In **Journal of Intelligent and Robotic Systems: Theory and Applications.** *Special Issue on Computational Aspects of Robot Kinematics, Dynamics, and Control* Vol. 9, pp. 121-148, 1994.

Computational Considerations in the Implementation of Force Control Strategies

Richard Volpe* and Pradeep Khosla†

Abstract

This paper discusses computational and experimental details necessary for successfully implementing and evaluating a wide variety of force control strategies. First, a review of both explicit force and impedance control strategies is provided. Second, the basic computational requirements of these schemes are discussed, and the hardware and timing information for our implementation is provided. Third, computational problems such as noise filtering and sampling rates are explained and discussed in detail. Finally, a review of the experimental results obtained is provided. These results support the previous discussions by demonstrating the importance of fully considering the implementational details required for successful force control of robotic manipulators.

1 Introduction

It has generally been recognized that there is a class of tasks for which manipulators need to be force controlled [18, 38, 16]. Examples of such tasks include pushing, pulling, scraping, polishing, inserting, etc. Since these tasks inherently require not just the exertion but the control of forces, force sensing and force control algorithms must be employed. This paper addresses control architecture and computational methods we have employed to implement and experimentally compare a wide variety of force control methods. This testing has revealed the relative efficacy of each control strategy, as well as the computational considerations necessary for the algorithms which implement each strategy.

In this paper we first review the wide spectrum force control strategies that we have tested [29, 35]. Second, the computational requirements of each strategy will be presented. Third, we describe the architecture of the CMU DD Arm II system, and discuss the implementation considerations which limit the effectiveness of some schemes. Fourth, we present some experimental results from our implementations.

Previously proposed force control strategies are usually divided into two categories: explicit force control and impedance control [38, 7, 23]. The explicit force control strategies are typically

*Currently at The Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109. This work was completed while the author was a member of the Department of Physics, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213

†Department of Electrical and Computer Engineering, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213

some variation of PID control of exerted forces. The impedance control methodology controls the apparent dynamics of the manipulator, as experienced by the environment with which it is contact. We have previously shown through analysis and experimentation that impedance control is equivalent to proportional gain force control [33]. We have also demonstrated and explained the superiority of integral gain force control for force trajectory tracking [35]. Further, we have developed and demonstrated an impact controller that provides stable and bounceless impact with the environment [32].

The computational requirements of each of these strategies can be discussed in terms of the multiplies and additions required, or in terms of the experimentally determined cycle times for each algorithm on our system. To put this in perspective, we will describe the system and give these computational times. Among the most intensive operations are the force sensor interrupt servicing and the manipulator inverse dynamics computation. Some control schemes do not employ the dynamics computation and, therefore, have faster cycle times. Other operations requiring CPU bandwidth are joystick servicing, trajectory queue reading and data logging, manipulator kinematics and Jacobian calculations, tool and manipulator Jacobian multiplications, joint velocity computations, the operator interface, etc.

Five of these algorithmic operations inherently limit the robustness of the controllers. These are: the velocity signal computation, the force signal computation, the force derivative computation, the hybrid position/force control switching, and impact transient handling. Each of these will be extensively discussed, and experimental data will be used to demonstrate the limitations inherent in these computations.

Finally, we will present some experimental results obtained, and discuss how computational costs have effected these results.

2 Explicit Force Control Strategies

This section describes the forms of explicit force control that were implemented and experimentally evaluated [29]. Explicit force control strategies utilize direct evaluation of desired and measured forces. Two types of output are possible from this type of controller: forces or positions. In the former, forces are commanded directly, and then translated into manipulator joint torques. In the latter, position setpoints are given to an inner-loop position controller. We have previously shown that position-based methods may be recast as force-based methods [34]. Therefore, only the force-based methods are reviewed here.

Force-based explicit force control describes a method that compares the reference and measured force signals, processes them, and provides an actuation signal directly to the plant. The reference force may also be fedforward and added to the signal going to the plant. The plant is the arm / sensor / environment system, comprised of the CMU DD Arm II , a Lord force sensor, and several different environmental surfaces. It has previously been shown that the plant is best represented by a fourth order transfer function [4, 29, 31]. To control this plant some subset of PID control (i.e. P, I, PD, etc.) is usually chosen. We have extensively investigated these control methods through both analytical and experimental means [35, 34]. The specific forms of these controllers will be discussed next.

2.1 Strategies for Force-Based Explicit Force Control

This section presents force-based explicit force control strategies that have been implemented. From a computational perspective, all are approximately equal in complexity. The strategies presented here are generalizations of schemes previously proposed, as indicated below. In all cases, the joint torques commanded by these schemes are obtained through the transpose of the Jacobian, and gravity compensation is employed. The parameters f and $\dot{x}$ are Cartesian force and velocity. K is a gain for either proportional (subscript fp), integral (fi), or derivative (fd) force control. K_v is the velocity gain and provides active damping that is incorporated directly into the plant [34]. The subscripts c and m denote commanded and measured quantities. The variables s and a are Laplace domain complex numbers.

Proportional Control [20, 11, 1, 4, 5, 40, 36]

$$f = f_c + K_{fp}(f_c - f_m) - K_v \dot{x}_m \tag{1}$$

Integral Control [27, 41, 3, 28]

$$f = K_{fi} \int (f_c - f_m) dt - K_v \dot{x}_m \tag{2}$$

Proportional–Integral Control [23, 17, 5]

$$f = K_{fp}(f_c - f_m) + K_{fi} \int (f_c - f_m) dt - K_v \dot{x}_m \tag{3}$$

Proportional–Derivative Control [5, 39, 28]

$$\text{Unfiltered :} \quad f = f_c + K_{fp}(f_c - f_m) + K_{fd} \frac{d}{dt}(f_c - f_m) - K_v \dot{x}_m \tag{4}$$

$$\text{Filtered :} \quad F(s) = F_c(s) + [K_{fp} + K_{fd} s] \left[F_c(s) - \left(\frac{a}{s+a} \right) F_m(s) \right] - K_v s X_m(s) \tag{5}$$

Second Order Low Pass Filter Control [29, 34]

$$F(s) = \frac{K_{fp}}{s(s+a)} (F_c(s) - F_m(s)) + K_v s \, X_m(s) \tag{6}$$

3 Impedance Control Strategies

Unlike explicit force control, impedance control is designed to make the manipulator have a specific dynamic behavior. This section reviews impedance control with and without model-based dynamics compensation.

3.1 Model Based Control

Model based control involves the use of a dynamic model of the manipulator to compute the actuation torques [2, 12]. This Computed Torque strategy provides a way to compensate for the non-linearities of the manipulator dynamics. Thus, a linear control signal, u, will provide the desired joint accelerations in the manipulator:

$$\tau_A = D(\theta)u + h(\theta, \dot{\theta}) + g(\theta) - \Gamma \tag{7}$$

where τ_A is the actuator torque vector, D is the manipulator inertia matrix, h is the Coriolis and centripetal torque vector, g is the gravitational torque vector, and Γ is the reaction torque vector from environmental interactions. With this formulation the problem becomes one of choosing u.

For a Cartesian space position controller, the desired joint acceleration must be obtained from the desired Cartesian acceleration. First, it is known that $\dot{x} = J\dot{\theta}$, where J is the manipulator Jacobian. Taking the derivative of this equation and solving for the angular acceleration gives the control signal:

$$u = \ddot{\theta} = J^{-1}\left(\ddot{x} - \dot{J}\dot{\theta}\right) \tag{8}$$

It is still necessary to define the desired Cartesian acceleration. One choice is the acceleration due to gravity: $\ddot{x}_g = [0, 0, -9.8m/s^2, 0, 0, 0]$. Considering the static, free-space situation ($\tau = h = \Gamma = 0$), a gravity compensation vector of joint torques may be computed:

$$g = -DJ^{-1}\ddot{x}_g \tag{9}$$

The simple form of $\ddot{x}_g$ allows for streamlining of the computation of this vector.

Another way to obtain the Cartesian acceleration is to specify the desired behavior of the manipulator by a second order impedance relation [7]:

$$M\ddot{x} - C\Delta\dot{x} - K\Delta x = f \tag{10}$$

The impedance parameters, M, C, and K must be chosen by the user. The variable x and its derivatives are obtained from the transformation of the corresponding angular values:

$$\begin{aligned} \Delta x &= x_c - x_m &&= x_c - \mathcal{F}(\theta_m) && (11) \\ \Delta\dot{x} &= \dot{x}_c - \dot{x}_m &&= \dot{x}_c - J\dot{\theta}_m && (12) \end{aligned}$$

where $\mathcal{F}$ represents the forward kinematics, and c and m denote the commanded and measured quantities. Finally, the force, f, corresponds to the physical force exerted on the manipulator: $J^T f = \Gamma$. Since the force experienced is the negative of the force imparted, $f = -f_m$. The measured force may then be substituted into Equation (10) for the desired behavior and Equation (7) for the arm model.

Summarizing,

$$\begin{aligned} \tau &= D(\theta)u + h(\theta, \dot{\theta}) + g(\theta) && (13) \\ \tau_A &= \tau + J^T f_m && (14) \\ u &= J^{-1}\left(\ddot{x}_u - \dot{J}\dot{\theta}_m\right) && (15) \\ \ddot{x}_u &= M^{-1}\left[\{C\left(\dot{x}_c - \dot{x}_m\right) + K\left(x_c - x_m\right)\} - f_m\right] && (16) \\ x_m &= \mathcal{F}(\theta_m) && (17) \\ \dot{x}_m &= J\dot{\theta}_m && (18) \\ J &= J(\theta_m) && (19) \end{aligned}$$

The following two sections will illustrate the use of these equations in impedance control. The next section shows the use of the full dynamics implementation, while the last section shows the use of a steady state simplification.

3.2 Second Order Impedance Control

This section presents second order impedance control with dynamics compensation [7]. From Equations (15) and (16) in (13) and (14), we have:

$$\tau \;=\; DJ^{-1}\left(M^{-1}\left(C\Delta\dot{x} + K\Delta x - f_m\right) \;-\; \dot{J}\dot{\theta}\right) \;+\; h \;+\; g \;+\; J^T f_m \tag{20}$$

This is the best form to use for the real time computation in a second order impedance controller, since it utilizes the efficiency of the Computed Torque technique directly. However, for the purposes of discussion it is useful to utilize the relation $D(\theta) = J^T\Lambda(x)J$, where Λ is the Cartesian space manipulator inertia at the end effector of the arm [9]. This yields:

$$\tau \;=\; J^T\Lambda M^{-1}\left(C\Delta\dot{x} + K\Delta x - f_m\right) \;-\; J^T\Lambda\dot{J}\dot{\theta} \;+\; h \;+\; g \;+\; J^T f_m \tag{21}$$

Note that second order impedance control utilizes force feedback in two ways. First, this feedback is used in the physical model of the arm dynamics, Equations (13) and (14). Second, the feedback is used in the impedance relation, Equation (16). While Equations (13) and (14) effectively linearize the arm, Equation (16) modifies the Impedance Control signal to compensate for the experienced force.

3.3 Second Order Impedance Control with a Steady State Approximation

For the steady state case, all velocities in the dynamics equations are assumed zero and the inertia is assumed constant [8, 6]. Therefore, Equation (21) gives:

$$\tau \;=\; J^T\Lambda M^{-1}\left[(C\Delta\dot{x} + K\Delta x) - f_m\right] \;+\; J^T f_m + g \tag{22}$$

Note that Equation (13) is not used to calculate the inverse dynamics, and there is no use of the inverse of the Jacobian. Rather, the matrices ΛM^{-1}, $\Lambda M^{-1}C$, and $\Lambda M^{-1}K$ are chosen, usually as diagonal gain matrices.

While not as accurate as the full dynamics representation, this approach has one major advantage: there is no inversion of the Jacobian, which is computationally intensive and can become singular. Therefore, while this scheme is faster and more robust than the previous, it is also less accurate, especially for situations in which the velocity terms cannot be neglected.

4 Position Control

When the measured force is zero for the previous impedance control algorithms, they reduce to position controllers. However, there are other methods of position control. It is important to review these position control methods, because when the manipulator interacts with an environment some degrees of freedom will not be constrained and must remain position controlled. This section reviews the most common position control methods employed.

4.1 Joint Space Control

Joint space PID position control has the following form:

$$\tau \;=\; K_\theta(\theta_c - \theta_m) \;+\; K_\omega(\dot{\theta}_c - \dot{\theta}_m) \;+\; K_i\int(\theta_c - \theta_m)dt \tag{23}$$

While this strategy has been proposed for use in conjunction with force control [19], it is not truly compatible with Cartesian Hybrid Position/Force Control [16], on which all of the previously reviewed force control strategies rely. Further, this form of control neglects dynamic coupling of joint motions. The results of an implementation of this scheme are presented later, mainly for comparison purposes.

The Computed Torque, previously introduced in Section 3.1, was proposed to account for dynamic coupling. It was the first scheme to use Equation (13) to calculate the nonlinear dynamics of a manipulator [15], using a control law expressed in joint space as:

$$u \;=\; \ddot{\theta}_c \;+\; K_\omega(\dot{\theta}_c - \dot{\theta}_m) \;+\; K_\theta(\theta_c - \theta_m) \tag{24}$$

This joint space form of Computed Torque was not implemented in our study, and is presented here for completeness. Instead, a Cartesian Space formulation was employed, as will be reviewed in Section 4.3.

4.2 First Order Impedance Control

First order impedance control is essentially PD position control in Cartesian space. Unlike Equation (10), there is no acceleration term: $f = -C\Delta\dot{x} - K\Delta x$. Therefore, the commanded torque can be obtained directly through the transpose of the Jacobian [37, 24, 38]:

$$\tau \;=\; J^T(K_v\Delta\dot{x} + K_p\Delta x) \;+\; g \tag{25}$$

This formulation is equivalent to second order impedance control without dynamics compensation, Equation (22), with $f_m = 0$. We have previously discussed the need for *either* force or acceleration measurements to implement second order impedance control [34].

This controller suffers from large tracking error during free space motion, but works well in static situations. Further, it is much less computationally intensive than other Cartesian schemes. For this reason it was used extensively to control the unconstrained degrees of freedom during our force control experiments.

4.3 Resolved Acceleration Control

Resolved Acceleration Control was the first scheme to use Equation (15) to resolve the desired Cartesian acceleration into joint space, for use in the Computed Torque scheme [14]. Operational Space Control [9] provides an equivalent result, but represents the dynamics equations in Cartesian Space [13, 29]. The Cartesian acceleration is specified as:

$$\ddot{x}_u \;=\; \ddot{x}_c \;+\; K_v(\dot{x}_c - \dot{x}_m) \;+\; K_p(x_c - x_m) \tag{26}$$

It can be seen that this equation corresponds to Equation (16) with $\ddot{x}_c = -M^{-1}f_m$. This term seems strange at first, since Resolved Acceleration Control is not designed to interact with the environment ($f_m = 0$). However, the feedforward acceleration can be thought of as the result of an *artificial force*. This force is exerted by the environment only in computer model of the world, and is calculated from the gradient of a modelled *artificial potential field*. One use of artificial forces is for obstacle avoidance strategies in which objects to be avoided are surrounded by repulsive potentials [10, 30].

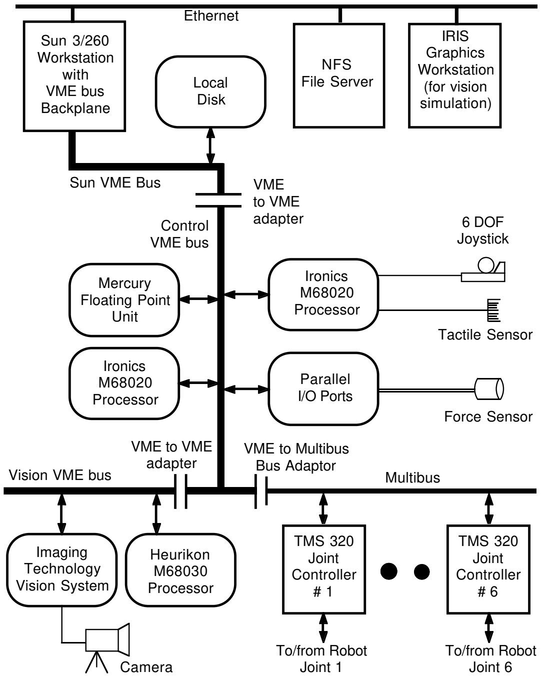


Figure 1: System architecture for the CMU DD Arm II.

5 Support Function Computations

To implement the above strategies, a basic set of computational modules are needed. Recently, we have rigorously structured this module relationship to readily enable reconfiguration of the control software [26]. Here we will present the modules and the computational costs in terms of execution time. To ground these speeds in common units, we present the processor times required for simple operations. Also, we discuss the interprocessor communication sequence, as well as the interrupt servicing overhead. First, however, we will describe the computational hardware for the system.

5.1 Computation Environment

All of the previously discuss algorithms were implemented on the CMU DD Arm II system which consists of over ten microprocessors, a force sensor controller, a tactile sensor, a joystick, a camera, joint resolver hardware, and six joint motor amplifiers. The microprocessor boards are of main concern in this discussion. They are located in four backplanes (three VME and one multibus) as shown in Figure 1. The Sun's VME backplane contains a Sun 3/260 Host computer and one end of a VME/VME bus repeater. The second VME backplane contains a Mercury MC3200 floating point board (with a Weitex 13132 processor), an Ironics M68020 processor (with 2 serial ports), a parallel IO board, two ends of VME/VME bus repeaters, and one end of a VME/Multibus repeater. The third VME backplane (not utilized for the force control implementations) contains an Imaging Technology vision system, a Heurikon M68030 processor, and one end of a VME/VME repeater. The Multibus contains six Texas Instruments TMS32010 joint controllers (320s), one TMS32010 master controller, and one end of the VME/Multibus repeater.

The real-time control of the CMU DD Arm II is performed by the M68020 processor residing on a VME backplane that is connected to the Sun through a repeater. This processor operates under the *Chimera II* real-time operating system [25]. A real-time control and user interface program runs on the M68020. The interface utilizes Chimera II system calls to communicate with the Sun, the Mercury floating point processor, the force sensor, the tactile sensor, the joystick, and the vision system. It also downloads control code to the Mercury and the 320s, as well as controlling execution rates on these boards. The Mercury floating point processor can be accessed by the user through the Sun for code development, or by the interface program for real-time control code execution. The six 320 joint controllers calculate the three phase torque values needed by the motors, as well as obtaining the joint position values from the joint resolvers.

5.2 Basic Computations

Since the Mercury MC3200 serves as the main computational engine for the control algorithms, this section presents the times needed for execution of basic operations and routines running on this board. All software is written in C, and compiled with the *Green Hills*^{TM} compiler provided by Mercury. The computation times for basic operations are shown in Figure 2. At first glance it appears that the MC3200 is only capable of 2 Mflops. However, pipelining of multiple floating point math commands can achieve speeds of up to 7 Mflops for optimized C-code. (The advertised claim of 20 Mflops is possible only for special operations, programmable in Assembly Language.) Note that memory access for an '=' operation takes almost as much time as an add or multiply. Further, because subscripted access of matrices or vectors requires offset calculations, it should be avoided. In the case where the subscripts are global integers, the costs go up drastically. (Access of matrix or vector elements by 'hardcoded' offsets is fine however, since the addresses are pre-computed by the compiler.) Finally, trigonometric and type-casting operations are also costly and should be avoided.

5.3 Basic Algorithms

Our experience has shown that all control code should utilize as much in-line expansion as possible with as few global variable accesses as possible. One other rule of thumb has been to not use more than four arguments to a function call, since only four are passed in registers. Using these programming rules, we have developed software on the MC3200 for all of the outlined force control strategies. The computation times of these algorithms are shown in Figure 3. Some discussion of these computation times is useful.

- The routine sgtoft() has five functions: subtraction of a bias vector (obtained by calibration) from the eight measured strain gauge values, multiplication of the result by a calibration matrix to obtain six forces/torques, conversion of units from English to SI, calculation and subtraction of the geometrically dependent torque caused by the end effector, and transformation of the forces/torques into the end effector frame. Some of these operations, such as the calibration matrix multiply and units conversion, were combined to save time.

- The invdynamics() routine was the most computational costly routine implemented. It has previously been shown that this scheme can be optimized for the CMU DD Arm II such that there are 303 multiplications and 226 additions [12]. Using the previously determined time of 0.5 μs for these 529 operations, it might be expected that the inverse dynamics would take a maximum of 215 μs. However, the structure of the inverse dynamics computation requires that intermediate results be saved during the forward recursion, and accessed during the backward recursion. Since the number of intermediate variables is larger than the registers available in the MC3200, these saves require memory accesses. A close look at the code shows

Operation	Time (μs)	Variable Types
`for(i=0;i<N;i++)`	0.80	*per iteration*
`a = b;`	0.41	`float a, b;`
`i = j;`	0.53	`int i,j;`
`a + b;`	0.50	`float a,b;`
`a - b;`	0.50	`float a,b;`
`a * b;`	0.50	`float a,b;`
`a / b;`	2.30	`float a,b;`
`a * b; e + f;`	0.82	*local floats, parallel processing capability*
`c[j];`	0.41	`float c[N]; extern int j;`
`c[j];`	0.20	`float c[N]; int j;`
`d[j][k];`	0.61	`float d[N][M]; extern int j,k;`
`d[j][k];`	0.53	`float d[N][M]; extern int j; int k;`
`d[j][k];`	0.51	`float d[M][N]; extern int k; int j;`
`d[j][k];`	0.50	`float d[M][N]; int j,k;`
`c[j] = d[j][k] * g[k];`	30.00	*global variables;* $j, k < 3$
`c[j] = d[j][k] * g[k];`	11.80	*math library call*
`c[j] = d[j][k] * g[k];`	7.60	*subroutine with complete expansion*
`c[j] = d[j][k] * g[k];`	6.60	*complete expansion, local variables*
`sin(a);`	5.30	float a;
`cos(a);`	5.50	float a;
`a = (float)i;`	5.20	float a; int i;
`i = (int)a;`	5.60	float a; int i;

Figure 2: Computation times for basic operations the MC3200 processor.

Routine	(μs)	Description	Equations
`sincos()`	72	Obtain sine and cosine for all 6 arm joints.	
`sgtoft()`	60	Change 8 strain gauge values to 6 forces/torques.	
`invdynamics`	714	Newton-Euler customized inverse dynamics.	13
`gravcomp()`	130	Gravity compensation. Special case of above.	9
`fkj()`	43	Forward kinematics and Jacobian.	17, 19
`fkjij()`	151	Forward kinematics, Jacobian, and inverse Jacobian.	17, 19
`pd()`	25	Joint space PD control.	23
`pid()`	35	Joint space PID control.	23
`pdg()`	160	Joint space PD control with gravity compensation.	23, 9
`x()`	236	Cartesian PD control. Includes fkj(), sincos().	25
`xg()`	382	Addition of gravcomp().	25, 9
`xfg()`	578	Addition of hybrid force control and sgtoft().	25, 9, 14, 1-6
`xfct()`	1261	Replacement of gravcomp() with invdynamics().	26, 13-15, 1-6
`imp()`	550	Steady-state impedance control. Similar to xfg()	22, 9
`impct()`	1200	Impedance control with dynamics. Similar to xfct()	20
`from320s()`	63	Obtain joint angles from joint controllers.	
`to320s()`	48	Check torque limits and send out commanded values.	

Figure 3: Computation times for basic algorithms on the MC3200 processor.

over 1000 such accesses, requiring close to 500 μs. This accounts for the bulk of the execution time.

- The fkjij() routine employs Paul's method for forward kinematics and Jacobian computation [21]. The inversion of the Jacobian was done with the Strassen inversion technique [22], which utilizes the block structure of the Jacobian due to the spherical wrist of the CMU DD Arm II . The tool Jacobian is computed and inverted separately due to its simple structure.
- The times for the joint space algorithms are included for completeness and comparison purposes. As mentioned earlier, they were not used during force control.
- The impedance control algorithms were not timed separately since they are so similar in computational structure to their explicit force control counterparts.

While the Mercury processor executes the above control algorithms, it does not decide the execution starting time. Instead, it acts as a slave to the M68020 which triggers it. This control process, as well as all others, are discussed next.

5.4 Required Processes

The control system implemented utilizes at most four processes while executing force control algorithms. These are the main control loop, the user interface, the joystick reader, and the joystick interpolator. There is also an interrupt handler for the force sensor data.

The main control loop is the core of the system and assumes highest possible priority while executing. The philosophy followed is that the main control loop must execute periodically, and all other processes must make due with the remaining bandwidth of the CPU. If there is not enough time left over, the processor is saturated, and control is impossible at this rate.

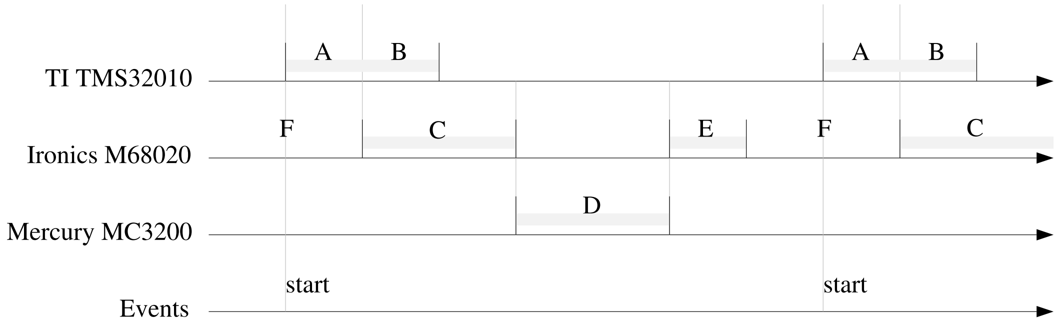
The functions of the main loop process are: read all measured and reference values and transfer them to the MC3200, trigger the MC3200, wait, transfer out the computational results from the MC3200, and log data. The measured values consist of joint angles obtained from the 320 boards, and strain gauge force values place in memory by an interrupt handler. The reference values consist of positions, velocities, and forces that are entered by the user, obtained from the joystick, or read from a trajectory cache that has been read from a file. The logged data consists of the measured and reference values, as well as the computed values from the MC3200. Logging can absorb a significant amount of time since it requires extensive data transfer both from the MC3200 and within the Ironics board. (Logging does not rely on real-time disk access, however. All data is stored in memory, and uploaded after an experiment is completed.)

The user interface process, the only aperiodic process, is also extremely important. This process parses user commands, downloads the MC3200 and TMS320 boards, permits selection of control modes and configures the system appropriately, permits the selection of reference and gain values, creates separate processes for the reading of joystick values, etc. The two joystick processes are necessary to read the joystick at its maximum rate of about 30 Hz, and to provide servo commands by interpolation at a rate of 150 Hz. Without the interpolation, the manipulator motion is not smooth.

5.5 Interprocessor Communication and Timing

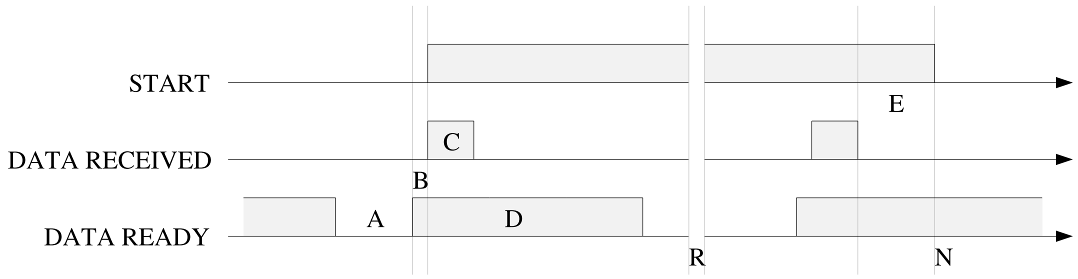
To illustrate the role of the various processors and processes, it is valuable to trace the transfer of data through the system during one control cycle. A timing diagram for interprocessor communication in the system is shown in Figure 4. It should be noted that the size of segment D is dependent on the control algorithm chosen to run on the MC3200. Also, the size of segment F is dependent on the selected control rate. During F, the user interface and joystick processes may execute. Interrupt servicing, which takes place throughout, is not shown.

The timing for the Lord Force sensor communication with the 68020 interrupt handler is shown in Figures 5 and 6. While the interrupt handler is only executing on the 68020 for about 20 μs for each piece of data, there is 60 μs of overhead. Since the data is arriving ($8 \times 416 =$) 3328 times a second, force sensor interrupt servicing steals at least 25% of the processor bandwidth. This is a significant portion and has forced us to decrease our operation rate for force control algorithms. For instance, First Order Impedance Control can run at 400 Hz without force data, but the control rate must be reduced to 300 Hz if it is used in conjunction with a force controller in a Hybrid Control framework. Considering that this control rate reduction results from only one sensor, it is obvious that there is an intrinsic likelihood of data overload in sensor based control systems. We have begun work on distributed sensor processing and control to address this issue [26].



A - get joint angles from A/D boards, add offsets
B - calculate three phase torques, send to D/A boards
C - transfer joint angles to Mercury board, perform logging
D - run control law
E - transfer torque values to 320s, perform logging
F - run other processes, service interrupts

Figure 4: A timing diagram showing the sequence of interprocessor communication in the CMU DD Arm II system. The segments are not to scale.



A - 25 to 150 microseconds (interrupt triggered)
B - 5 microseconds
C - 15 microseconds
D - 75 microseconds
E - 25 microseconds
R - repeat A,B,C,D six times
N - new cycle triggered by falling edge

Figure 5: A timing diagram showing the communication details between the Lord force sensor and the interrupt handler on the Ironics 68020. The START signal indicates the beginning of the transfer cycle for eight strain gauge values. The DATA signals perform handshaking between the parallel port interrupt servicer on the 69020, and the Lord force sensor hardware.

6 Algorithm Implementation Considerations

In addition to the above system issues, there are many problems and issues associated with the implementation of the previously reviewed algorithms. Some are only minor annoyances, while others can effect the stability or range of operation of a particular controller. Among the issues discussed in this section are velocity signal calculation, force signal noise and filtering, the force signal derivative, hybrid control switching, and impact transient handling.

Much of the following discussion utilizes graphed data to illustrate and validate the topic. In

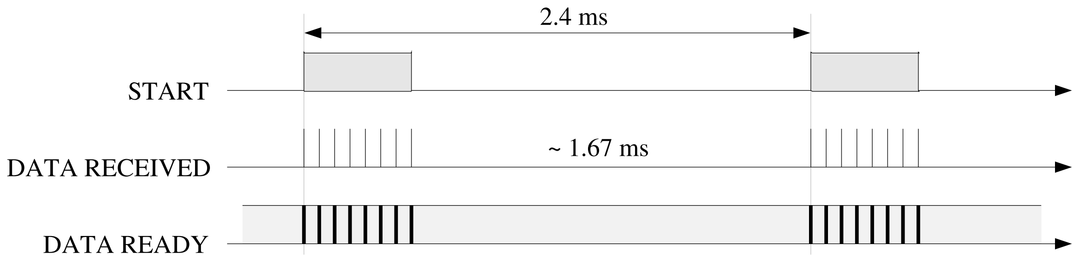


Figure 6: A timing diagram showing a larger time slice than in Figure 5. The 416 Hz (2.4 ms) cycle of the force sensor controller hardware is apparent. Also the clumping of the incoming eight strain gauge values is evident. Each of the spikes on the DATA RECEIVED line correspond to letter C in Figure 5. The dark bands on the DATA READY line correspond to letter A.

the legends of the graphs, the reference value of the applied force is called 'RefForc'; the measured value of the experienced force is called 'MezForc_wd'; the filtered value of this measured force is called 'Filter_Forc'; the derivative of this filtered force is called 'Fdot'; the measured value of the Cartesian velocity is called 'MezXVel_wd'; the measured value of the end effector position is call 'MezP'; and the hybrid control selection parameter is called 'SHybrid' [23]. All of these variables are vectors and the indices follow the conventions of the C computer language.

6.1 The Velocity Signal

The angular velocity signal for the joints of the CMU DD Arm II is obtained by differencing and averaging the angular position signal. The position signal is a 16 bit absolute position value obtained from pancake resolvers located at each joint. Every control cycle, the position is obtained and placed in a stack. A velocity signal averaged over the past n control cycles can be obtained by simple differencing of the current position with the one n cycles before it:

$$\begin{aligned} v_{avg} &= \frac{1}{n}\left[v(t) + v(t-T) + \cdots + v(t-nT)\right] \\ &= \frac{1}{n}\left[\frac{p(t) - p(t-nT)}{T} + \frac{p(t-T) - p(t-2T)}{T} + \cdots + \frac{p(t-(n-1)T) - p(t-nT)}{T}\right] \\ &= \frac{1}{nT}\left[p(t) - p(t-nT)\right] \end{aligned} \tag{27}$$

Good results are obtained for the CMU DD Arm II with $3 \leq n \leq 10$. The lower number provides a velocity signal with less lag and more noise, and the higher number just the opposite. For free space motion with the CMU DD Arm II , the natural frequency of the system is determined by the stiffness provided by the position gain. This frequency is usually low enough that the velocity signal lag is not significant. However, when the arm is in contact with the environment, the natural frequency of the system is largely determined by the environmental stiffness. This frequency is much higher than in the free space motion case. Therefore, the velocity signal lag can become a major portion of the oscillation cycle. As the delay approaches 90° the velocity signal will be in phase with the position signal. In this case, the velocity gain will not damp, but rather add to the already large stiffness of the system, driving it toward instability.

For the tests conducted in contact with the environment, a velocity averaging factor of $n = 3$ was used for joints 4, 5, and 6. A factor of $n = 5$ was used for joints 1, 2, and 3. For tests involving free space motion, the natural frequency is smaller and a factor of $n = 10$ for joints 1, 2, and 3 is usually used. The value $n = 3$ for the last three joints tends to be sufficient at all times.

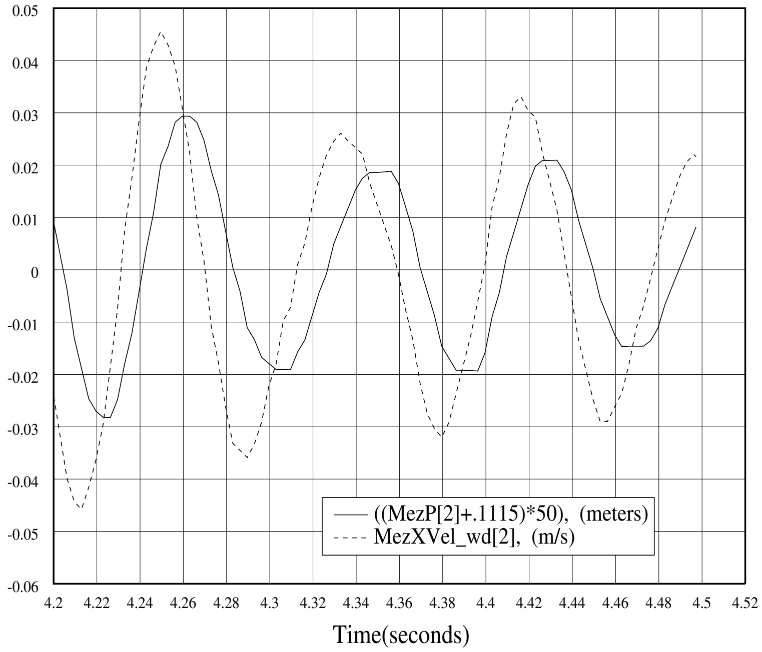


Figure 7: Velocity phase lag due to averaging. It can be seen that the velocity signal lags its ideal value by about 0.16s, or 45 degrees.

Since the Cartesian velocity signal contains components from all of the joint signals, the delay will be between three and five cycles. For the control rate of 300 Hz, the delay is between 0.01 and 0.016 seconds. Figure 7 shows the velocity and position signals during proportional gain explicit force control ($K_{fp} = 0.75$), after a step input. The delay of the velocity signal is about 0.01 seconds, or a 45° phase lag for the 12 Hz oscillation. This also explains why an averaging factor of $n = 10$ is unacceptable. This delay would put the velocity signal in phase with the position signal.

Note that active damping when the manipulator is in contact with the environment must be used with caution. The time delay from the velocity calculation/filtering is always present. If this delay is a significant part of the natural frequency of the system, then the velocity signal will act as a position signal and add to instability. Further, stiffer environments have higher oscillation frequencies, making the velocity signal least reliable when it would be most useful. Therefore, the damping intrinsic to impedance control, and sometimes used in explicit force control, is always suspect.

Finally, it is worth mentioning that these problems with delay only apply to active damping. Passive damping, as supplied by some soft sensors or end effector covers, will provide damping without time delay [39, 1]. These devices also lower the natural frequency of the system, making active damping possible.

6.2 The Force Signal

A Lord 15-50 force sensor was used in all of the experiments. In its factory configuration, it supplies eight strain gauge values at 416 Hz, as shown in Figure 6. However, the controllers used often ran at only 300 Hz, as previously discussed. Since the Lord sensor controller has its own internal clock, there is no way to easily change the update rate. Alternatively, individual request for data can be made, but only with a maximum rate of 250 Hz, due to clock skew. Therefore, we chose to receive the data at the faster rate of 416 Hz and ignore one of every four sets. This has the added effect

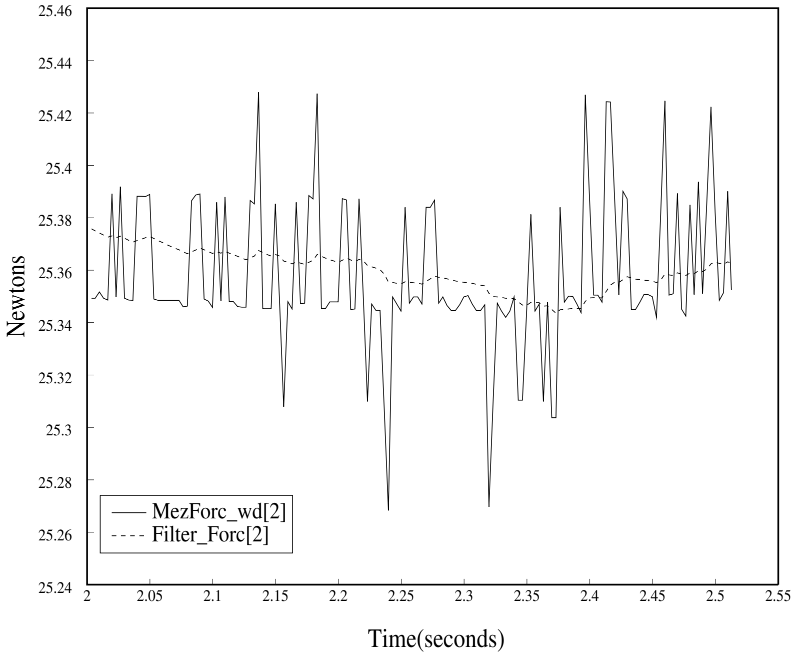


Figure 8: Filtered and unfiltered force signals.

that each data set could be as old as one 416 Hz cycle, or 2.4 ms. This asynchronous sampling has no appreciable effect on the stability of the controllers since force oscillations were an order of magnitude slower than the control rates and sampling time.

What does drastically effect control stability is the noise in the resultant force signal. As shown in Figure 8, this can be substantial. Filtering of the force signal is partially effective, but introduces lag as can be seen in Figure 9, where the measured force signal is a solid line and the filtered force signal is a short dash line. This effect is very detrimental for PD force control [35], as will be reviewed in the Section 6.3.

The force signal noise has several contributors, discussed below: intrinsic noise, kinematic fluctuations, kinematic inaccuracies, and inertial effects. All of these factors contribute to a noise amplitude of ~ 0.1 Newtons. This is an order of magnitude above the sensor resolution.

Intrinsic Noise This is present in the analog and digital electronics of the sensor system as well as the joint position resolvers. The joint position measurement noise contributes because of the need for transformation of the measured force to the control frame. Considering that the CMU DD Arm II has 16-bit absolute positioning resolvers, fluctuation in the last bit typically causes angular errors on the order of $2(2\pi/2^{16}) = 96\mu$radians. The additive effect of the six joints will then cause a worst case error of 0.57 milliradians in orientation. Multiplying this by the 1.25 meter reach of the arm, gives a position error of 0.71 mm.

Kinematic Fluctuations The kinematics of the arm are based on the assumption that the links are completely rigid. However, bending or oscillations in the arm structure lead to erroneous calculations of the sensor frame position and orientation, and therefore the measured force. This is especially true when there are forces exerted on the arm, such as during impact and force control.

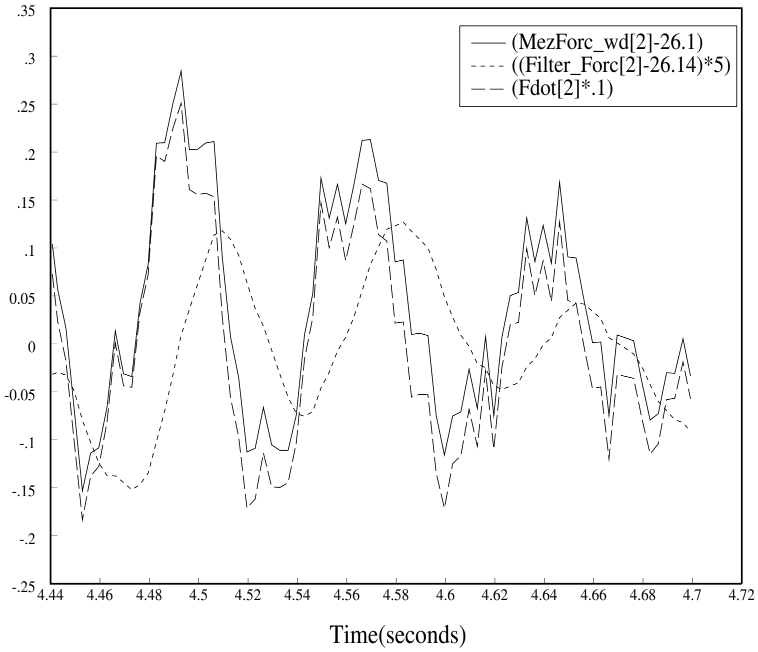


Figure 9: The lag of the filtered force causes the force derivative to be in phase with the measured force.

End Effector Weight Compensation Since the orientation of the end effector with respect to the direction of gravity (down) changes, the torque imparted on the wrist sensor changes as the orientation changes. Figure 10 shows the geometry of this problem. Assuming that the end effector consists of all material effectively beyond the strain gauge sensing elements, it has an effective mass, m_e, and length, l_e. If $\hat{a}$ is the approach vector, we have:

$$|\tau| = l_e m_e g \sin\theta \tag{28}$$

$$\tau_x = \tau \sin\phi \tag{29}$$

$$\tau_y = -\tau \cos\phi \tag{30}$$

where

$$\sin\theta = \sqrt{a_x^2 + a_y^2} \tag{31}$$

$$\sin\phi = \sqrt{a_x^2 + a_y^2} \tag{32}$$

$$\cos\phi = \frac{a_x}{\sqrt{a_x^2 + a_y^2}} \tag{33}$$

For the end effector we utilized during force control experiments, $l_e = 4.1$cm and $m_e = 0.57$kg. This yielded a gravitational bias force of 5.6 N and a maximum bias torque of 0.22 N $\cdot$ m.

Since compensation for the weight of the end effector is position dependent, it is subject to position measurement noise and kinematic fluctuations. Further, the end effector weight compensation is subject to kinematic inaccuracies which can cause the calculated direction of 'down' to be different from the direction of the measured gravitational force. For instance, a 1° error in the D-H parameter, α_1, was detected in this way by noting a discrepancy in the direction of the measured force throughout the workspace of the CMU DD Arm II .

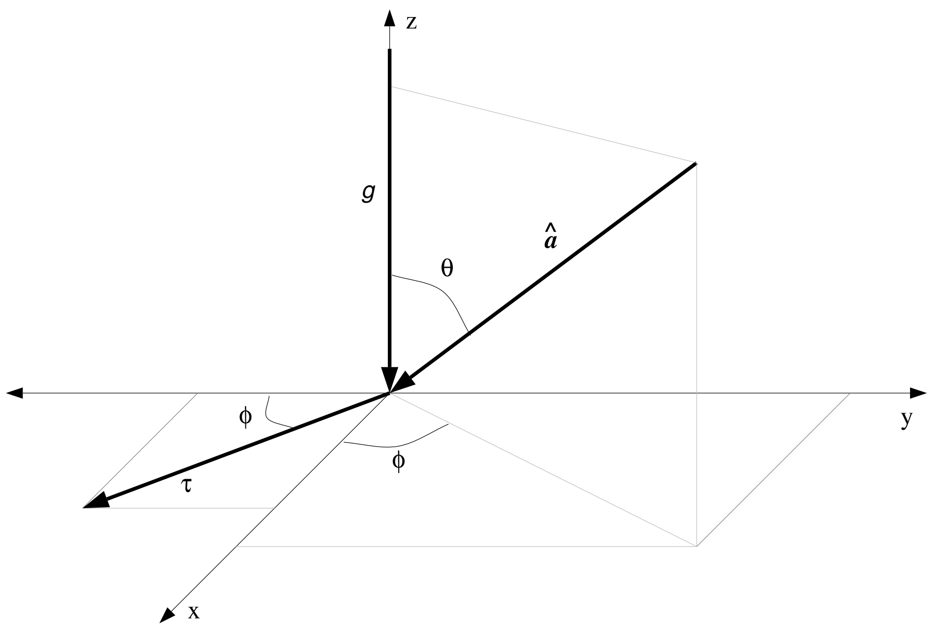


Figure 10: The geometry of the end effector bias torque (τ) as a function of the approach vector ($\hat{a}$) and gravity (g) in the world frame.

Inertial Effects Inertia causes the measured force not to equal the applied force if the sensor is accelerating. Since most environments are stationary, zero acceleration usually implies that the arm has zero velocity as well.

Many manipulators (including the CMU DD Arm II) are capable of rapid acceleration, and the inertial forces can be considerable. We have observed the inertia of the end effector causing problems with both impedance and explicit force control schemes. For impedance control, the manipulator drifts, since it is attempting to apply an impedance to an external force, when one is not actually present. For explicit force control, the hybrid controller will switch from position to force control in the direction of the inertial force.

6.3 The Derivative of the Measured Force

The previous section described the noise that is present in the force signal. This noise makes it essentially impossible to use PD force control. Even with lowpass filtering, the system was unstable for appreciable derivative gain values. Figure 11 shows the response of the system (solid), as well as the reference force (short dash), and filtered force (long dash), for $K_{fp} = 0.5$, $K_{fd} = 0.01$, $K_v = 10$, and $a = 10$ in Equation (5). The results are not much better than for proportional gain alone [35]. As will be described below, improvements in the performance of this controller can not be made by varying the gains given here.

First, increasing the derivative gain does not improve the response of the system because the amplified low frequency noise can still drive the system unstable. While Figure 11 seems to show a fairly smooth filtered force signal, a close-up view of the same data has already been shown in Figure 8. Much of the noise has been removed, but with a large enough gain the noise will dominate. Making the cut-off frequency of the filter lower ($a < 10$) will eliminate this noise, but it introduces a more serious problem of lag.

Figure 12 shows that the calculated derivative (solid curve) appears accurate. (The dotted curve is the measured force.) However, it is apparent from this figure and Figure 11 that there is lag

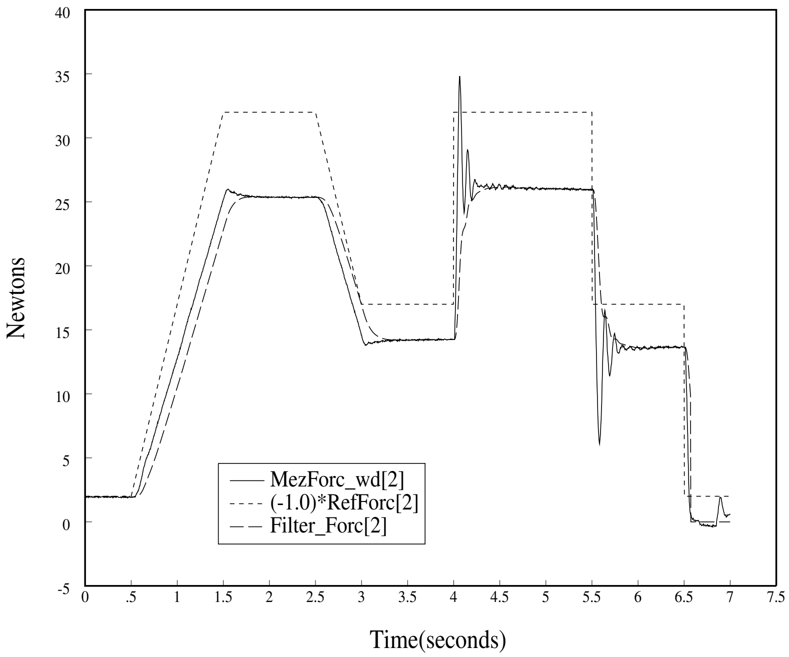


Figure 11: Experimental data from PD control with $K_{fp} = 0.5$ and $K_{fd} = 0.01$.

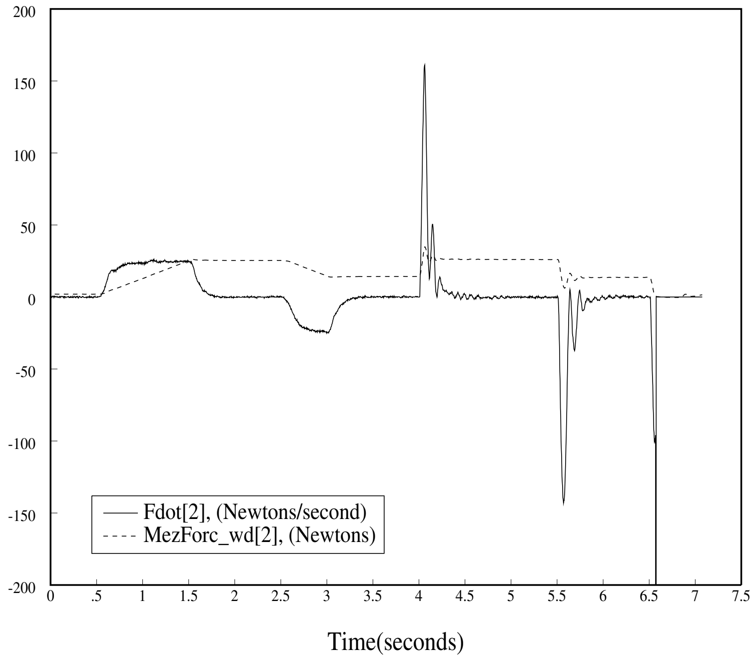


Figure 12: Calculated force derivative and measured force signal used in PD control.

introduced by the filtering process. This lag becomes extremely important when it is a significant portion of the period of oscillation of the system. Figure 9 shows the original force signal (solid), the filtered force signal (short dash), and the derivative of the filtered signal (long dash). For this oscillation frequency, the filtering process causes the filtered force to lag the measured force by one quarter cycle. This makes the force signal 180° out of phase with the ideal derivative signal. Thus, the proportional gain acts as a destabilizing negative derivative gain. Further, the derivative of the filtered signal leads it by one quarter cycle. Thus, the derivative is in phase with the originally measured force and the derivative gain acts as a proportional gain. Increasing the derivative gain causes greater oscillations exactly when the effective damping is being reduced by the proportional gain. This obviously will cause the system to go unstable.

It can be concluded from this discussion than the filter pole should be significantly larger than the natural frequency of the system, However, it also must be small enough to effectively filter the noise of the force sensor. These two criteria could not be met with our system. To be fair, most systems will never meet this criteria. Force controlled systems are most challenged by stiff environments that have high natural frequencies. It is unlikely that a sensor can be built that has noise only at frequencies much greater than the natural frequencies of these environments.

One solution, however, is to use a soft force sensor or compliant covering on the sensor. The compliance acts as a lowpass filter with no time delay. In this way, the derivative of the force signal may be used under the condition that the time necessary to calculate it is not significant. In this case, without a noisy force signal, simple differencing of the current and most recent force samples will usually suffice. Thus, all that is required is that the force sampling frequency is not of the same order of magnitude as the natural frequency of the system. Successful PD force control with a soft force sensor has been reported elsewhere [39]. Soft sensors, however, have other drawbacks such as smaller operation range, mechanical fatigue, unactuated degrees of freedom, nonlinearities, etc.

6.4 Hybrid Control Switching

For an explicit force controller it is necessary to switch from position to force control when using a Hybrid Control framework. One way to achieve this is to switch to force control when a measured force threshold is exceeded. To prevent force signal noise from causing the switch, a value of 2 N was used for the threshold value for switching *to* force control. Also, since the noise still exists while in force control mode, the measured force may drop below 2 N inadvertently. Thus, a lower threshold value of 1 N was chosen for switching *from* force control. The switching strategy was implemented in the second joystick process running at 150 Hz. Because the switching was done by the joystick controller, the joystick values could be interpreted as commanded velocity (free space motion), or commanded force (constrained motion).

Another aspect of the switching strategy is that it could be made unidirectional — permitting only switching to force control. When unidirectional, force control will remain in effect even if the measured force is reduced below the threshold, as when the manipulator leaves the surface. The behavior of the controller for this case of contact loss can be quite interesting and illustrative [32, 35]. Experience showed that some of the controllers tested were sure to become unstable when surface contact was lost. To prevent damage to the system, bidirectional switching was usually used. (If the end effector lost contact with the environment, the controller reverted to position mode, as can be seen at the tail the data in Figure 11.) To prevent the manipulator from loosing contact with the environment, Impact Control proved extremely effective [32].

6.5 The Impact Transient

The transition from free space motion to contact with the environment provides the greatest test to the stability of the chosen control strategy. This is because of the almost instantaneous exertion of reaction forces upon the arm. Some researchers have addressed this problem by utilizing soft force sensors, or a soft 'skin' over the force sensor surface [39, 1]. The introduction of extra compliance extends the period of impact and absorbs some of the energy. As mentioned in Section 6.3, we have chosen not to use passive compliance because of its intrinsic problems such as mechanical fatigue and nonlinearities.

Considering the case of hard surface to hard surface impact, the transient time is very short. For a manipulator end-effector moving at 1 m/s impacting a surface with stiffness of 10^4 N/m, the force will initially increase at a rate of 10^4 N/s. If a resolution of 1 N is considered adequate for control, the sampling rate must be 10^4 Hz. This is 25 times faster than our sampling rate. Further, a required sampling rate of over 10^6 Hz would be necessary for robustness to some of the surfaces/speeds that were tried in our experimentation.

This impact transient is further complicated by the available torque of the actuators. Even if the sampling rate were fast enough to adequately detect the rise in external forces due to the impact, the joint torque necessary to substantially soften the impact is not available. In other words, the arm cannot stop itself instantaneously to prevent the impact. Assuming a maximum allowed impact force of 10 N, the arm would have to stop in $1mm$. To stop this suddenly, an acceleration of 500 m/s^2 is required (over 50 g!). For a manipulator with an effective Cartesian Space space mass of 1 kg, 500 N is required. At least 98% of this force must be provided by the arm itself. Obviously this is not feasible for conventional actuators and manipulators.

A third problem is the kinematic fluctuations of the arm during impact. This is mainly due to the compression and flexion caused by the impact forces. For instance, vibrations in the links will cause changes in the end effector position, although no change in joint position is measured. This can cause problems for schemes that rely on accurate measurement of the surface position or velocity, such as stiff impedance controllers [24] or impact damping strategies [11].

Other strategies that do not depend on position measurements are still effected by low sampling rate and limited motor torque. These two problems lead to the failure of positive proportional gain and integral gain force control during impacts. Since the measured force overshoot is large and immediate, the first error signals to the controller tend to drive the manipulator with a force directed away from the surface (to reduce the error). This actuated force, coupled with the reaction force from the surface, drives the manipulator off the surface. Once off the surface, the measured force drops to zero, and the controller drives the arm into the surface again. Bouncing ensues.

We have proposed a the use of negative gain proportional force control to address this problem [32]. Instead of retreating from the large reaction force of the surface, the controller matches it in the opposite direction. Thus, the manipulator 'sticks' to the surface until the impact transient is over. After this period, and with stable contact with the environment established, a switch is made to a robust force tracking strategy such as integral gain force control.

7 Discussion of Results

This section presents the force control results we have obtained [35], given the implementational considerations previously discussed.

The best force tracking results were obtained with an integral gain explicit force controller, Equation 2. Figure 13 shows the results for an integral gain of $K_{fi} = 22.5$, where the reference

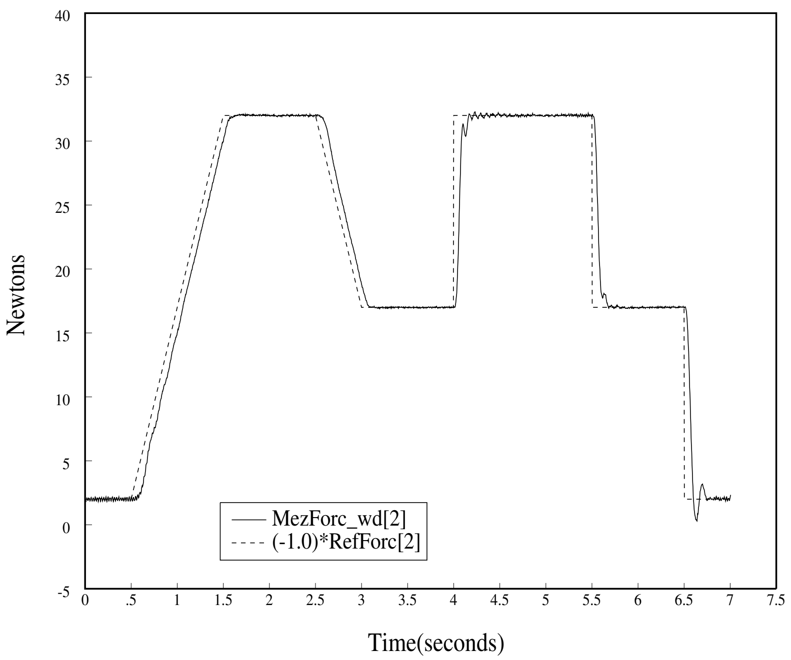


Figure 13: Experimental data of integral gain explicit force control with feedforward and $K_{fi} = 22.5$.

force is the dotted curve and the measured force is the solid curve. Integral gain worked best for two main reasons: 1.) this strategy intrinsically acts as a low pass filter that very effectively filters the noise seen in Figure 8, and 2.) the lag of the controller prevents the manipulator from reacting faster than the environment in which it is in contact, and thereby losing contact with it. Both of these aspects successfully address rapid changes in the measured or commanded force. Finally, integral gain control compensated for inaccuracies in the actuator torques, and provided zero steady state error.

Contrary to integral gain force control, proportional gain force control was both less stable and less accurate. Figure 14 shows the result for a proportional gain of $K_{fp} = 0.5$, where the reference force is the dotted curve and the measured force is the solid curve. This controller had a large steady state error which cannot be eliminated, even for the highest stable gains. Further, for increasing gains the overshoot and oscillations increase markedly. We have previously shown experimentally and explained analytically that the proportional gain force controller is equivalent to second order impedance control, with and without dynamics compensation [33]. The response of the impedance controllers, therefore, will not be shown here.

Finally, we have developed and tested an impact controller for stable contact transition [32]. Figure 15 shows the result of this control strategy, where the dotted curve is the measured velocity, the dashed curve is the commanded force, and the solid curve is the measured force. This controller utilizes both negative gain proportional control, and positive gain integral control with a simple switching scheme. It has proven very successful in our tests.

8 Conclusion

This paper has presented the computational considerations we have found necessary while implementing a wide variety of force control strategies. Not only do these 'implementational details'

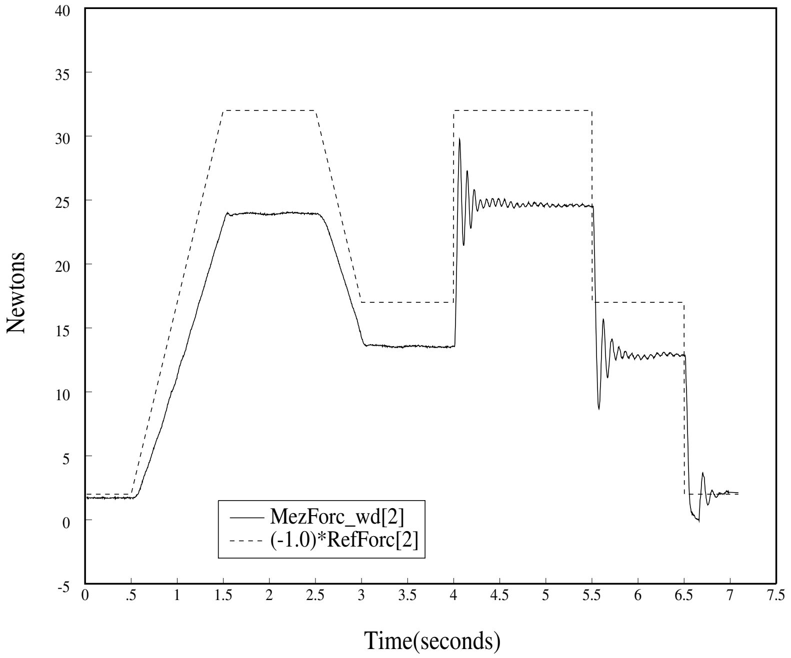


Figure 14: Experimental data of proportional gain explicit force control with feedforward and $K_{fp} = 0.5$.

require significant thought and engineering, they can govern whether a control strategy is feasible or not. For instance, in the case of derivative force control, system noise prevents its stable operation. Filtering of this noise is also one of the attractive aspects of integral force control, which we consider to be the best performer. Another example of a 'detail' is force sensor interrupt handling, which can drastically reduce the computational bandwidth, and therefore the sampling rate and stability margins of all control strategies. A third example is the velocity signal, which cannot be reliably used for damping in stiff contact operations because of computational lag, exacerbated by high natural frequencies. Each of these problems may be alleviated by cleaner measurement techniques and faster processing. However, it is apparent that force control techniques that are not sensitive to these problems are preferable. We have also demonstrated that such robust techniques are currently realizable.

9 Acknowledgements

This research was performed at Carnegie Mellon University and supported by an Air Force Graduate Laboratory Fellowship (for Richard Volpe), DARPA under contract DAAA-21-89C-0001, the Department of Electrical and Computer Engineering, and The Robotics Institute.

The writing and publication of this paper was supported by the above and the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

The views and conclusion contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Air Force, DARPA, or the U.S. Government. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California

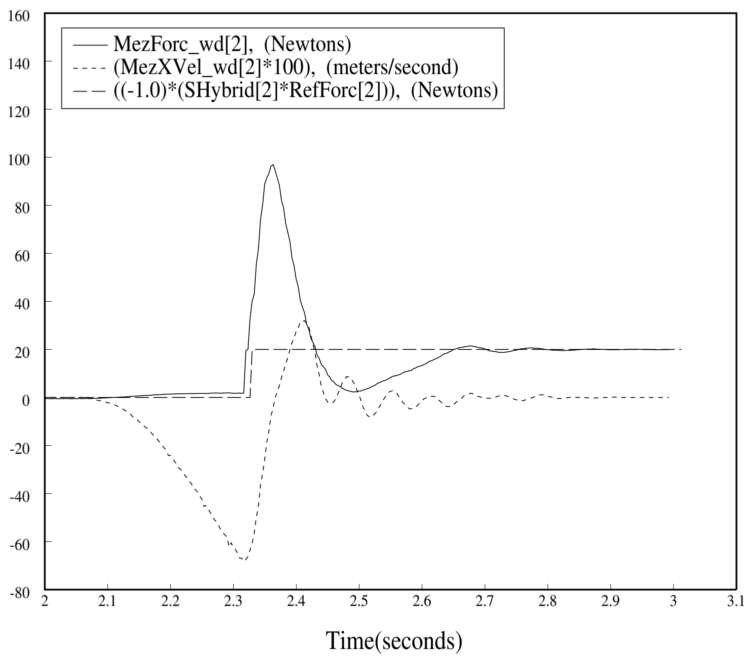


Figure 15: Experimental data of impact control with transition to integral gain force control. The impact control phase lasts for 0.15s after the beginning of the impact. This is followed by a period of transition from impact control to integral gain force control which lasts 0.15s. Beyond 0.3s after impact, integral gain force control is used.

Institute of Technology.

References

[1] C. An and J. Hollerbach. Dynamic Stability Issues in Force Control of Manipulators. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 890–896, 1987.

[2] A. Bejczy. Robot Arm Dynamics and Control. Technical Memorandum 33-669, Jet Propulsion Laboratory, Pasadena, CA, February 1974.

[3] E. Colgate and N. Hogan. An Analysis of Contact Instability In Terms of Passive Physical Equivalents. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 404–409, 1989.

[4] S. Eppinger and W. Seering. On Dynamic Models of Robot Force Control. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 29–34, 1986.

[5] S. Eppinger and W. Seering. Understanding Bandwidth Limitations on Robot Force Control. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 904–909, Raleigh, N.C., 1987.

[6] W. Hamilton. Globally Stable Compliant Motion Control For Robotic Assembly. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1179–1184, 1988.

[7] N. Hogan. Impedance Control: An Approach to Manipulation: Parts I, II, and III. *Journal of Dynamic Systems, Measurement, and Control*, 107:1–24, March 1985.

[8] H. Kazerooni, T. Sheridan, and P. Houpt. Robust Compliant Motion for Manipulators, Parts I and II. *IEEE Journal of Robotics and Automation*, RA-2(2):83–105, June 1986.

[9] O. Khatib. *Commande Dynamique dans l'Espace Operationnel des Robots Manipulateurs en Presence d'Obstacles.* PhD thesis, Ecole Nationale Superieure de l'Aeronautique et del'Espace (ENSAE), December 1980.

[10] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, 5(1), 1986.

[11] O. Khatib and J. Burdick. Motion and Force Control of Robot Manipulators. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1381–1386, 1986.

[12] P. Khosla. Effect of Sampling Rates on the Performance of Model-Based Manipulator Control Schemes. In G. Schweitzer, editor, *Dynamics of Controlled Mechanical Systems*, pages 271–284. Springer-Verlag, August 1988.

[13] K. Kreutz. On Manipulator Control by Exact Linearization. *IEEE Transactions on Automatic Control*, 34(7):763–767, July 1989.

[14] J. Luh, M. Walker, and R. Paul. Resolved-Acceleration Control of Mechanical Manipulators. *IEEE Transactions on Automatic Control*, 25(3):468–474, June 1980.

[15] B. Markiewicz. Analysis of the Computed-Torque Drive Method and Comparison with the Conventional Position Servo for a Computer-Controlled Manipulator. Technical Memorandum 33-601, Jet Propulsion Laboratory, Pasadena, CA, March 1973.

[16] M. Mason. Compliance and Force Control for Computer Controlled Manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6):418–432, June 1981.

[17] F. Miyazaki and S. Arimoto. Sensory Feedback for Robot Manipulators. *Journal of Robotic Systems*, 2(1):53–71, 1985.

[18] R. Paul. Problems and Research Issues Associated with the Hybrid Control of Force and Displacement. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1966–1971, June 1987.

[19] R. Paul and B. Shimano. Compliance and Control. In *Proceedings of the Joint Automatic Control Conference*, pages 694–699. American Automatic Control Council, 1976.

[20] R. Paul and C. Wu. Manipulator Compliance Based on Joint Torque. In *IEEE Conference on Decision and Control*, pages 88–94, New Mexico, 1980.

[21] R. P. Paul. *Robot Manipulators : Mathematics, Programming and Control.* MIT Press, Cambridge, MA, 1981.

[22] William Press and etal. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge Univesity Press, Cambridge, 1988.

[23] M. Raibert and J. Craig. Hybrd Position/Force Control of Manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 103(2):126–133, June 1981.

[24] J. K. Salisbury. Active Stiffness Control of a Manipulator in Cartesian Coordinates. In *IEEE Conference on Decision and Control*, pages 95–100, New Mexico, 1980.

[25] D. Stewart, D. Schmitz, and P. Khosla. Implementing Real-Time Robotic Systems Using Chimera II. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 598–603, May 1990.

[26] D. Stewart, R. Volpe, and P. Khosla. Integration of Real-Time Software Modules for Reconfigurable Sensor-Based Control Systems. In *Proceedings of the 1992 IEEE International Conference on Intelligent Robots and Systems*, July 1992.

[27] W. Townsend and J. Salisbury. The Effect of Coulomb Friction and Stiction on Force Control. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 883–889, 1987.

[28] D. Vischer and O. Khatib. Design and Development of Torque-Controlled Joints. In V. Hayward and O. Khatib, editors, *Experimental Robotics I*, pages 271–286. Springer-Verlag Berlin Heidelberg, 1990.

[29] R. Volpe. *Real and Artificial Forces in the Control of Manipulators: Theory and Experiments.* PhD thesis, Carnegie Mellon University, Department of Physics, September 1990.

[30] R. Volpe and P. Khosla. Manipulator Control with Superquadric Artificial Potential Functions: Theory and Experiments. *IEEE Transactions on Systems, Man, and Cybernetics; Special Issue on Unmanned Vehicles and Intelligent Systems*, November/December 1990.

[31] R. Volpe and P. Khosla. Theoretical Analysis and Experimental Verification of a Manipulator / Sensor / Environment Model for Force Control. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Los Angeles, November 1990.

[32] R. Volpe and P. Khosla. Experimental Verification of a Strategy for Impact Control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991.

[33] R. Volpe and P. Khosla. The Equivalence of Second Order Impedance Control and Proportional Gain Explicit Force Control: Theory and Experiments. In *Proceedings of the Second Annual International Symposium on Experimental Robotics*, Toulouse, France, June 1991.

[34] R. Volpe and P. Khosla. An Analysis of Manipulator Force Control Strategies Applied to an Experimentally Derived Model. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Raleigh, North Carolina, July 7-10 1992.

[35] R. Volpe and P. Khosla. An Experimental Evaluation and Comparison of Explicit Force Control Strategies for Robotic Manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France, May 10-15 1992.

[36] D. Wedel and Saridis G. An Experiment in Hybrid Position/Force Control of a Six DOF Revolute Manipulator. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1638–1642, 1988.

[37] D. Whitney. Force Feedback Control of Manipulator Fine Motions. *Journal of Dynamic Systems, Measurement, and Control*, pages 91–97, June 1977.

[38] D. Whitney. Historical Perspective and State of the Art in Robot Force Control. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 262–268, 1985.

[39] Y. Xu and R. Paul. On Position Compensation and Force Control Stability of a Robot with a Compliant Wrist. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1173–1178, 1988.

[40] K. Youcef-Toumi. Force Control of Direct-Drive Manipulators For Surface Following. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 2055–2060, 1987.

[41] K. Youcef-Toumi and D. Gutz. Impact and Force Control. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 410–416, 1989.