissertation has identified characteristics of human demonstration data for accommodation tasks and determined several data processing options that can improve the chances of success in using an ANN controller to acquire demonstrated human skills. The ANN controller was shown to work robustly in both simulation and on a PUMA 562 manipulator when the ANN was trained on synthesized data sets. In addition, the most profitable future refinements have been identified as those which address improving the integrity of the I/O mapping exhibited by the training data. With these insights we are now one step closer to the goal of having an easy technique to transfer human skills to robots in support of increasing the level of local autonomy for telerobotic systems.

## *VIII. Recommendations*

The results of this dissertation have shown that more work remains before an ANN controller can reliably acquire even simple accommodation skills from human demonstrations of a task. Though the road to success seems arduous, the most promising directions of travel have been identified in the feasibility studies of the present work. This chapter delineates the first few steps along each path which should be taken to further our understanding.

Alternative controller architectures should be explored within the present ANN paradigm. Our selection of $\vec{F}$ as the input and $\vec{V}$ as the output and expressing those vectors in tool-frame coordinates was motivated by a desire to have the ANN controller capable of operating at any point in the workspace. That selection of variables made all the force measurements and motion commands relative to the current position of the peg and dictated much of the remaining architecture and methodology explored in the present work. Section VII identified several problems with DIDO training data that hindered the success of ANN controllers trained on that data. One of the key faults identified was the lack of accommodation exhibited in the $Y$-axis. One possible way to cope with the lack of $Y$-axis accommodation is to modify the controller architecture to decouple the motion commands that are learned for each axis and thereby decompose the overall task into smaller subtasks. The outputs of the subtask controllers could then be superimposed to generate the overall controller command.

Decoupling the motion commands would enable one to implement certain subtasks as hard-coded controllers if enough a priori information existed about the subtasks. For instance, the troubling $Y$-axis accommodation that was not learned from the DIDO data could be hard-coded as an accommodation, while the $Z$-axis angular compliance could be learned for the edge-mating task. This would provide a degree of assured safety in the requisite compliant behavior of the manipulator while the ANN subtask was simply to determine the necessary angular alignment motions. For tasks other than edge-mating, some analysis of the task would be required to determine the proper decomposition such that only subtasks that are exhibited in the DIDO data were assigned to the ANN controller.

Another suggestion is to leave the nominal velocity, ${}^{T}\vec{V}_n$, on during the entire contact phase of the task rather than turning it off via the blending function presented in Eq (5.47).

From the DIDO data collected, the ANN controllers were trained to produce small magnitude velocity components in the $Y$-axis direction which would tend to stop peg motion. Referencing Figure 5.8, we note that keeping ${}^{T}\vec{V}_n$ turned on during contact would be equivalent to commanding a moderate constant bias force for the terminal state as opposed to the present controller configuration which tends to have a lower bias force upon task completion. Another byproduct of this configuration of the controller is that it might tend to keep the peg in contact with the table surface better during the alignment process, whereas the present controller typically allows the peg to bounce in and out of contact.

Another quality concern for the DIDO data that was identified in Section VII was that the human operator was probably using sensory information and a mental model which are unavailable to the ANN controller. This means that the DIDO data may not contain enough information for the ANN controller to complete the task, since it was presumably insufficient for the human. To increase the accommodation fidelity in the DIDO data, a better system should be configured to collect the DIDO training data. A device such as the PHANToM™ (built by SensAble Devices, Inc., Vanceburg, KY) should be used to interface the operator with a computer model of the peg and the task environment. The low-mass and easy back-driveability of the PHANToM™ would ensure that higher-fidelity force information was available to the operator when demonstrating the task. Using this system, the operator would be blindfolded to deny visual feedback and wear headphones emitting masking noise to deny auditory information. Further, the position and orientation of the table surface could be randomly varied to deny the operator the benefit of working from a mental model of where the table was located. This system would then be used to conduct a study to conclusively determine how much effect the additional sensory information and mental model had on the quality of the DIDO data. In addition, the computer model of the task would simplify parameter matching between DIDO data collection and controller implementation and testing in simulation.

Some of the data integrity concerns that were mentioned in Section 5.6 may be approached by applying the time shifting and time-delayed inputs feature vectors as mentioned in Section 5.6 and described in detail in Appendix D. It is possible that the time shifting technique can improve the integrity of the I/O mapping in the DIDO data by ensuring that the

proper cause-effect relationship is restored. To properly apply time shifting, one would first have to determine the correct number of data samples to shift the data. This is a nontrivial parameter to determine since there are many factors that can affect the signal propagation and mental processing delay of the human while completing even the simple edge-mating task. As a first-order approximation, one can assume that the causal time delay is constant throughout the task execution. If the inertia of the demonstration collection system is small, then one could begin testing by using the average human choice reaction time to tactile stimulation, which is approximately 300-400 milliseconds [55]. The time-delayed inputs are a significant modification to the control algorithm which might preserve the context of the input force vectors and provide information about the change in the force as well as the current force vector. The current results give no particular reason that the context of the force is important for the edge-mating task, but it may become important for other tasks such as the chamferless peg insertion which can transition from single-point contact to multiple-point contact during insertion.

Some difficulty was encountered when training the ANN controllers. This was largely due to the simplicity of the ANN architecture and training algorithm. The back-propagation training algorithm used in this work is among the simplest approaches to ANN training in existence. As a result, it is not a very robust algorithm as compared with some enhanced back-propagation training techniques which are available. One particular algorithm that offers much more rapid training is Levenberg-Marquardt optimization which is available as a function in the Neural Network Toolbox [17] that is part of MATLAB®. It offers much more rapid training convergence than the gradient decent of back-propagation at the expense of large memory consumption for large network architectures. For the ANN architecture used in this research, the network size should not challenge the memory capacity of most common workstations. In addition to using Levenberg-Marquardt optimization for training, other methods of terminating the training session should be explored. The present work simply used a certain total number of training exposures[1] which could prematurely end training

[1]A training exposure is defined here as the presentation of a single I/O pair of data vectors and the corresponding adjustment of ANN weights based on the output error. Thus, 100 training vectors, each seen 200 times would constitute 20,000 training exposures.

or, just as likely over train a network. The combination of these two enhancements could significantly ease the task of training successful controllers.

The ideas presented here represent several avenues of future exploration and enhancement of the present algorithm. They could be pursued individually, or one might be so ambitious as to attempt them all in concert. The probability that the DIDO-trained ANN controllers resulting from these enhancements will be more successful than those trained and tested in this dissertation is very good.

*Bibliography*


1. Albus, J. "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller," *ASME J. of Dynamic Systems, Measurement and Control*, *97*:270–277 (1975).
2. Asada, H. "Teaching and Learning of Compliance Using Neural Nets: Representation and Generation of Nonlinear Compliance." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 1237–1244. May 1990.
3. Asada, H. and H. Hanahusa. "Playback Control of Force Teachable Robots," *Trans. of Society of Instrument and Control Engineers, 15-3* (1979).
4. Asada, H. and H. Izumi. "Direct Teaching and Automatic Program Generation for the Hybrid Control of Robot Manipulators." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 1401–1406. 1987.
5. Asada, H. and S. Liu. "Transfer of Human Skills to Neural Net Robot Controllers." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 2442–2448. 1991.
6. Asada, H. and B.-H. Yang. "Skill Acquisition From Human Experts Through Pattern Processing of Teaching Data." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 1302–1307. 1989.
7. Barto, A. G., et al. "Associative Search Networks: A Reinforcement Learning Associative Memory," *Biological Cybernetics*, *40*:201–211 (1981).
8. Benady, M., et al. "Robot Learning of Contact Tasks." *Robotic Systems and AMT: Proceedings of the Int. Conf. on CAD/CAM and AMT*, edited by G. Halevi. 147–159. Dec 1990. NTIS Accession No: N93-15303/9/XAB.
9. Buckley, S. J. "Teaching Compliant Motion Strategies," *IEEE Trans. on Robotics and Automation*, *5*(1):112–118 (February 1989).
10. Burden, R. L. and J. D. Faires. *Numerical Analysis* (3rd Edition). Boston: Prindle, Weber, & Schmidt, 1985.
11. Caine, M. E. *Chamferless Assembly of Rectangular Parts in Two and Three Dimensions*. MS thesis, Massachussetts Inst. of Tech., 1985. Dept. of Mech. Eng.
12. Caine, M. E., et al. "Assembly Strategies for Chamferless Parts." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 472–477. 1989.
13. Chan, L.-W. and F. Fallside. "An Adaptive Training Algorithm for Back Propagation Networks," *Computer Speech and Language*, *2*:205–218 (1987).
14. Craig, J. J. *Introduction to Robotics: Mechanics and Control* (2nd Edition). Addison-Wesley, 1989.
15. Cybenko, G. "Approximations by Superpositions of a Sigmoidal Function," *J. on Mathematics of Control, Signals, and Systems*, *2*:303–313 (1989).
16. De Fazio, T. L., et al. "The Instrumented Remote Center Compliance," *The Industrial Robot*, *11*(4):238–242 (1984).

17. Demuth, H. and M. Beale. *Neural Network TOOLBOX User's Guide*. Natick, MA: The MathWorks, Inc., 1994.

18. DSP Development Corp., One Kendall Square, Cambridge MA 02139. *The DADiSP Worksheet Reference Manual*, November 1991.

19. DSP Development Corp., One Kendall Square, Cambridge MA 02139. *The DADiSP Worksheet Workstation User Manual*, July 1992.

20. Fu, K. S., et al. *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill, 1987.

21. Gullapalli, V. "A Stochastic Reinforcement Algorithm for Learning Real-Valued Functions," *Neural Networks*, *3*:671–692 (1990).

22. Gullapalli, V. "Learning Control Under Extreme Uncertainty." *Advances in Neural Information Processing Systems 5*. San Mateo, CA: Morgan Kaufmann, 1993.

23. Gullapalli, V., et al. "Learning Admittance Mappings for Force-Guided Assembly." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 2633–2638. May 1994.

24. Gullapalli, V., et al. "Learning Reactive Admittance Control." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 1475–1480. May 1992.

25. Handelman, D. A., et al. "Integrating Neural Networks and Knowledge-Based Systems for Robotic Control." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 1454–1460. 1989.

26. Hecht-Nielson, R. *Neurocomputing*. Addison-Wesley, 1990.

27. Houpis, C. H. and G. B. Lamont. *Digital Control Systems: Theory, Hardware, Software*. New York: McGraw-Hill, 1985.

28. Hush, D. R. and B. G. Horne. "Progress in Supervised Neural Networks: What's New Since Lippmann," *IEEE Signal Processing Magazine*, *10*(1):8–39 (January 1993).

29. Hush, D. R., et al. "Error Surfaces for Multi-Layer Perceptrons," *IEEE Trans. on Systems, Man and Cybernetics*, *22*(5) (1992).

30. JR3, Inc., Woodland, CA. *JR$^3$ Universal Force-Moment Sensor System Operation Manual*, August 1989.

31. Kitagaki, K., et al. "Methods to Detect Contact State by Force Sensing in an Edge Mating Task." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 701–706. 1993.

32. Lang, K.J., et al. "A Time Delay Neural Network Architecture for Isolated Word Recognition," *Neural Networks*, *3*:23–43 (1990).

33. Leahy, Jr., M. B. *ARCADE Users Guide: Version 2.0*. Technical Report ARSL-89-4, Air Force Inst. of Tech., August 1989. Dept. of Elect. and Comp. Eng.

34. Leahy, Jr., M.B., et al. "Evaluation of Dynamic Models for PUMA Robot Control," *IEEE Trans. on Robotics and Automation*, *5*(2):242–244 (April 1989).

35. Lippmann, R. "An Introduction to Computing with Neural Nets," *IEEE Acoustics, Speech, and Signal Processing Magazine*, *4*(2):4–22 (April 1987).

36. McCarragher, B. J. and H. Asada. "A Discrete Event Controller Using Petri Nets Applied to Robotic Assembly: The Desired Velocity Commands." *Proc. of the American Controls Conf.*. 2473–2478. 1992.

37. McCarragher, B. J. and H. Asada. "A Discrete Event Approach to the Control of Robotic Assembly Tasks." *Proc. of the IEEE Int. Conf. on Robotics and Automation1*. 331–336. 1993.

38. Nakamura, Y. *Theoretical Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

39. Narendra, K. S. and K. Parthasarathy. "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. on Neural Networks*, *1*(1):4–27 (March 1990).

40. Parsons, T. W. *Voice and Speech Processing*. New York, NY: McGraw-Hill, 1987.

41. Payandeh, S. "Causality and Robotic Contact Tasks Problem," *IEEE Trans. on Systems, Man, and Cybernetics*, *22*(5):1210–1214 (Sept/Oct 1992).

42. Peshkin, M. A. "Programmed Compliance for Error Corrective Assembly," *IEEE Trans. on Robotics and Automation*, *6*(4):473–482 (August 1990).

43. Press, W. H., et al. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge Univ. Press, 1988.

44. Rogers, S. K. and M. Kabrisky. *An Introduction to Biological and Artificial Neural Networks for Pattern Recognition, TT 4*. SPIE Optical Engineering Press, 1991.

45. Rummelhart, D. E. and J. L. McClelland, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, 1986.

46. Shahinpoor, M. and H. Zohoor. "Analysis of Dynamic Insertion-Type Assembly for Manufacturing Automation." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 2458–2464. 1991.

47. Sheridan, T. B. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, 1992.

48. Spong, M. W. and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, 1989.

49. Tarn, T.J. and A.K. Bejczy. *Dynamic Equations for PUMA-560 Robot Arm*. Technical Report SSM-RL-85-02, Washington Univ., St. Louis, MO, July 1985. Dept. of Syst. Sci. and Math.

50. Unimation Inc. *PUMA Mark II Robot 500 Series Equipment Manual*, March 1985.

51. Vaaler, E. G. and W. P. Seering. "A Machine Learning Algorithm for Automated Assembly." *Proc. of the IEEE Int. Conf. on Robotics and Automation*. 2231–2237. 1991.

52. Waibel, A. "Consonant Recognition by Modular Construction of Large Phonemic Time-Delay Networks." *Proc. of IEEE Conf. on Neural Information Processing Systems*. 1988.

53. Waibel, A. "Modular Construction of Time-Delay Neural Networks for Speech Recognition," *Neural Computation*, *1*(1):39–46 (Spring 1989).

54. Wasserman, P. D. *Neural Computing: Theory and Practice*. Van Nostrand Reinhold, 1989.

55. Whitlow, J. W., et al. "Operator Motor Control." *Engineering Data Compendium: Human Perception and Performance* edited by K. R. Boff and J. E. Lincoln, 1841–2011, Wiley & Sons, 1988.

56. Whitney, D. E. "Quasi-static Assembly of Compliantly Supported Rigid Parts," *ASME J. of Dynamic Systems, Measurement, and Control, 104*(3):65–77 (1982).

57. Whitney, D. E. "Part Mating in Assembly." *Handbook of Industrial Robots* edited by S. Y. Nof, 1084–1116, Wiley & Sons, 1985.

58. Whitney, D. E. "The Remote Center Compliance." *The Encyclopedia of Robotics* edited by S. Y. Nof, New York: John Wiley and Sons, 1988.

59. Whitney, D. E. "A Survey of Manipulation and Assembly." *Robotics Science* edited by M. Brady, chapter 8, 291–348, MIT Press, 1989.

60. Whitney, D. E., et al. "Designing Chamfers," *Int. J. of Robotics Research, 2*(4):3–18 (1983).

61. Whitney, D. E. and J. M. Rourke. "Mechanical Behavior and Design Equations for Elastomer Shear Pad Remote Center Compliances," *ASME J. of Dynamic Systems, Measurement, and Control, 108*(3):223–232 (1986).

62. Yang, J., et al. *Hidden Markov Model Approach to Skill Learning and Its Application to Telerobotics*. Technical Report CMU-RI-TR-93-01, Robotics Institute, Carnegie Mellon Univ., January 1993. Available via DTIC as A266-989.

*Vita*

Captain Paul V. Whalen was born on 30 July 1962 in Louisville, Kentucky. In 1980 he graduated as Salutatorian from Jesse Stuart High School in Louisville. In that same year he entered Purdue University's School of Engineering in West Lafayette, Indiana on a four-year Air Force ROTC scholarship. While attending Purdue, he was initiated into the mechanical engineering honorary fraternity, Pi Tau Sigma. He received the degree of Bachelor of Science in Mechanical Engineering from Purdue in May 1984. Upon graduation he was designated a distinguished ROTC graduate and received a Regular Air Force commission. His first assignment was to Eglin AFB, Florida in July 1984 where he served as a Research and Development Test Engineer in the Terminal Effects Branch of the Munitions Test Division under the 3246th Test Wing. While stationed at Eglin he was awarded the AF Commendation Medal and the AF Achievement Medal for outstanding service. After three years of service at Eglin AFB, Capt Whalen was selected to enter the masters degree program at the Air Force Institute of Technology (AFIT), Wright-Patterson AFB, OH. He received the master of science degree in aeronautical engineering in December 1988 and immediately entered the doctoral degree program at AFIT. In January 1992 Capt Whalen was assigned to the Bioacoustics and Biocommunications Division of the Crew Systems Directorate at the Armstrong Laboratory located at Wright-Patterson AFB. While at Armstrong Laboratory, he directed the Human Sensory Feedback for Telepresence program conducting research into the man-machine interface issues of providing force and tactile feedback information to operators of telerobotic systems and users of virtual reality systems. Capt Whalen is a registered Professional Engineer in the state of Ohio.

Permanent address: 575 Milan Road
Payneville, KY 40157

## *Appendix A. Artificial Neural Network Computations.*

This appendix will present the feedforward calculations one uses to compute the outputs of a MLP ANN and the backward error propagation algorithm used for training the ANN. The algorithms are presented here in summary form. For a more detailed explanation of how ANN s operate and explanations of other training algorithms, see [54], [35], [44] or [28].

Figure 3.5 shows a simple schematic of the MLP ANN used for this dissertation. It is a fully-connected, two-layer ANN. *Fully-connected* means that all of the input nodes are connected to each of the hidden nodes, and all of the hidden nodes are connected to each of the output nodes. Some authors might call the network depicted in Figure 3.5 a three-layer ANN. However, this author refers to it as a *two-layer* ANN because only two of the three "apparent" layers perform transformations on the data. The input nodes are simply connections for the input features to be applied and do not perform any transformations on the data. Networks with additional hidden layers are possible, and in some cases desirable, but Cybenko showed that any nonlinear mapping can be approximated to an arbitrary accuracy by choosing enough hidden layer nodes in a simple two-layer network [15]. With this in mind, we have chosen to limit the architecture of the present network to two layers.

### *A.1 Feedforward Computations.*

When a vector of input features is presented to the ANN, simple feedforward equations are used to compute the network output. Since the layers of the network are cascaded, the computations of each layer can be considered one at a time in turn. Consequently, for the two-layer network under discussion, the hidden layer computations will be described first and then those of the output layer will follow.

*A.1.1 Hidden Layer Computations.* Each neuron in the hidden layer forms a weighted sum of the inputs to the neural network. The vector of inputs, $\vec{F}$, has $k$ components, $\{f_1, f_2, f_3, \cdots, f_k\}$. For each input, $f_i$, to the $j$th hidden layer node there is a *weight*, $w_{ji}$, that is multiplied by $f_i$ and added to a sum for the $j$th node. If there are $m$ hidden nodes and $k$ input nodes, the weights can be assembled into a *weight matrix*, $\mathcal{W}_h$, with $m$ rows

and $k$ columns while the inputs can be concatenated into a vector, $\vec{F}$, of length $k$. Then the $m$-length vector of weighted sums, $\vec{s}$, can be written as

$$\vec{s} = \mathcal{W}_h \vec{F} \tag{A.1}$$

An $m$-length vector of *bias* or *offset* values, $\vec{\psi}_h$, is then added to $\vec{s}$ to get a new $\vec{s}^*$. Finally, in order for Cybenko's theorem mentioned above to be true, at least one layer of the network must have a nonlinear transfer function applied to its output. For the hidden layer, a nonlinear *activation function*, $g_h(\cdot)$, operates on each element of $\vec{s}^*$ to generate the $n$-length vector of *hidden layer outputs*, $\vec{\varsigma}$. Mathematically, the output of the hidden layer is

$$\vec{\varsigma} = g_h(\vec{s}^*) = g_h(\mathcal{W}_h \vec{F} + \vec{\psi}_h) \tag{A.2}$$

The nonlinear activation function used for the hidden layer is the sigmoid function[1] which can be expressed as

$$g_h(\chi) = \frac{1}{1 + e^{-\chi}} = \frac{1}{2}(1 + \tanh \frac{1}{2}\chi) \tag{A.3}$$

and is shown in Figure A.1. This nonlinear function is selected because it has a continuous derivative with a simple expression and it has been used successfully in many applications.

*A.1.2 Output Layer Computations.* The computations for the output layer nodes differ in two ways from those of the hidden layer. First, the dimensions of the output weight matrix, $\mathcal{W}_o$, and the output bias vector, $\vec{\psi}_o$, differ from those of the corresponding variables for the hidden layer. If there are $n$ output nodes, then the output weight matrix, $\mathcal{W}_o$, is $n$ rows by $m$ columns while the length of the output layer bias vector, $\vec{\psi}_o$, is $n$. Thus, the $n$-length vector of *output layer outputs*, $\vec{V}$, is

$$\vec{V} = \mathcal{W}_o \vec{\varsigma} + \vec{\psi}_o \tag{A.4}$$

Comparing Eqs (A.2) and (A.4), we note that $\vec{\varsigma}$, rather than $\vec{F}$, is the input for the output layer.

[1] Sometimes the sigmoid function is incorrectly called a logistic function.

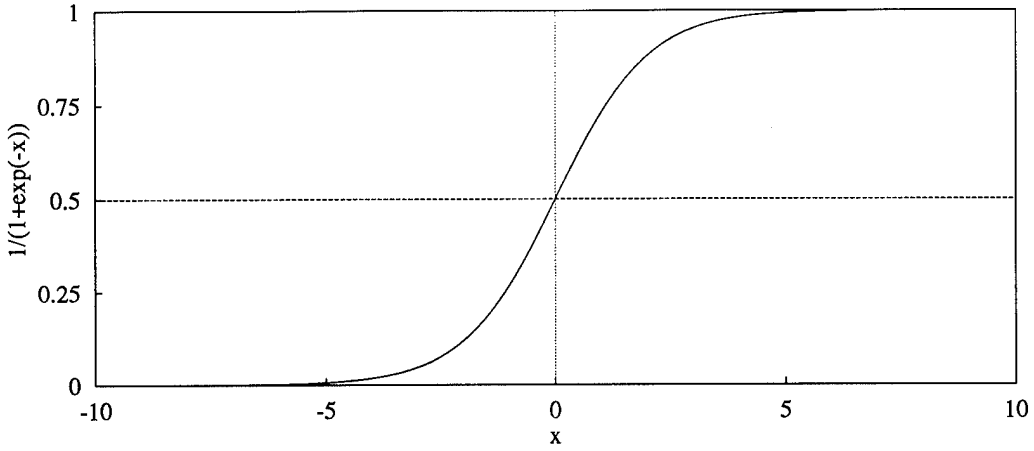



Figure A.1 Plot of the nonlinear sigmoid activation function

The second difference in the output layer computations is that the nonlinear activation function used on the hidden layer is replaced with a unity gain. Networks which try to classify the input features into one of several discrete output classes commonly use nonlinear activation functions on the output layer to squash the output and improve class separation. A unity gain (which is a form of linear activation function) is used on the output of our network because we want a continuously variable output which is not distorted across its range of values. Consequently, Eq (A.4), without modification, reflects the final output of the output layer.

A simple mathematical manipulation can combine Eqs (A.2) and (A.4) to get

$$\vec{V} = \mathcal{W}_o \left[ g_h(\mathcal{W}_h \vec{F} + \vec{\psi}_h) \right] + \vec{\psi}_o \tag{A.5}$$

which is the overall vector equation for computing the output of the two-layered network.

### *A.2 Back Error Propagation Training Algorithm.*

Before an ANN controller can be implemented, the weights and biases must be determined. The process of determining the weights and biases which provide the best input-output mapping is called training. Training algorithms fall into one of two categories;

unsupervised and supervised. Unsupervised training does not require apriori knowledge of the correct network output for each input. Input vectors are presented to the network, and the network *self-organizes* by adjusting its weights and biases according to a well-defined algorithm. For more information on unsupervised training algorithms see [54] or [35].

For supervised training, the correct output vector must be known apriori for each input feature vector used during training. All supervised training algorithms follow the same basic process:

- a sample input feature vector is presented to the ANN
- the ANN output is computed with the current weights and biases
- the computed output is compared to the known correct output
- the weights and biases are then adjusted in such a way as to reduce the difference between the computed and correct outputs

Prior to beginning the actual training session, training data must be generated. The training data is generated in the form of input training vectors, $\vec{F}^*$, and their corresponding output training vectors, $\vec{V}^*$. An input training vector and its corresponding output training vector concatenated together constitute an *exemplar* vector, $\vec{\phi}_i$, which has a length of $(k+ n)$ elements.

$$\vec{\phi}_i = \left(\vec{F}^{*T}, \vec{V}^{*T}\right) \tag{A.6}$$

Note that the $T$ superscript indicates the transpose operator. If there are $p$ training exemplars available, then the *training data matrix*, $\Phi$, can be formed which has $p$ rows and $(k+ n)$ columns.

$$\Phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix} \tag{A.7}$$

Once the training data have been generated and assembled into the training data matrix, one can begin the supervised training session. For a fully-connected multi-layered perceptron network, there are several supervised training algorithms available. The supervised training algorithm chosen for this research is called back error propagation; or

*backprop* for short. The backprop algorithm is the most commonly used and the best characterized algorithm. The backprop training algorithm begins by setting all the weights and biases to random initial values ranging from -1 to +1. Each exemplar vector, $\vec{\phi}_i$, is then presented to the network one at a time. When each $\vec{\phi}_i$ is presented, Eq (A.5) is used to compute $\vec{V}$. The output error vector, $\vec{\delta}_o$, is then computed as

$$\vec{\delta}_o = \vec{V}^* - \vec{V} \tag{A.8}$$

At each training iteration, the interconnecting weights must be adjusted to reduce error. At the $n$th training iteration, the relationship used to adjust the weights to the $j$th node of the output layer is

$$\vec{w}_j(n+1) = \vec{w}_j(n) + \eta\, \vec{\delta}_o^{\,T} \odot \vec{\varsigma} + \alpha\,[\vec{w}_j(n) - \vec{w}_j(n-1)] \tag{A.9}$$

where $\eta$ is the *training rate coefficient*, $\alpha$ is the *training momentum coefficient*, $j$ takes on index values from zero to $n$, and $\odot$ operator represents an element-by-element vector multiplication exemplified by

$$\begin{Bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{Bmatrix} \odot \begin{Bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{Bmatrix} = \begin{Bmatrix} a_1^2 \\ a_2^2 \\ \vdots \\ a_n^2 \end{Bmatrix}$$

The value of $\eta$ controls how quickly the network will adjust its weights at each iteration while the value of $\alpha$ controls the network's tendency to continue adjusting the weights in the same direction. The values for $\eta$ and $\alpha$ are typically between zero and one. Guidance on choosing good values for $\eta$ and $\alpha$ can be found in [13] or [45]. The bias update relationship for the output layer is

$$\vec{\psi}_o(n+1) = \vec{\psi}_o(n) + \eta\vec{\delta}_o + \alpha\left(\vec{\psi}_o(n) - \vec{\psi}_o(n-1)\right) \tag{A.10}$$

As with the feedforward computations, the weight and bias adjustment equations for the hidden layer differ from those of the output layer. For the hidden layer, the error-related term, $\vec{\delta}_h$, is computed by

$$\vec{\delta}_h = \vec{\Omega} \odot \left\{ \vec{\delta}_o{}^T \mathcal{W}_o \right\}^T \tag{A.11}$$

where

$$\vec{\Omega} = (\vec{\varsigma} - \vec{\varsigma} \odot \vec{\varsigma}) \tag{A.12}$$

The weights connecting the inputs to the $j$th node of the hidden layer are then updated according to

$$\vec{w}_j(n+1) = \vec{w}_j(n) + \eta\, \vec{\delta}_h{}^T \odot \vec{F} + \alpha\, [\vec{w}_j(n) - \vec{w}_j(n-1)] \tag{A.13}$$

while the bias update relationship for all the nodes of the hidden layer is

$$\vec{\psi}_h(n+1) = \vec{\psi}_h(n) + \eta\, \vec{\delta}_h + \alpha \left[ \vec{\psi}_h(n) - \vec{\psi}_h(n-1) \right] \tag{A.14}$$

To train the ANN, then, one must use the feedforward calculations in Eq (A.5) to compute the output for each training vector input. The output error is then computed using Eq (A.8) and the output layer weights and biases are updated according to Eqs (A.9) and (A.10), respectively. Finally, the hidden layer weights and biases are updated using Eqs (A.13) and (A.14), respectively.

## *Appendix B. Data Tables*

This appendix contains the tabulated values for many of the data plots presented in the body of this dissertation. They are included for completeness of the results.

Table B.1 Implementation results (best metric) of various distributions of SISO training data after ensuring a consistent training data exposure was maintained between the data sets

| Ranges (x,y,z) | Divisions (x,y,z) | Spacing Function | | | | |
|---|---|---|---|---|---|---|
| | | Even | Sine | Cubic | Complex | Mirrored |
| Figure Number $\Longrightarrow$ | | 6.12 | 6.13 | 6.14 | 6.15 | 6.16 |
| .5,.5,.5 | 5,5,5 | 0.999 | 0.994 | 0.999 | 1.000 | N/A |
| .5,.5,.5 | 4,4,4 | 0.994 | 0.981 | 0.992 | 0.995 | N/A |
| .5,.5,.5 | 3,3,3 | 0.994 | 0.986 | 0.995 | 0.995 | N/A |
| .5,.5,.5 | 2,2,2 | 0.990 | 0.990 | 0.982 | 1.022 | N/A |
| 5,5,5 | 5,5,5 | 1.345 | 0.995 | 1.049 | 0.983 | 0.984 |
| 5,5,5 | 4,4,4 | 1.011 | 0.975 | 0.988 | 1.074 | 1.010 |
| 5,5,5 | 3,3,3 | 0.982 | 0.997 | 0.991 | 0.987 | 0.988 |
| 5,5,5 | 2,2,2 | 1.027 | 0.983 | 0.980 | 0.0 | 0.996 |
| 10,10,10 | 5,5,5 | 0.971 | 0.991 | 1.006 | 1.350 | 0.986 |
| 10,10,10 | 4,4,4 | 0.982 | 0.984 | 1.054 | 1.308 | 1.104 |
| 10,10,10 | 3,3,3 | 0.994 | 0.988 | 0.982 | 1.014 | 0.994 |
| 10,10,10 | 2,2,2 | 1.008 | 0.993 | 0.994 | 0.0 | 0.987 |
| 15,15,15 | 5,5,5 | 1.046 | 1.010 | 1.115 | 1.994 | 0.989 |
| 15,15,15 | 4,4,4 | 0.985 | 0.983 | 1.077 | 1.071 | 1.014 |
| 15,15,15 | 3,3,3 | 0.975 | 0.986 | 1.016 | 1.003 | 0.984 |
| 15,15,15 | 2,2,2 | 1.032 | 0.982 | 0.973 | 0.0 | 1.422 |
| 20,20,20 | 5,5,5 | 1.023 | 0.978 | 1.329 | 1.180 | 1.030 |
| 20,20,20 | 4,4,4 | 1.011 | 1.088 | 1.005 | 0.999 | 0.990 |
| 20,20,20 | 3,3,3 | 1.068 | 0.978 | 1.116 | 0.994 | 0.996 |
| 20,20,20 | 2,2,2 | 1.276 | 0.982 | 0.978 | 0.0 | 1.052 |

Table B.2 Matrix *structural* similarity indexes of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on various distributions of SISO training data.

| Ranges (x,y,z) | Divisions (x,y,z) | Spacing Function | | | | |
|---|---|---|---|---|---|---|
| | | Even | Sine | Cubic | Complex | Mirrored |
| Figure Number $\Longrightarrow$ | | 6.26 | 6.29 | 6.32 | 6.35 | 6.38 |
| .5,.5,.5 | 5,5,5 | 0.002 | 0.001 | 0.001 | 0.003 | 0.000 |
| .5,.5,.5 | 4,4,4 | 0.001 | 0.005 | 0.001 | 0.019 | 0.000 |
| .5,.5,.5 | 3,3,3 | 0.002 | 0.000 | 0.000 | 0.032 | 0.000 |
| .5,.5,.5 | 2,2,2 | 0.003 | 0.000 | 0.000 | 0.006 | 0.000 |
| 5,5,5 | 5,5,5 | 0.098 | 0.052 | 0.002 | 1.032 | 0.094 |
| 5,5,5 | 4,4,4 | 0.003 | 0.023 | 0.001 | 0.656 | 0.102 |
| 5,5,5 | 3,3,3 | 0.106 | 0.019 | 0.000 | 3.707 | 0.024 |
| 5,5,5 | 2,2,2 | 0.003 | 0.015 | 0.005 | 0.399 | 0.068 |
| 10,10,10 | 5,5,5 | 0.354 | 25.478 | 0.042 | 5.651 | 1.205 |
| 10,10,10 | 4,4,4 | 0.156 | 0.320 | 0.069 | 0.979 | 0.559 |
| 10,10,10 | 3,3,3 | 0.057 | 0.043 | 0.011 | 11.196 | 0.211 |
| 10,10,10 | 2,2,2 | 0.012 | 0.000 | 0.107 | 2.167 | 0.112 |
| 15,15,15 | 5,5,5 | 3.885 | 0.214 | 0.024 | 5.794 | 13.558 |
| 15,15,15 | 4,4,4 | 0.010 | 0.166 | 0.024 | 12.162 | 0.508 |
| 15,15,15 | 3,3,3 | 0.310 | 0.100 | 0.005 | 17.138 | 1.683 |
| 15,15,15 | 2,2,2 | 0.065 | 0.057 | 0.007 | 4.336 | 1.222 |
| 20,20,20 | 5,5,5 | 0.145 | 2.480 | 0.109 | 10.076 | 7.890 |
| 20,20,20 | 4,4,4 | 0.004 | 0.030 | 0.380 | 12.079 | 0.500 |
| 20,20,20 | 3,3,3 | 0.184 | 0.075 | 0.630 | 48.649 | 0.333 |
| 20,20,20 | 2,2,2 | 0.022 | 0.014 | 0.127 | 7.975 | 0.018 |

Table B.3 Matrix *gain* similarity indexes of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on various distributions of SISO training data.

| Ranges (x,y,z) | Divisions (x,y,z) | Spacing Function | | | | |
|---|---|---|---|---|---|---|
| | | Even | Sine | Cubic | Complex | Mirrored |
| Figure Number $\Longrightarrow$ | | 6.27 | 6.30 | 6.33 | 6.36 | 6.39 |
| .5,.5,.5 | 5,5,5 | 1.870 | 1.591 | 2.284 | 2.417 | 0.000 |
| .5,.5,.5 | 4,4,4 | 1.834 | 1.577 | 2.204 | 2.342 | 0.000 |
| .5,.5,.5 | 3,3,3 | 1.775 | 1.548 | 2.077 | 2.033 | 0.000 |
| .5,.5,.5 | 2,2,2 | 1.666 | 1.495 | 1.850 | 2.114 | 0.000 |
| 5,5,5 | 5,5,5 | 18.735 | 29.044 | 8.339 | 6.046 | 9.693 |
| 5,5,5 | 4,4,4 | 19.867 | 29.776 | 9.952 | 7.245 | 10.483 |
| 5,5,5 | 3,3,3 | 21.863 | 31.026 | 12.864 | 13.941 | 16.774 |
| 5,5,5 | 2,2,2 | 26.006 | 33.378 | 19.352 | 12.016 | 11.847 |
| 10,10,10 | 5,5,5 | 113.444 | 163.124 | 60.471 | 47.672 | 67.585 |
| 10,10,10 | 4,4,4 | 119.181 | 166.772 | 69.040 | 54.594 | 72.029 |
| 10,10,10 | 3,3,3 | 128.868 | 172.686 | 84.144 | 89.586 | 103.897 |
| 10,10,10 | 2,2,2 | 148.816 | 183.711 | 116.554 | 79.745 | 78.979 |
| 15,15,15 | 5,5,5 | 288.389 | 406.587 | 160.377 | 129.095 | 339.555 |
| 15,15,15 | 4,4,4 | 301.960 | 415.100 | 181.280 | 145.830 | 359.823 |
| 15,15,15 | 3,3,3 | 324.973 | 428.904 | 217.858 | 230.888 | 501.178 |
| 15,15,15 | 2,2,2 | 372.413 | 455.055 | 295.772 | 207.072 | 390.656 |
| 20,20,20 | 5,5,5 | 543.030 | 758.838 | 308.108 | 250.178 | 178.570 |
| 20,20,20 | 4,4,4 | 568.155 | 774.343 | 346.630 | 281.904 | 189.173 |
| 20,20,20 | 3,3,3 | 610.338 | 799.816 | 414.082 | 437.940 | 265.332 |
| 20,20,20 | 2,2,2 | 696.750 | 847.335 | 556.866 | 394.777 | 205.578 |

Table B.4 Matrix *ratio* similarity indexes of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on various distributions of SISO training data.

| Ranges (x,y,z) | Divisions (x,y,z) | Spacing Function | | | | |
|---|---|---|---|---|---|---|
| | | Even | Sine | Cubic | Complex | Mirrored |
| Figure Number $\Longrightarrow$ | | 6.28 | 6.31 | 6.34 | 6.37 | 6.40 |
| .5,.5,.5 | 5,5,5 | 2.873e-05 | 2.358e-04 | 1.071e-06 | 7.659e-06 | 0.000e+00 |
| .5,.5,.5 | 4,4,4 | 9.600e-09 | 1.924e-04 | 9.632e-07 | 1.124e-04 | 0.000e+00 |
| .5,.5,.5 | 3,3,3 | 2.022e-06 | 7.186e-06 | 8.015e-08 | 3.998e-05 | 0.000e+00 |
| .5,.5,.5 | 2,2,2 | 2.867e-06 | 4.311e-06 | 2.822e-06 | 7.308e-05 | 0.000e+00 |
| 5,5,5 | 5,5,5 | 9.851e-05 | 2.357e-04 | 3.766e-08 | 5.032e-07 | 1.382e-06 |
| 5,5,5 | 4,4,4 | 8.663e-07 | 2.037e-04 | 3.766e-09 | 4.090e-04 | 7.621e-05 |
| 5,5,5 | 3,3,3 | 8.092e-06 | 3.145e-05 | 1.284e-08 | 1.488e-04 | 2.266e-06 |
| 5,5,5 | 2,2,2 | 4.972e-08 | 1.489e-06 | 1.978e-07 | 1.010e-05 | 7.133e-06 |
| 10,10,10 | 5,5,5 | 5.628e-07 | 3.510e-04 | 2.616e-08 | 1.996e-05 | 3.450e-06 |
| 10,10,10 | 4,4,4 | 2.159e-06 | 1.043e-04 | 9.422e-08 | 4.584e-04 | 2.936e-05 |
| 10,10,10 | 3,3,3 | 2.243e-07 | 4.357e-08 | 8.007e-08 | 1.462e-04 | 1.077e-06 |
| 10,10,10 | 2,2,2 | 4.951e-08 | 4.132e-08 | 1.526e-06 | 1.203e-05 | 3.035e-06 |
| 15,15,15 | 5,5,5 | 2.852e-05 | 1.518e-15 | 4.650e-08 | 5.600e-08 | 2.296e-05 |
| 15,15,15 | 4,4,4 | 2.667e-08 | 1.798e-07 | 3.766e-07 | 2.049e-04 | 3.103e-05 |
| 15,15,15 | 3,3,3 | 1.822e-05 | 1.746e-07 | 3.559e-08 | 3.377e-08 | 1.363e-06 |
| 15,15,15 | 2,2,2 | 1.081e-06 | 2.941e-07 | 4.337e-07 | 3.797e-05 | 4.132e-06 |
| 20,20,20 | 5,5,5 | 6.237e-08 | 9.284e-07 | 2.855e-14 | 2.836e-07 | 2.248e-05 |
| 20,20,20 | 4,4,4 | 1.215e-06 | 1.227e-05 | 2.344e-08 | 3.700e-04 | 3.630e-07 |
| 20,20,20 | 3,3,3 | 5.610e-08 | 4.358e-08 | 7.712e-15 | 9.190e-06 | 1.077e-06 |
| 20,20,20 | 2,2,2 | 4.469e-07 | 4.133e-08 | 2.440e-07 | 5.345e-06 | 3.375e-07 |

Table B.5 Performance metrics of $\mathcal{A}_a{}'$ extracted using the LSMF method from ANN controllers trained on various distributions of SISO training data.

| Ranges (x,y,z) | Divisions (x,y,z) | Spacing Function | | | | |
|---|---|---|---|---|---|---|
| | | Even | Sine | Cubic | Complex | Mirrored |
| Figure Number $\Longrightarrow$ | | 6.21 | 6.22 | 6.23 | 6.24 | 6.25 |
| .5,.5,.5 | 5,5,5 | 1.206 | 1.176 | 1.255 | 1.275 | 0.000 |
| .5,.5,.5 | 4,4,4 | 1.203 | 1.172 | 1.246 | 1.234 | 0.000 |
| .5,.5,.5 | 3,3,3 | 1.190 | 1.173 | 1.227 | 1.216 | 0.000 |
| .5,.5,.5 | 2,2,2 | 1.194 | 1.169 | 1.200 | 1.232 | 0.000 |
| 5,5,5 | 5,5,5 | 3.544 | 5.110 | 2.498 | 2.024 | 3.067 |
| 5,5,5 | 4,4,4 | 4.258 | 6.105 | 3.135 | 2.732 | 2.714 |
| 5,5,5 | 3,3,3 | 3.527 | 5.796 | 3.049 | 3.774 | 3.406 |
| 5,5,5 | 2,2,2 | 4.418 | 6.271 | 4.409 | 3.222 | 2.947 |
| 10,10,10 | 5,5,5 | 17.191 | 30.145 | 8.410 | 7.814 | 10.937 |
| 10,10,10 | 4,4,4 | 16.343 | 14.979 | 8.461 | 10.207 | 11.646 |
| 10,10,10 | 3,3,3 | 7.513 | 17.760 | 9.565 | 15.944 | 7.947 |
| 10,10,10 | 2,2,2 | 16.150 | 16.614 | 12.476 | 9.626 | 11.504 |
| 15,15,15 | 5,5,5 | 29.093 | 33.445 | 15.616 | 21.830 | 18.550 |
| 15,15,15 | 4,4,4 | 15.747 | 73.018 | 17.980 | 20.599 | 25.033 |
| 15,15,15 | 3,3,3 | 33.454 | 84.674 | 31.700 | 21.427 | 28.282 |
| 15,15,15 | 2,2,2 | 69.851 | 14.172 | 28.491 | 15.027 | 27.858 |
| 20,20,20 | 5,5,5 | 139.331 | 586.678 | 36.162 | 17.180 | 37.454 |
| 20,20,20 | 4,4,4 | 136.517 | 33.944 | 17.942 | 25.266 | 38.968 |
| 20,20,20 | 3,3,3 | 222.767 | 122.624 | 17.052 | 30.068 | 235.405 |
| 20,20,20 | 2,2,2 | 441.046 | 557.987 | 103.667 | 33.499 | 74.757 |

Table B.6 Implementation results (best metric) of various configurations of RISO training data after ensuring a consistent training data exposure was maintained between the data sets

| LP cut-off Freq. (Hz) | Subsample Pts. Skipped | Raw Data | Vel. Pruned @ 0.05 m/s | Vel. Pruned @ 0.1 m/s | Hem. Pruned @ 86.5 deg | Coll. Pruned @ 0.05 N |
|---|---|---|---|---|---|---|
| Figure Number ⟹ | | 6.47 | 6.48 | 6.49 | 6.50 | 6.51 |
| None | None | 0.997 | 0.996 | 0.997 | 0.999 | 0.997 |
| 200. | None | 1.046 | 1.043 | 1.046 | 1.043 | 1.048 |
| 100. | None | 1.199 | 1.212 | 1.090 | 1.198 | 1.129 |
| 50. | None | 1.208 | 1.207 | 1.355 | 1.132 | 1.206 |
| 25. | None | 1.220 | 1.257 | 0.0 | 1.195 | 1.289 |
| None | 1. | 1.000 | 0.997 | 0.998 | 0.999 | 0.998 |
| 200. | 1. | 1.049 | 1.043 | 1.051 | 1.040 | 1.044 |
| 100. | 1. | 1.129 | 1.246 | 1.109 | 1.124 | 1.162 |
| 50. | 1. | 1.154 | 1.199 | 1.197 | 1.115 | 1.143 |
| 25. | 1. | 1.236 | 1.261 | 0.0 | 1.290 | 1.333 |
| None | 2. | 0.998 | 0.999 | 1.000 | 1.000 | 0.998 |
| 200. | 2. | 1.055 | 1.044 | 1.051 | 1.050 | 1.053 |
| 100. | 2. | 1.131 | 1.169 | 1.206 | 1.111 | 1.172 |
| 50. | 2. | 1.212 | 1.303 | 1.214 | 1.167 | 1.282 |
| 25. | 2. | 1.170 | 1.155 | 1.402 | 1.434 | 1.214 |
| None | 3. | 0.998 | 0.999 | 0.997 | 0.998 | 1.001 |
| 200. | 3. | 1.038 | 1.041 | 1.034 | 1.042 | 1.040 |
| 100. | 3. | 1.145 | 1.188 | 1.182 | 1.137 | 1.174 |
| 50. | 3. | 1.184 | 1.211 | 1.233 | 1.278 | 1.452 |
| 25. | 3. | 1.168 | 1.257 | 1.214 | 1.237 | 1.316 |
| None | 4. | 1.003 | 1.002 | 1.001 | 1.000 | 0.998 |
| 200. | 4. | 1.041 | 1.048 | 1.043 | 1.036 | 1.043 |
| 100. | 4. | 1.126 | 1.139 | 1.134 | 1.118 | 1.151 |
| 50. | 4. | 1.155 | 1.167 | 1.167 | 1.297 | 1.311 |
| 25. | 4. | 1.273 | 1.328 | 1.230 | 1.276 | 1.330 |

Table B.7 Included angle standard deviations of various distributions of RISO training data

| LP cut-off Freq. (Hz) | Subsample Pts. Skipped | Raw Data | Vel. Pruned @ 0.05 m/s | Vel. Pruned @ 0.1 m/s | Hem. Pruned @ 86.5 deg | Coll. Pruned @ 0.05 N |
|---|---|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.52 | 6.53 | 6.54 | 6.55 | 6.56 |
| None | None | 0.162 | 0.170 | 0.169 | 0.159 | 0.105 |
| 200. | None | 1.137 | 1.112 | 0.915 | 1.310 | 1.105 |
| 100. | None | 8.400 | 3.010 | 1.321 | 9.784 | 8.523 |
| 50. | None | 8.019 | 3.560 | 1.318 | 9.516 | 8.198 |
| 25. | None | 11.204 | 3.150 | 0.517 | 13.275 | 11.471 |
| None | 1. | 0.081 | 0.074 | 0.072 | 0.052 | 0.102 |
| 200. | 1. | 0.998 | 0.990 | 1.001 | 1.151 | 0.993 |
| 100. | 1. | 8.686 | 2.896 | 1.241 | 10.106 | 8.884 |
| 50. | 1. | 8.283 | 3.519 | 1.271 | 9.841 | 8.469 |
| 25. | 1. | 11.285 | 3.055 | 0.588 | 13.370 | 11.559 |
| None | 2. | 0.072 | 0.083 | 0.083 | 0.094 | 0.107 |
| 200. | 2. | 1.025 | 0.962 | 0.970 | 1.182 | 0.962 |
| 100. | 2. | 8.977 | 2.628 | 1.390 | 10.445 | 9.192 |
| 50. | 2. | 7.819 | 3.432 | 1.210 | 9.293 | 8.002 |
| 25. | 2. | 11.226 | 3.145 | 0.393 | 13.307 | 11.505 |
| None | 3. | 0.111 | 0.109 | 0.109 | 0.086 | 0.104 |
| 200. | 3. | 1.037 | 1.028 | 1.037 | 1.196 | 1.032 |
| 100. | 3. | 7.873 | 3.184 | 1.273 | 9.162 | 8.061 |
| 50. | 3. | 7.600 | 3.167 | 1.104 | 9.023 | 7.777 |
| 25. | 3. | 10.767 | 3.018 | 0.685 | 12.774 | 11.040 |
| None | 4. | 0.117 | 0.117 | 0.117 | 0.093 | 0.106 |
| 200. | 4. | 1.672 | 1.684 | 0.903 | 1.949 | 1.684 |
| 100. | 4. | 7.612 | 3.046 | 1.326 | 8.868 | 7.806 |
| 50. | 4. | 7.154 | 2.873 | 1.350 | 8.498 | 7.327 |
| 25. | 4. | 10.683 | 3.076 | 0.459 | 12.694 | 10.943 |

Table B.8 Matrix structural similarity indexes, $\Upsilon_s$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on various distributions of RISO training data

| LP cut-off Freq. (Hz) | Subsample Pts. Skipped | Vel. Pruned @ 0.05 m/s | Hem. Pruned @ 86.5 deg | Coll. Pruned @ 0.05 N |
|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.57 | 6.60 | 6.63 |
| None | None | 0.0016 | 0.0000 | 0.0001 |
| 200 | None | 0.0000 | 0.0000 | 0.0000 |
| 100 | None | 0.0000 | 0.0000 | 0.0000 |
| 50 | None | 0.0000 | 0.0000 | 0.0000 |
| 25 | None | 0.0000 | 0.0003 | 0.0001 |
| None | 1 | 0.0003 | 0.0000 | 0.0000 |
| 200 | 1 | 0.0000 | 0.0000 | 0.0000 |
| 100 | 1 | 0.0000 | 0.0000 | 0.0000 |
| 50 | 1 | 0.0000 | 0.0002 | 0.0001 |
| 25 | 1 | 0.0000 | 0.0008 | 0.0003 |
| None | 2 | 0.0007 | 0.0001 | 0.0001 |
| 200 | 2 | 0.0002 | 0.0002 | 0.0002 |
| 100 | 2 | 0.0001 | 0.0000 | 0.0000 |
| 50 | 2 | 0.0000 | 0.0003 | 0.0001 |
| 25 | 2 | 0.0000 | 0.0028 | 0.0008 |
| None | 3 | 0.0013 | 0.0000 | 0.0000 |
| 200 | 3 | 0.0003 | 0.0004 | 0.0003 |
| 100 | 3 | 0.0000 | 0.0001 | 0.0001 |
| 50 | 3 | 0.0000 | 0.0006 | 0.0002 |
| 25 | 3 | 0.0000 | 0.0042 | 0.0018 |
| None | 4 | 0.0102 | 0.0009 | 0.0011 |
| 200 | 4 | 0.0004 | 0.0005 | 0.0004 |
| 100 | 4 | 0.0001 | 0.0001 | 0.0001 |
| 50 | 4 | 0.0000 | 0.0006 | 0.0002 |
| 25 | 4 | 0.0000 | 0.0066 | 0.0018 |

Table B.9 Matrix gain similarity indexes, $\Upsilon_g$, of $\mathcal{A}_a{}'$ extracted from ANN controllers trained on various distributions of RISO training data

| LP cut-off Freq. (Hz) | Subsample Pts. Skipped | Vel. Pruned @ 0.05 m/s | Hem. Pruned @ 86.5 deg | Coll. Pruned @ 0.05 N |
|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.58 | 6.61 | 6.64 |
| None | None | 3.9395 | 3.9578 | 3.9542 |
| 200 | None | 3.9859 | 3.9839 | 3.9859 |
| 100 | None | 3.9929 | 3.9905 | 3.9924 |
| 50 | None | 3.9935 | 3.9916 | 3.9928 |
| 25 | None | 3.9946 | 3.9918 | 3.9932 |
| None | 1 | 3.9416 | 3.9550 | 3.9513 |
| 200 | 1 | 3.9857 | 3.9837 | 3.9857 |
| 100 | 1 | 3.9928 | 3.9903 | 3.9922 |
| 50 | 1 | 3.9934 | 3.9911 | 3.9926 |
| 25 | 1 | 3.9944 | 3.9912 | 3.9927 |
| None | 2 | 3.9405 | 3.9568 | 3.9524 |
| 200 | 2 | 3.9854 | 3.9837 | 3.9855 |
| 100 | 2 | 3.9929 | 3.9903 | 3.9923 |
| 50 | 2 | 3.9934 | 3.9911 | 3.9926 |
| 25 | 2 | 3.9943 | 3.9906 | 3.9924 |
| None | 3 | 3.9401 | 3.9576 | 3.9535 |
| 200 | 3 | 3.9863 | 3.9845 | 3.9863 |
| 100 | 3 | 3.9926 | 3.9901 | 3.9922 |
| 50 | 3 | 3.9935 | 3.9907 | 3.9923 |
| 25 | 3 | 3.9942 | 3.9898 | 3.9920 |
| None | 4 | 3.9553 | 3.9556 | 3.9528 |
| 200 | 4 | 3.9857 | 3.9836 | 3.9856 |
| 100 | 4 | 3.9926 | 3.9899 | 3.9922 |
| 50 | 4 | 3.9934 | 3.9906 | 3.9922 |
| 25 | 4 | 3.9941 | 3.9891 | 3.9919 |

Table B.10 Matrix ratio similarity indexes, $\Upsilon_r$, of $\mathcal{A}_a'$ extracted from ANN controllers trained on various distributions of RISO training data

| LP cut-off Freq. (Hz) | Subsample Pts. Skipped | Vel. Pruned @ 0.05 m/s | Hem. Pruned @ 86.5 deg | Coll. Pruned @ 0.05 N |
|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.59 | 6.62 | 6.65 |
| None | None | 8.9437 | 9.2398 | 9.2373 |
| 200 | None | 9.0715 | 9.0830 | 9.0753 |
| 100 | None | 8.8768 | 9.0335 | 8.8514 |
| 50 | None | 8.8115 | 8.7052 | 8.7759 |
| 25 | None | 8.8618 | 8.6438 | 8.7068 |
| None | 1 | 9.0180 | 9.2395 | 9.2392 |
| 200 | 1 | 9.0553 | 9.0698 | 9.0561 |
| 100 | 1 | 8.8758 | 9.0316 | 8.8511 |
| 50 | 1 | 8.8217 | 8.7628 | 8.8046 |
| 25 | 1 | 8.8659 | 8.7567 | 8.7574 |
| None | 2 | 9.0018 | 9.2317 | 9.2274 |
| 200 | 2 | 9.0205 | 9.0351 | 9.0175 |
| 100 | 2 | 8.8138 | 9.0106 | 8.8048 |
| 50 | 2 | 8.8182 | 8.7799 | 8.8154 |
| 25 | 2 | 8.8481 | 8.8458 | 8.8433 |
| None | 3 | 8.9835 | 9.2347 | 9.2299 |
| 200 | 3 | 9.0189 | 9.0405 | 9.0128 |
| 100 | 3 | 8.8115 | 8.9977 | 8.7680 |
| 50 | 3 | 8.8028 | 8.8016 | 8.8205 |
| 25 | 3 | 8.7791 | 8.9110 | 8.8940 |
| None | 4 | 9.1936 | 9.2578 | 9.2575 |
| 200 | 4 | 9.0177 | 9.0404 | 9.0138 |
| 100 | 4 | 8.7442 | 8.9603 | 8.7323 |
| 50 | 4 | 8.7485 | 8.7859 | 8.8051 |
| 25 | 4 | 8.7861 | 8.9535 | 8.9088 |

Table B.11 Overall $\mathcal{A}_a{}'$ extracted from each of the PUMA DIDO raw training data files using the LSMF interrogation technique

| Demo. No. | $\mathcal{A}_a{}'$ |
|---|---|
| 1 | $\begin{bmatrix} -0.00057 & -0.00030 & -0.00082 \\ 0.00023 & -0.00005 & -0.00004 \\ -0.00209 & 0.00136 & 0.00224 \end{bmatrix}$ |
| 2 | $\begin{bmatrix} 0.00044 & 0.00015 & -0.00725 \\ -0.00011 & -0.00002 & 0.00105 \\ 0.00037 & -0.00086 & 0.03444 \end{bmatrix}$ |
| 3 | $\begin{bmatrix} 0.00007 & -0.00029 & -0.00392 \\ 0.00089 & -0.00003 & 0.00072 \\ -0.00766 & 0.00147 & 0.00985 \end{bmatrix}$ |
| 4 | $\begin{bmatrix} 0.00103 & 0.00068 & -0.00344 \\ -0.00018 & -0.00010 & 0.00065 \\ -0.00263 & -0.00286 & 0.02196 \end{bmatrix}$ |
| 5 | $\begin{bmatrix} -0.00266 & 0.00008 & -0.00185 \\ -0.00046 & 0.00012 & -0.00043 \\ 0.00665 & 0.00043 & -0.01454 \end{bmatrix}$ |
| 6 | $\begin{bmatrix} 0.00106 & 0.00021 & -0.00655 \\ -0.00017 & -0.00003 & 0.00104 \\ -0.00351 & -0.00069 & 0.02422 \end{bmatrix}$ |
| 7 | $\begin{bmatrix} 0.00317 & -0.00026 & 0.00496 \\ 0.00054 & -0.00006 & 0.00095 \\ -0.01561 & 0.00136 & -0.02451 \end{bmatrix}$ |
| 8 | $\begin{bmatrix} -0.00013 & 0.00018 & -0.00743 \\ 0.00000 & -0.00002 & 0.00119 \\ 0.00470 & -0.00093 & 0.04162 \end{bmatrix}$ |
| 9 | $\begin{bmatrix} 0.00100 & -0.00028 & -0.00414 \\ 0.00005 & -0.00004 & -0.00093 \\ 0.00359 & 0.00126 & 0.03123 \end{bmatrix}$ |
| 10 | $\begin{bmatrix} 0.00032 & 0.00009 & -0.00426 \\ -0.00005 & -0.00001 & 0.00077 \\ 0.00025 & -0.00053 & 0.02483 \end{bmatrix}$ |

Table B.12 Overall $\mathcal{A}_a{}'$ extracted from each of the PLIMMS DIDO training data files using the LSMF interrogation technique

| Demo. No. | $\mathcal{A}_a{}'$ |
|---|---|
| 1 | $\begin{bmatrix} -0.00135 & 0.00005 & 0.00142 \\ 0.00736 & -0.00027 & -0.00980 \\ 0.15067 & -0.00627 & -0.16557 \end{bmatrix}$ |
| 2 | $\begin{bmatrix} -0.00369 & 0.00008 & 0.00769 \\ -0.00319 & -0.00001 & 0.00269 \\ 0.10864 & -0.00047 & -0.11291 \end{bmatrix}$ |
| 3 | $\begin{bmatrix} -0.00518 & 0.00015 & 0.01520 \\ 0.01120 & -0.00035 & -0.02568 \\ 0.15823 & -0.00544 & -0.26036 \end{bmatrix}$ |
| 4 | $\begin{bmatrix} 0.00225 & 0.00011 & -0.00840 \\ -0.00964 & -0.00047 & 0.02196 \\ 0.13838 & 0.00707 & -0.20568 \end{bmatrix}$ |
| 5 | $\begin{bmatrix} -0.00989 & 0.00039 & 0.02243 \\ 0.02134 & -0.00100 & -0.04684 \\ 0.41555 & -0.02037 & -0.83286 \end{bmatrix}$ |
| 6 | $\begin{bmatrix} -0.00061 & -0.00005 & 0.00159 \\ -0.00416 & -0.00033 & 0.00639 \\ 0.12793 & 0.00984 & -0.20883 \end{bmatrix}$ |
| 7 | $\begin{bmatrix} 0.00072 & -0.00007 & -0.00506 \\ 0.00215 & 0.00006 & 0.00284 \\ 0.04073 & -0.00025 & 0.05430 \end{bmatrix}$ |
| 8 | $\begin{bmatrix} 0.00116 & 0.00004 & -0.00445 \\ -0.00487 & -0.00017 & 0.01092 \\ 0.10292 & 0.00274 & -0.13587 \end{bmatrix}$ |
| 9 | $\begin{bmatrix} -0.00080 & 0.00002 & 0.00213 \\ 0.00604 & -0.00014 & -0.01278 \\ 0.11712 & -0.00281 & -0.22897 \end{bmatrix}$ |
| 10 | $\begin{bmatrix} 0.00054 & 0.00012 & -0.00473 \\ -0.00219 & -0.00017 & 0.00435 \\ 0.04856 & 0.00193 & -0.02489 \end{bmatrix}$ |

Table B.13 Matrix similarity index data for PUMA DIDO training data files

| Config. Code | Demo. Number | Structural Index ($\Upsilon_s$) | Gain Index ($\Upsilon_g$) | Sign Index ($\Upsilon_\pm$) | Ratio Index ($\Upsilon_r$) |
|---|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.76(a) | 6.76(b) | 6.76(c) | 6.76(d) |
| N/A | 1 | 7.385e-06 | 3.911e-02 | 2 | 5.746e+04 |
| N/A | 2 | 5.482e-05 | 2.741e-02 | 2 | 2.727e+06 |
| N/A | 3 | 7.754e-05 | 3.616e-02 | 2 | 4.350e+02 |
| N/A | 4 | 2.891e-05 | 3.170e-02 | 2 | 3.787e+03 |
| N/A | 5 | 5.529e-05 | 4.603e-02 | 4 | 2.699e+04 |
| N/A | 6 | 5.797e-05 | 3.090e-02 | 2 | 3.767e+05 |
| N/A | 7 | 2.814e-04 | 5.041e-02 | 6 | 2.316e+04 |
| N/A | 8 | 7.970e-05 | 2.508e-02 | 2 | 2.041e+06 |
| N/A | 9 | 3.358e-05 | 2.848e-02 | 2 | 1.995e+05 |
| N/A | 10 | 1.922e-05 | 3.068e-02 | 2 | 2.750e+06 |

Table B.14 Matrix similarity index data for PLIMMS DIDO training data files

| Config. Code | Demo. Number | Structural Index ($\Upsilon_s$) | Gain Index ($\Upsilon_g$) | Sign Index ($\Upsilon_\pm$) | Ratio Index ($\Upsilon_r$) |
|---|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.79(a) | 6.79(b) | 6.79(c) | 6.79(d) |
| N/A | 1 | 2.289e-02 | 1.336e-01 | 6 | 1.115e+05 |
| N/A | 2 | 1.189e-02 | 9.791e-02 | 6 | 2.218e+08 |
| N/A | 3 | 2.611e-02 | 2.119e-01 | 6 | 2.035e+05 |
| N/A | 4 | 1.985e-02 | 1.646e-01 | 6 | 2.334e+04 |
| N/A | 5 | 1.763e-01 | 1.067e+00 | 6 | 3.006e+05 |
| N/A | 6 | 1.652e-02 | 1.671e-01 | 6 | 1.254e+05 |
| N/A | 7 | 1.698e-03 | 2.123e-02 | 0 | 3.338e+05 |
| N/A | 8 | 1.076e-02 | 1.128e-01 | 6 | 2.648e+05 |
| N/A | 9 | 1.392e-02 | 1.840e-01 | 6 | 1.814e+06 |
| N/A | 10 | 2.409e-03 | 5.058e-02 | 6 | 1.957e+04 |

Table B.15 Mean values for PLIMMS DIDO training data files

| Config. | Demo. | $F_x$ | $F_y$ | $M_z$ |
|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.88 | 6.89 | 6.90 |
| 1 | 1 | 0.41511 | -23.91887 | 0.26592 |
| 1 | 2 | -1.13065 | -22.29583 | -0.19077 |
| 1 | 3 | 0.90491 | -31.77127 | 0.60192 |
| 1 | 4 | -0.20284 | -23.78240 | -0.38983 |
| 1 | 5 | -0.15275 | -23.13363 | 0.34250 |
| 1 | 6 | -0.16093 | -23.00774 | -0.64685 |
| 1 | 7 | 2.43631 | -25.26437 | 0.75924 |
| 1 | 8 | -1.27663 | -23.74945 | -0.43093 |
| 1 | 9 | 1.74000 | -30.15075 | 0.74141 |
| 1 | 10 | -2.83249 | -26.65357 | -0.86312 |
| 2 | 1 | 0.59802 | -34.01005 | 0.38317 |
| 2 | 2 | -1.65727 | -33.60800 | -0.27252 |
| 2 | 3 | 1.08509 | -38.24192 | 0.72456 |
| 2 | 4 | -0.26466 | -37.08613 | -0.59360 |
| 2 | 5 | -0.28003 | -38.45847 | 0.56232 |
| 2 | 6 | -0.22083 | -32.23373 | -0.90160 |
| 2 | 7 | 3.26566 | -33.94432 | 1.01528 |
| 2 | 8 | -1.87447 | -34.49514 | -0.62658 |
| 2 | 9 | 2.22526 | -39.12806 | 0.95782 |
| 2 | 10 | -3.63958 | -34.30371 | -1.10854 |
| 3 | 1 | -0.26479 | -28.40546 | 0.10227 |
| 3 | 2 | -1.29116 | -29.11979 | -0.20550 |
| 3 | 3 | 0.54646 | -38.95196 | 0.58512 |
| 3 | 4 | -0.04887 | -28.41571 | -0.43518 |
| 3 | 5 | -0.39217 | -24.39460 | 0.28984 |
| 3 | 6 | 0.12399 | -28.84749 | -0.71108 |
| 3 | 7 | 2.23139 | -27.37019 | 0.75396 |
| 3 | 8 | -1.33185 | -34.54142 | -0.57732 |
| 3 | 9 | 1.32059 | -34.70308 | 0.69860 |
| 3 | 10 | -2.82659 | -31.74863 | -1.01372 |
| 4 | 1 | 0.51298 | -29.04021 | 0.33549 |
| 4 | 2 | -1.59568 | -28.21244 | -0.29492 |
| 4 | 3 | 1.14438 | -37.81334 | 0.75337 |
| 4 | 4 | -0.33245 | -28.64749 | -0.50662 |
| 4 | 5 | -0.11132 | -25.72684 | 0.40784 |
| 4 | 6 | -0.19389 | -28.96230 | -0.80904 |
| 4 | 7 | 2.86399 | -30.13680 | 0.90176 |
| 4 | 8 | -1.92670 | -32.68736 | -0.64270 |
| 4 | 9 | 1.93027 | -34.25500 | 0.81459 |
| 4 | 10 | -3.73715 | -32.09216 | -1.07557 |
| 5 | 1 | 0.41506 | -23.91890 | 0.26590 |
| 5 | 2 | -1.13059 | -22.29588 | -0.19075 |
| 5 | 3 | 0.90485 | -31.77125 | 0.60191 |
| 5 | 4 | -0.20289 | -23.78226 | -0.38984 |
| 5 | 5 | -0.15280 | -23.13343 | 0.34248 |
| 5 | 6 | -0.16088 | -23.00774 | -0.64683 |
| 5 | 7 | 2.43629 | -25.26432 | 0.75924 |
| 5 | 8 | -1.27658 | -23.74955 | -0.43092 |
| 5 | 9 | 1.73999 | -30.15069 | 0.74141 |
| 5 | 10 | -2.83286 | -26.65369 | -0.86328 |

Table B.15 Mean values for PLIMMS DIDO training data files (**continued**)

| Config. | Demo. | $F_x$ | $F_y$ | $M_z$ |
|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.88 | 6.89 | 6.90 |
| 6 | 1 | 0.41516 | -23.91957 | 0.26591 |
| 6 | 2 | -1.13005 | -22.29626 | -0.19050 |
| 6 | 3 | 0.90486 | -31.77098 | 0.60192 |
| 6 | 4 | -0.20312 | -23.78230 | -0.38987 |
| 6 | 5 | -0.15260 | -23.13105 | 0.34244 |
| 6 | 6 | -0.16102 | -23.00813 | -0.64686 |
| 6 | 7 | 2.43709 | -25.26409 | 0.75950 |
| 6 | 8 | -1.27602 | -23.75022 | -0.43076 |
| 6 | 9 | 1.74044 | -30.15007 | 0.74157 |
| 6 | 10 | -2.83193 | -26.65486 | -0.86288 |
| 7 | 1 | 1.53921 | -19.34545 | 0.59258 |
| 7 | 2 | -1.41774 | -14.41531 | -0.35631 |
| 7 | 3 | 1.80048 | -29.75069 | 0.85656 |
| 7 | 4 | -0.89790 | -20.76633 | -0.60062 |
| 7 | 5 | 0.27738 | -20.69675 | 0.46187 |
| 7 | 6 | -0.71260 | -22.05044 | -0.79852 |
| 7 | 7 | 2.94967 | -24.88305 | 0.87424 |
| 7 | 8 | -1.84113 | -19.41758 | -0.53055 |
| 7 | 9 | 2.77545 | -28.55056 | 1.01424 |
| 7 | 10 | -3.40058 | -27.63096 | -0.92933 |
| 8 | 1 | -0.31903 | -36.43895 | 0.13540 |
| 8 | 2 | -1.57819 | -36.45421 | -0.24682 |
| 8 | 3 | 0.58999 | -42.08617 | 0.63190 |
| 8 | 4 | -0.04836 | -39.41137 | -0.59491 |
| 8 | 5 | -0.63347 | -39.83915 | 0.46793 |
| 8 | 6 | 0.15173 | -33.48775 | -0.82200 |
| 8 | 7 | 2.74584 | -33.85761 | 0.92673 |
| 8 | 8 | -1.52687 | -39.60338 | -0.66148 |
| 8 | 9 | 1.61940 | -43.52693 | 0.87076 |
| 8 | 10 | -3.30047 | -37.16550 | -1.18462 |
| 9 | 1 | 0.64047 | -35.41785 | 0.41224 |
| 9 | 2 | -1.93346 | -34.70420 | -0.35417 |
| 9 | 3 | 1.20972 | -39.98307 | 0.79638 |
| 9 | 4 | -0.41706 | -37.56630 | -0.65702 |
| 9 | 5 | -0.16442 | -39.11510 | 0.61528 |
| 9 | 6 | -0.21725 | -33.37898 | -0.92938 |
| 9 | 7 | 3.31867 | -35.03005 | 1.04404 |
| 9 | 8 | -2.15755 | -36.54513 | -0.71877 |
| 9 | 9 | 2.25847 | -40.53931 | 0.96035 |
| 9 | 10 | -4.11287 | -35.36126 | -1.18391 |
| 10 | 1 | 0.59779 | -34.00834 | 0.38308 |
| 10 | 2 | -1.65716 | -33.60705 | -0.27248 |
| 10 | 3 | 1.08499 | -38.24069 | 0.72452 |
| 10 | 4 | -0.26442 | -37.08355 | -0.59353 |
| 10 | 5 | -0.28014 | -38.45724 | 0.56228 |
| 10 | 6 | -0.22051 | -32.23217 | -0.90150 |
| 10 | 7 | 3.26549 | -33.94335 | 1.01525 |
| 10 | 8 | -1.87405 | -34.49213 | -0.62647 |
| 10 | 9 | 2.22508 | -39.12643 | 0.95777 |
| 10 | 10 | -3.63997 | -34.30289 | -1.10878 |

Table B.15 Mean values for PLIMMS DIDO training data files **(concluded)**

| Config. | Demo. | $F_x$ | $F_y$ | $M_z$ |
|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.88 | 6.89 | 6.90 |
| 11 | 1 | 0.59699 | -34.00050 | 0.38283 |
| 11 | 2 | -1.65566 | -33.60176 | -0.27193 |
| 11 | 3 | 1.08489 | -38.23153 | 0.72431 |
| 11 | 4 | -0.26329 | -37.07680 | -0.59321 |
| 11 | 5 | -0.28114 | -38.43648 | 0.56142 |
| 11 | 6 | -0.21814 | -32.21921 | -0.90090 |
| 11 | 7 | 3.26477 | -33.92800 | 1.01512 |
| 11 | 8 | -1.87131 | -34.48117 | -0.62584 |
| 11 | 9 | 2.22458 | -39.12154 | 0.95778 |
| 11 | 10 | -3.63792 | -34.30177 | -1.10825 |
| 12 | 1 | 2.46812 | -30.93132 | 0.95416 |
| 12 | 2 | -2.54148 | -26.58026 | -0.63502 |
| 12 | 3 | 2.20582 | -36.55795 | 1.05274 |
| 12 | 4 | -1.45613 | -34.89838 | -0.99354 |
| 12 | 5 | 0.46441 | -36.50570 | 0.80799 |
| 12 | 6 | -1.04094 | -32.18103 | -1.16179 |
| 12 | 7 | 4.28875 | -36.25613 | 1.26795 |
| 12 | 8 | -2.88041 | -30.05200 | -0.82339 |
| 12 | 9 | 3.49099 | -36.23822 | 1.28390 |
| 12 | 10 | -4.03741 | -32.78540 | -1.10218 |
| 13 | 1 | 1.74056 | -20.77117 | 0.66804 |
| 13 | 2 | -1.48372 | -14.76277 | -0.37369 |
| 13 | 3 | 1.85057 | -29.50552 | 0.86942 |
| 13 | 4 | -0.88980 | -22.03649 | -0.62205 |
| 13 | 5 | 0.44153 | -22.55467 | 0.56118 |
| 13 | 6 | -0.87821 | -20.97415 | -0.82756 |
| 13 | 7 | 3.02808 | -25.31989 | 0.86814 |
| 13 | 8 | -1.81273 | -19.03978 | -0.52125 |
| 13 | 9 | 2.87309 | -30.48693 | 1.03728 |
| 13 | 10 | -3.69419 | -28.89049 | -0.98126 |
| 14 | 1 | 2.71362 | -32.26109 | 1.04398 |
| 14 | 2 | -2.62367 | -26.82959 | -0.65765 |
| 14 | 3 | 2.31684 | -37.04186 | 1.09192 |
| 14 | 4 | -1.43244 | -36.55409 | -1.01699 |
| 14 | 5 | 0.71072 | -37.26303 | 0.92286 |
| 14 | 6 | -1.33547 | -31.86502 | -1.25342 |
| 14 | 7 | 4.38530 | -36.70755 | 1.25319 |
| 14 | 8 | -2.86427 | -29.75610 | -0.81681 |
| 14 | 9 | 3.42328 | -36.57929 | 1.24239 |
| 14 | 10 | -4.44194 | -34.71557 | -1.17878 |

Table B.16 Matrix similarity index data for PLIMMS DIDO training data files

| Config. Code | Demo. Number | Structural Index ($\Upsilon_s$) | Gain Index ($\Upsilon_g$) | Sign Index ($\Upsilon_\pm$) | Ratio Index ($\Upsilon_r$) |
|---|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.80 | 6.81 | 6.82 | 6.83 |
| 1 | 1 | 0.023 | 40066 | 6 | 144632 |
| 1 | 2 | 0.012 | 40045 | 6 | 201076800 |
| 1 | 3 | 0.026 | 40104 | 6 | 69235 |
| 1 | 4 | 0.020 | 40082 | 6 | 315299 |
| 1 | 5 | 0.176 | 40334 | 6 | 27558 |
| 1 | 6 | 0.017 | 40084 | 6 | 129703 |
| 1 | 7 | 0.002 | 39978 | 0 | 18631 |
| 1 | 8 | 0.011 | 40054 | 6 | 39892 |
| 1 | 9 | 0.014 | 40092 | 6 | 400311 |
| 1 | 10 | 0.002 | 40010 | 6 | 729634 |
| 2 | 1 | 0.023 | 40066 | 6 | 128770 |
| 2 | 2 | 0.011 | 40042 | 4 | 23486470 |
| 2 | 3 | 0.026 | 40103 | 6 | 54661 |
| 2 | 4 | 0.020 | 40083 | 6 | 284697 |
| 2 | 5 | 0.179 | 40337 | 6 | 28694 |
| 2 | 6 | 0.017 | 40084 | 6 | 174542 |
| 2 | 7 | 0.002 | 39979 | 0 | 109435 |
| 2 | 8 | 0.011 | 40055 | 6 | 34884 |
| 2 | 9 | 0.014 | 40092 | 6 | 516554 |
| 2 | 10 | 0.002 | 40010 | 6 | 730552 |
| 3 | 1 | 0.017 | 40054 | 6 | 15890210 |
| 3 | 2 | 0.011 | 40044 | 4 | 3751 |
| 3 | 3 | 0.022 | 40098 | 6 | 7021267 |
| 3 | 4 | 0.024 | 40094 | 6 | 8839521 |
| 3 | 5 | 0.103 | 40237 | 4 | 332844 |
| 3 | 6 | 0.022 | 40095 | 6 | 4084057 |
| 3 | 7 | 0.002 | 39984 | 0 | 631947 |
| 3 | 8 | 0.013 | 40065 | 6 | 139824400 |
| 3 | 9 | 0.010 | 40077 | 6 | 12783970 |
| 3 | 10 | 0.003 | 40024 | 4 | 2943 |
| 4 | 1 | 0.023 | 40061 | 6 | 108674 |
| 4 | 2 | 0.012 | 40042 | 4 | 44292 |
| 4 | 3 | 0.026 | 40102 | 6 | 109 |
| 4 | 4 | 0.021 | 40085 | 6 | 223123 |
| 4 | 5 | 0.152 | 40295 | 6 | 177192 |
| 4 | 6 | 0.019 | 40085 | 6 | 14518 |
| 4 | 7 | 0.002 | 39982 | 0 | 226728 |
| 4 | 8 | 0.011 | 40052 | 6 | 8525 |
| 4 | 9 | 0.014 | 40090 | 6 | 486786 |
| 4 | 10 | 0.002 | 40013 | 6 | 131527 |
| 5 | 1 | 0.021 | 40051 | 6 | 254751 |
| 5 | 2 | 0.012 | 40044 | 6 | 203851600 |
| 5 | 3 | 0.023 | 40090 | 6 | 139773 |
| 5 | 4 | 0.019 | 40074 | 6 | 355185 |
| 5 | 5 | 0.171 | 40321 | 6 | 46237 |
| 5 | 6 | 0.016 | 40078 | 6 | 123921 |
| 5 | 7 | 0.002 | 39970 | 0 | 7288 |
| 5 | 8 | 0.010 | 40047 | 6 | 81084 |
| 5 | 9 | 0.014 | 40088 | 6 | 314833 |
| 5 | 10 | 0.002 | 40001 | 6 | 966912 |

Table B.16 Matrix similarity index data for DIDO training data files **(continued)**

| Config. Code | Demo. Number | Structural Index ($\Upsilon_s$) | Gain Index ($\Upsilon_g$) | Sign Index ($\Upsilon_\pm$) | Ratio Index ($\Upsilon_r$) |
|---|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.80 | 6.81 | 6.82 | 6.83 |
| 6 | 1 | 0.018 | 40031 | 6 | 479470 |
| 6 | 2 | 0.011 | 40042 | 6 | 831231800 |
| 6 | 3 | 0.018 | 40059 | 6 | 318147 |
| 6 | 4 | 0.017 | 40059 | 6 | 424839 |
| 6 | 5 | 0.160 | 40294 | 6 | 64698 |
| 6 | 6 | 0.016 | 40070 | 6 | 123001 |
| 6 | 7 | 0.001 | 39947 | 0 | 7030 |
| 6 | 8 | 0.010 | 40035 | 6 | 142178 |
| 6 | 9 | 0.013 | 40081 | 6 | 193570 |
| 6 | 10 | 0.002 | 39982 | 2 | 119159 |
| 7 | 1 | 0.019 | 40058 | 6 | 159618 |
| 7 | 2 | 0.007 | 40020 | 6 | 614347 |
| 7 | 3 | 0.023 | 40088 | 6 | 120399 |
| 7 | 4 | 0.020 | 40076 | 6 | 465965 |
| 7 | 5 | 0.159 | 40307 | 6 | 66366 |
| 7 | 6 | 0.017 | 40097 | 6 | 162819 |
| 7 | 7 | 0.002 | 39977 | 0 | 629040 |
| 7 | 8 | 0.010 | 40039 | 6 | 441468 |
| 7 | 9 | 0.013 | 40073 | 6 | 22615760 |
| 7 | 10 | 0.002 | 40000 | 6 | 993200 |
| 8 | 1 | 0.017 | 40058 | 6 | 2982878 |
| 8 | 2 | 0.011 | 40044 | 4 | 183 |
| 8 | 3 | 0.022 | 40098 | 6 | 6729135 |
| 8 | 4 | 0.025 | 40097 | 6 | 1578862 |
| 8 | 5 | 0.124 | 40266 | 6 | 202372 |
| 8 | 6 | 0.022 | 40096 | 6 | 1543526 |
| 8 | 7 | 0.002 | 39984 | 0 | 595311 |
| 8 | 8 | 0.013 | 40065 | 6 | 143805900 |
| 8 | 9 | 0.011 | 40078 | 6 | 21272100 |
| 8 | 10 | 0.003 | 40024 | 4 | 4019 |
| 9 | 1 | 0.024 | 40062 | 6 | 114428 |
| 9 | 2 | 0.012 | 40043 | 4 | 52350 |
| 9 | 3 | 0.026 | 40102 | 6 | 83 |
| 9 | 4 | 0.022 | 40087 | 6 | 240956 |
| 9 | 5 | 0.169 | 40313 | 6 | 39562 |
| 9 | 6 | 0.019 | 40086 | 6 | 35577 |
| 9 | 7 | 0.002 | 39983 | 0 | 104906 |
| 9 | 8 | 0.011 | 40052 | 6 | 10064 |
| 9 | 9 | 0.014 | 40090 | 6 | 539434 |
| 9 | 10 | 0.002 | 40013 | 6 | 86590 |
| 10 | 1 | 0.020 | 40050 | 6 | 237524 |
| 10 | 2 | 0.011 | 40040 | 4 | 20113810 |
| 10 | 3 | 0.023 | 40089 | 6 | 122843 |
| 10 | 4 | 0.019 | 40074 | 6 | 323543 |
| 10 | 5 | 0.173 | 40324 | 6 | 49866 |
| 10 | 6 | 0.016 | 40079 | 6 | 170925 |
| 10 | 7 | 0.002 | 39970 | 0 | 609751 |
| 10 | 8 | 0.010 | 40047 | 6 | 74724 |
| 10 | 9 | 0.014 | 40088 | 6 | 440862 |
| 10 | 10 | 0.002 | 40001 | 6 | 968060 |

Table B.16 Matrix similarity index data for DIDO training data files **(concluded)**

| Config. Code | Demo. Number | Structural Index ($\Upsilon_s$) | Gain Index ($\Upsilon_g$) | Sign Index ($\Upsilon_\pm$) | Ratio Index ($\Upsilon_r$) |
|---|---|---|---|---|---|
| Figure Number $\Longrightarrow$ | | 6.80 | 6.81 | 6.82 | 6.83 |
| 11 | 1 | 0.017 | 40030 | 6 | 467674 |
| 11 | 2 | 0.011 | 40038 | 4 | 12790180 |
| 11 | 3 | 0.017 | 40058 | 6 | 301678 |
| 11 | 4 | 0.017 | 40059 | 6 | 389109 |
| 11 | 5 | 0.162 | 40297 | 6 | 72904 |
| 11 | 6 | 0.016 | 40071 | 6 | 164792 |
| 11 | 7 | 0.001 | 39948 | 0 | 373805 |
| 11 | 8 | 0.010 | 40035 | 6 | 134994 |
| 11 | 9 | 0.013 | 40082 | 6 | 330879 |
| 11 | 10 | 0.002 | 39982 | 2 | 155536 |
| 12 | 1 | 0.019 | 40058 | 6 | 113142 |
| 12 | 2 | 0.006 | 40015 | 6 | 684125 |
| 12 | 3 | 0.022 | 40086 | 6 | 104337 |
| 12 | 4 | 0.020 | 40077 | 6 | 468748 |
| 12 | 5 | 0.160 | 40308 | 6 | 71357 |
| 12 | 6 | 0.018 | 40098 | 6 | 199204 |
| 12 | 7 | 0.002 | 39978 | 0 | 573320 |
| 12 | 8 | 0.010 | 40039 | 6 | 411195 |
| 12 | 9 | 0.013 | 40073 | 6 | 9181656 |
| 12 | 10 | 0.002 | 40000 | 6 | 992373 |
| 13 | 1 | 0.019 | 40064 | 6 | 199703 |
| 13 | 2 | 0.007 | 40019 | 6 | 505815 |
| 13 | 3 | 0.019 | 40071 | 6 | 215178 |
| 13 | 4 | 0.021 | 40088 | 6 | 361039 |
| 13 | 5 | 0.158 | 40301 | 6 | 90919 |
| 13 | 6 | 0.029 | 39989 | 0 | 983908 |
| 13 | 7 | 0.001 | 39967 | 0 | 270716 |
| 13 | 8 | 0.009 | 40039 | 6 | 357455 |
| 13 | 9 | 0.012 | 40080 | 6 | 240272 |
| 13 | 10 | 0.002 | 39995 | 2 | 880085 |
| 14 | 1 | 0.019 | 40063 | 6 | 154392 |
| 14 | 2 | 0.007 | 40015 | 6 | 590524 |
| 14 | 3 | 0.018 | 40069 | 6 | 199106 |
| 14 | 4 | 0.021 | 40087 | 6 | 370708 |
| 14 | 5 | 0.160 | 40304 | 6 | 100159 |
| 14 | 6 | 0.022 | 40011 | 6 | 938468 |
| 14 | 7 | 0.001 | 39968 | 0 | 22642390 |
| 14 | 8 | 0.009 | 40039 | 6 | 319979 |
| 14 | 9 | 0.012 | 40079 | 6 | 162615 |
| 14 | 10 | 0.002 | 39995 | 2 | 881359 |

## *Appendix C. Supplemental Data Plots*

This appendix contains a catalog of supplemental data plots. Pages C-2 through C-11 depict the ten sets of DIDO data collected on the PUMA manipulator. Pages C-12 through C-21 depict the ten sets of DIDO collected on the PLIMMS. Pages C-22 through C-31 depict comparisons between the output vectors ,$\vec{V}$, of the PLIMMS DIDO data shown in Figures C.11 through C.20 and the corresponding computed DISO outputs for the same input vectors.

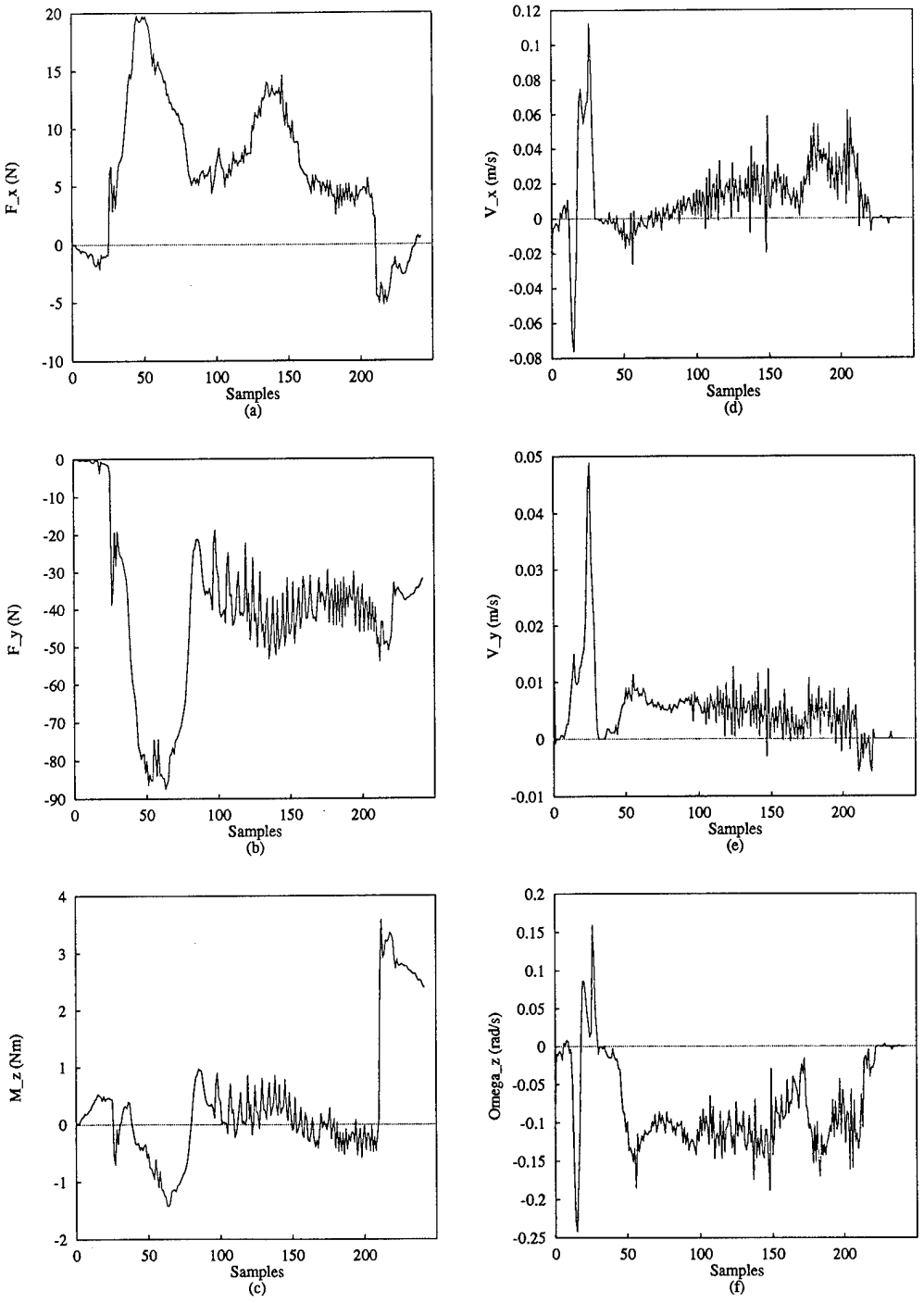



Figure C.1 Demonstration number 1 of DIDO training data collected on the PUMA manipulator

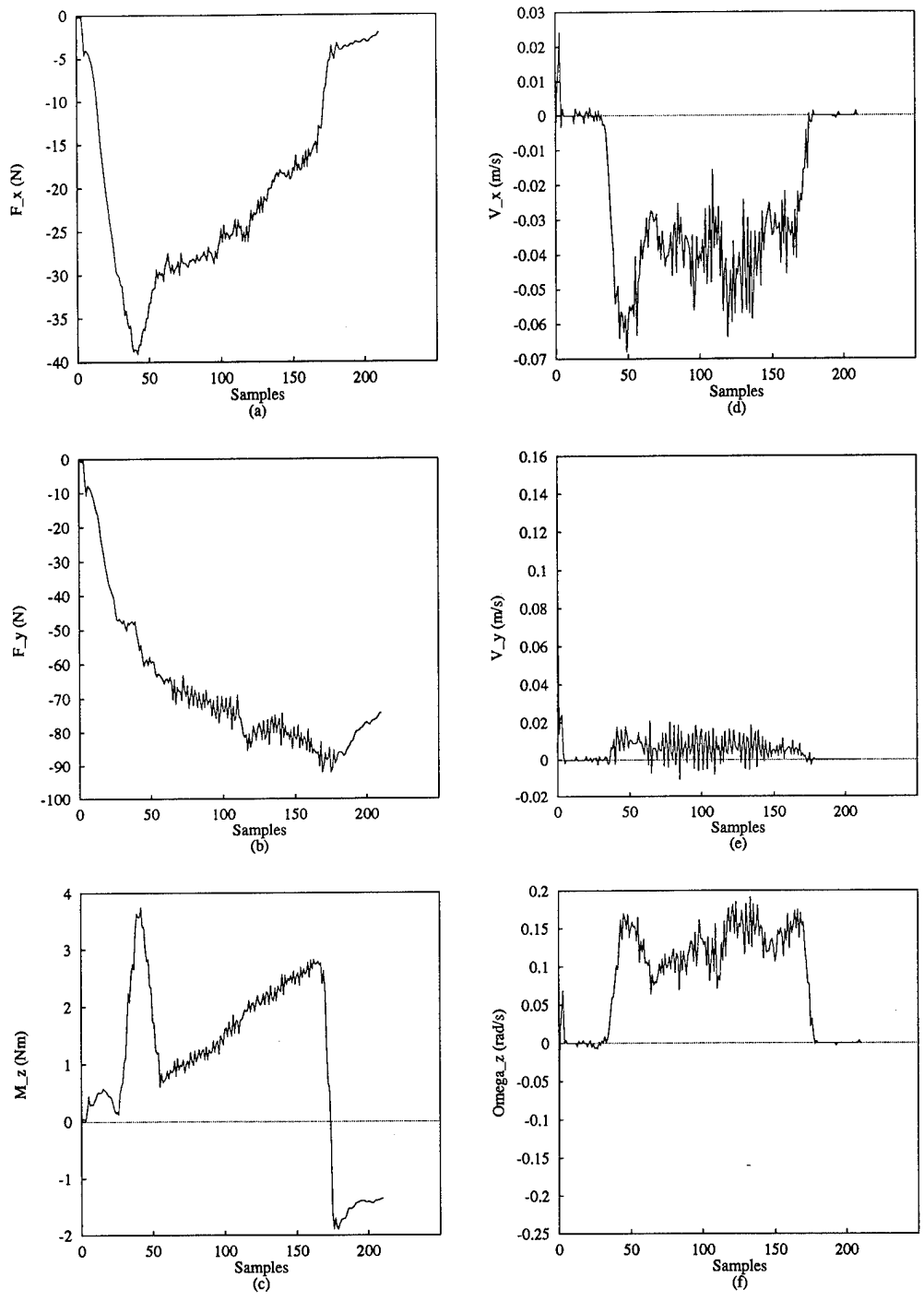



Figure C.2 Demonstration number 2 of DIDO training data collected on the PUMA manipulator

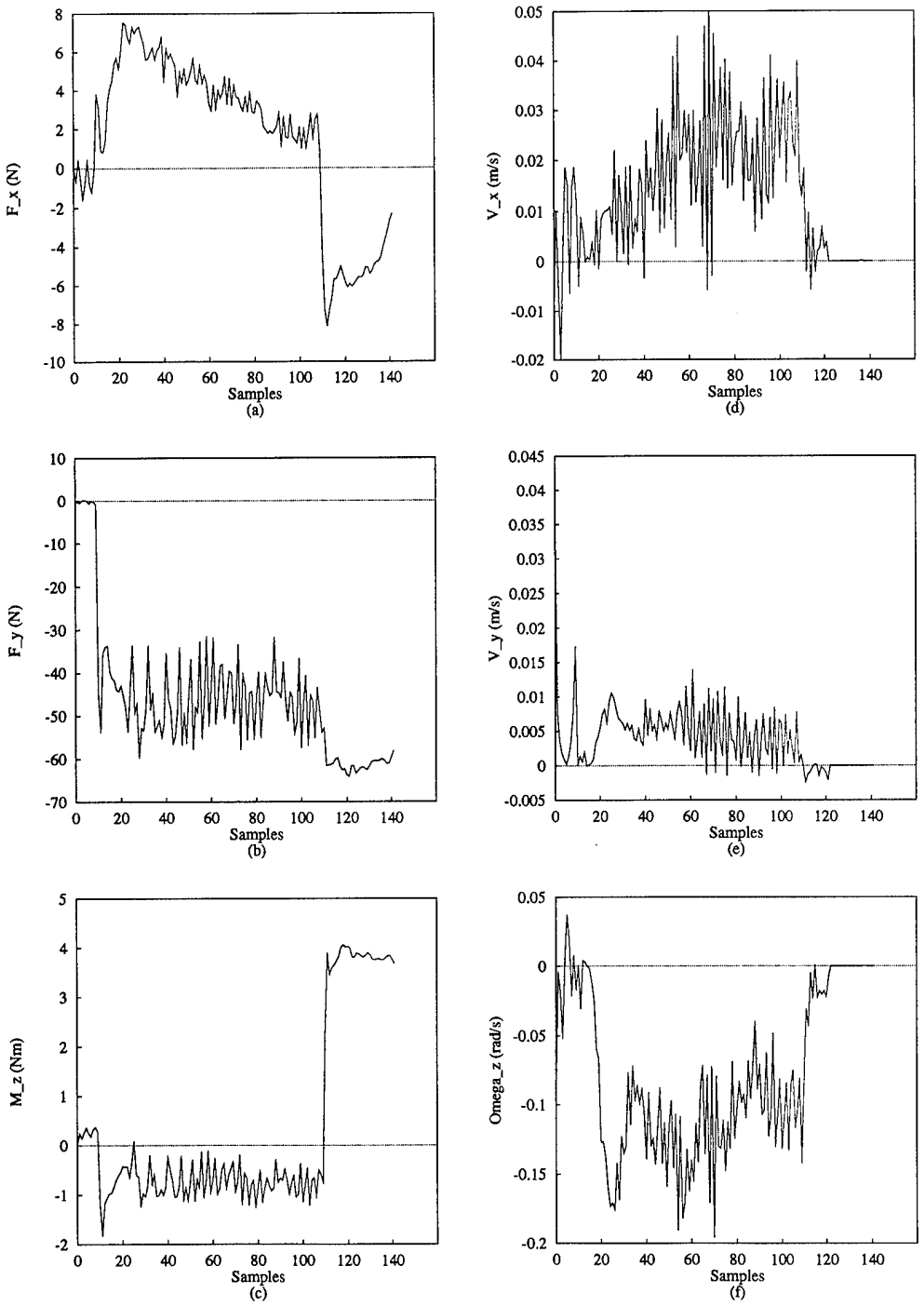



Figure C.3 Demonstration number 3 of DIDO training data collected on the PUMA manipulator

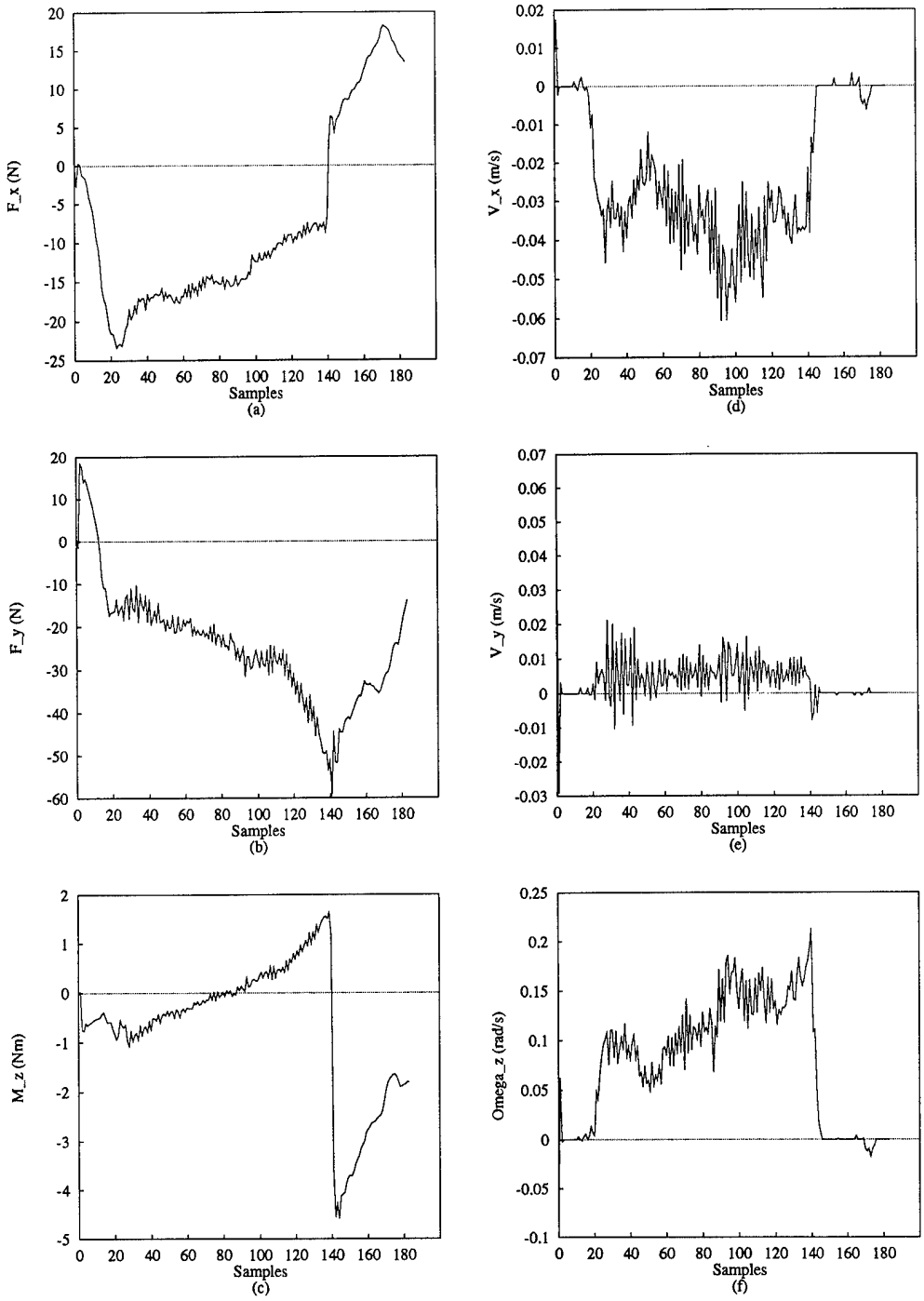



Figure C.4 Demonstration number 4 of DIDO training data collected on the PUMA manipulator

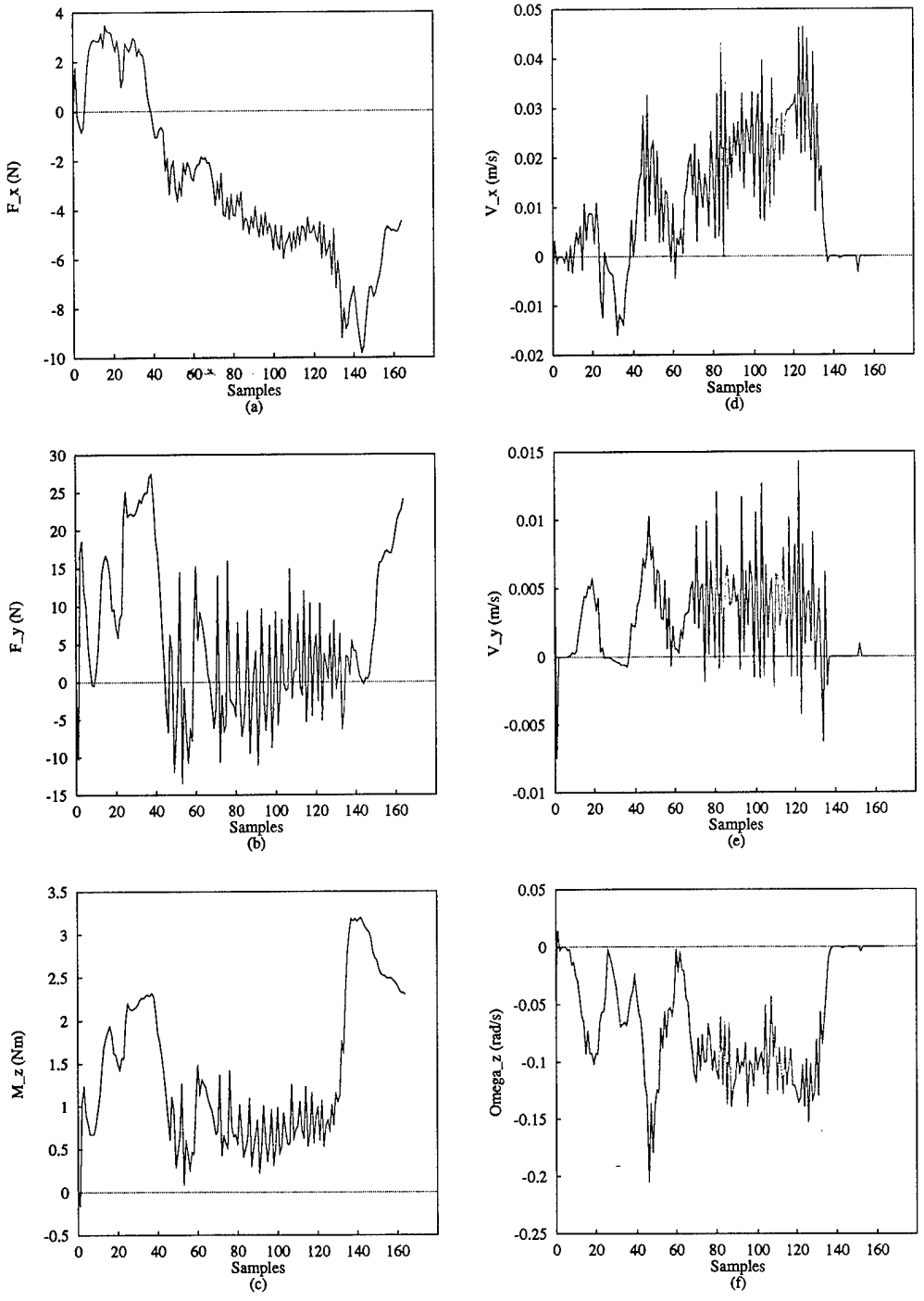



Figure C.5 Demonstration number 5 of DIDO training data collected on the PUMA manipulator

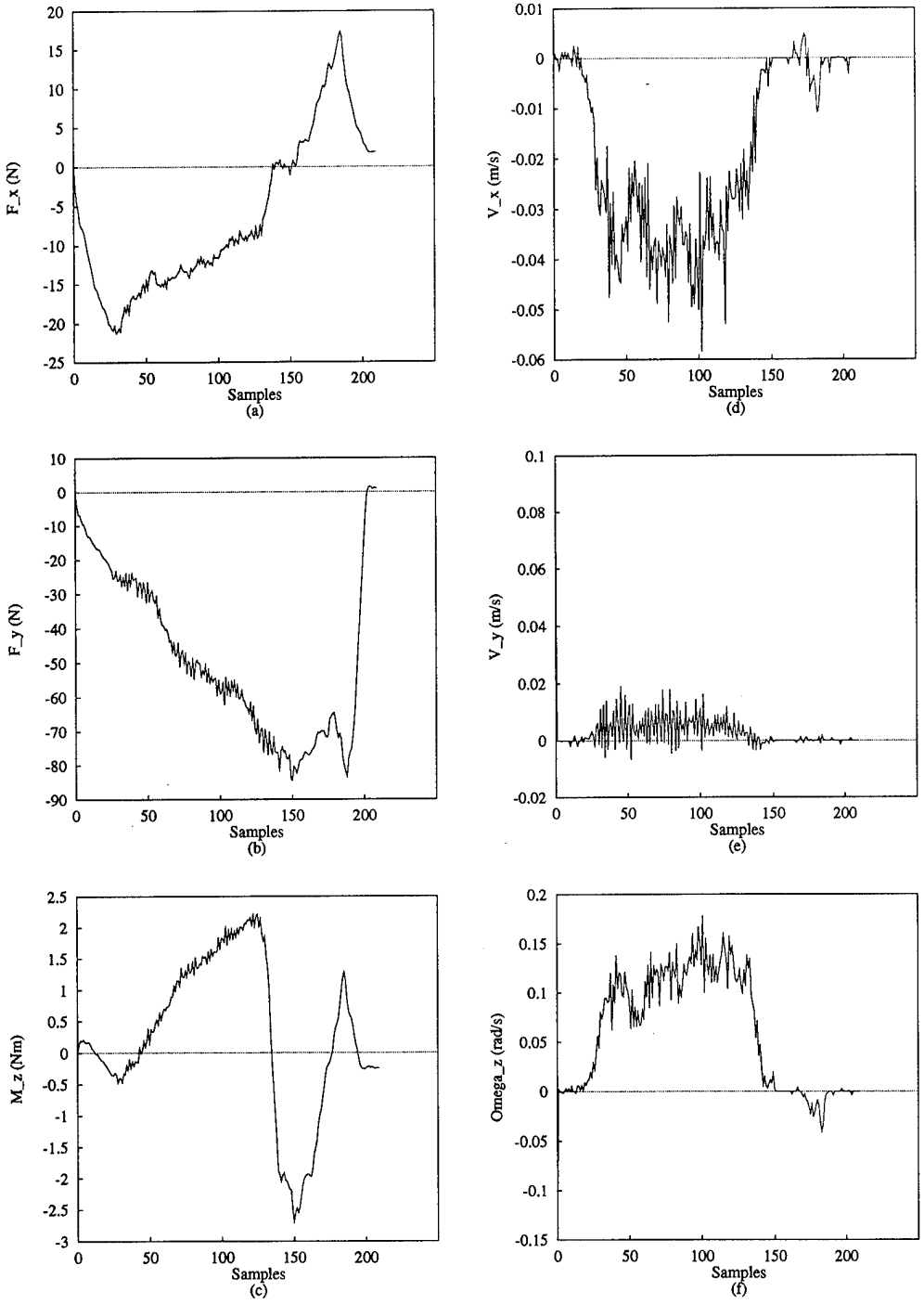



Figure C.6 Demonstration number 6 of DIDO training data collected on the PUMA manipulator

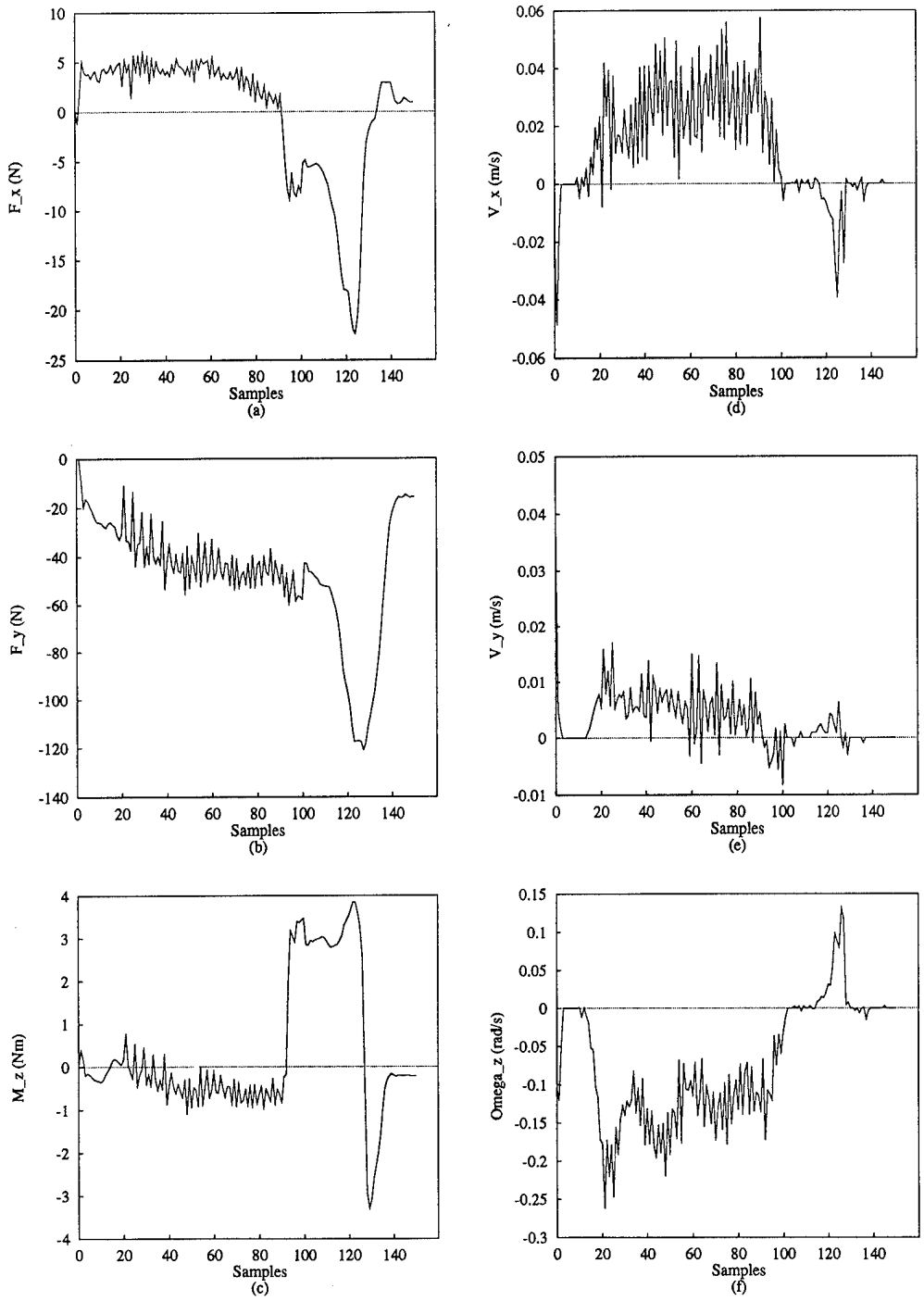



Figure C.7 Demonstration number 7 of DIDO training data collected on the PUMA manipulator

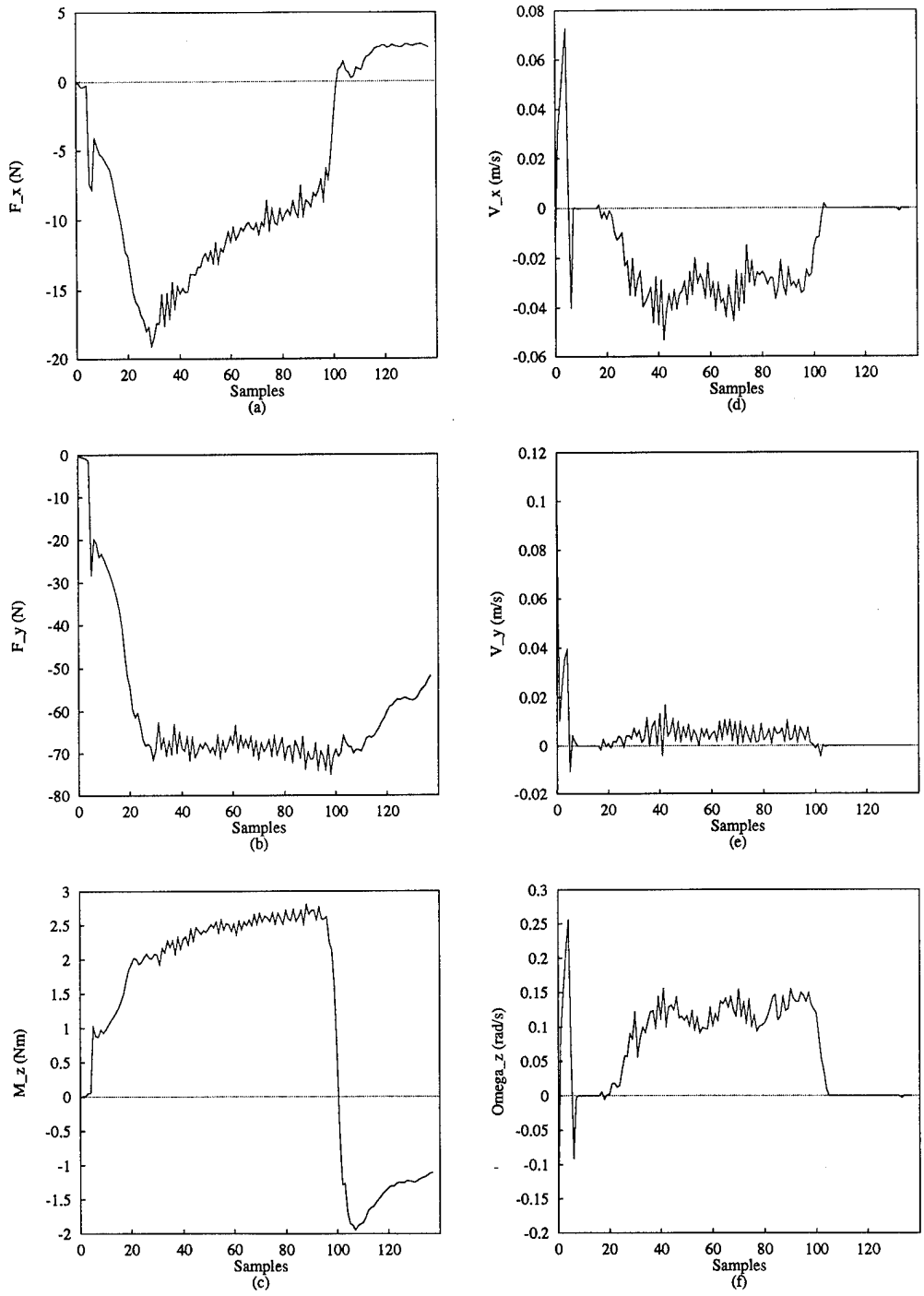



Figure C.8 Demonstration number 8 of DIDO training data collected on the PUMA manipulator

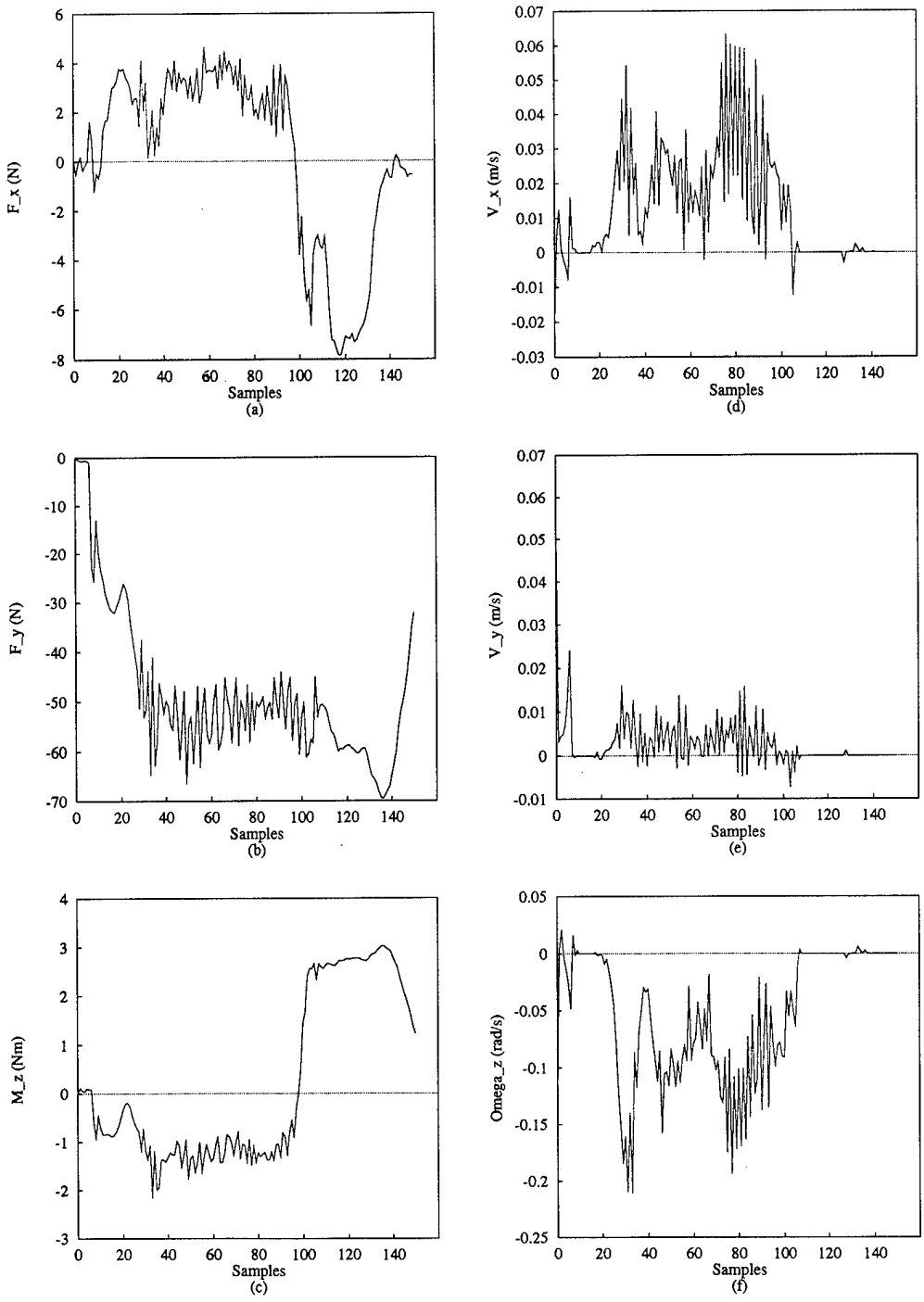



Figure C.9 Demonstration number 9 of DIDO training data collected on the PUMA manipulator

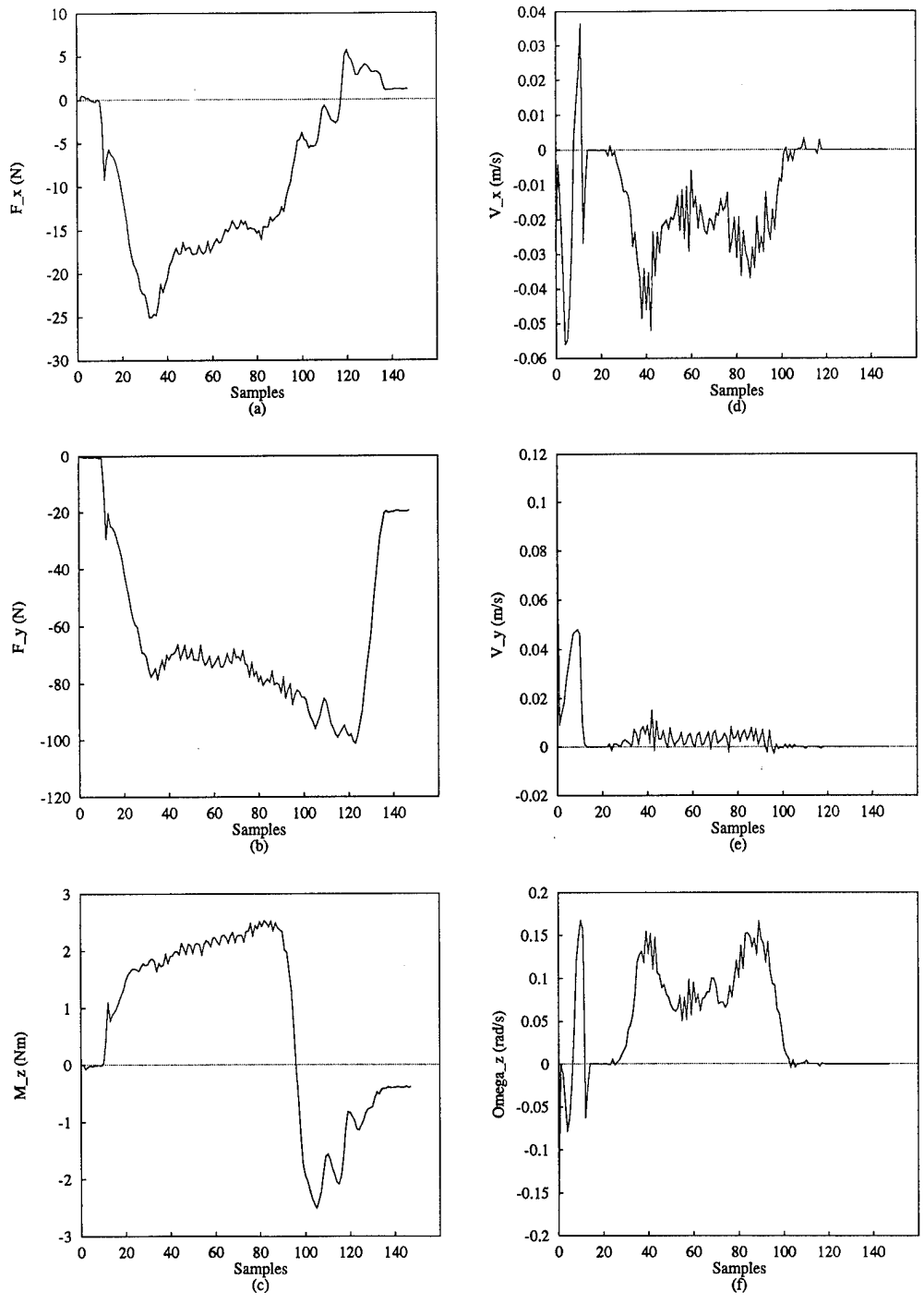



Figure C.10 Demonstration number 10 of DIDO training data collected on the PUMA manipulator

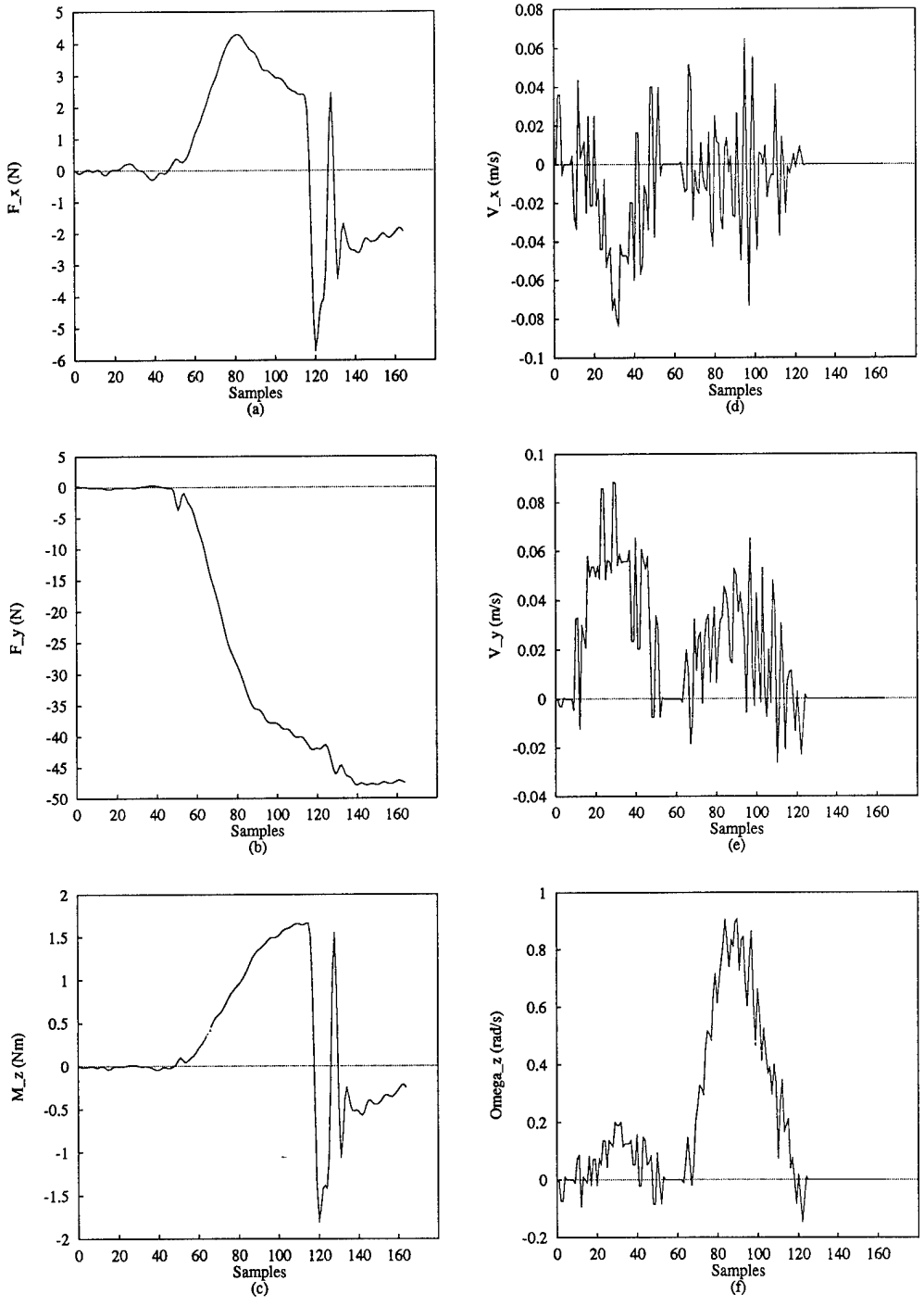



Figure C.11 Demonstration number 1 of DIDO training data collected on the PLIMMS

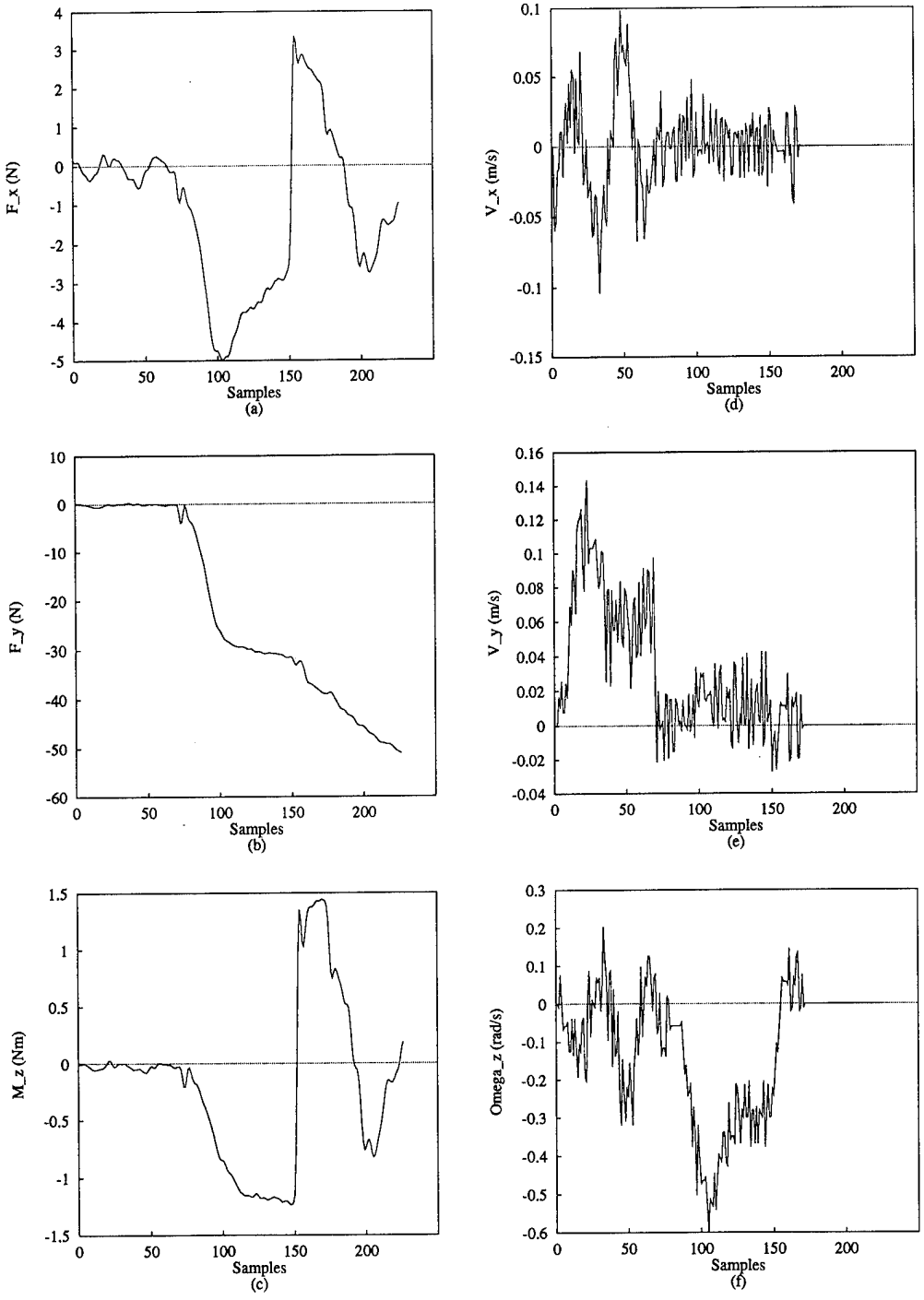



Figure C.12 Demonstration number 2 of DIDO training data collected on the PLIMMS

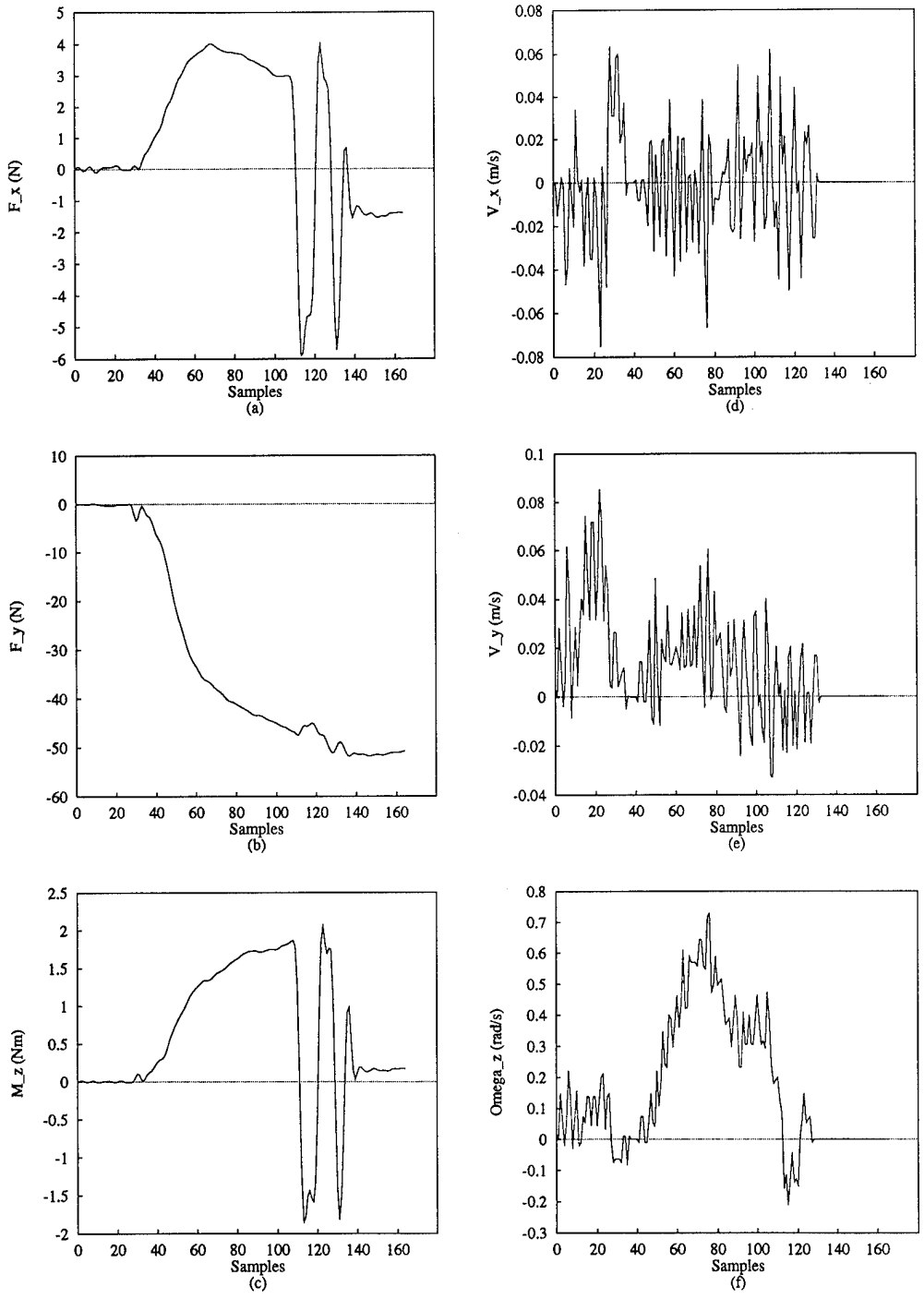



Figure C.13 Demonstration number 3 of DIDO training data collected on the PLIMMS

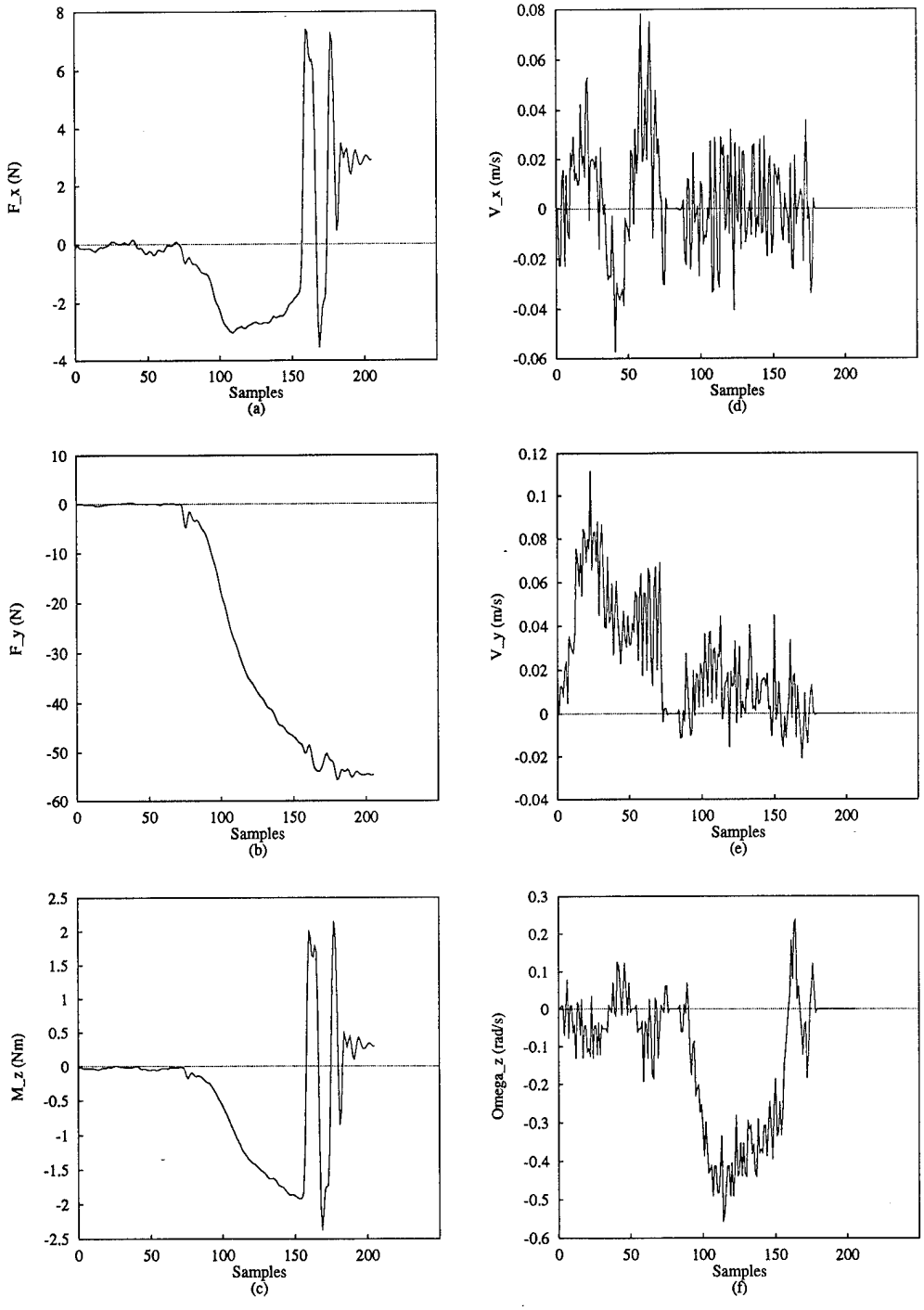



Figure C.14 Demonstration number 4 of DIDO training data collected on the PLIMMS

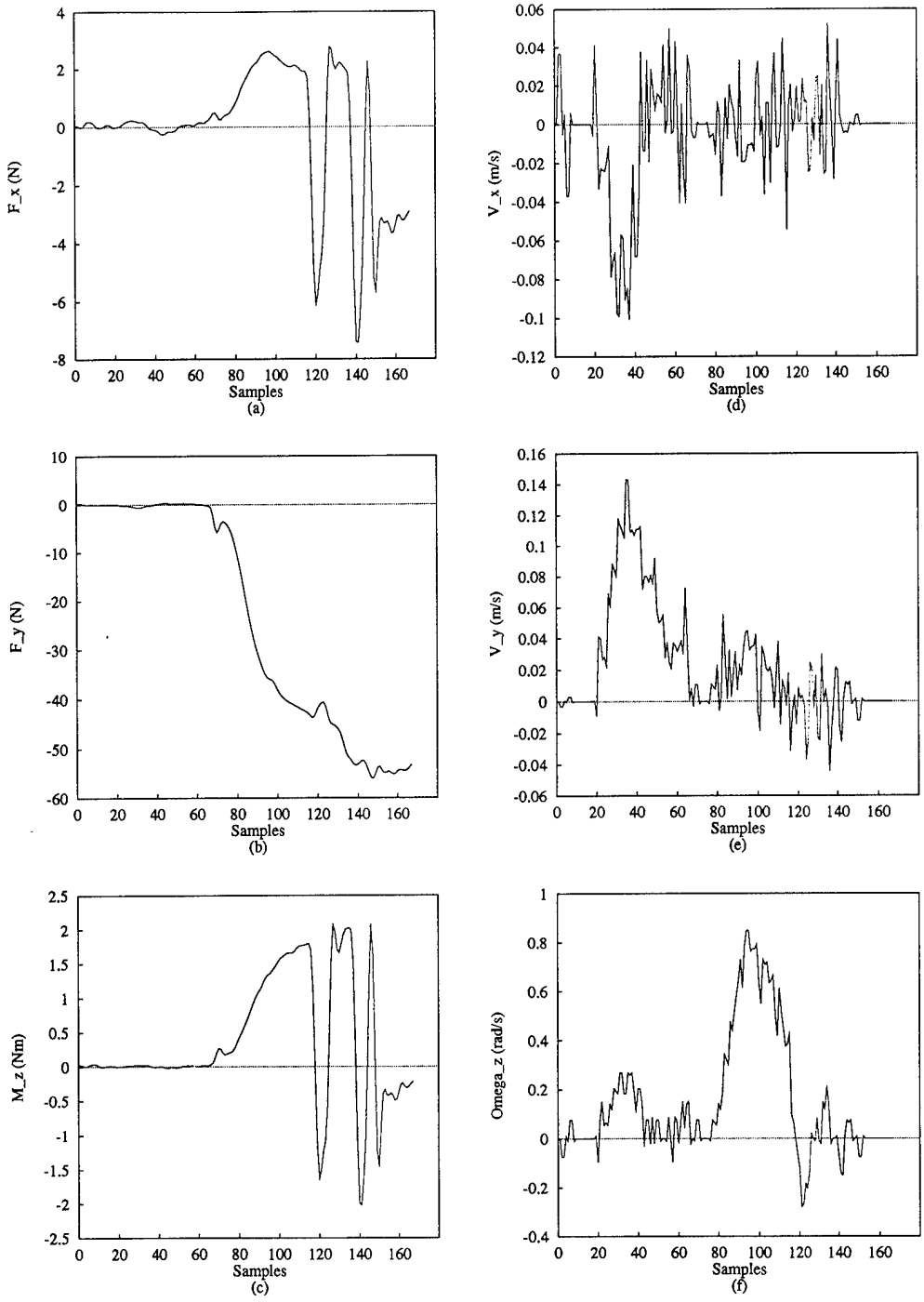



Figure C.15 Demonstration number 5 of DIDO training data collected on the PLIMMS

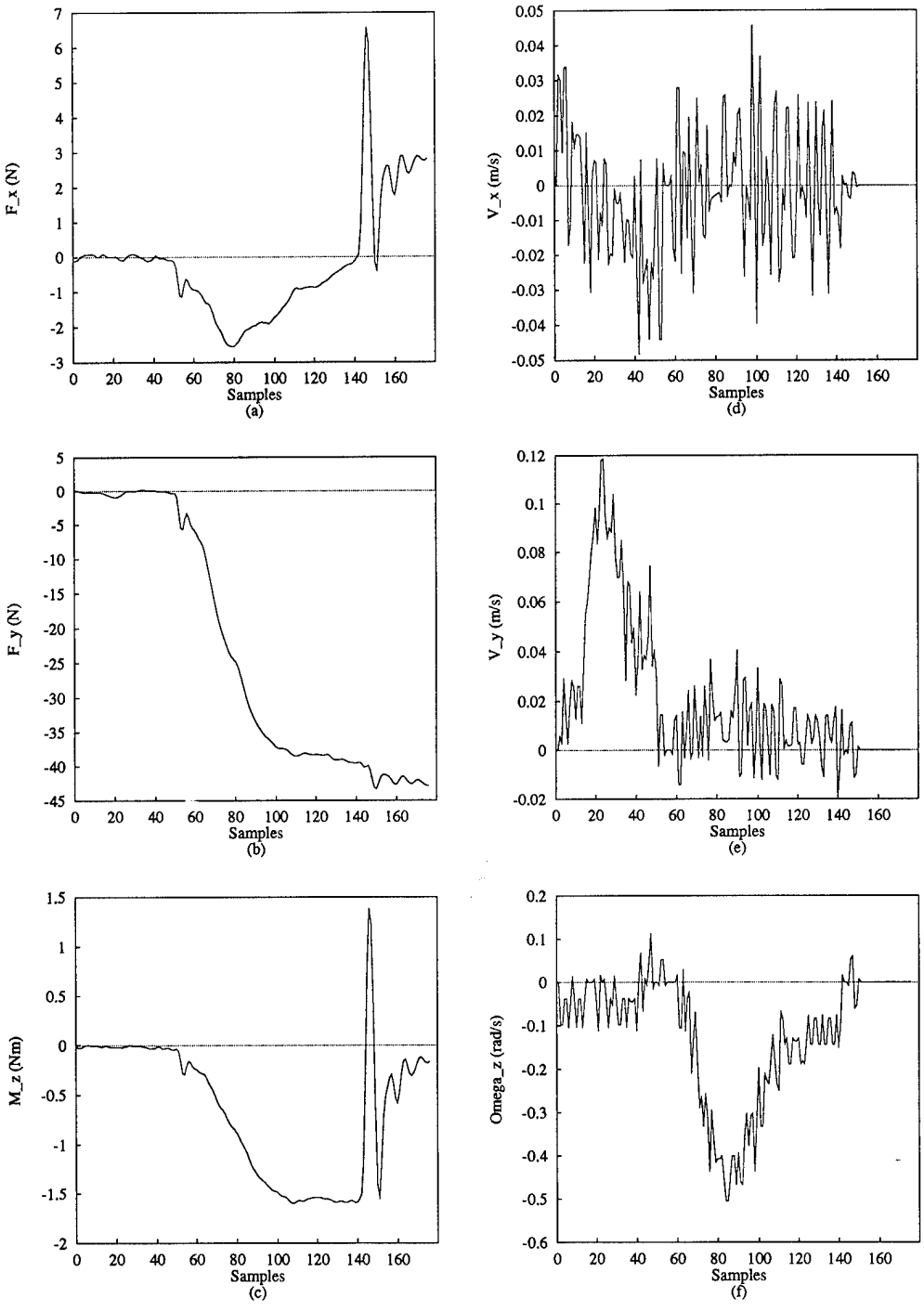



Figure C.16 Demonstration number 6 of DIDO training data collected on the PLIMMS

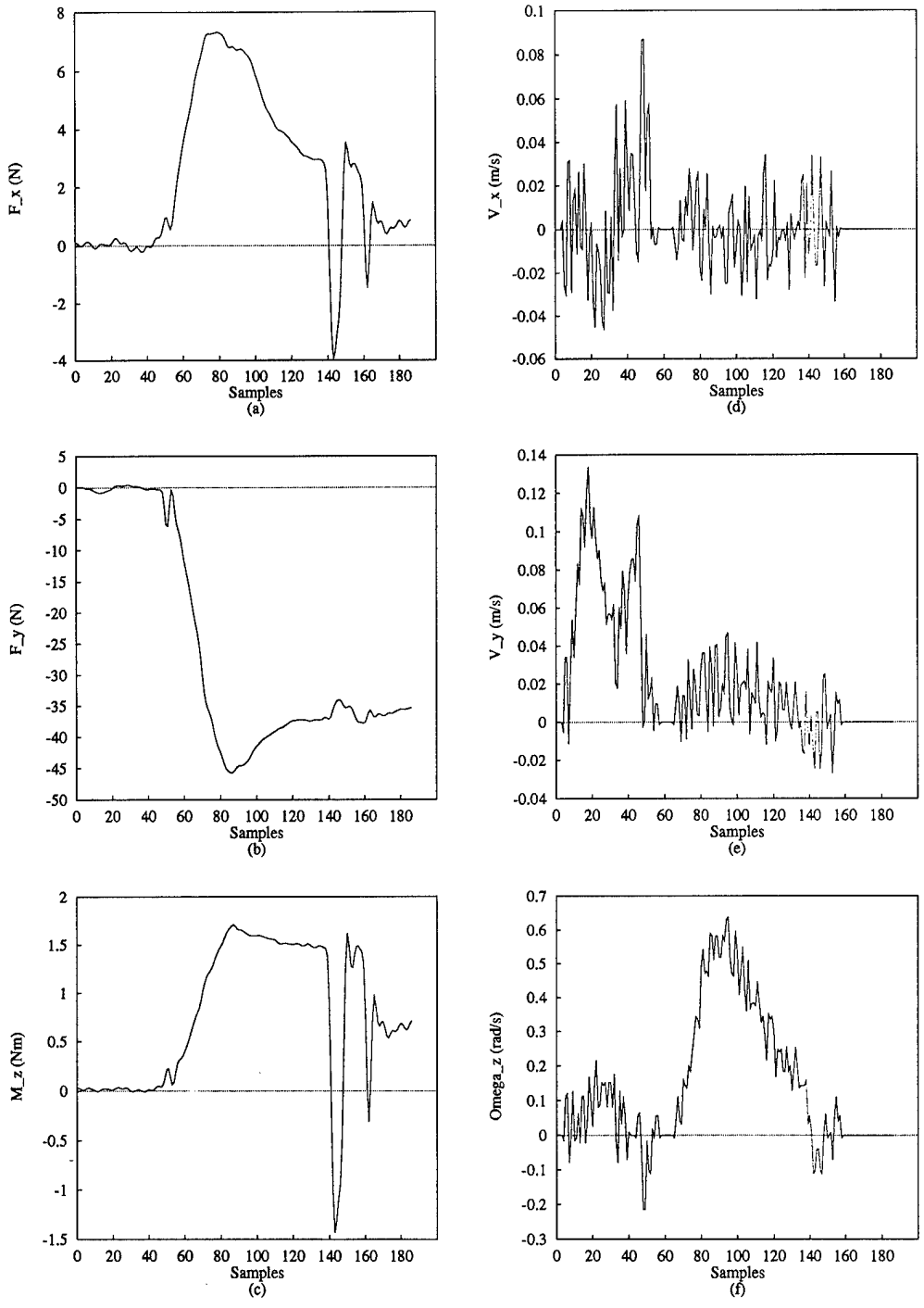



Figure C.17 Demonstration number 7 of DIDO training data collected on the PLIMMS

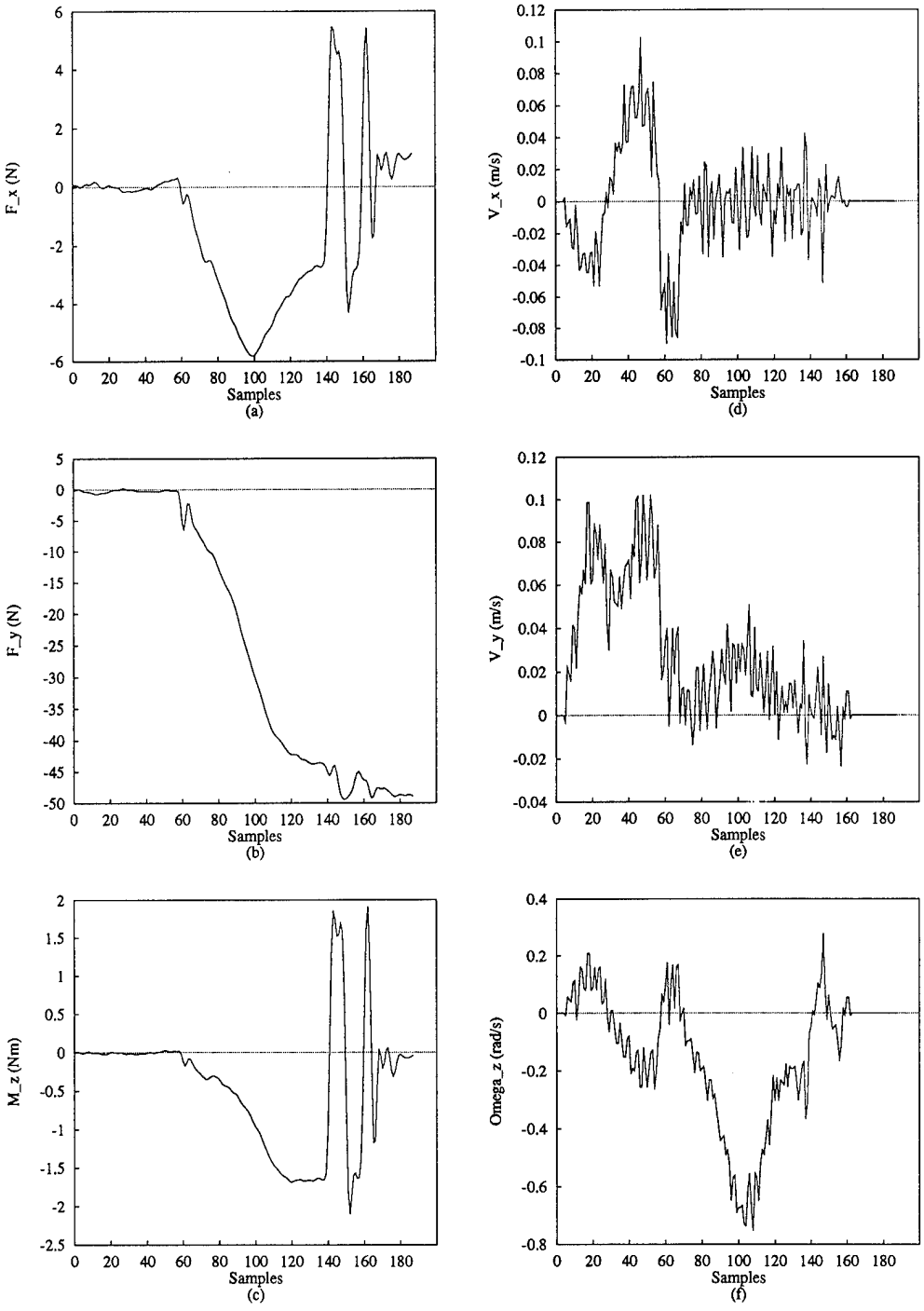



Figure C.18 Demonstration number 8 of DIDO training data collected on the PLIMMS

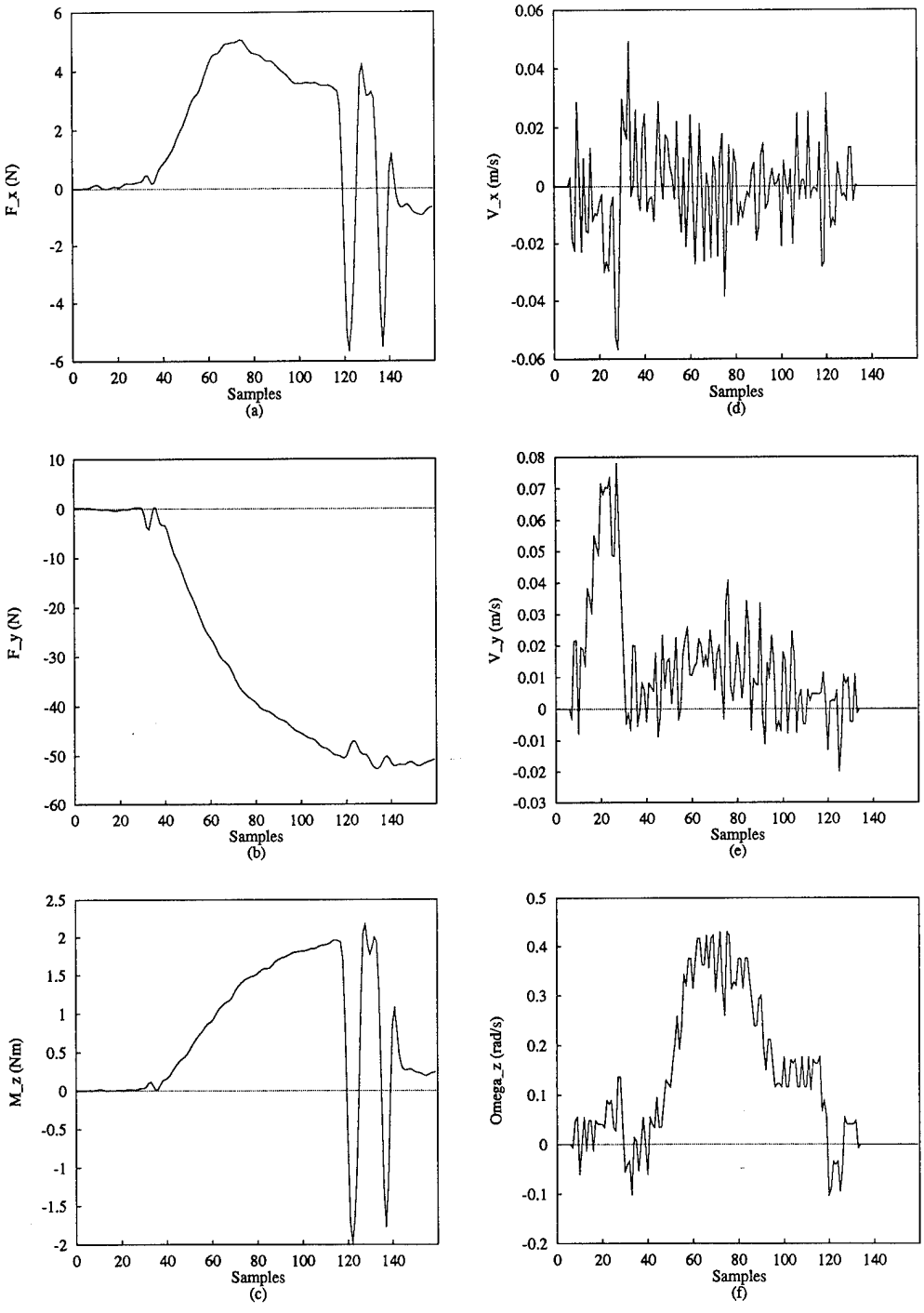



Figure C.19 Demonstration number 9 of DIDO training data collected on the PLIMMS

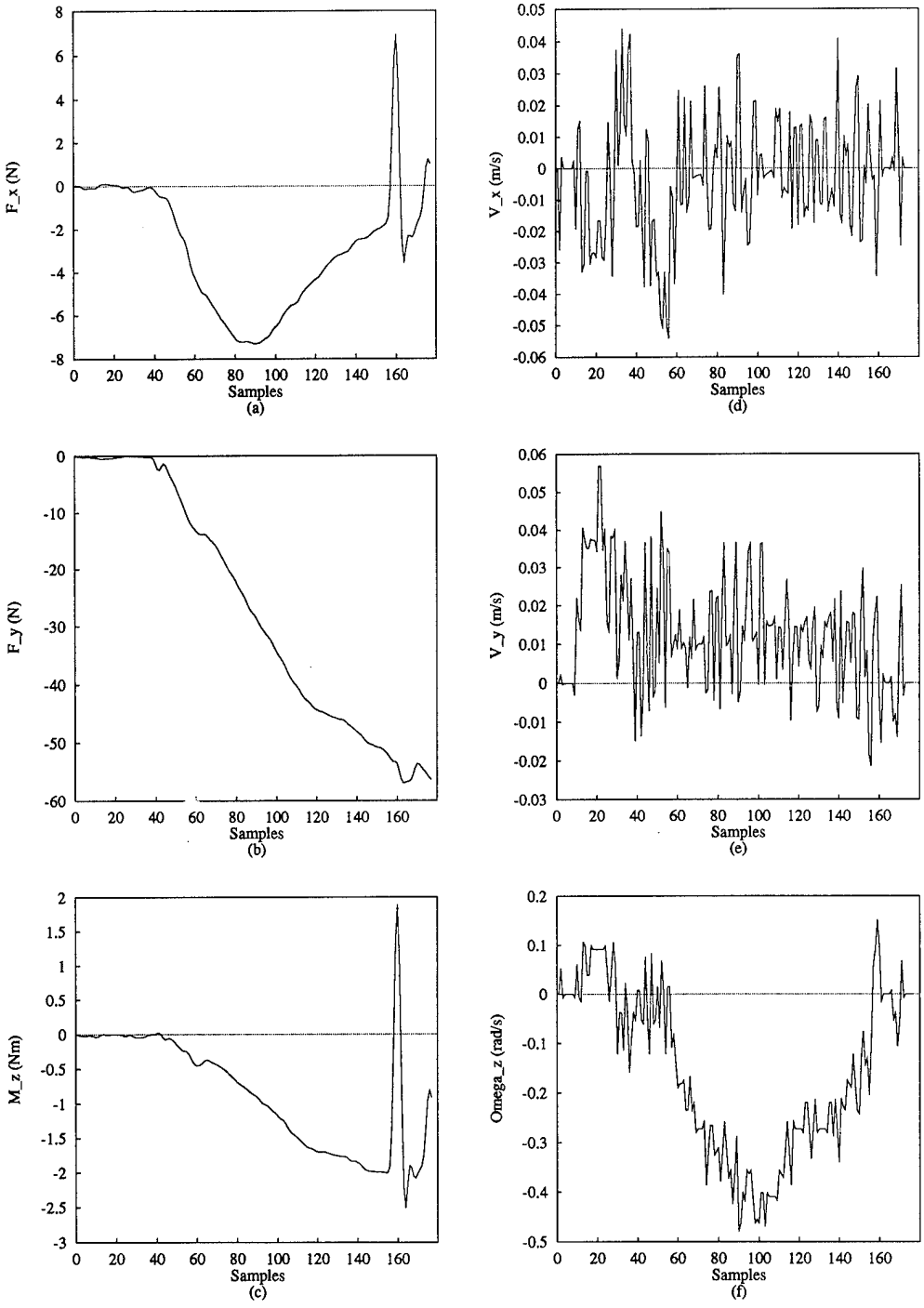



Figure C.20 Demonstration number 10 of DIDO training data collected on the PLIMMS

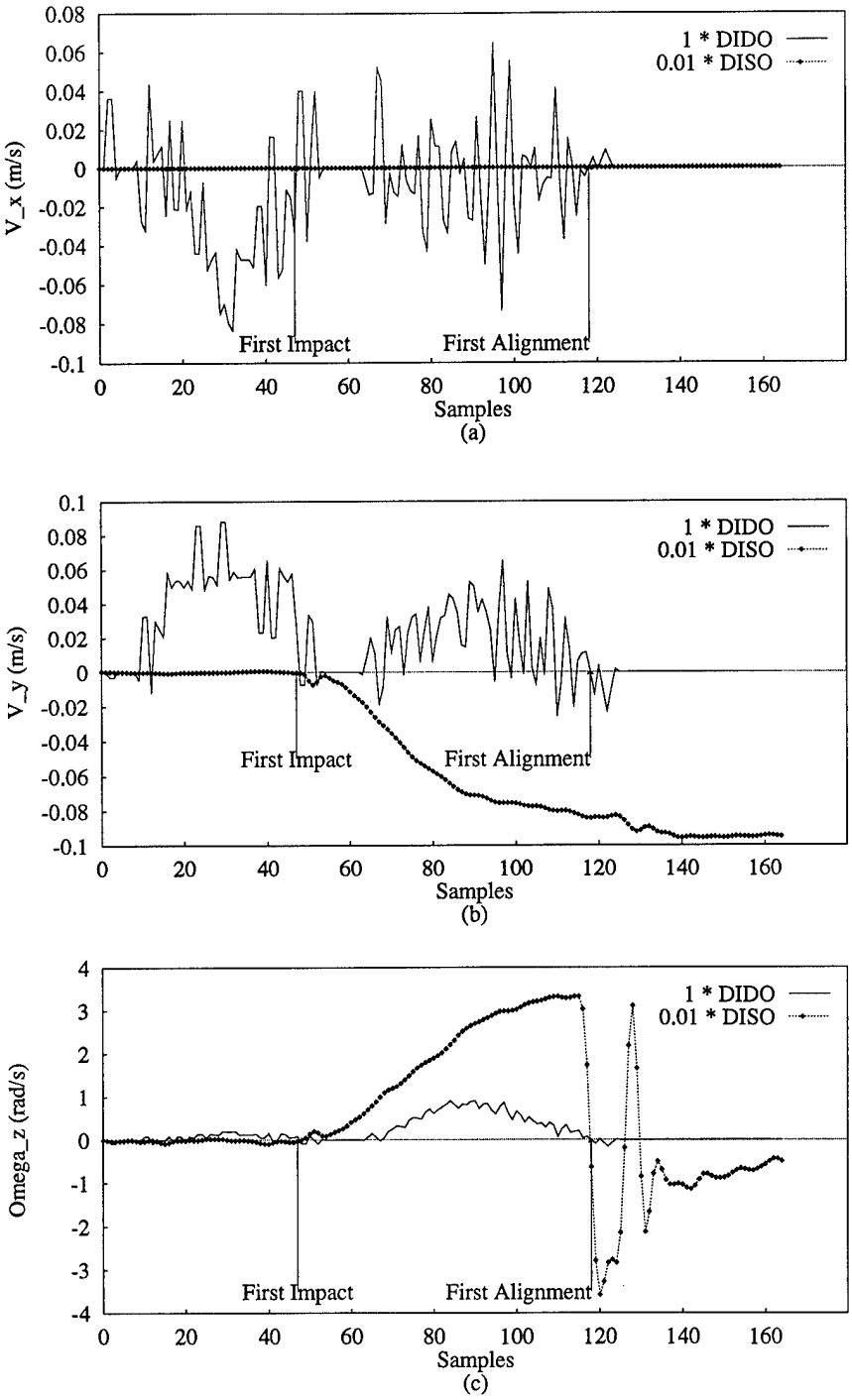



Figure C.21 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 1 of the PLIMMS DIDO training data (ref. Figure C.11) and their corresponding DISO outputs

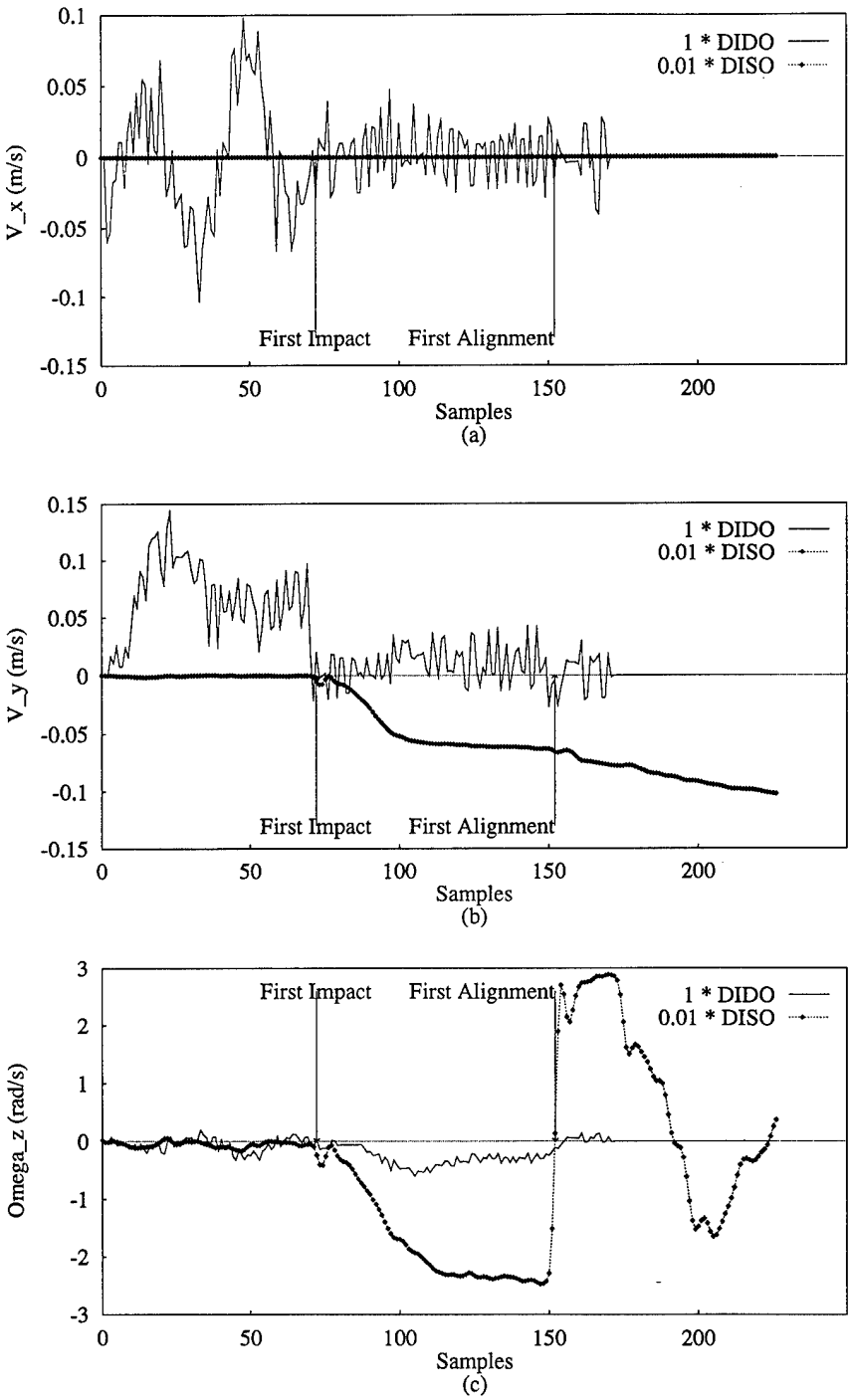



Figure C.22 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 2 of the PLIMMS DIDO training data (ref. Figure C.12) and their corresponding DISO outputs

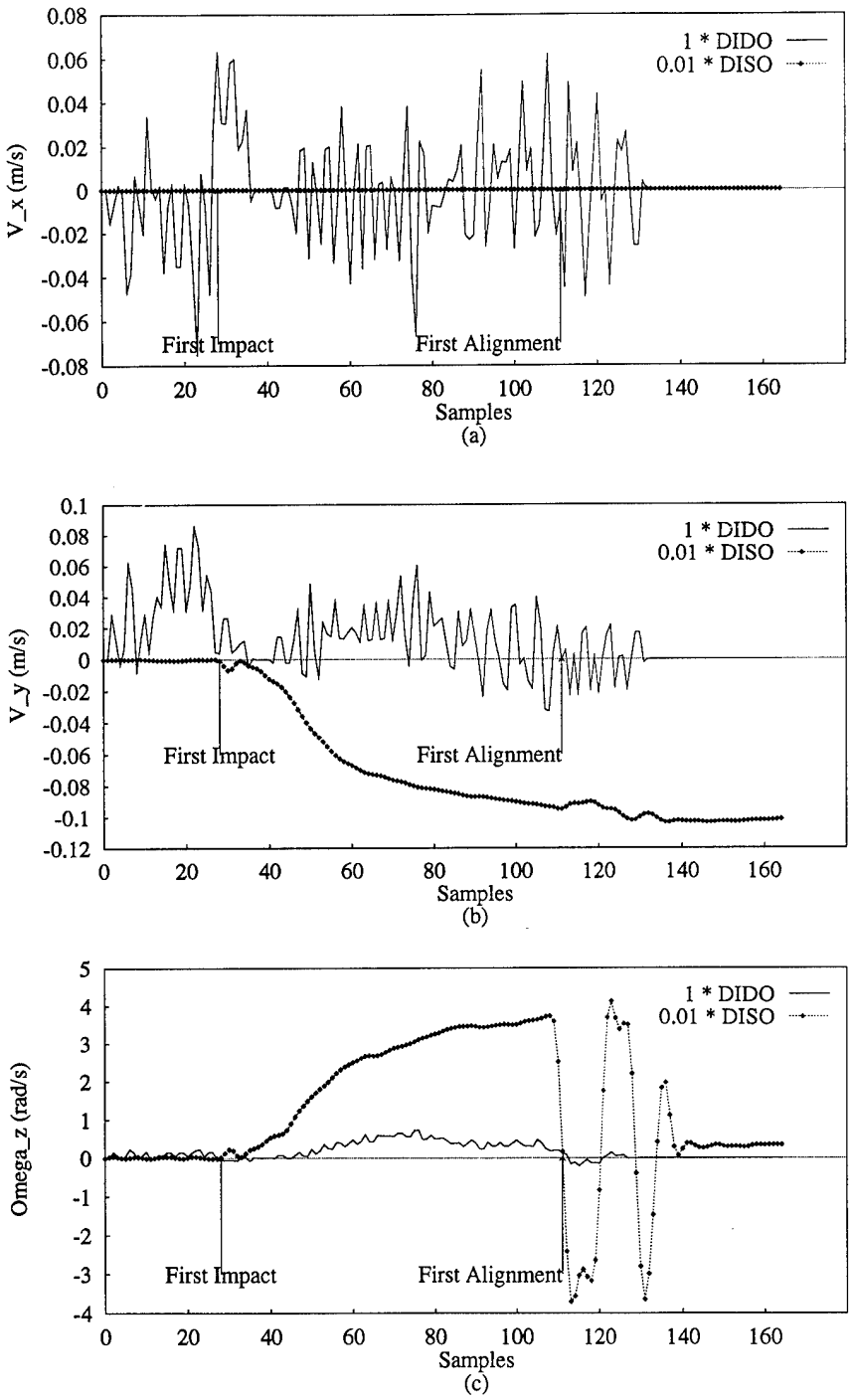



Figure C.23 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 3 of the PLIMMS DIDO training data (ref. Figure C.13) and their corresponding DISO outputs

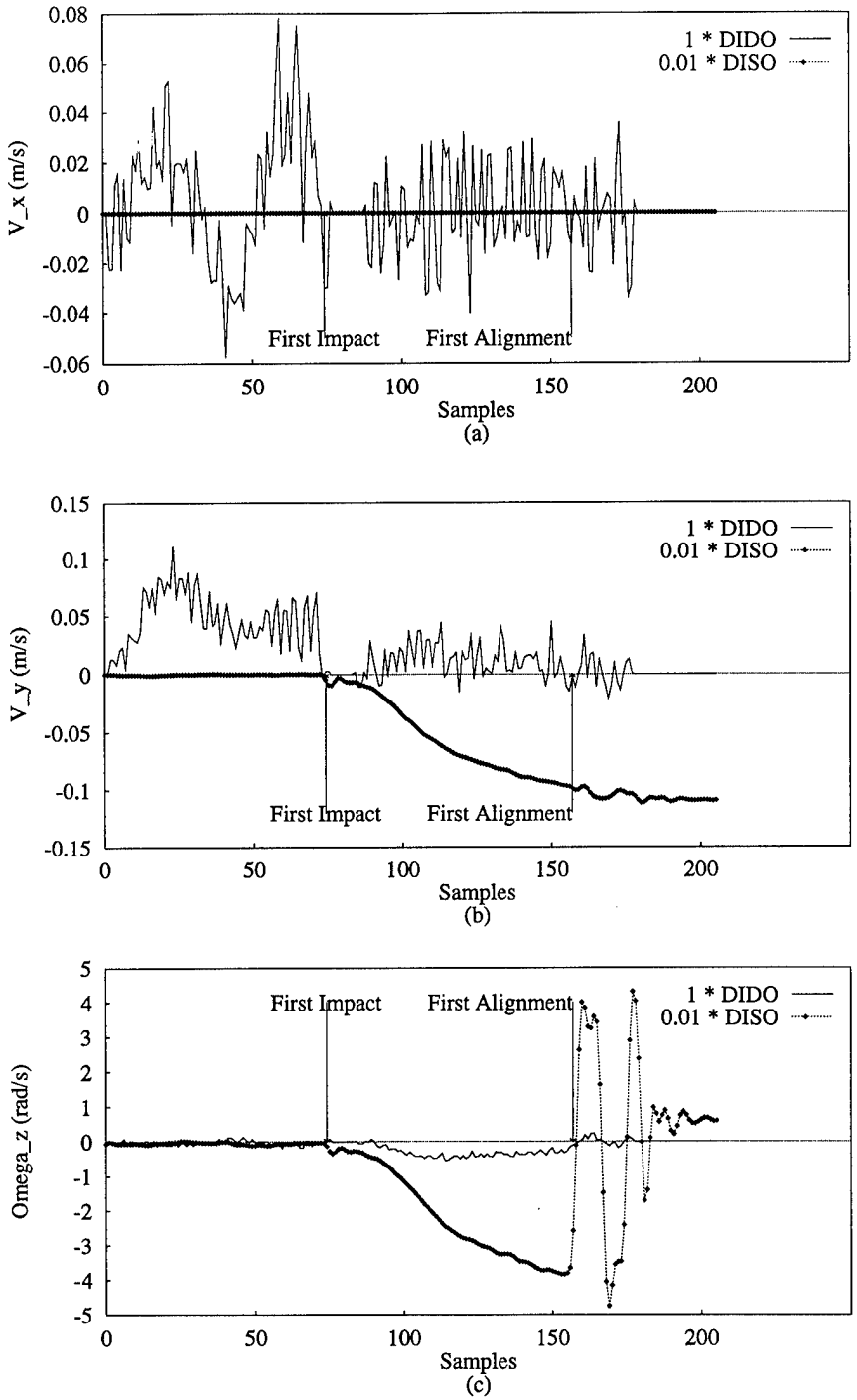



Figure C.24 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 4 of the PLIMMS DIDO training data (ref. Figure C.14) and their corresponding DISO outputs

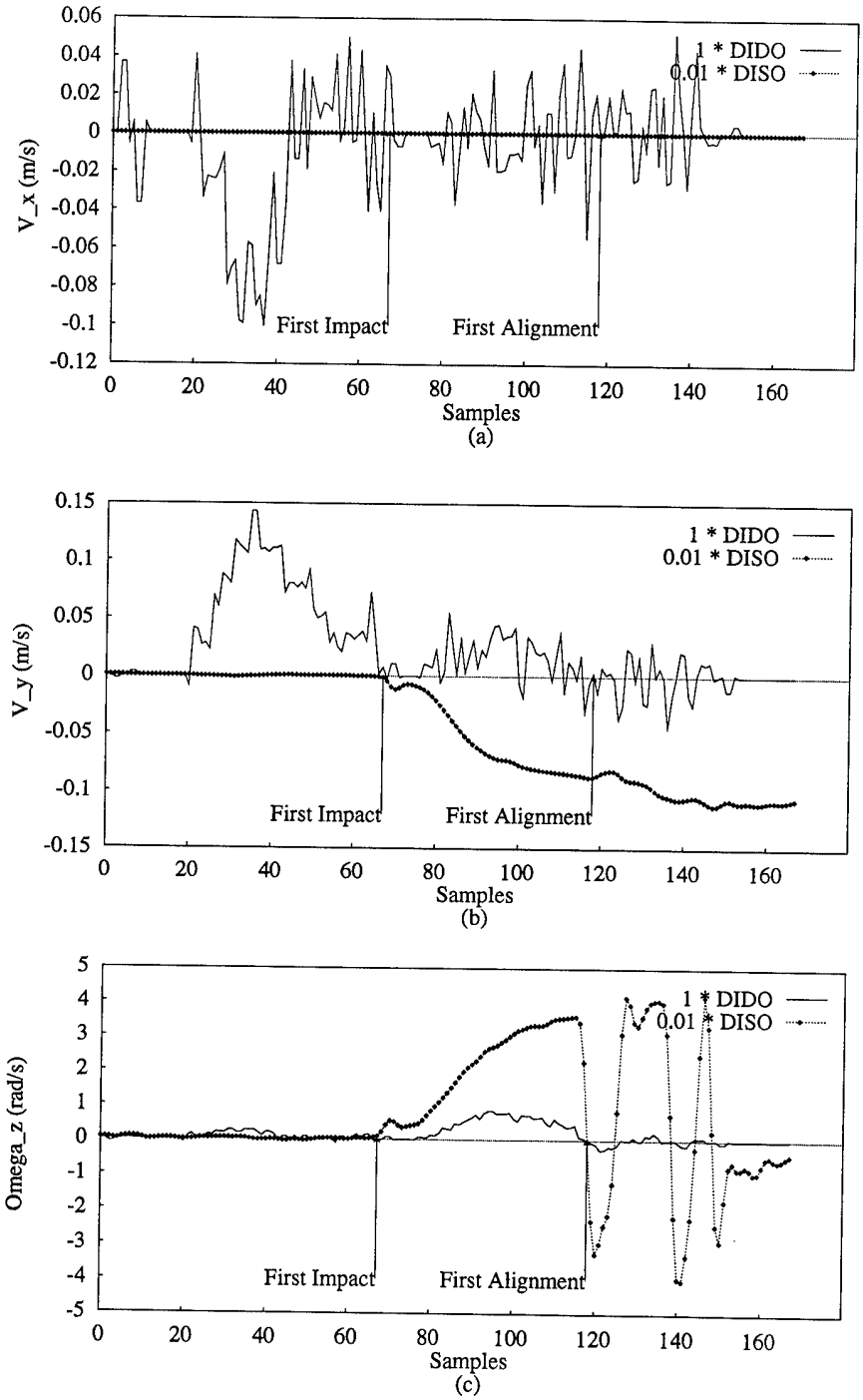



Figure C.25 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 5 of the PLIMMS DIDO training data (ref. Figure C.15) and their corresponding DISO outputs

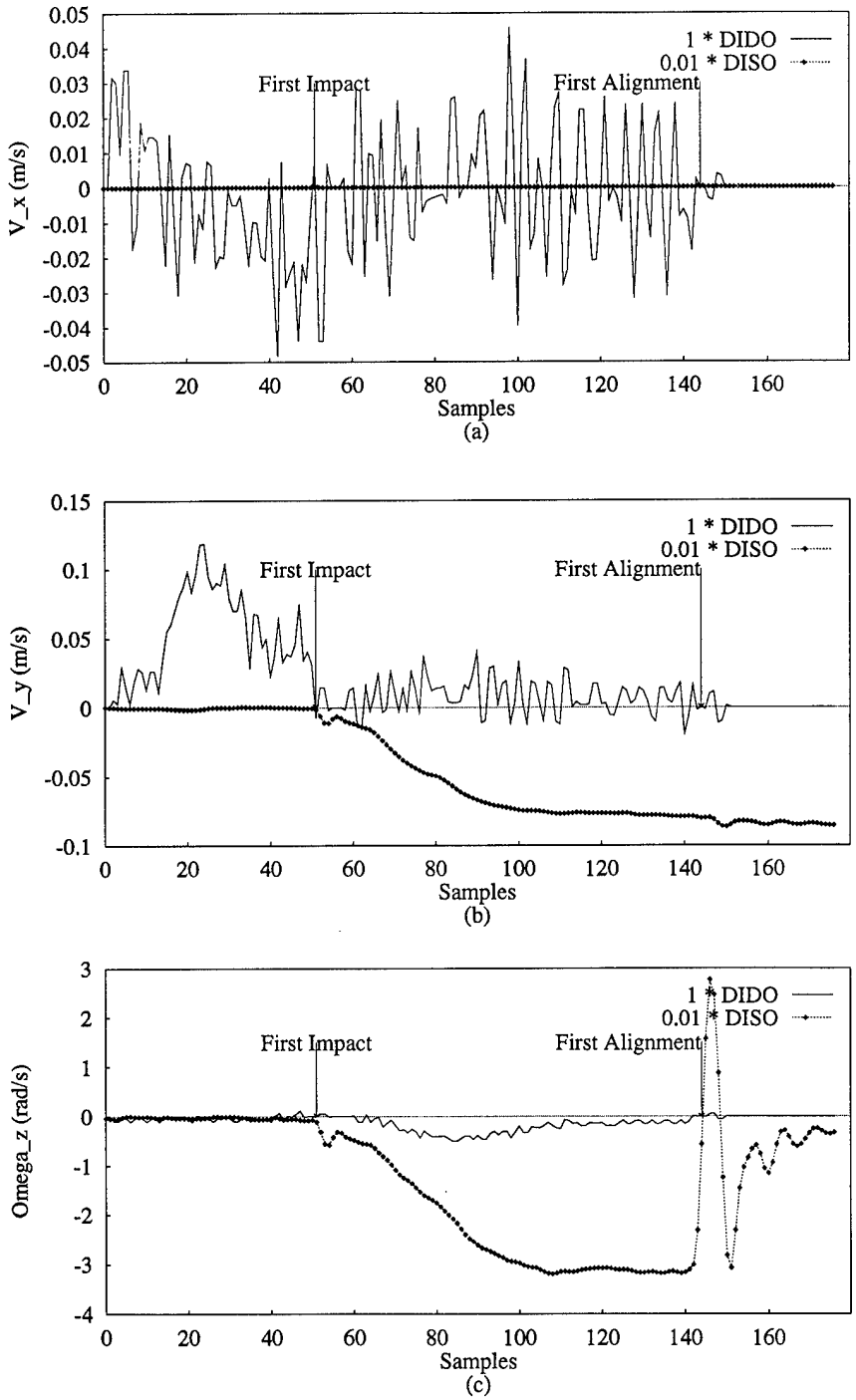



Figure C.26 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 6 of the PLIMMS DIDO training data (ref. Figure C.16) and their corresponding DISO outputs

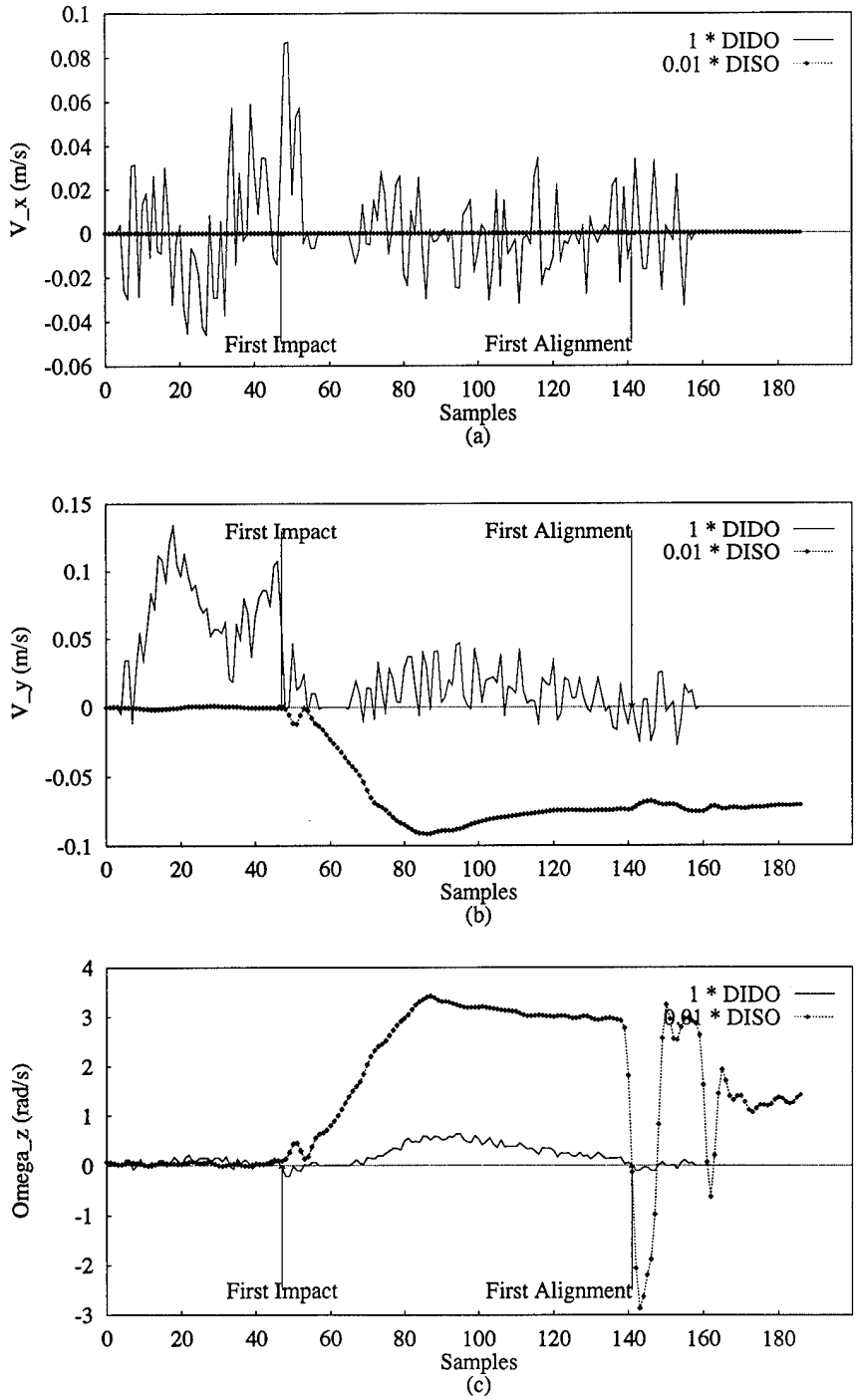



Figure C.27 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 7 of the PLIMMS DIDO training data (ref. Figure C.17) and their corresponding DISO outputs

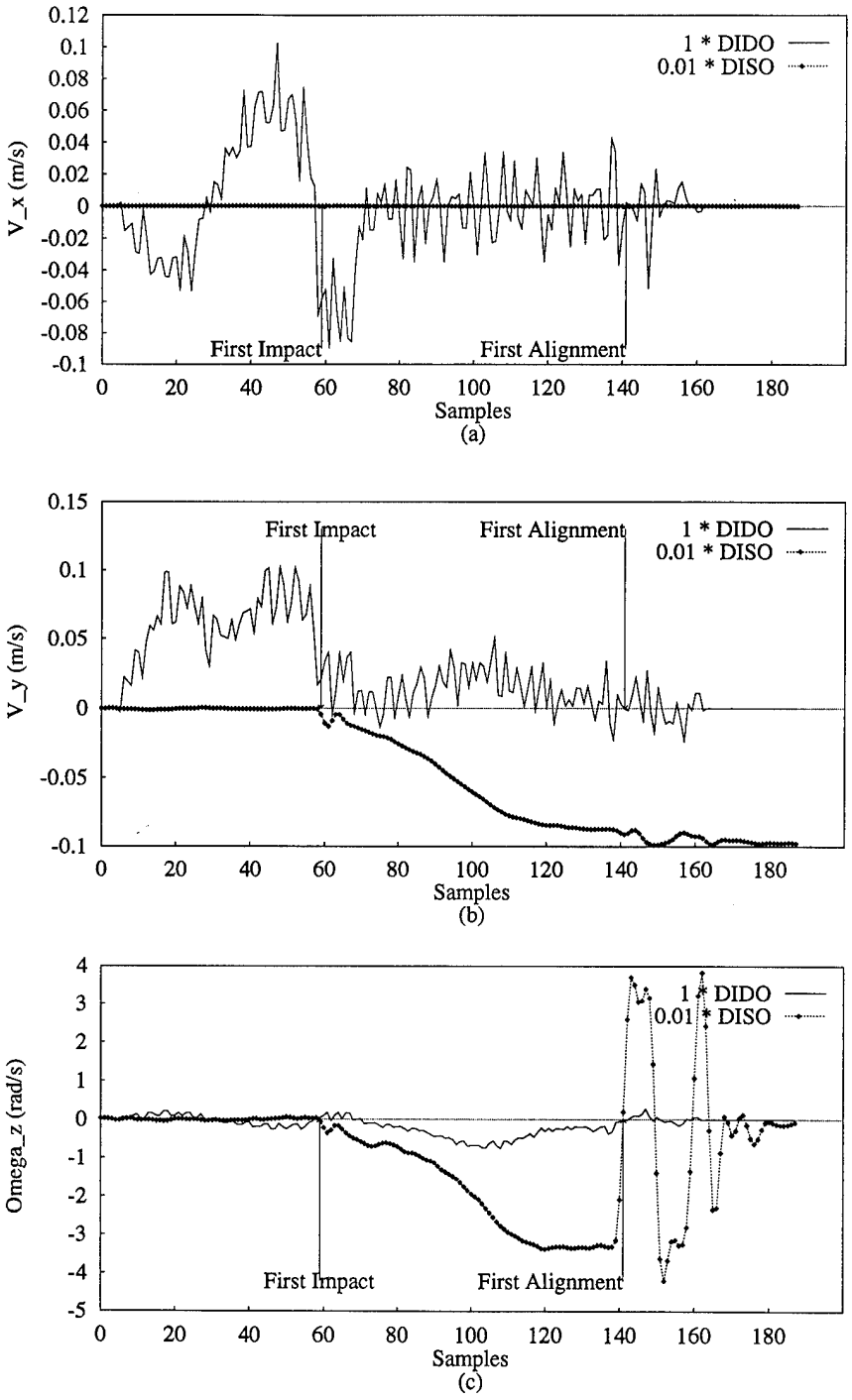



Figure C.28 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 8 of the PLIMMS DIDO training data (ref. Figure C.18) and their corresponding DISO outputs

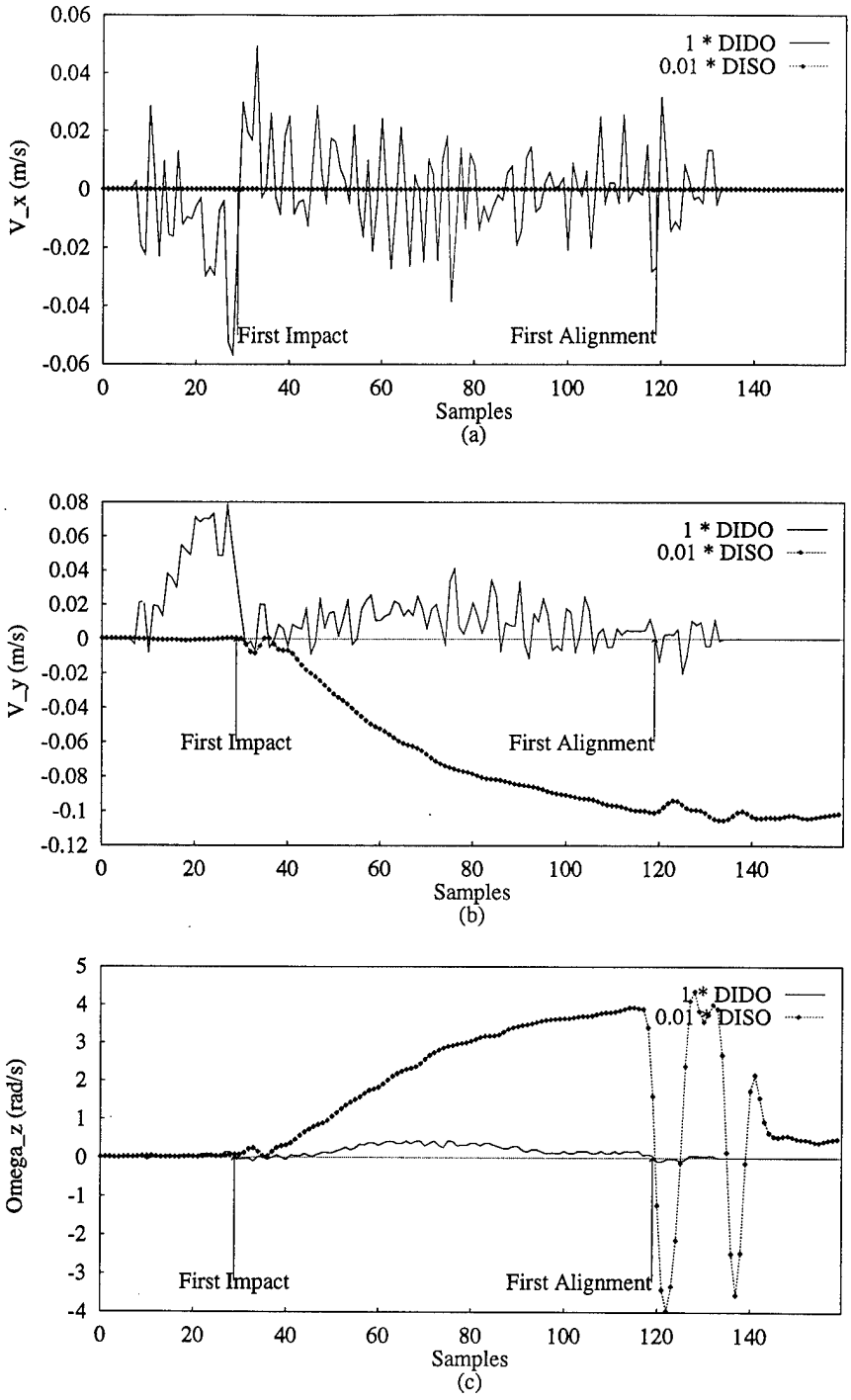



Figure C.29 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 9 of the PLIMMS DIDO training data (ref. Figure C.19) and their corresponding DISO outputs

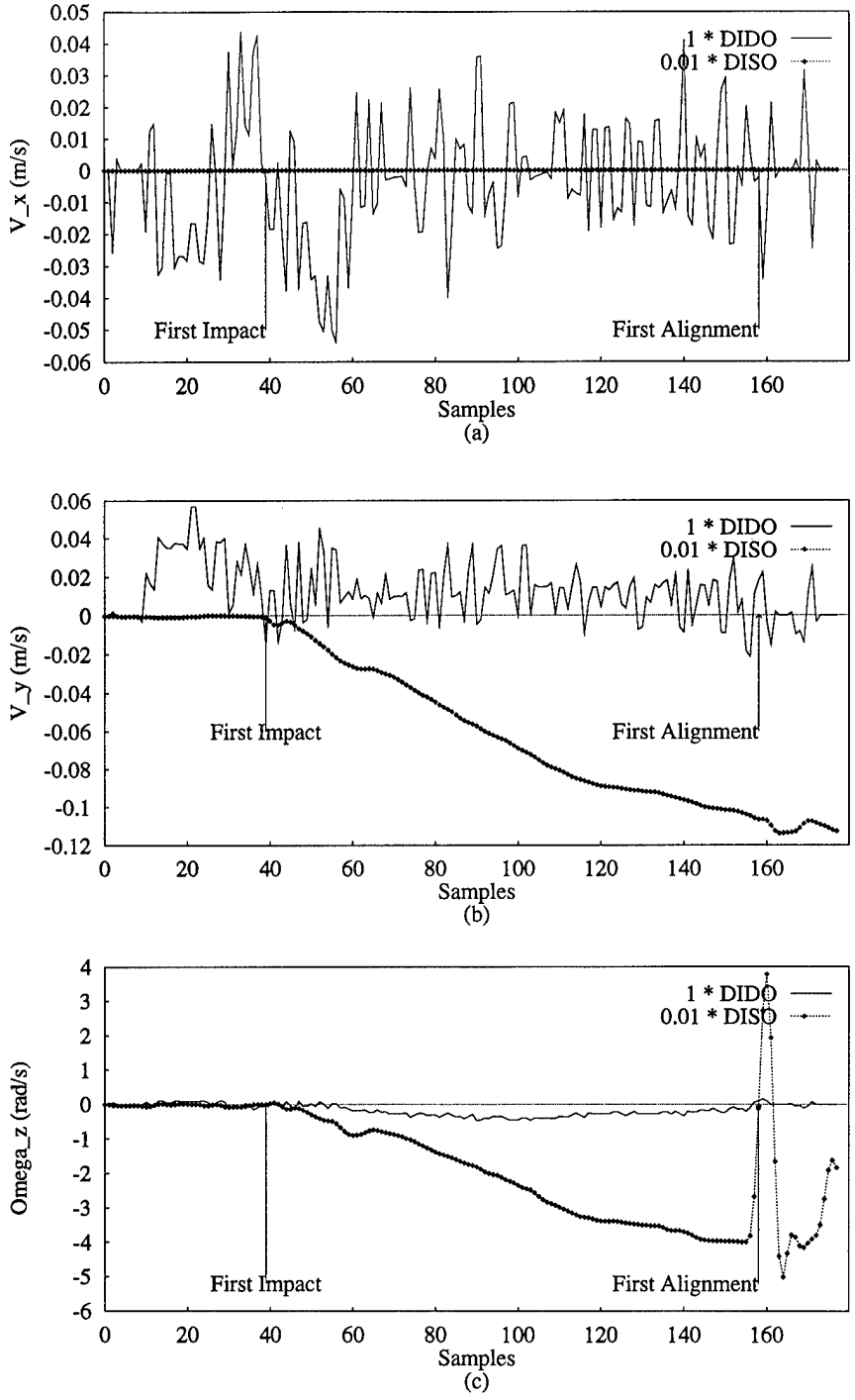



Figure C.30 Comparison of (a) $V_x$, (b) $V_y$, and (c) $\omega_z$ outputs for demonstration number 10 of the PLIMMS DIDO training data (ref. Figure C.20) and their corresponding DISO outputs

## *Appendix D. Untried Processing Options.*

This Appendix describes three data processing options which were conceived but not investigated and evaluated within the scope of the present work. They are items for future work.

### *D.1 Time Shifting.*

Time shifting is used to compensate for processing delay between measured force and reaction-commanded velocity in the collected training data. For example, if the human demonstrator takes one second to react to a feature in the force time history, then a processing delay of one second exists. To capture the causal relationship in the data, the sampling period must be equal to or less than the processing delay. When presenting the measured pairs of desired input-output training vectors to a static ANN, one must then correct for the processing time delay in order for the ANN to extract the desired command mapping relationship. See Section 5.6.2 for a more detailed explanation of the causal time delay.

To correct for processing time delay, the raw measured data set is time shifted by an amount equal to the processing delay of the controller in place during the collection of the data. For a computed accommodation control law (as is the case when collecting RISO or DISO data), the processing time delay is clearly constant and a multiple of (possibly equal to one) the servo rate for the controller.

For data collected by observing a human controller, the processing time delay may not be constant, as the alertness and attention may vary. In this case, the processing time delay cannot be corrected by time shifting. As an approximation, however, it may be possible to alleviate the effect of processing time delay in the training data samples if the variation of the human processing time delay is small as compared to the sampling rate for the collected data.

### *D.2 Time Delaying.*

Since it is possible that the trends in the captured force/velocity data are important to discerning the task, the ANN may be presented with several previous time input force and/or

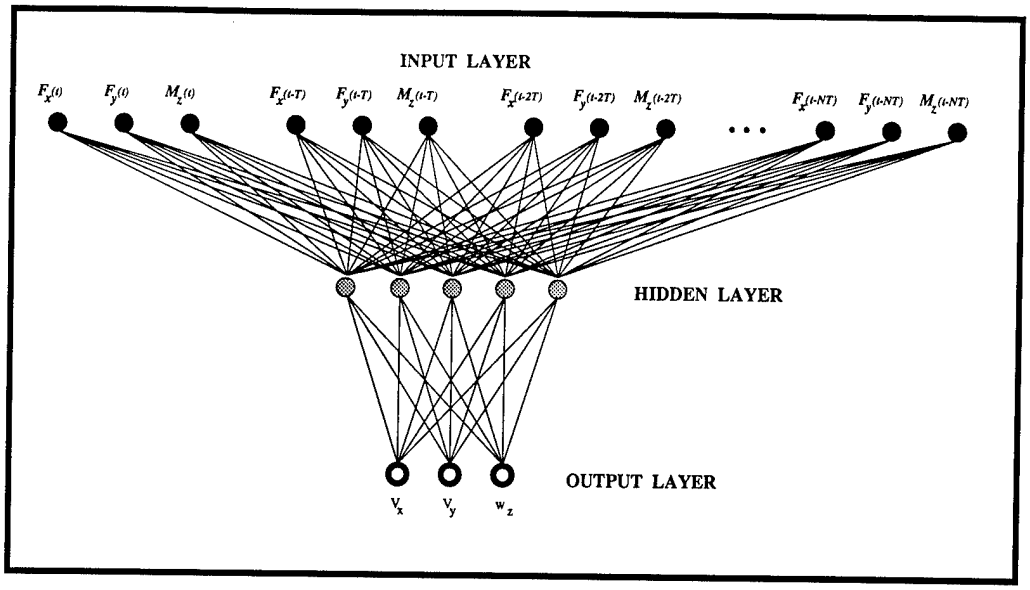



Figure D.1 Illustration of time-delayed force vectors used in the input feature of the ANN.

output velocity vectors to incorporate the context of time as input features. Such an ANN is called a time-delayed ANN (TDANN) [32, 52, 53]. In this case, the current components of force and some number, $N$, of previous forces are included in the input feature vector as shown in Figure D.1.

Other than appending the previous force vectors together for a larger input feature vector, the data vectors are not modified. Note that Figure D.1 illustrates only applying time-delayed forces to the input, while the input feature vector can also include previous velocity output vectors. In such a case, the ANN adopts a recurrent architecture which is discussed in [26].

### *D.3 Angle Features.*

The angle features option can be considered an extension of the magnitude normalization. If the raw vectors have been magnitude normalized, they no longer contain magnitude information, so the three components can be thought of as the endpoint coordinates of a unit-length vector in three-space. This unit-length vector can be described by two independent parameters which are the angles formed between the vector and the coordinate axes.

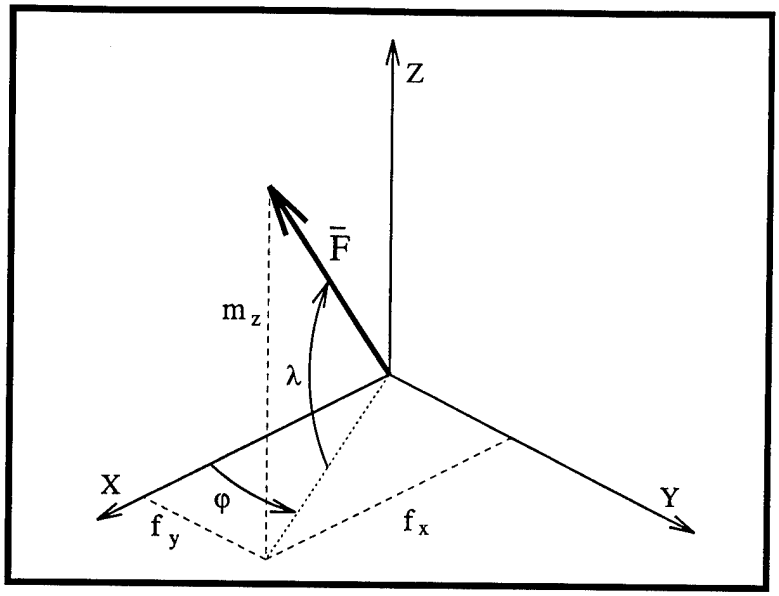



Figure D.2 Vector diagram for derivation of angle feature parameters $\varphi_f$ and $\lambda_f$ from normalized cartesian vector components.

If the raw vectors have not been normalized, the three-space vector will not be unit-length; the resulting angle features are unaffected by the normalization, so the results are the same either way. These desired angles are depicted in Figure D.2 as $\varphi_f$ and $\lambda_f$.

Given a vector $\vec{F}$ which has components $(f_x, f_y, m_z)$, a new vector, $\vec{F}_a$, composed of angle features $(\varphi_f, \lambda_f)$ can be derived from the following expressions:

$$\varphi_f = \text{atan2}\left(\frac{f_y}{f_x}\right) \tag{D.1}$$

$$\lambda_f = \text{atan}\left(\frac{m_z}{\sqrt{f_y^2 + f_x^2}}\right) \tag{D.2}$$

One should note that if the vector $\vec{F}$ has been normalized, then $\lambda_f$ is simply $\text{asin}(m_z)$. The range of $\varphi_f$ will be $\left(0 \leq \varphi_f < 2\pi\right)$, while the range of $\lambda_f$ will be $\left(\frac{-\pi}{2} \leq \lambda_f \leq \frac{\pi}{2}\right)$. To prepare data for training, the $\vec{V}^*$ must also be converted using Eqs (D.1) and (D.2) to get $(\varphi_v, \lambda_v)$.

When the ANN controller is implemented after being trained on angle features, its computed output, $\vec{V}$, is composed of $(\varphi_v, \lambda_v)$ and must be converted back to a cartesian vector, ${}^T\vec{V}_c$, having components $\left\{{}^T\dot{x}_p, {}^T\dot{y}_p, \dot{\theta}_p\right\}$. To do that conversion, the following relationships are

used:

$$ {}^{T}\dot{x}_p = \cos\varphi_v \cos\lambda_v \tag{D.3} $$

$$ {}^{T}\dot{y}_p = \sin\varphi_v \cos\lambda_v \tag{D.4} $$

$$ \dot{\theta}_p = \sin\lambda_v \tag{D.5} $$

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | March 1995 | *Doctoral Dissertation* |

**4. TITLE AND SUBTITLE**
Teaching Accommodation Task Skills: From Human Demonstration to Robot Control Via Artificial Neural Networks

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**
Paul V. Whalen, Capt, USAF

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Air Force Institute of Technology, WPAFB OH 45433-7765

**8. PERFORMING ORGANIZATION REPORT NUMBER**
AFIT/DS/AA/95-01

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for Public Release; Distribution Unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

The edge mating task was used to study the feasibility of using readily available force and velocity data collected during a human demonstration to train an artifical neural network (ANN) to control a PUMA robot for similar tasks. The planar edge mating task served as the basis for the investigation. A simple multilayered perceptron ANN was trained using backpropagation mapped contact forces to commanded velocities, all in the tool-frame for a configuration-independent solution. An accommodation matrix controller provided a performance baseline for simulation and on the PUMA manipulator. The same matrix was used to synthesize various forms of training data during an analysis of the function and vulnerabilities of the proposed control scheme. ANN controllers trained on these synthesized data were shown to easily complete the task in both simulation and on the PUMA. Human demonstration data collected using either the PUMA robot or a custom-built system were found to be poor examples of accommodation mappings for reasons that are discussed. Seven data processing algorithms were investigated independently and in combinations for their ability to improve the usefulness of the demonstration data. None were found to ensure controller success, although data mirroring improved controller performance significantly.

**14. SUBJECT TERMS**
robotics, artificial neural networks, skill acquisition, force control, accommodation, telerobotics, edge mating, artificial intelligence

**15. NUMBER OF PAGES**
281

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |