BOND GRAPH MODEL BASED CONTROL OF ROBOTIC MANIPULATORS

A DISSERTATION

SUBMITTED TO THE FACULTY OF ENGINEERING

OF GLASGOW UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

David Wynn Roberts

June 1993

Dedication

This thesis is dedicated to my parents, Mabel Rita and Arthur Wynn Roberts, for their unfailing support and encouragement throughout my life and education. It is also dedicated to Kay for keeping me sane whilst writing the thesis.

Abstract

The performance of robotic manipulators is critical to their widespread use in industry. As manipulators become faster, their potential productivity can rise thus improving the return on the investment required to purchase them. Improving accuracy, on the other hand, increases the range of tasks for which the manipulator is suitable.

The speed and accuracy of a manipulator is partly determined by the capability of the algorithm used to control it. Whilst being a highly non-linear multiple input, multiple output device, however, most industrial controllers are derived on the basis that the robot is a series of independent, linear actuator+link subsystems. The resulting independent joint controller is simple to design and implement but is limited in its performance as link interactions and the non-linear effects of centrifugal and Coriolis forces degrade the accuracy at high manipulator velocities.

Improvements in the control of manipulators may be made by incorporating a mathematical model of the manipulator in the control algorithm. Control schemes such as 'computed torque' incorporate an inverse model of the manipulator to calculate the input torques required to force the end-effector to follow a desired trajectory. The equations of motion required to implement these controllers are large and complex even for relatively simple manipulators.

This thesis explores how bond graph representations of robotic manipulators may be used to automate the implementation of model based controllers. To provide a practical basis for this research the bond graph derived controllers are

tested on an experimental rigid, planar, direct drive two-link manipulator. It is shown how the bond graph for this manipulator, including d.c.motor actuators, can be constructed and used to derive the equations of motion of the manipulator automatically. The bond graph model is then validated by comparing simulations obtained using these equations of motion with experimental data.

Two approaches to model based control are investigated: a model based observer and inverse model based control.

The model based observer is one way of tackling the problem of noise contaminated joint angular velocity measurements obtained through tachometers. By modifying the standard form of the two-link manipulator bond graph into an observer format, the equations and software required to implement a full order non-linear model based observer can be created automatically. With a linear feedback loop implemented around the observer, the observed state vector can be made to track the state vector of the manipulator accurately allowing observed angular velocities to replace measured angular velocities in an independent joint feedback controller. As the observed velocities are less contaminated by noise, the gains can be increased significantly thus increasing the bandwidth of the controller and improving the performance of the manipulator.

The basic bond graph can also be modified to construct the inverse system bond graph and this is demonstrated for the two-link manipulator. From this bond graph, the equations and software required to implement a 'computed torque' controller can be derived automatically. In practice, this automatically derived controller considerably improves the available speed and accuracy of the experimental manipulator over standard independent joint controllers.

Acknowledgements

My thanks go primarily to my supervisor, Professor P.J. Gawthrop, for his guidance, support and encouragement throughout the course of my Ph.D. and to Dr John Howell for his advice on writing the thesis and for proof reading the manuscript.

I would also like to thank my fellow members in the Control Group of Glasgow University for their friendship and help in solving numerous problems. In particular, I thank Dr Donald Ballance and Mrs Yasmine Mather for providing the excellent computing facilities, Dr George Worship for his help in using MTT and Mr Mohammed Abderrahim for the discussions in the field of robotics. I especially thank Dr Richard Jones for his technical guidance and encouragement on a day to day basis.

Finally, I would like to thank the technicians and support staff of Glasgow University Mechanical Engineering department for constructing the experimental apparatus used for my research, and the Science and Engineering Research Council for funding it.

Contents

List of Tables

List of Figures

Chapter 1

Introduction.

The present use of robotic manipulators in industry is limited mainly to simple, repetitive 'pick and place' tasks [1] that require low precision. To maintain, or attain, commercial viability and to extend their usefulness, robotic manipulators must be developed to perform precision tasks at high speeds and with the ability to interact with their environment. In 1988, An, Atkeson and Hollerbach [2] listed the properties desirable for industrial manipulators as:

- fast speed, adequate payload capability
- accurate joint torque control
- accurate position sensing
- accurate velocity sensing
- force control capability
- adequate bandwidth
- adequate computing power

They noted that existing commercial manipulators met few of these requirements.

Whilst much can be done to improve the physical constituents of manipulators, for example in actuator capability, sensor design, link materials and design, real scope for improvement lies within the field of control. A robotic manipulator is a highly non-linear, multiple input, multiple output (MIMO) system which most existing industrial controllers treat as a series of independent, linear systems; so called independent joint control [3] [4]. In this type of control, force and moment interactions between links together with non-linearities such as Coriolis and centrifugal force are ignored and must be considered as system disturbances.

The presence of gear-boxes in most industrial manipulators helps in this respect by increasing the effective inertia of the motor rotor by the square of the gear-box ratio. With a typical gear-box ratio of 100:1, the rotor inertia is magnified 10000 times hence the effects of link inertia are dominated by rotor inertia. The higher motor speeds and the presence of the gear-box also causes high friction. Link dynamics, and the interaction between links, are therefore dominated by actuator dynamics. As the actuators are independent, independent joint control seems to be the sensible way to control the manipulator.

The use of gear-boxes limits the performance of robots, however. Gear backlash and flexibility reduce the ability to accurately control joint position and torque. As these effects are non-linear and vary with manipulator configuration, they are extremely difficult to model. Furthermore, the increase in friction, especially static friction, reduces the capability of the manipulator to be used in a force control mode [2].

To overcome the limitations imposed by gear-boxes, it is possible to design manipulators so that the links are driven directly by the actuators, usually electric motors. The reduction in friction, elimination of backlash and the ability to control joint torque accurately allows for a fast, potentially high precision manipulator but the elimination of gears also means that the full non-linear MIMO characteristics are reflected directly back to the actuators. The reduction in motor speed also

has the secondary effect of decreasing the signal to noise ratio of the tachometers thus restricting controller bandwidth: it is not feasible to increase the gains of independent joint controllers to compensate for the increased effect of the non-linearities.

A more sophisticated control scheme is therefore desirable for directly driven manipulators. This can be done using model-based control schemes which compensate for link interactions and non-linearities by calculating the torques required to counter-act them. Two such control schemes are feed-forward control[5] [6] [7] and 'inverse system' or 'computed torque' control[8] [9] [3] [10].

To implement model based control schemes, a mathematical model of the specific manipulator to be controlled must be derived. The two most commonly quoted modelling techniques in the robot control literature are the energy conservation based Lagrange-Euler formulation [3] [11] [12] and the force-balance based Newton-Euler formulation [13]. These techniques calculate a vector containing the force or torque required at each joint to attain a specified trajectory of joint positions, velocities and accelerations. These techniques are reviewed in the next section.

The main dis-advantages of the above modelling techniques are their complexity and lack of versatility. For a full six-degree of freedom manipulator, the computation of the terms in the equations of motion becomes very complicated and time-consuming [4] necessitating the use of simplification techniques [14] to reduce the equations to more manageable proportions. Furthermore, these mathematical models do not in general include actuator dynamics or joint fiction which would add to the complexity and may invalidate the simplification techniques.

Bond-graphs represent a powerful approach to modelling robotic manipulators and in the subsequent generation of model based controllers. Bond graphs were introduced by Paynter [15] as a graphical representation for dynamic energy exchanging systems. The power of bond graphs lies in the fact that they

provide an unambiguous graphical representation of a system from which other representations, for example the set of system state-space equations, may be derived automatically by computers. It is therefore possible to augment or alter systems without getting involved with the mathematical complexities of the dynamic equations of motion although these are easily obtainable in a range of formats and in human readable form.

The use of bond-graphs to model robotic manipulators has been attempted by several authors [16] and of particular relevance here is the work of Gawthrop [17] [18] [19] to model two-dimensional SCARA type rigid robotic manipulators. This research itself has been part of a project to develop generic techniques to automatically model specific robotic manipulators; a process termed 'meta modelling'.

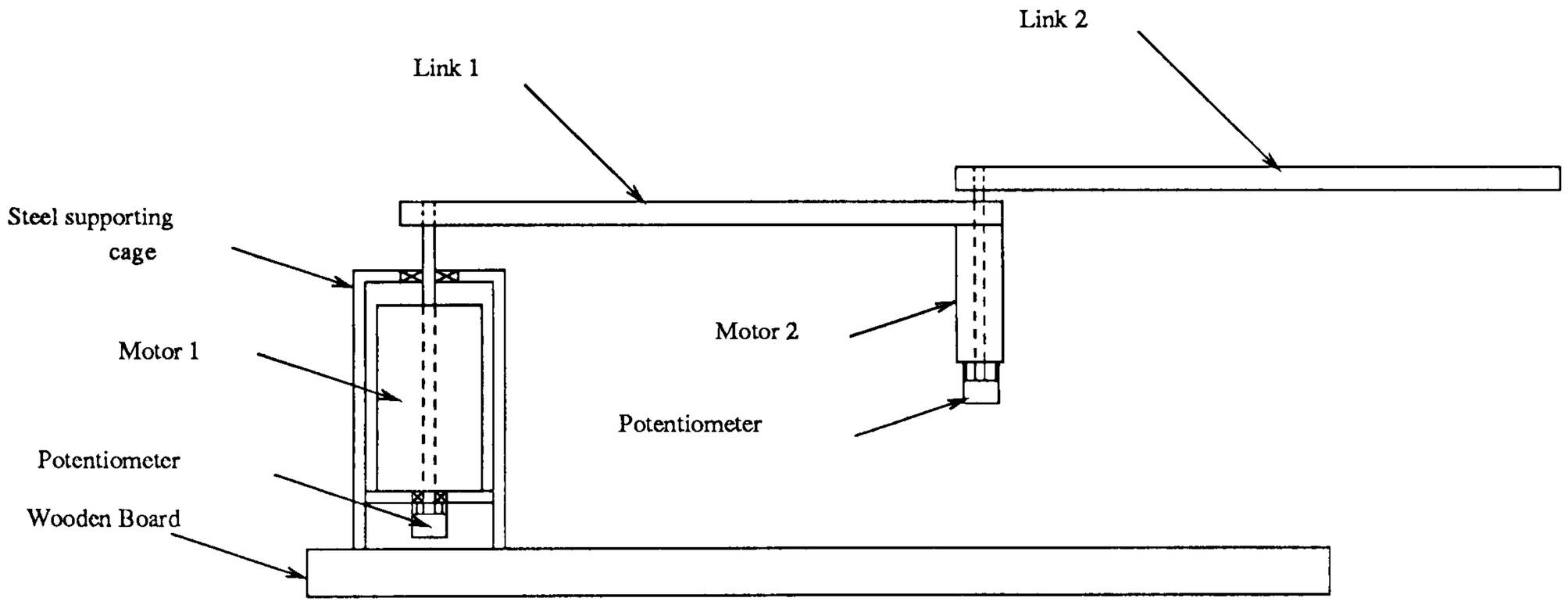


Figure 1.1: Construction of the Two-Link Manipulator.

To investigate how bond-graph models can be used to improve the control of robotic manipulators, it was decided that an experimental, rigid, planar, directly driven two link manipulator (DD2lm, see figure 1.1) be constructed on which to test the new techniques. In the literature, many control schemes have been proposed and tested with the aid of computer simulations but few have been

implemented on real systems. Using the ideas developed by Gawthrop, a specific bond-graph for the experimental DD2lm was constructed, including actuators, from which the dynamic equations of motion were derived automatically using the Model Transformation Toolbox, also developed by Gawthrop [20]. This model could then be validated by comparing simulations obtained using the bond-graph derived equations of motion in the simulation package SIMULAB [21] against real data obtained from the experimental manipulator.

The bond-graph was then augmented, using techniques developed by Karnopp [22], to create a full, non-linear, model based observer. By creating a linear feed-back loop around the observer, it was found that the states of the model could be made to find and track the states of the system. The outputs of the model, which are not prone to measurement noise, could then be used in the feed-back loop of a conventional independent joint controller. The ability to use observed values of angular velocity rather than the poorly conditioned signals from tachometers allows the derivative gain to be increased thus allowing the proportional gain, and hence the speed of the manipulator, to be increased. In effect, the use of the observer allows the bandwidth of the controller to be extended.

The next stage of the research was to investigate how the basic bond-graph for the DD2lm could be used to implement inverse system type controllers such as feed-forward control and computed torque. This was done by modifying the input/output space to give joint torques as output with joint angular accelerations as inputs. The way in which the inverse model interacts with the feed-back loop of the controller defines whether the controller is termed a feed-forward controller or computed torque. The computed torque controller tested using simulations together with practical implementations on the experimental arm.

In summary, I believe the original aspects of this work have been:

- the construction of a bond-graph for a specific two-dimensional, two-degree of freedom, rigid, directly driven manipulator including actuators.

- validation of this model against data obtained from an experimental manipulator.

- development and implementation of a full, non-linear, bond-graph observer and its use to improve the control of the experimental manipulator.

- the development of software to automatically create code to implement observers from a bond-graph representation.

- the use of the specific bond-graph for the two-link manipulator to implement 'inverse system' type controllers such as feed-forward control and computed torque.

Whilst the system used as the test bed for this research, the DD2lm, is relatively simple, the ideas are generic and could, in principle, be used to improve the control of three dimensional, multi-degree of freedom direct drive manipulators once generic techniques to model three dimensional manipulators have been developed.

1.1 Literature Survey.

1.1.1 Robotic Modelling.

The creation of the dynamic equations of motion of a robotic manipulator may be done for various reasons including:

- computer simulation of robot arm motion

- analysis of manipulator design and performance
- evaluation of controller design
- constituent part of controller algorithm

The method by which the equations of motion are generated depends to a large extent on the desired use of the model. With dynamic mechanical modelling packages, the accuracy of the model may be more important than computational complexity as simulation need not be carried out in real time.

For model-based control of robotic manipulators the equations of motion must be of a sufficiently concise form to allow the model to be run in real time at the same 'speed' as the actual system. In practice, this means that the computer running the model must be capable of computing the generalised joint torque/force vector $\boldsymbol{\tau}$ at each stage of the desired trajectory $(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$, where $\boldsymbol{\theta}$ is a vector of joint angles, in a sufficiently short time to allow the sample rate of the controller to exceed the highest natural frequency of the manipulator, preferably by a factor of at least six.

The two approaches most commonly used to model robot arm dynamics are the Lagrange-Euler (L-E) and Newton-Euler (N-E) methods although others such as Recursive Lagrangian [23] and Generalised D'Alembert [24] have also been used.

The general form of the dynamic equations of motion most useful for control purposes is

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{\theta})(\ddot{\boldsymbol{\theta}}) + \mathbf{V}(\dot{\boldsymbol{\theta}}) + \mathbf{f}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta}; i, j = 1, 2...n) + \boldsymbol{g}(\boldsymbol{\theta}) \qquad (1.1)$$

where

$$
\begin{aligned}
\mathrm{n} &= \text{number of degrees of freedom of the manipulator} \\
\boldsymbol{\tau} &= \mathrm{n} \times 1 \text{ vector of generalised torques/forces} \\
\boldsymbol{J}(\boldsymbol{\theta}) &= \mathrm{n} \times \mathrm{n} \text{ inertia matrix} \\
\mathbf{V} &= \mathrm{n} \times \mathrm{n} \text{ viscous friction matrix} \\
\mathbf{f}(\theta_i\theta_j, \boldsymbol{\theta}) &= \mathrm{n} \times 1 \text{ vector of Coriolis and centrifugal terms} \\
\boldsymbol{g}(\boldsymbol{\theta}) &= \mathrm{n} \times 1 \text{ vector of gravity terms}
\end{aligned}
$$

Uicker [11], Paul [3] and Lewis [12] have derived the equations of motion for manipulators having n joints using Lagrangian generalised co-ordinates. With friction omitted, the equations of motion can be written [13]

$$
\begin{aligned}
\tau_i = & \sum_{j=1}^{n} \left\{ \sum_{k=1}^{j} Tr\left[\mathbf{U}_{jk}\mathbf{J}_j(\mathbf{U}_{ji})'\right] \ddot{\theta}_k \right\} \\
& + \sum_{j=i}^{n} \left\{ \sum_{k=1}^{j} \sum_{m=1}^{j} Tr\left[\mathbf{U}_{jkm}\mathbf{J}_j(\mathbf{U}_{ji})'\right] \dot{\theta}_m \dot{\theta}_k \right\} \\
& - \sum_{j=1}^{n} m_j \hat{\mathbf{g}}' \mathbf{U}_{ji} \hat{\mathbf{r}}_j{}' \qquad (1.2)
\end{aligned}
$$

for 1=1,2...n, where

$$
\begin{aligned}
\mathrm{Tr} &= \text{trace operator,} \\
(\ )' &= \text{transpose of } (\ ) \\
\tau_i &= \text{input generalised force for joint i,} \\
m_j &= \text{mass of link j,} \\
\hat{\mathbf{r}}_j{}' &= \text{vector describing centre of mass of link j} \\
&\quad \text{with respect to } j^{th} \text{ co-ordinate system,} \\
\hat{\mathrm{g}}' &= [0, 0, 9.8ms^{-2}], \text{ gravitational acceleration vector,} \\
\mathbf{J}_j &= \text{inertia matrix for link j,} \\
\mathbf{U}_{jk} \text{ and } \mathbf{U}_{jkm} &= 4 \times 4 \text{ matrices which transform vectors and matrices} \\
&\quad \text{among various joint co-ordinate systems.}
\end{aligned}
$$

For a specific manipulator, the computation of the terms of equation 1.2 is very complicated and time consuming, as illustrated by Luh [4]. The required number

of matrix multiplications for the first term of equation 1.2, representing forces due to accelerations, is of $O(n^3)$ whilst that of the second term, representing Coriolis and centrifugal terms, is of $O(n^4)$. This complexity requires large execution times to compute the joint torques.

In experiments with the six degree of freedom Stanford manipulator, Paul [3] found the contribution from the Coriolis and centrifugal terms to be relatively insignificant, especially at the low velocities required around the 'goal' positions of point to point moves. Consequently, the execution time could be significantly reduced by dropping the second term of equation 1.2 thus reducing the required number of multiplications to $O(n^3)$. By finding approximations to the acceleration and gravity terms of equation 1.2, Bejczy [14] was able to further reduce the execution time to an acceptable level for control, but only for a specific manipulator at low velocities.

Due to the inability of the Lagrange-Euler derived equations of motion to calculate the generalised torque/force vector $\boldsymbol{\tau}$ in a time suitable for on-line control, Luh, Walker and Paul [13] formulated a computational scheme to calculate $\boldsymbol{\tau}$ using the Recursive Newton-Euler technique. Given the joint trajectory $(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$ at any instant of time, the rotational velocity and linear and rotational acceleration of the centre of mass of each link can be calculated iteratively starting from the basal link out to the distal link using the mathematics of moving co-ordinate systems and a knowledge of the kinematics of the manipulator. The joint torques required to attain this trajectory can then be calculated iteratively from the distal joint back to the basal joint by writing force and moment balance equations at each joint, given a knowledge of the inertial parameters for each link. A concise form for the Iterative Newton-Euler dynamic formulation is given in Craig [1].

The use of the Newton-Euler Recursive equations reduces the computational complexity to $O(n)$ thus providing an efficient formulation but at the loss of the structure of the dynamic equations of motion (1.1) which can make it difficult to

use this formulation in advanced model based control schemes.

To summarise, the Lagrange-Euler method produces a closed form set of equations of motion which, due to their complexity, are generally unsuited for real time control. They can be simplified by ignoring centrifugal and Coriolis contributions but at the expense of making accurate control of fast manipulator motion impossible.

The Recursive Newton-Euler method provides an efficient set of equations whose complexity varies linearly with the number of joints n of the manipulator but whose lack of structure complicates the development of advanced control techniques.

Other formulation methods include the Generalised D'Alembert (G-D) scheme [24], a force-balance based method which retains the closed form structure of the equations but at a complexity of $O(n^3)$ and a Recursive Lagrangian formulation [23] which has similar capabilities and dis-advantages to that of Recursive Newton-Euler.

1.1.2 Robotic Control.

The purpose of robotic control is to modify the inputs to joint actuators to force an end-effector to follow a desired trajectory as closely as possible. This trajectory may include force interaction with the manipulators' environment but here we consider only positional control.

The trajectory for the end effector to follow is more usefully defined in task co-ordinates which, for example, may be cartesian co-ordinates based on the workspace in which the manipulator operates. It is, however, most straightforward to control the robot using joint angles, velocities and accelerations which necessarily involves a transformation from task to joint co-ordinates using the inverse kinematics of the manipulator. For a pre-planned trajectory, this transformation can be executed off-line but with inevitable errors introduced by

discrepancies between the true and measured kinematics of the manipulator.

We consider here three joint space controllers

- Independent Joint Control
- Feed-Forward Control
- Computed Torque Control

Independent Joint Controller.

The independent joint controller [25] [4] is the simplest and most commonly used controller for commercial manipulators. As its name suggests, the controller treats each actuator-joint-link as an independent system to which it applies a proportional and derivative (PD) feed-back controller.

Each joint of a robotic manipulator is usually equipped with a position sensing device, such as an optical encoder or potentiometer, and a tachometer to measure joint velocity. For an n degree of freedom manipulator, the control input is calculated by

$$\boldsymbol{\tau} = \mathrm{k}_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}) - \mathrm{k}_v\dot{\boldsymbol{\theta}} \tag{1.3}$$

where

$\boldsymbol{\tau}$ = $n \times 1$ vector of generalised input torques/forces,

$\boldsymbol{\theta}$ = $n \times 1$ vector of joint positions,

$\dot{\boldsymbol{\theta}}$ = $n \times 1$ vector of joint velocities,

$\boldsymbol{\theta}_d$ = $n \times 1$ vector of desired joint positions,

$\mathrm{k}_p, \mathrm{k}_v$ = proportional and derivative gains respectively.

The velocity term of equation 1.3 is introduced to increase damping in order to stabilise the system. At high velocities, however, this term may introduce unnecessarily high damping reducing the speed of the manipulator. This can be

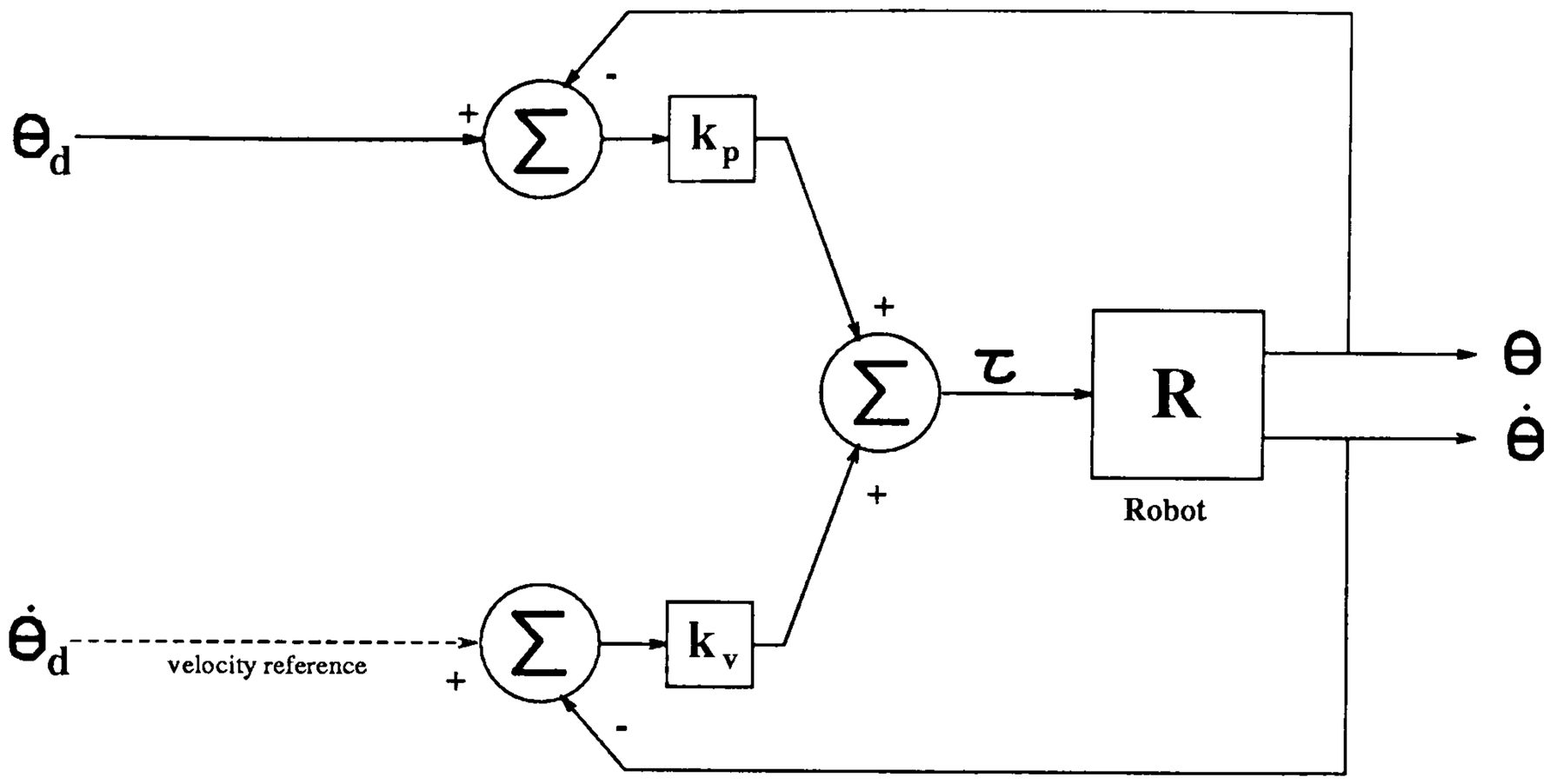


Figure 1.2: Schematic of Independent Joint Controller.

remedied by including trajectory velocity reference as shown in figure 1.2 where $\dot{\theta}_d$ is a vector of desired velocities. The control law then becomes

$$\tau = \mathrm{k}_p(\theta_d - \theta) - \mathrm{k}_v(\dot{\theta}_d - \dot{\theta}) \tag{1.4}$$

The choice of the gains $\mathbf{k}_p$ and $\mathbf{k}_v$ is largely empirical. The higher the values of the gains, the more closely the robot will follow the desired trajectory. The gains are limited, however, by actuator capability and instabilities caused by command and measurement noise and delays. Whilst it is possible to choose position sensing devices which give well-conditioned signals, velocity sensors such as tachometers are prone to significant measurement noise [26]. Directly driven manipulators are particularly affected as the relatively low velocities of the drive motors cause a reduction in the signal to noise ratio from the tachometers. This measurement noise severely limits the differential gains of the controller and hence the damping which, in turn, limits the proportional gain and the attainable speed of the manipulator. Differentiation of position to obtain velocity amplifies the higher frequencies of noise and hence does not provide an easy solution.

A more sophisticated way to derive the feed-back gains is detailed in Luh [4]. For each joint, the dynamics of the actuator are modelled to give the transfer

function between demanded position θ_d and actual position θ. For the example of a d.c. motor this results in the second order system

$$\frac{\Theta(s)}{\Theta_d(s)} = \frac{nk_\theta k_I}{RJ_{eff}} \frac{1}{s^2 + [RB_{eff} + k_I(k_b + k_l k_t)]\frac{s}{RJ_{eff}} + \frac{nk_\theta k_I}{RJ_{eff}}} \tag{1.5}$$

where

$\Theta(s)$ = Laplace transform of link angular position,

$\Theta_d(s)$ = Laplace transform of desired link angular position,

n = gearbox ratio,

k_I = torque constant of motor,

k_b = back emf constant of motor,

k_t = tachometer gain,

k_l = amplifier gain (V/V),

k_θ = proportional gain of controller,

R = resistance of motor armature winding,

J_{eff} = effective inertia of link,

B_{eff} = effective damping coefficient,

The characteristic equation is therefore given by

$$s^2 + \{RB_{eff} + k_I(k_b + k_l k_t)\}\frac{s}{RJ_{eff}} + \frac{nk_\theta k_I}{RJ_{eff}} = 0 \tag{1.6}$$

which may be expressed

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \tag{1.7}$$

where ζ is the damping ratio and ω_n is the undamped natural frequency.

Since $n, k_I, k_t, k_b, R, J_{eff}$ and B_{eff} are either specified or can be measured, the dynamics of the system are determined by the selection of the positional gain k_θ and the amplifier gain k_l. To avoid exciting structural resonances, the undamped natural frequency is set to be no higher than half the structural resonant frequency leaving a safety factor of 200percent [4]. For an effective inertia J and measured structural resonant frequency ω,the proportional gain is constrained by

$$k_\theta \leq \frac{(J\omega^2)R}{4nk_I} \tag{1.8}$$

To avoid overshoots, the system must never be under-damped thus $\zeta \geq 1$ at all times. This leads to a lower bound for k_l given by

$$k_l \geq \frac{R(\omega\sqrt{JJ_{eff}} - B_{eff})}{k_I k_t} - \frac{k_b}{k_t} \tag{1.9}$$

For simplicity of controller design, the gains k_θ and k_t should be kept constant. J_{eff}, however, varies with manipulator configuration and load so, to ensure that the system is never under-damped, the largest value of J_{eff} should be used.

The independent joint controller is a relatively straightforward controller which relies on conservative design to avoid instabilities. This results in a system which does not exploit the physical capabilities of the manipulator to the full and is therefore slower than it could be. It relies on feed-back control to minimise the effect of link interactions and non-linearities such as Coriolis and centrifugal forces. The extent to which these 'disturbances' can be rejected is limited though by structural resonancies and measurement noise from the sensors. The main advantage of independent joint control is that it does not rely on the derivation of the equations of motion of the manipulator.

Feed-Forward Controller.

The feed-forward controller [5] is perhaps the simplest form of model-based controller. It uses the dynamic model of the manipulator, $\hat{R^{-1}}$, to predict what the generalised input torque/force vector $\boldsymbol{\tau}$ should be to achieve a desired joint trajectory $(\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d)$. This effectively compensates for all link interactions. Coriolis and centrifugal forces but there will still be trajectory errors caused by outside disturbances, measurement and command noise, un-modelled dynamics and discrepancies between the dynamic model and the true system. It is therefore

necessary to use a feed-back controller in conjunction with the feed-forward controller.

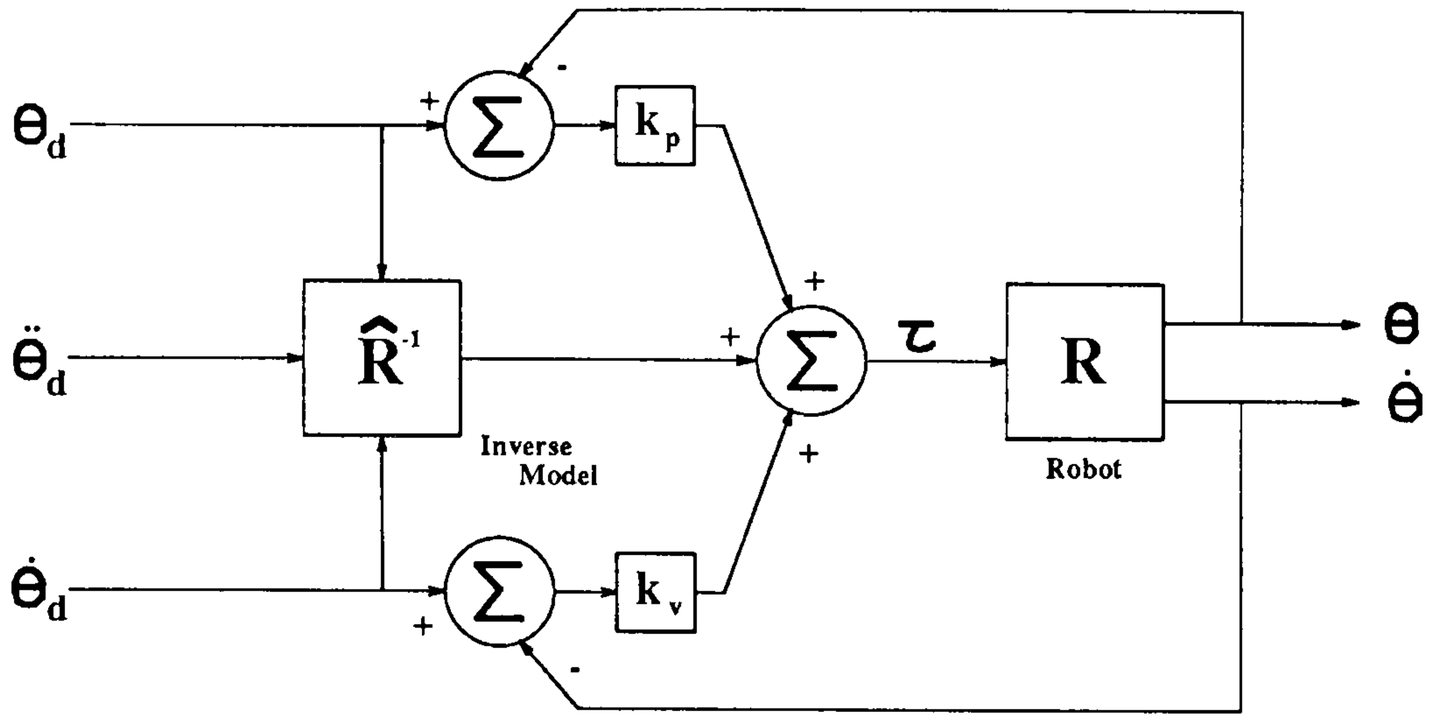


Figure 1.3: Schematic of Feed-Forward Controller.

The control law is defined as

$$\boldsymbol{\tau} = \hat{R^{-1}}(\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d) + \mathrm{k}_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}) + \mathrm{k}_v(\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}) \tag{1.10}$$

The gains k_p and k_v are essentially the same as for the independent joint controller and can be chosen in a similar way. The addition of the feed-forward term, however, considerably reduces the workload of the feed-back controller thus allowing the gains to be smaller to avoid potential instabilities.

A significant advantage of the feed-forward controller is that for a pre-planned trajectory $(\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d)$ the feed-forward terms of 1.10 may be calculated off-line leaving only the corrective torques of the feed-back controller to be calculated on-line. As a result the sample rate of the controller can be kept high.

The dis-advantage of the feed-forward controller is that corrective torques in the feed-back control of one joint will perturb all other joints.

Computed Torque Controller.

The computed torque technique [8] [14] is also known as the 'inverse problem technique' [3] [9]. It differs from the feed-forward controller in the way that the model interacts with the feed-back loop to avoid corrective torques for one joint perturbing all other joints.

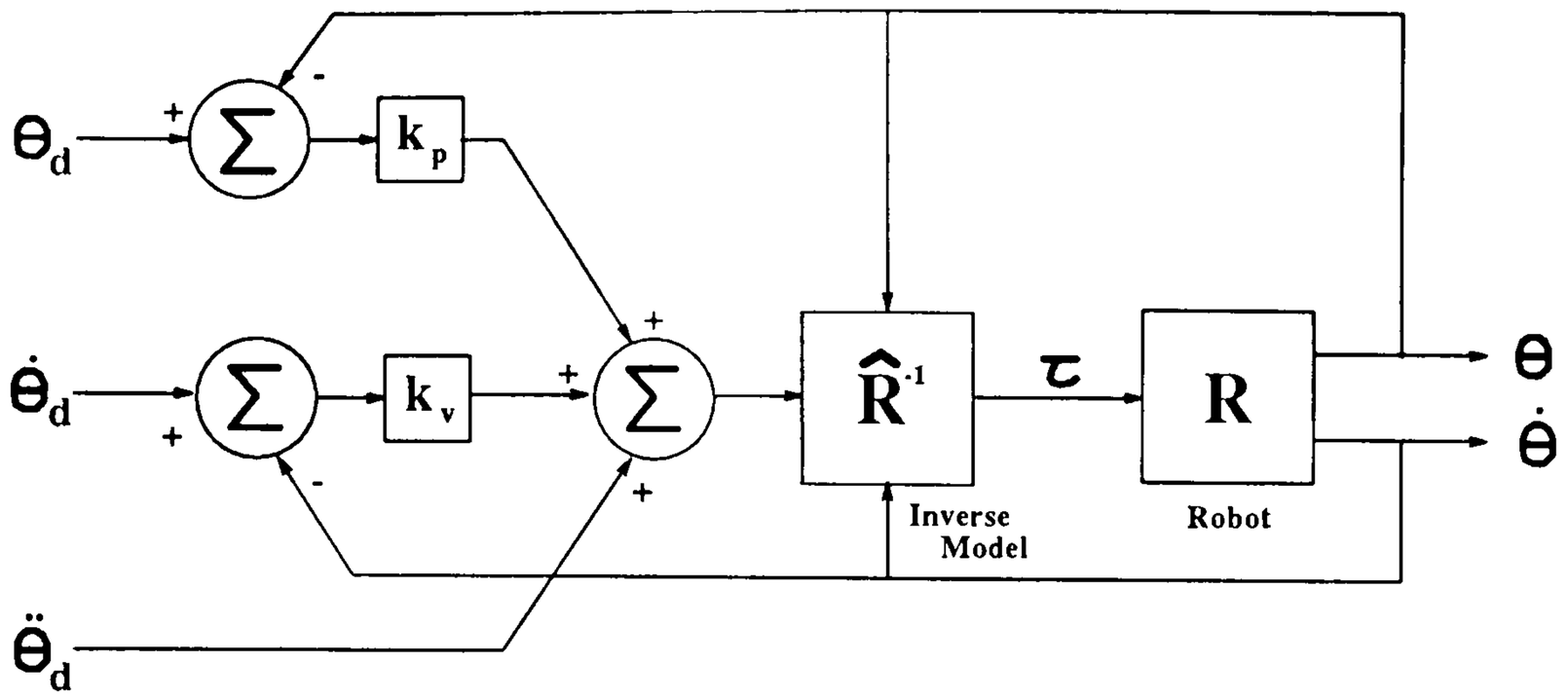


Figure 1.4: Schematic of Computed Torque Controller.

With reference to equation 1.1

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{\theta})(\ddot{\boldsymbol{\theta}}) + \mathbf{V}(\dot{\boldsymbol{\theta}}) + \mathbf{f}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta}; i, j = 1, 2...n) + \boldsymbol{g}(\boldsymbol{\theta}) \tag{1.11}$$

For the computed torque technique, the desired input torque is given by

$$\boldsymbol{\tau} = \hat{\mathbf{J}}(\boldsymbol{\theta})\left\{(\ddot{\boldsymbol{\theta}}_d + \mathrm{k}_v(\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}) + \mathrm{k}_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}))\right\} + \hat{\mathbf{V}}(\dot{\boldsymbol{\theta}}) + \hat{\mathbf{f}}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta}; i, j = 1, 2...n) + \hat{\mathrm{g}}(\boldsymbol{\theta}) \tag{1.12}$$

If the model is exact then

$$\hat{\mathbf{J}}\boldsymbol{\theta} = \mathbf{J}\boldsymbol{\theta} \tag{1.13}$$

$$\hat{\mathbf{V}}(\dot{\boldsymbol{\theta}}) = \mathbf{V}(\dot{\boldsymbol{\theta}}) \tag{1.14}$$

$$\hat{\mathbf{f}}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta}; i, j = 1, 2...n) = \mathbf{f}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta}; i, j = 1, 2...n) \tag{1.15}$$

$$\hat{\mathrm{g}}(\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}) \tag{1.16}$$

Equating 1.1 and 1.12 would then give

$$\mathbf{J}(\boldsymbol{\theta})\left\{(\ddot{\boldsymbol{\theta}}_d - \ddot{\boldsymbol{\theta}} + \mathrm{k}_v(\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}) + \mathrm{k}_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}))\right\} = 0 \tag{1.17}$$

Substituting $e = \boldsymbol{\theta}_d - \boldsymbol{\theta}$ and noting that $\mathbf{J}(\boldsymbol{\theta})$ is nonsingular yields

$$\ddot{\mathbf{e}} + \mathrm{k}_v\dot{\mathbf{e}} + \mathrm{k}_p\mathbf{e} = 0 \tag{1.18}$$

The characteristic roots of (1.18) can be assigned to have negative real parts through selection of appropriate gains k_v and k_p hence the trajectory error $\mathbf{e}$ will approach zero asymptotically.

This form of controller is known as non-linearity cancellation because, if the model is exact, the non-linear system equations (1.1) can be reduced to the set of decoupled linear equations (1.18) to which standard control techniques may be applied.

A dis-advantage of computed torque is that the input torque τ is calculated by the model using inputs based on the actual trajectory and not just the pre-planned trajectory. Consequently, the model must be run on-line thus reducing the sample rate for a given control computer. As the speed and capability of computing hardware and software increase, however, this drawback should become less significant.

An *et al* [27] [2] performed a comparison of independent joint control, feed-forward control and computed torque. They found that both model-based controllers considerably improved trajectory tracking over independent joint control but that there was no significant difference between the accuracy of the feed-forward and computed torque controllers. This finding was also supported by Khosla [28]. Leahy *et al* [29] evaluated the performance of computed-torque in controlling a PUMA-600 robot but found that simulation results did not accurately predict the real performance of the manipulator.

1.2 Organisation of Thesis.

The layout of the thesis is as follows.

Chapter 2 outlines the basic concepts of bond graphs and how they may be used to model rigid, planar, revolutionary joint manipulators. Using this generic method, the specific bond graph for a horizontal, rigid, planar two-link manipulator, driven directly by voltage controlled d.c.motors, is derived from which the dynamic equations of motion are generated automatically.

Chapter 3 gives details of the construction of the experimental two-link

manipulator used as the test bed for implementation of the results of the research. In addition to giving the physical description of the capabilities and specifications of the motors, amplifiers and links, the chapter also explains the data acquisition and computing facilities used to control the manipulator.

The bond graph model of the experimental manipulator is validated in chapter 4 by comparing simulations using the model with data gathered from the manipulator. The simulations are run with various joint friction models, the numerical parameters of which are identified from the measured data using least-squares identification routines.

In chapter 5, the observer form of the two-link manipulator bond graph is constructed. The software required to implement a full order non-linear model based observer is then produced automatically from this bond graph and used to replace the poorly conditioned measurements of link angular velocity with observed velocities in a standard independent joint feedback controller.

The inputs and outputs of the manipulator bond graph are modified in chapter 6 to produce the inverse model bond graph from which the equations required to implement an 'inverse model' type controller can be deriven automatically. These equations are implemented for the experimental manipulator in the form of a 'computed torque' controller. The performance of the manipulator controlled by the computed torque controller is compared with its performance using standard independent joint controllers.

Chapter 7 concludes the thesis and suggests avenues for further research.

Chapter 2

Derivation of the Bond Graph for the Experimental Two-Link Manipulator.

2.1 Introduction

Bond graphs provide a format for the graphic representation of dynamic energy-exchanging systems. This format is particularly suited for modelling robotic manipulators as they are predominantly electro-mechanical devices with mechanical links being driven by electric motors. As bond graphs deal with energy exchange, it is possible to represent the electrical part of the system, dealing with currents and voltages, and the mechanical part, dealing with torques and angular velocities, in one all-encompassing model. It is, of course, also possible to model hydraulically powered manipulators in a similar way.

The graphical nature of bond graphs allows us to augment large systems by joining together a set of sub-systems. This provides a methodical, step by step approach to modelling large systems without getting entrenched in the inherent complexity of such systems.

The real power behind the bond-graph approach, however, lies in the fact that a causally complete bond-graph provides an unambiguous system representation from which other representations may automatically be derived by computer. To this end, a bond graph toolbox, Model Transformation Tools (MTT), has been developed by Gawthrop [30] to take system bond graphs in a graphical format and transform them into number of different representations such as the set of state-space matrices or the set of differential algebraic equations etc. MTT makes use of the language PROLOG to logically decipher a bond-graph, and REDUCE to symbolically manipulate the resulting algebra. The package is also capable of producing system equations in a human readable format and of producing simulation software.

To summarise, the advantages of using bond graphs to model mechanical manipulators are:

- the ability to construct large, complex systems by joining together a series of relatively simple sub-systems.
- the ability to represent different physical domains in the same graphical model.
- ease of transformation between different system representations.
- the ability to provide unambiguous representations of physical systems.
- automatic derivation of complex equations of motion.

The main dis-advantage of using bond graphs is that, for most people, a new non-intuitive modelling technique must be learnt.

This chapter outlines the derivation of the bond graph for the experimental two-link manipulator (DD2lm). A brief background to the theory and practice of bond graph modelling is given before the generic technique for modelling planar rigid manipulators, developed by Gawthrop [17] [31], is explained. Finally, the

construction of the specific bond graph for the DD2lm is outlined incorporating bond graphs for d.c.motor actuators.

2.2 Bond Graph Modelling.

This section describes what bond graphs are and how they may be used to model dynamic systems. More comprehensive texts are given by Rosenberg and Karnopp [32] and Wellstead [33].

2.2.1 Bonds.

The basic element of bond graphs is the energy bond (see figure 2.1). Its main property is that it represents two variables; an effort and a flow variable, the product of which is power. The arrow on the bond denotes the direction of positive energy flow.

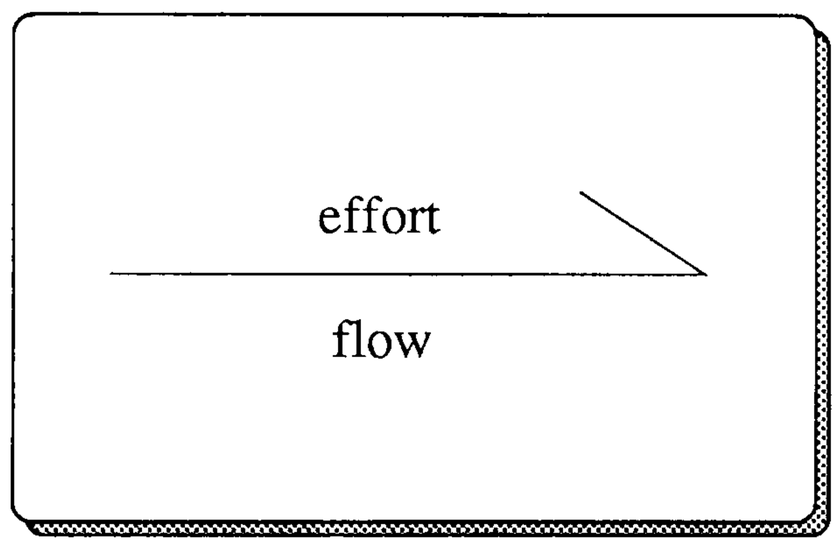


Figure 2.1: Energy Bond.

The physical meaning of the effort and flow variables depends upon the physical domain the bond represents. For example, in the electrical domain the effort variable is voltage and the flow variable is current. Voltage, or more accurately potential difference, multiplied by current gives the electrical power transferred by this bond. Table 2.2.1 gives examples of the effort and flow variables for a range of physical domains.

Domain	e (effort)	$\int e dt$	f (flow)	$\int f dt$
Mech.Translation	force	momentum	velocity	displacement
Mech.Rotation	torque	ang. momentum	ang. velocity	angle
Electrical	voltage	flux	current	charge
Hydraulic	pressure	fluid momentum	volumetric flow	volume

Table 2.1: Physical Meaning of Variables in a range of Domains.

2.2.2 Components.

Bonds are used to connect components. There are four types of components labelled **S,C,I,R** (see figure 2.2).

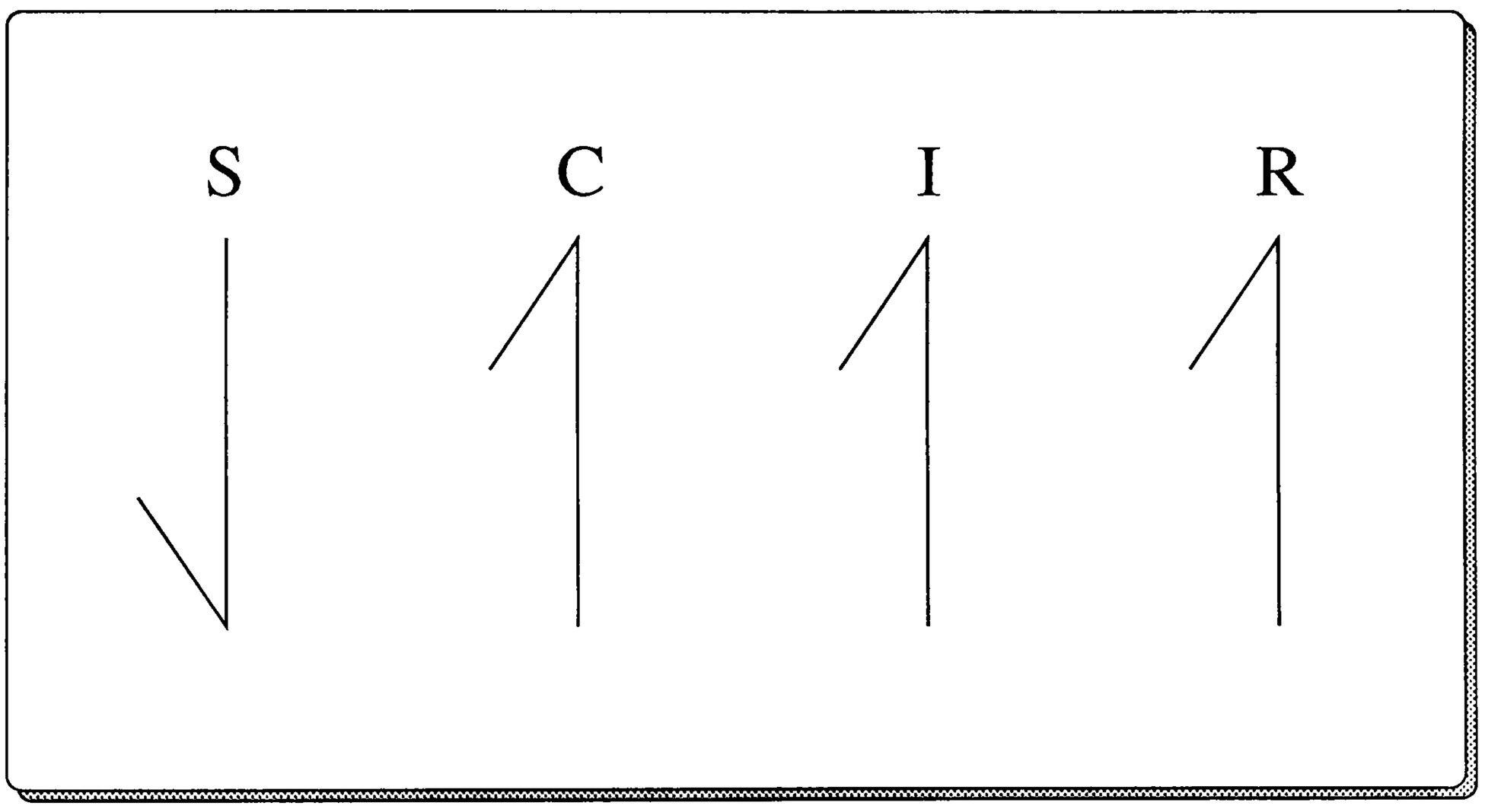


Figure 2.2: Bond Graph Components.

Again, the physical significance of the components depend upon the physical domain in which they operate but in general they can be remembered as **S**ource, **C**ompliance, **I**nertia and **R**esistance. Table 2.2.2 lists the actual meanings for a numbers of domains.

The components define how the effort and flow variables on the bond relate to each other. These relationships are known as the *Constitutive Laws* and have a specific form for each type of component as shown in figure 2.3 in which F and G are general functions which may be non-linear.

Figure 2.4 gives an example of the constitutive law for each component.

Domain	**I**	**C**	**R**
Mech.Translation	Inertia	Compliance	Damper
Mech.Rotation	rot. Inertia	rot. Compliance	rot. Damping
Electrical	Inductor	Capacitor	Resistor
Hydraulic	Fluid Inertia	Capacitor	Flow Resistance

Table 2.2: Physical Meaning of Components in a range of Domains.

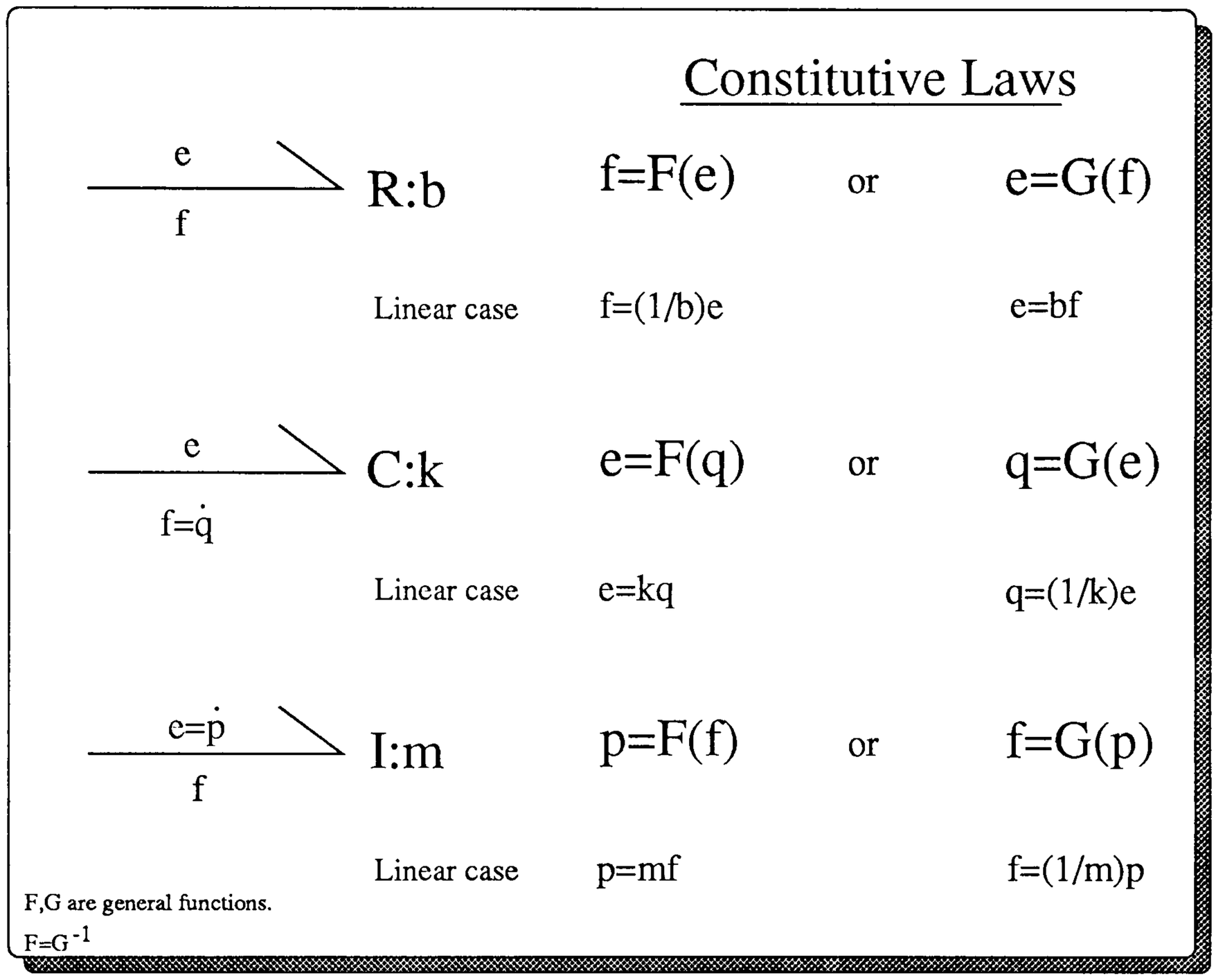


Figure 2.3: Component Constitutive Laws.

The source component provides a way of injecting energy into, or getting energy out of, a system. For example, electrical energy may be injected into an electrical circuit by the bond in figure 2.5.

Source components provide 'inputs' to the system.

2.2.3 Junctions.

Components are connected together using junctions. There are two types of junctions: a **0** or *common effort* junction and a **1** or *common flow* junction (see figure 2.6).

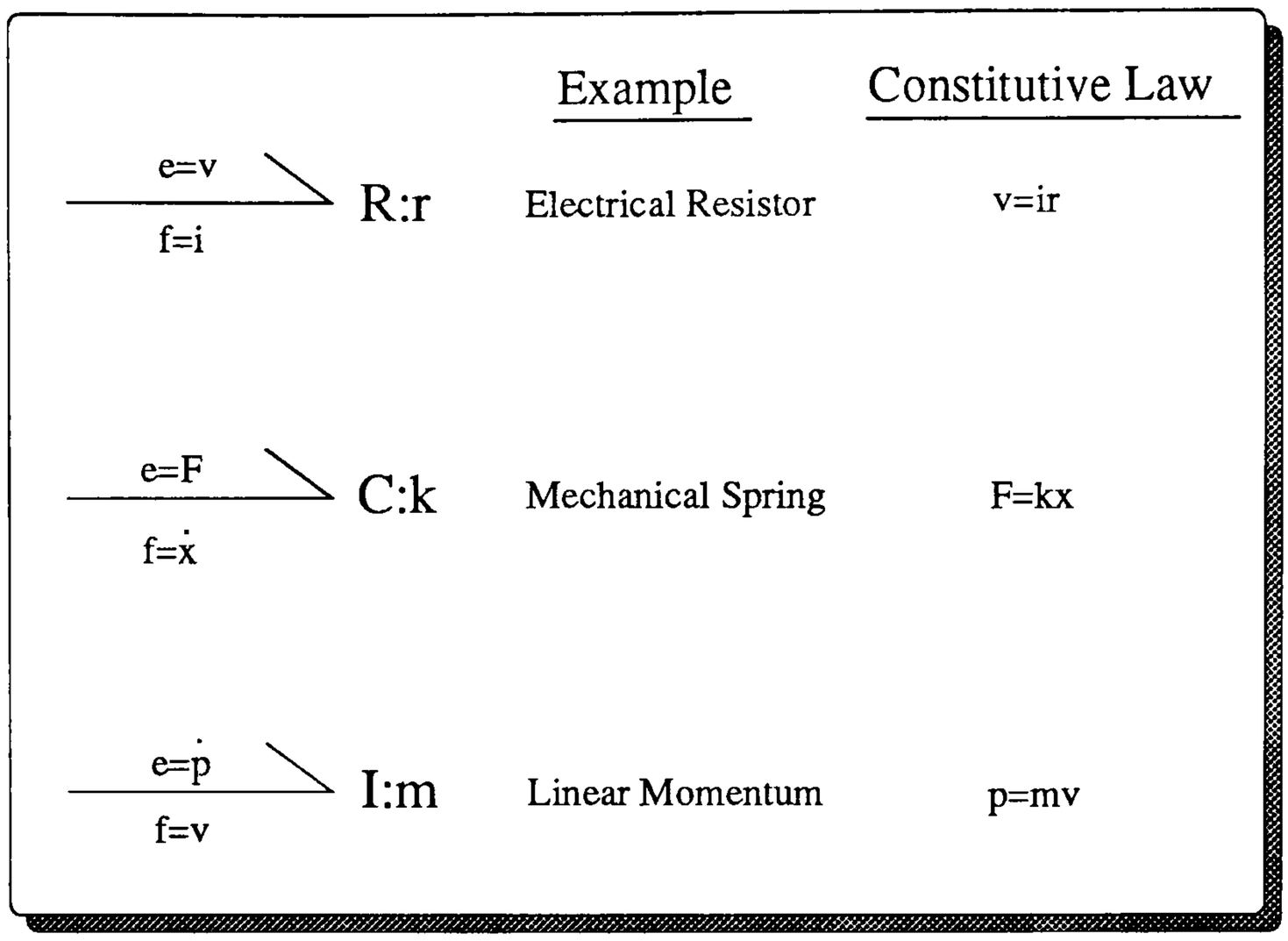


Figure 2.4: Examples of Component Constitutive Laws.

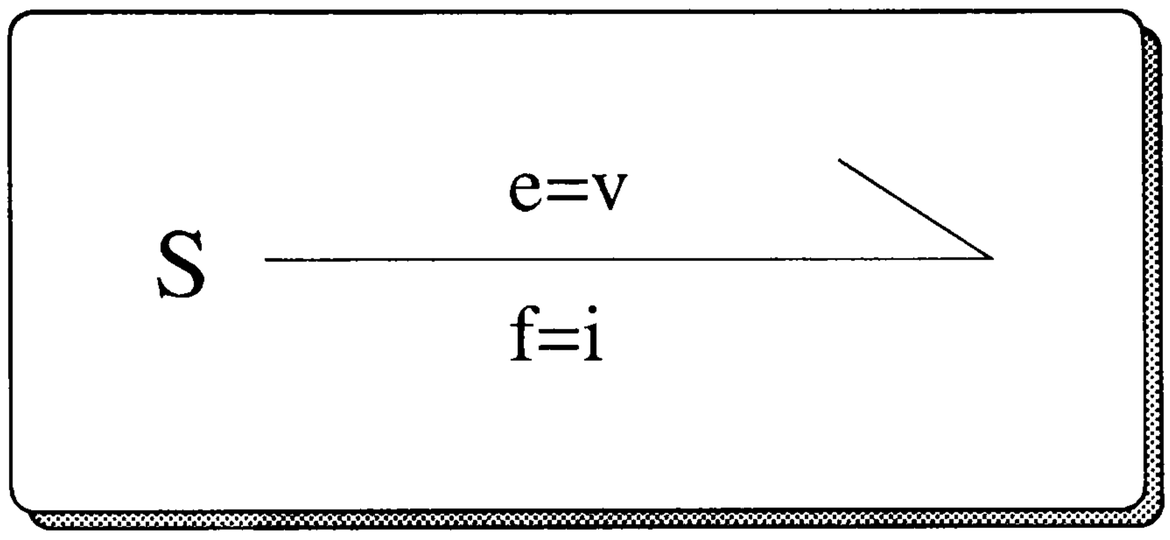


Figure 2.5: Source input.

The **0**, or common effort, junction has the following properties;

- all bonds impinging upon it have the same **effort** variable.
- all flows on attached bonds sum to zero.

With reference to figure 2.6 this results in

$$e_1 = e_2 = e_3 \tag{2.1}$$

$$f_1 + (-f_2) + (-f_3) = 0 \tag{2.2}$$

Similarly, the **1**, or common flow, junction has the properties:

- all bonds impinging upon it have the same **flow** variable.

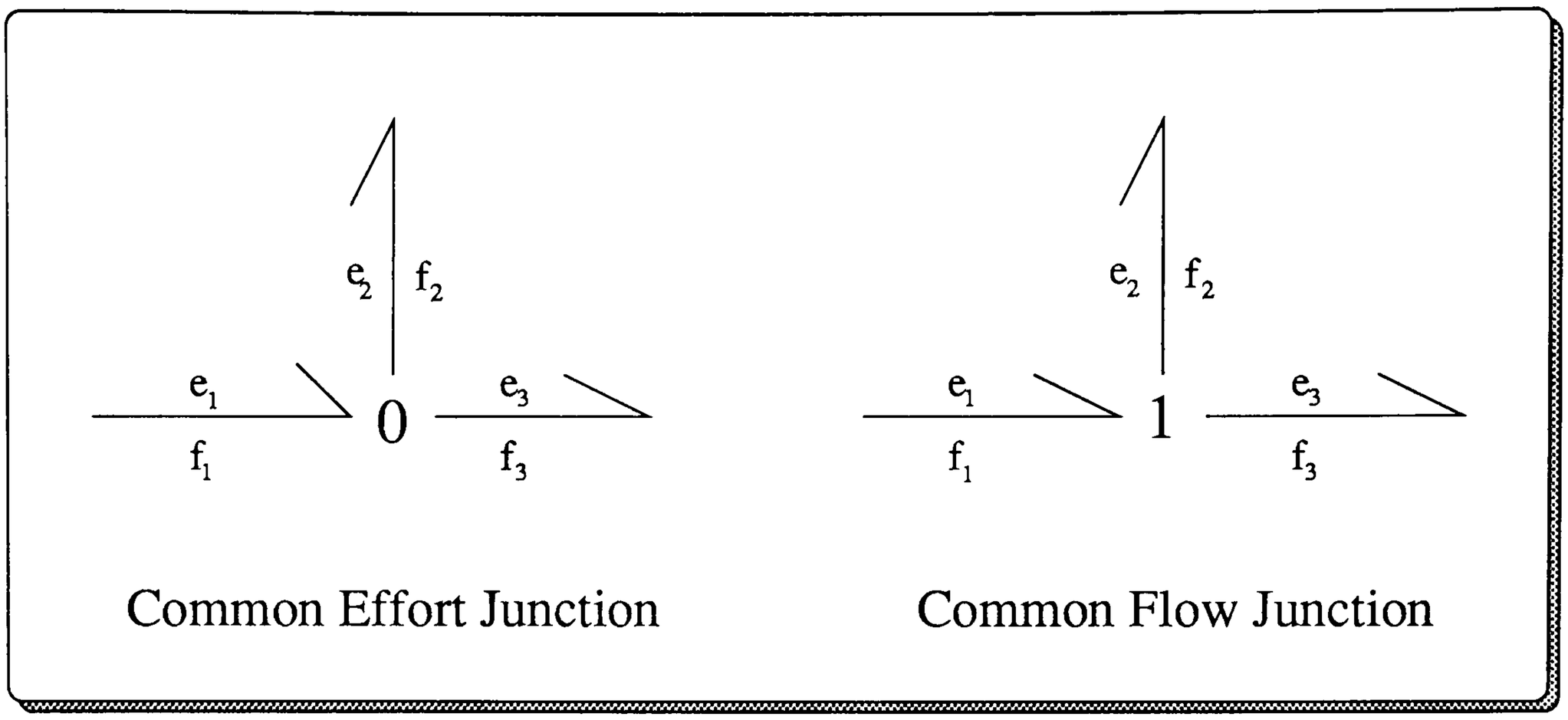


Figure 2.6: Junctions.

- all efforts on attached bonds sum to zero.

which results in

$$e_1 - e_2 - e_3 = 0 \tag{2.3}$$

$$f_1 = f_2 = f_3 \tag{2.4}$$

2.2.4 Connecting Physical Domains.

So far we have shown bonds, components and junctions and the laws by which they may be connected. Bond graphs containing only these elements would be constrained to only one physical domain, however, as the junctions allow only addition and subtraction of flows and efforts which must therefore be of the same physical units. To transfer between physical domains the ability to multiply must be included and bond graphs provide two means of accomplishing this: the *Transformer* and the *Gyrator* (see figure 2.7).

Note that the Gyrator and Transformer are energy conserving ($e_1 f_1 = \epsilon_2 f_2$).

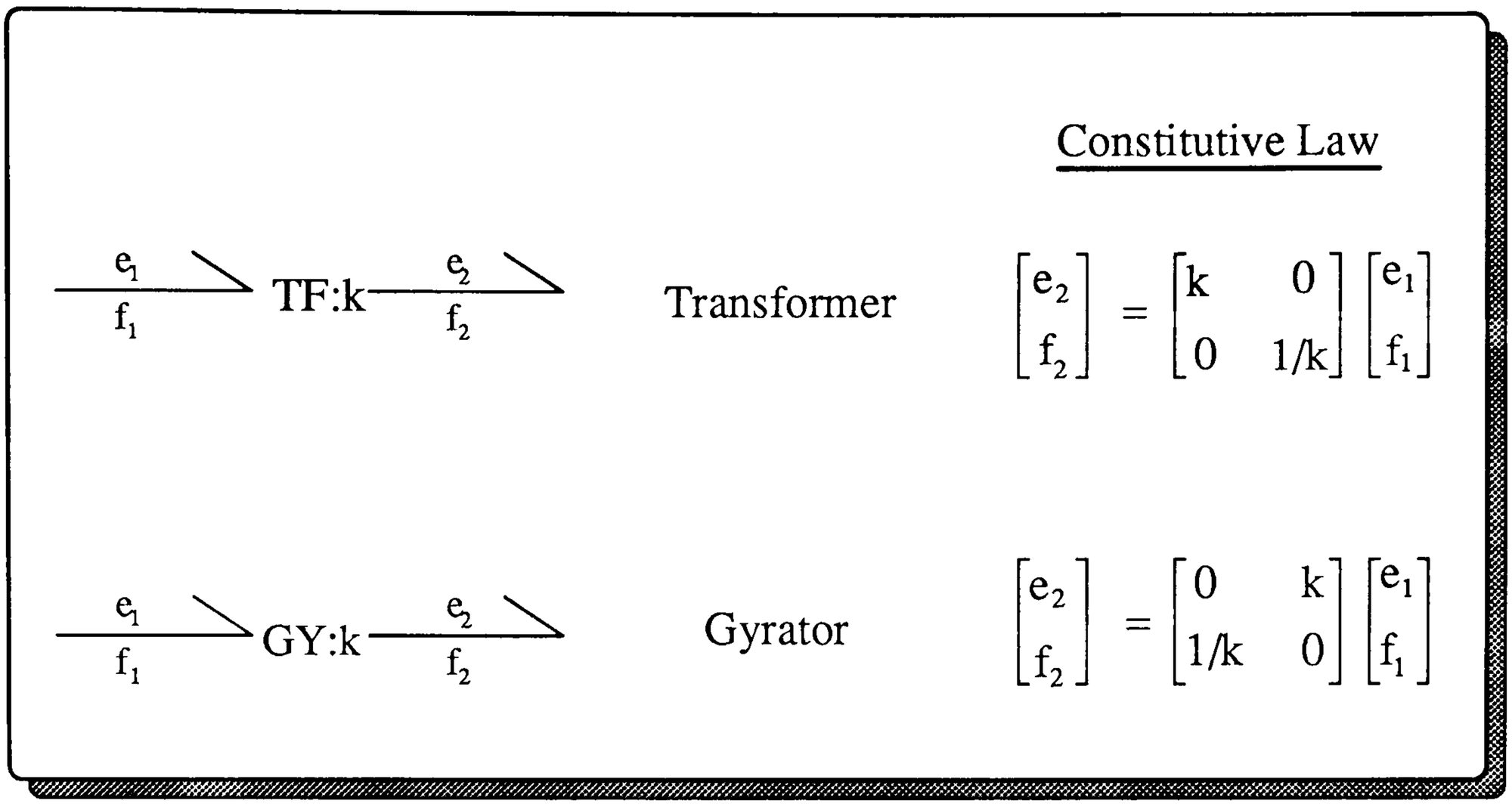


Figure 2.7: Elements to Couple Domains.

Modulated Transformers/Gyrators.

The multiplication factor k in figure 2.7 need not be a constant. It can be a function of some other variable(s) available within the bond graph in which case the transformation is said to be a *Modulated* Transformer or Gyrator.

2.2.5 Example: D.C. Motor.

As an example of how to represent a simple system we shall now consider how to model an ideal d.c.motor using bond graphs.

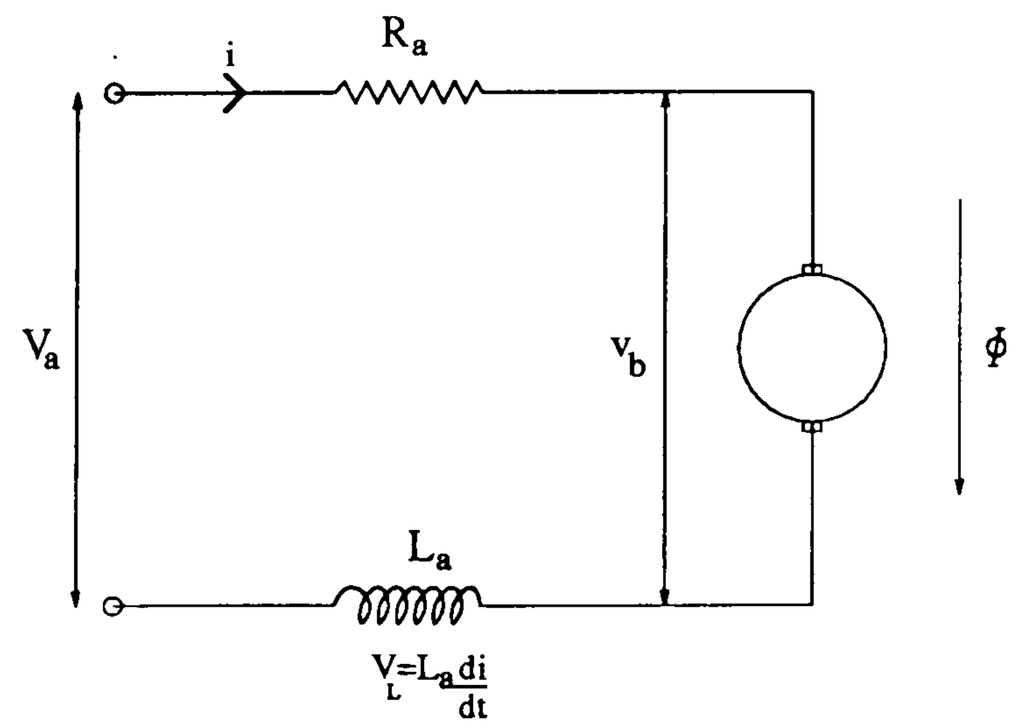


Figure 2.8: Armature Circuit for a D.C. Motor.

The armature circuit for a d.c.motor is shown in figure 2.8.

The variable common to all the components of the armature circuit is the current i. Consequently, the bond graph of the armature will be constructed around a common flow, or **1**, junction. Connected to this junction will be a **S**ource of either current or voltage, an **I**nertia corresponding to the armature inductance, L_a,and a **R**esistance corresponding to the armature resistance, R_a. The bond graph will therefore be as in figure 2.9.

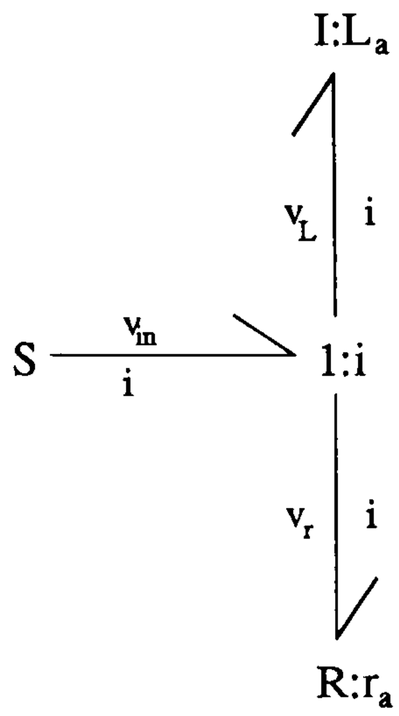


Figure 2.9: Bond Graph for Armature Circuit.

As the bond graph stands, and assuming linear components, the following equations are implied:

1 junction.

$$v_{in} - v_L - v_r = 0 \tag{2.5}$$

$$v_{in} = v_L + v_r \tag{2.6}$$

Inductance

$$v_L = L_a \frac{di}{dt} \tag{2.7}$$

Resistance

$$v_r = ri \tag{2.8}$$

These equations are as one would expect from figure 2.8 if the back e.m.f of the motor is ignored.

The torque produced by a d.c.motor is given by

$$\tau = k_2 i \tag{2.9}$$

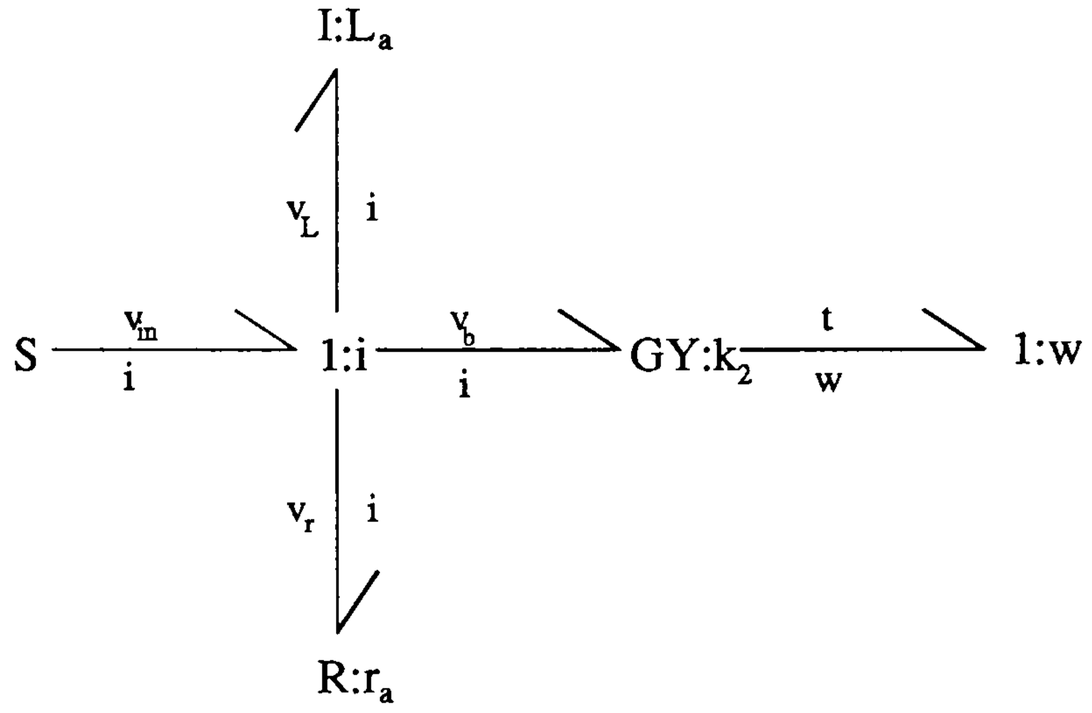


Figure 2.10: Bond Graph for D.C.Motor.

where k_2 is the torque constant of the motor. To implement this on the bond graph we note that τ, an effort variable represented by t on the bond graph, is produced through multiplication of i, a flow. This can be accomplished with the use of a gyrator as shown in figure 2.10.

Note that the addition of the motor characteristic implies the back e.m.f. voltage v_b such that

$$v_{in} = v_L + v_r + v_b \tag{2.10}$$

$$v_b = k_2\omega \tag{2.11}$$

where ω is the angular velocity of the motor rotor, represented by w on the bond graph.

The mechanical characteristics of the motor can now be added to the bond graph as a common flow junction corresponding to the angular velocity ω has been formed. The components to be added are the motor inertia J_m and the viscous friction, B, to form the final version of the bond graph shown in figure 2.11.

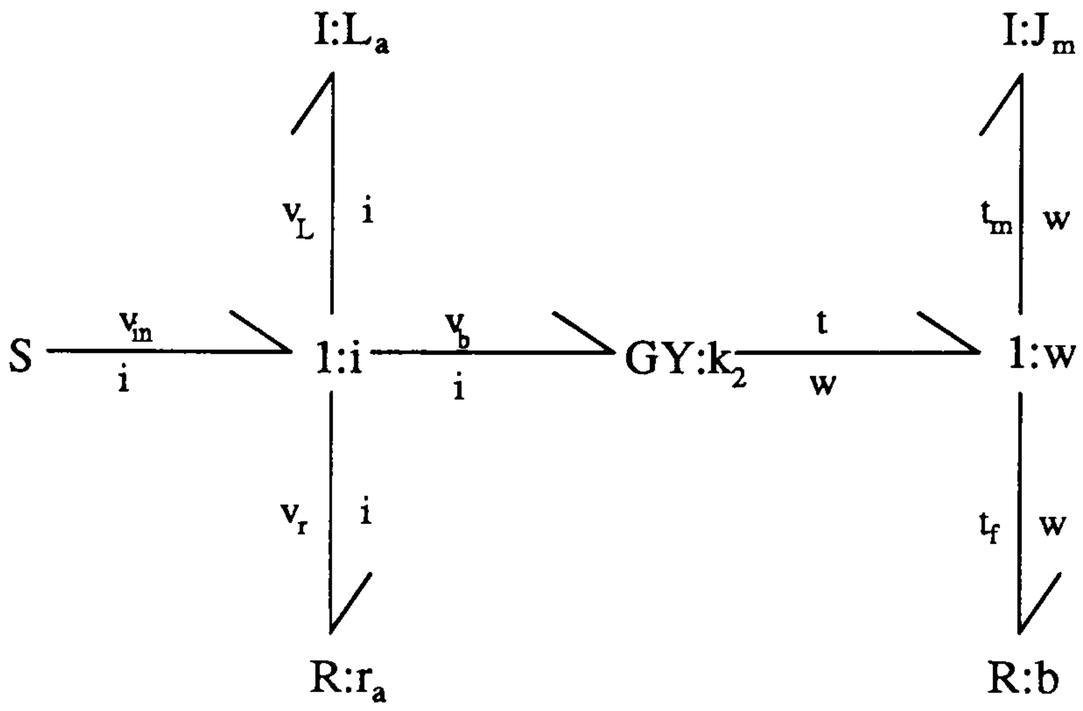


Figure 2.11: Complete Bond Graph for D.C.Motor.

The mechanical side of the bond graph implies the following equations

$$\tau - \tau_m - \tau_f = 0 \tag{2.12}$$

$$\tau = \tau_m + \tau_f \tag{2.13}$$

where

$$\tau_m = J_m \dot{\omega} \tag{2.14}$$

$$\tau_f = b\omega \tag{2.15}$$

The bond graph for the ideal d.c.motor is therefore complete. It is possible to account for non-ideal motor characteristics by modifying the constitutive equations of the components. For example, non-linear stiction could be incorporated into the model by modifying the linear constitutive equation 2.15 for viscous friction to the non-linear form for stiction friction given by

$$\tau_f = b\omega + k_s sign(\omega - |\omega|) \tag{2.16}$$

This does not alter the basic structure of the bond graph, only the constitutuve relationship for the $R : b$ element is changed. This illustrates the separation of system structure and component dynamics.

2.2.6 Causality.

Causality defines the cause and effect relationships within bond graphs. For example, in the bond graph of the d.c.motor it is not clear whether the source element is imposing a voltage or a current onto the armature circuit. For a voltage controlled motor the input would be a voltage; the current would then be dependent upon the other components in the armature circuit and the angular velocity of the motor. In this case, voltage would 'cause' current. This is handled within bond graph by assigning causal strokes to bonds.

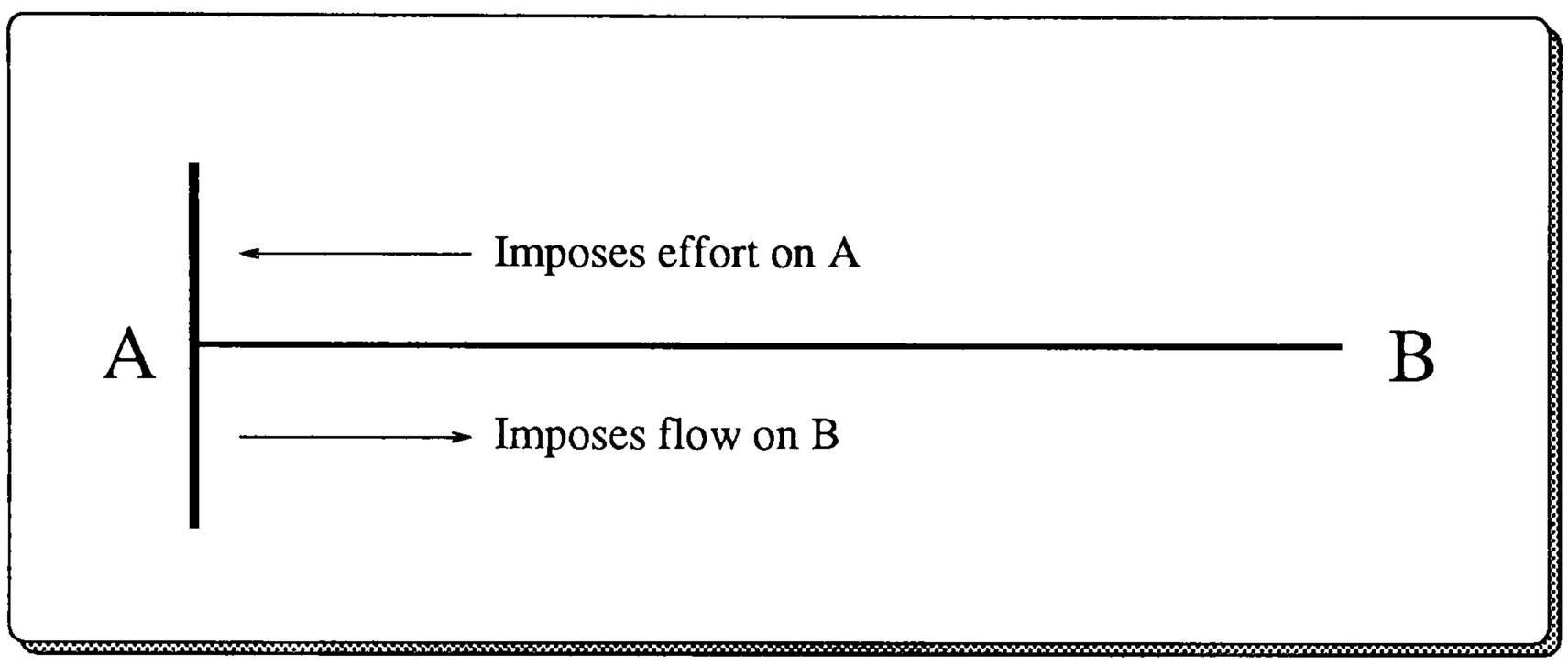


Figure 2.12: Causal Stroke.

The causal stroke imposes effort onto the side of the bond to which it is attached and, by implication, imposes flow onto the opposite end. Source components may therefore impose flow or effort into the system.

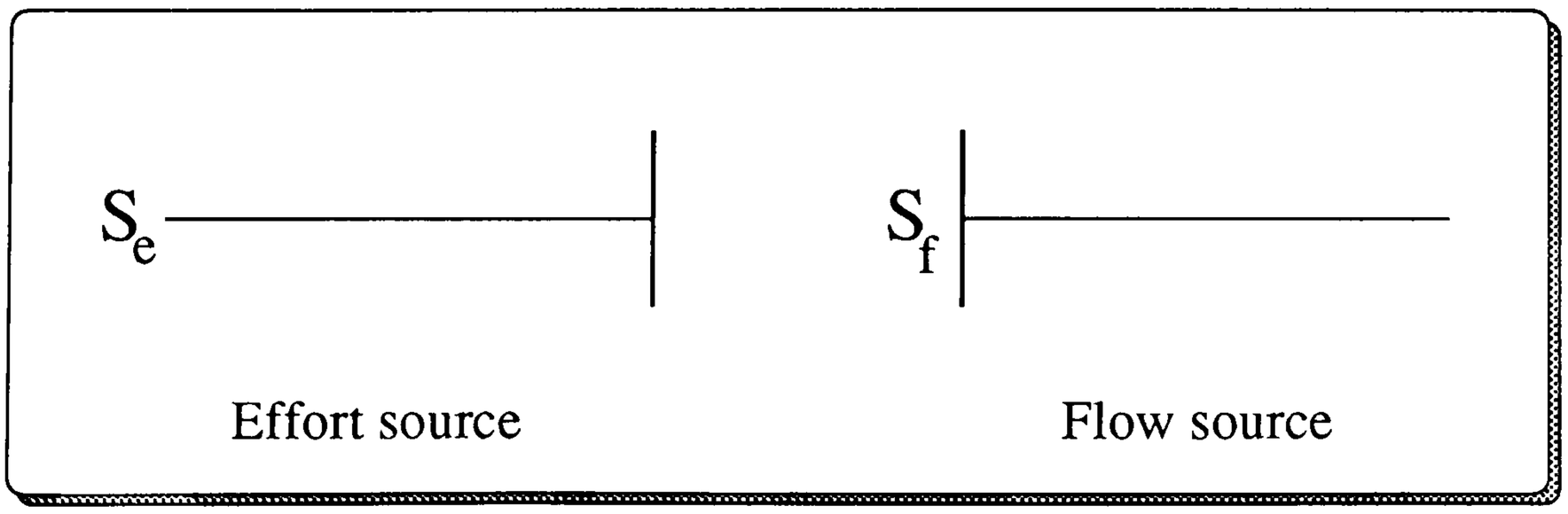


Figure 2.13: Source Elements.

The effect of causality on junctions is the following:

- one, and only one, bond must impose flow onto a common flow junction.
- one, and only one, bond must impose effort onto a common effort junction.

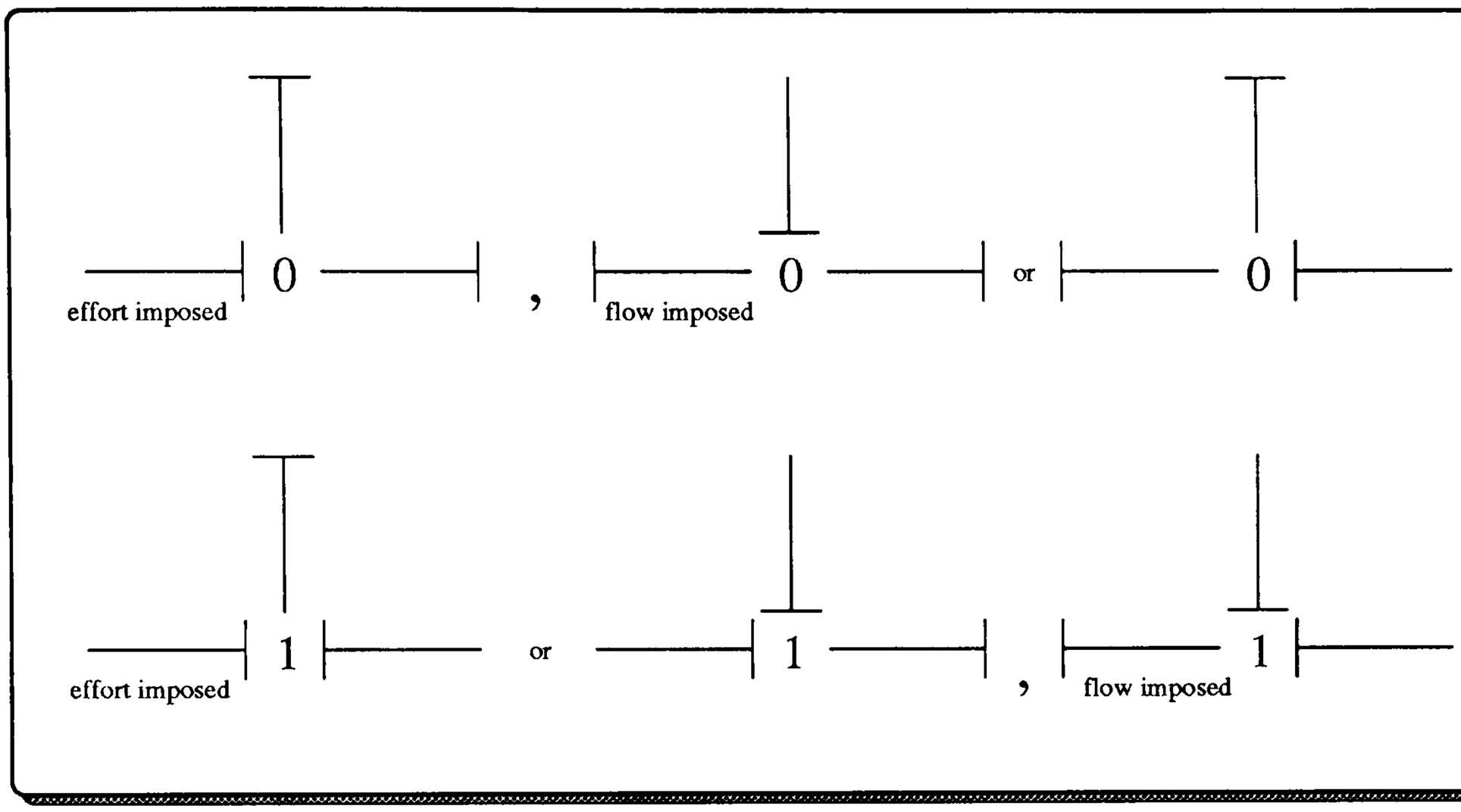


Figure 2.14: Junction Causality.

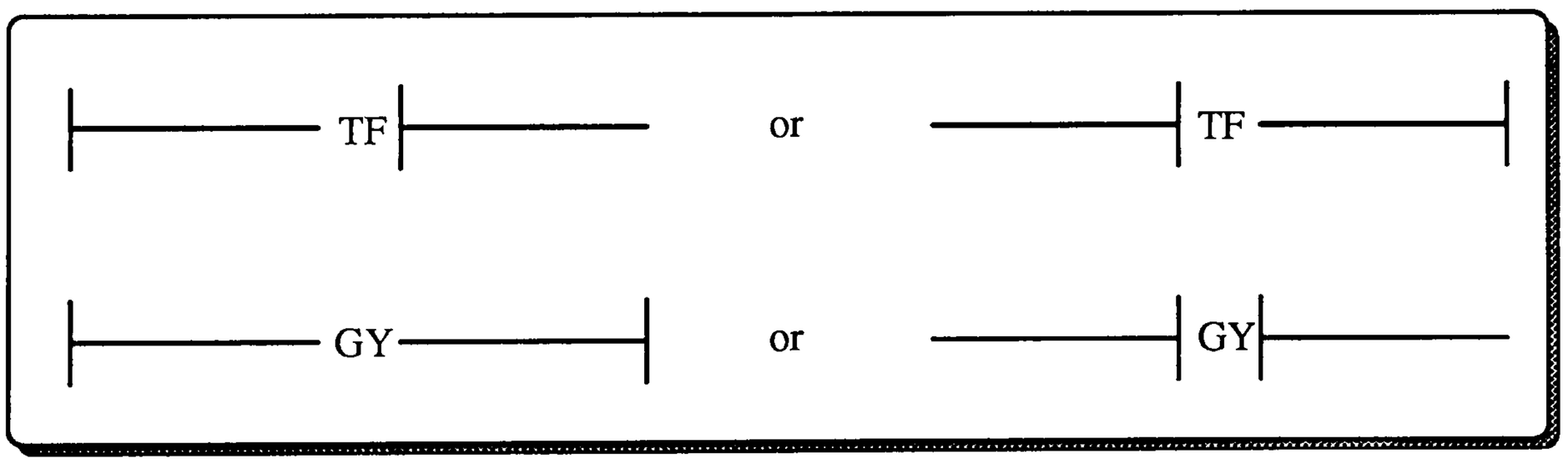


Figure 2.15: Transformer Causality.

Figures 2.14 and 2.15 show the permitted causality permutations for junctions and transformers respectively. It can be seen that causal strokes may, to a large extent, propagate automatically through a bond graph once the causality of the source components has been assigned. This mirrors the cause and effect relationships which exist within systems.

Applying a current source to the d.c.motor bond graph causes causality to propagate through the bond graph a shown in figure 2.16. The bond graph is

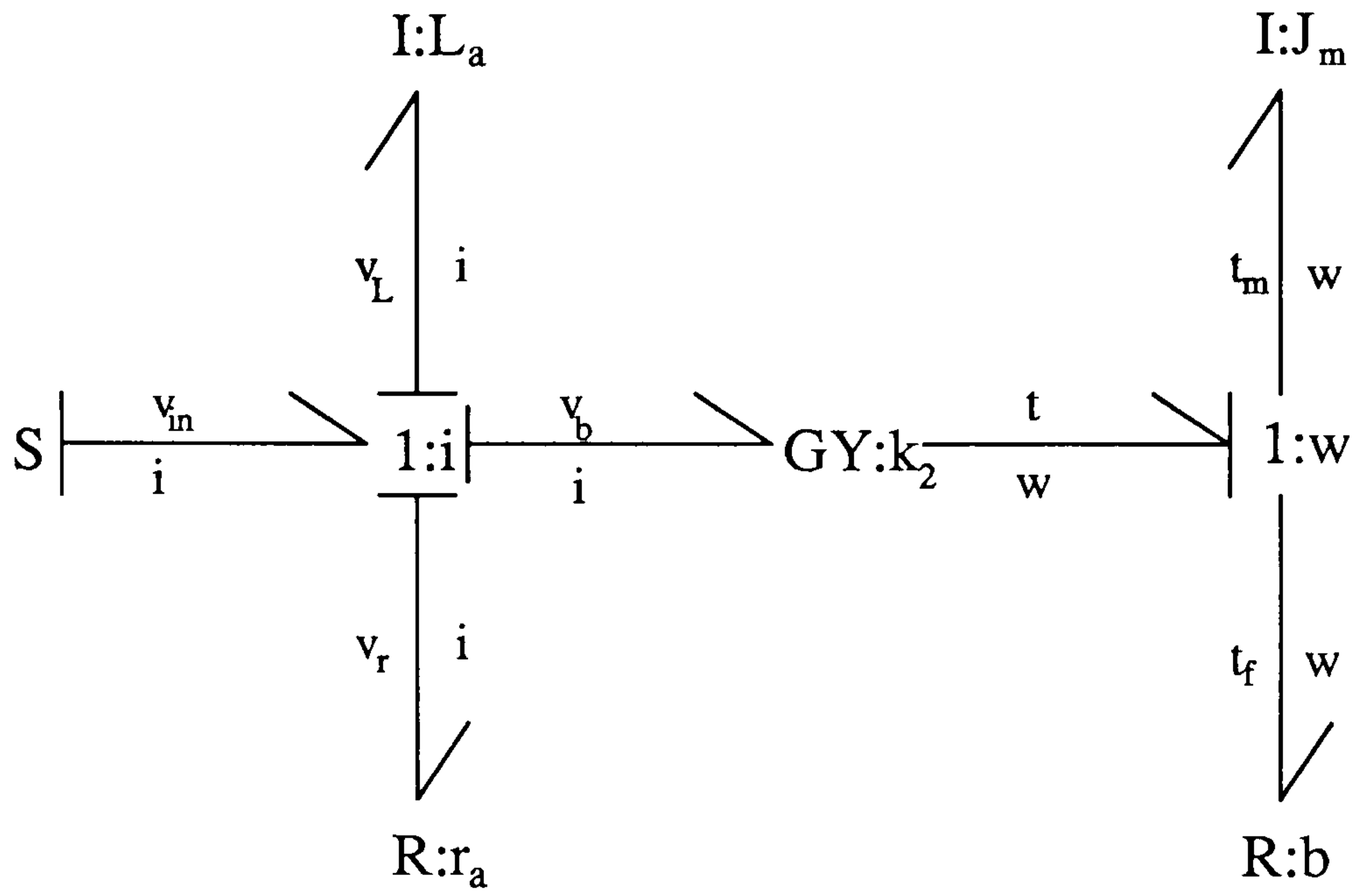


Figure 2.16: Current Sourced D.C.Motor.

not yet causally complete as the motor inertia and rotor friction components have not yet been causally assigned. One of these components must impose a flow, the angular velocity ω, onto the junction but according to the rules of causality it may be either one.

The purpose of creating bond graphs is to be able to automatically create the dynamic system equations from the causally complete bond graph. The creation of these equations is made simpler if the constitutive equations for the components are of the form

$$f = F(p) \tag{2.17}$$

or

$$e = F(q) \tag{2.18}$$

This form is known as *Integral Causality* and is imposed when **I**nertia and Compliance components have the form as shown in figure 2.17.

Hence, in a bond graph, if there is a choice we always assign **I** and **C** elements to have integral causality. Resistance components have arbitrary causality as the constitutive relationship is not a differential equation.

Following this convention, the current sourced d.c.motor has the causally complete bond graph as shown in figure 2.18 and the voltage controlled d.c.motor

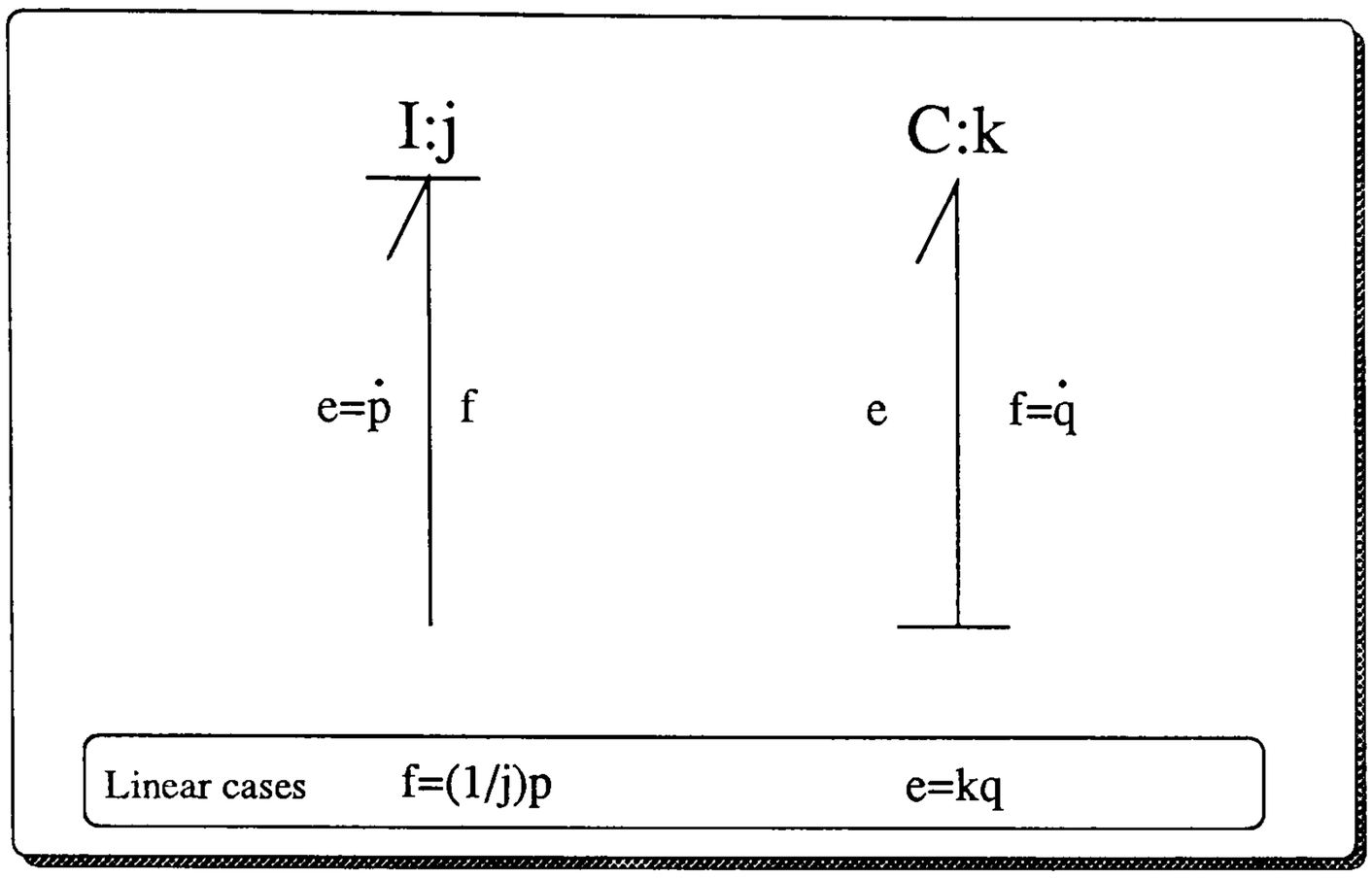


Figure 2.17: Integral Causality.

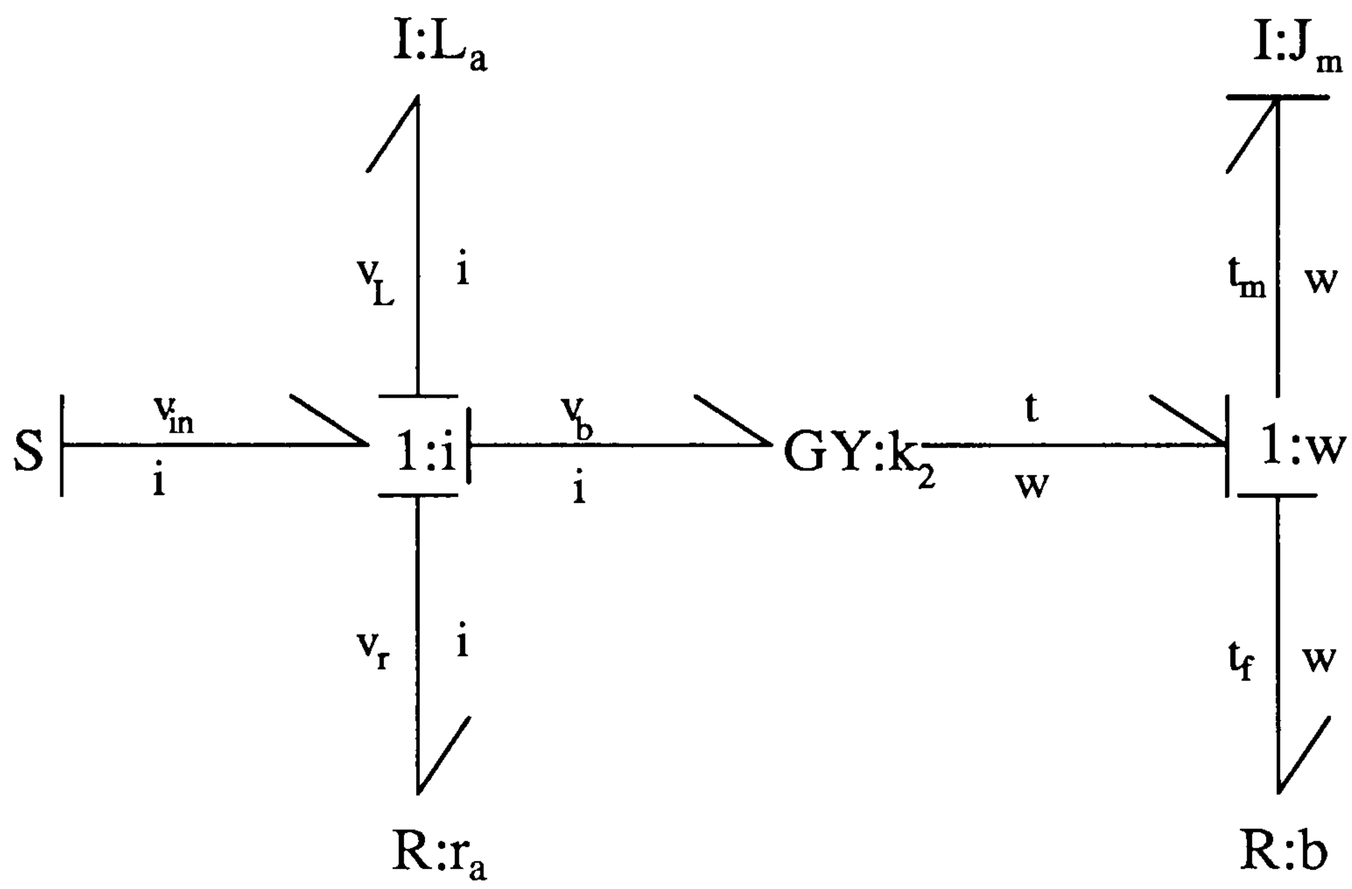


Figure 2.18: Causally Complete Current Sourced D.C.Motor.

has the causally complete bond graph as shown in figure 2.19.

States.

The states of a bond graph can now be defined as:

- the integral of the *effort* variable ($p = \int e dt$) on **I** elements which have integral causality.

- the integral of the *flow* variable ($q = \int f dt$) on **C** elements which have integral causality.

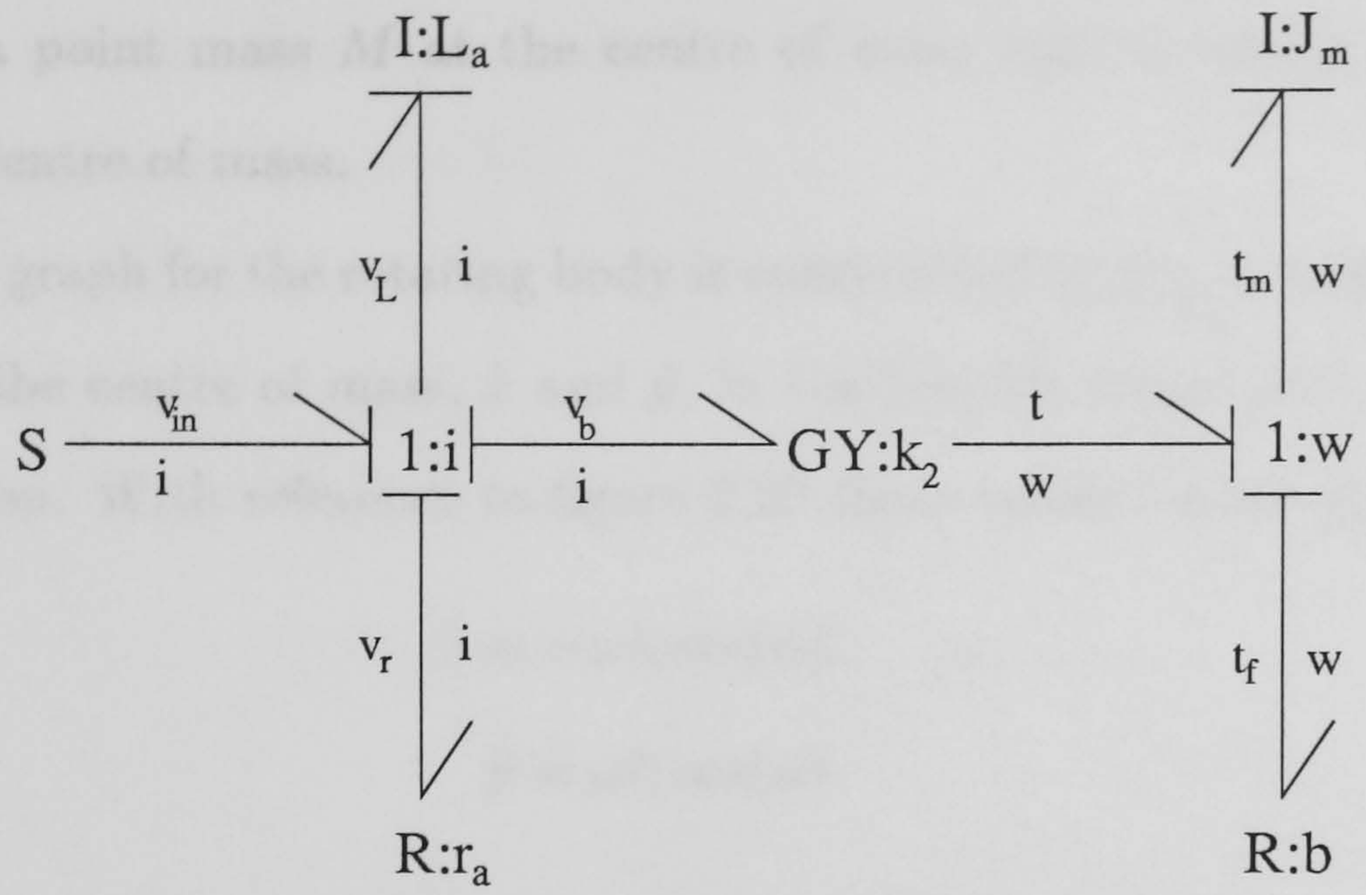


Figure 2.19: Causally Complete Voltage Sourced D.C.Motor.

2.3 Modelling Planar, Rigid Manipulators.

2.3.1 Planar Motion.

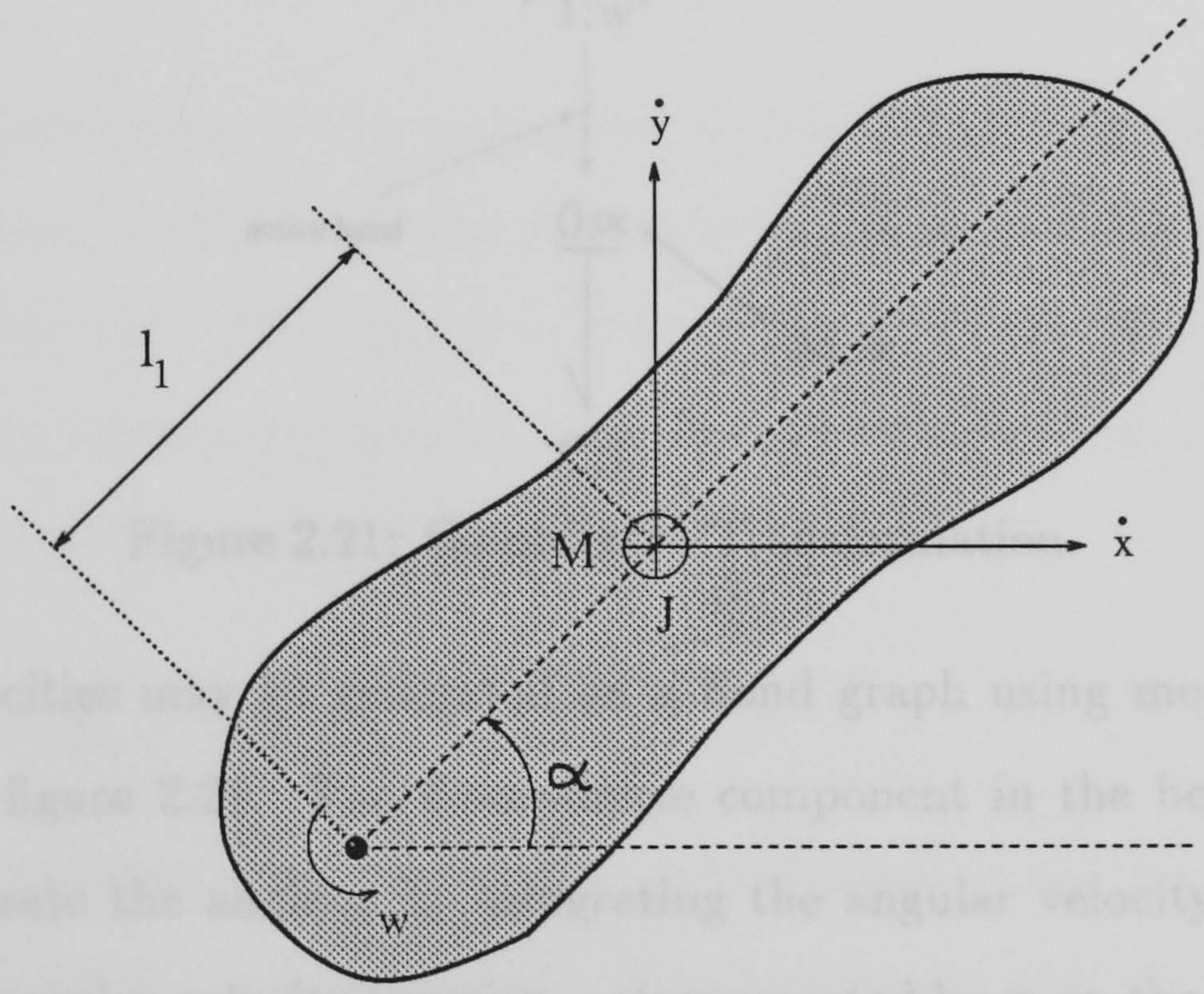


Figure 2.20: General Rotating Body.

The basis for modelling planar, rigid manipulators lies in modelling planar rotation of rigid bodies. The modelling of mechanical systems using bond graphs is explored by Tiernego and Bos [34]. Figure 2.20 shows a general body rotating in a plane around a pivot point at angular velocity ω. The body can be thought

of as being a point mass M at the centre of mass and as having an inertia J around this centre of mass.

The bond graph for the rotating body is constructed by generating the absolute velocities of the centre of mass, $\dot{x}$ and $\dot{y}$, in the inertial frame with origin at the axis of rotation. With reference to figure 2.20 these velocities are given by

$$\dot{x} = -\omega l_1 sin(\alpha) \tag{2.19}$$

$$\dot{y} = \omega l_1 cos(\alpha) \tag{2.20}$$

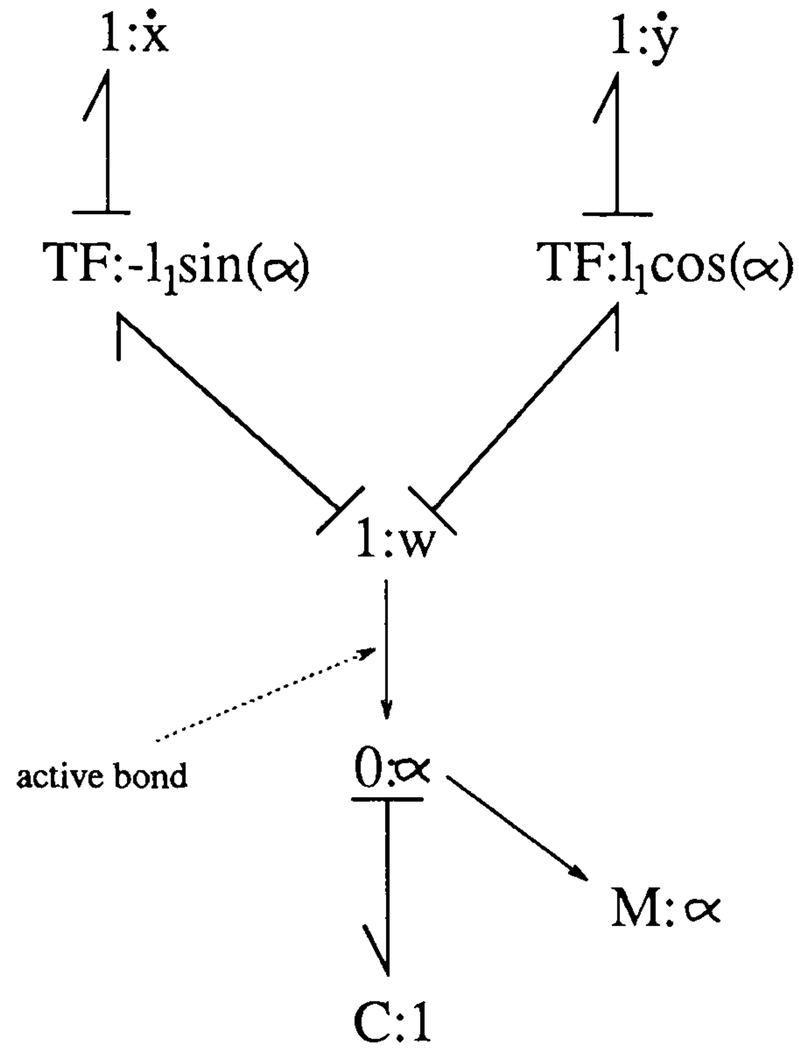


Figure 2.21: Co-ordinate Transformation.

These velocities may be generated on a bond graph using modulated transformers as in figure 2.21. The Compliance component in the bond graph is a 'trick' to generate the angle α by integrating the angular velocity ω. It is connected to the angular velocity junction, ω (represented by w on the bond graphs), by an *active bond* which conveys the angular velocity ω onto the **0** junction as a flow variable without having any effect on the angular velocity junction itself: it is a 'messenger'. The constitutive law of the compliance junction is

$$e = k \int f dt \tag{2.21}$$

$$\alpha = 1 \int \omega dt \tag{2.22}$$

The angle α is therefore made available for use in the modulated transformers using the measurement element **M**.

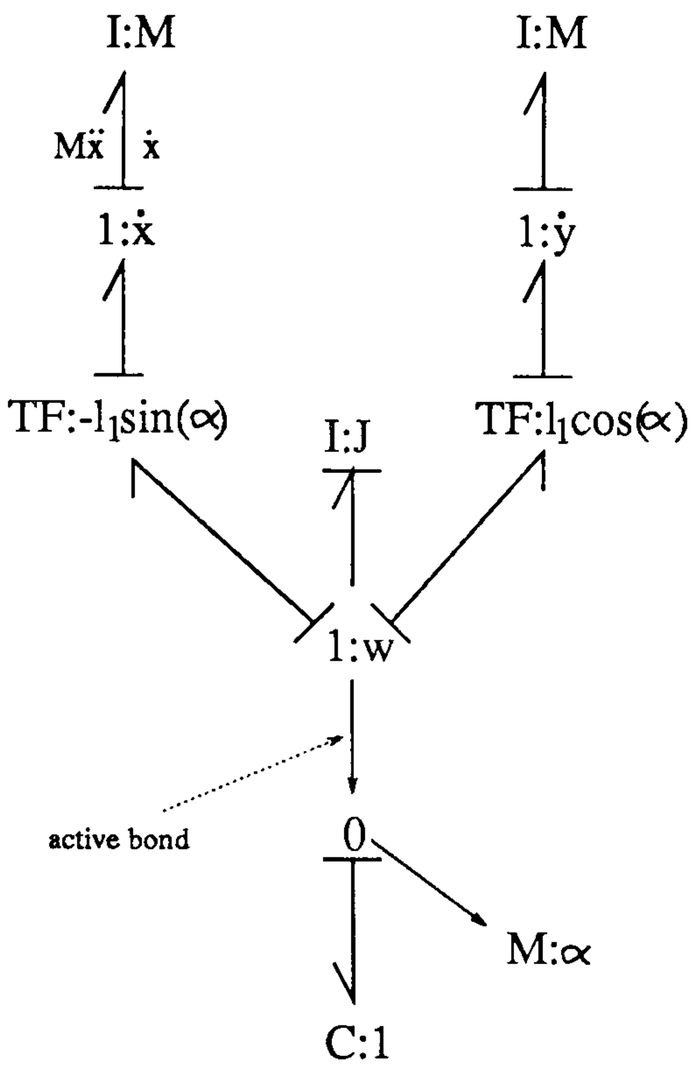


Figure 2.22: Bond Graph for Rotating Body.

Once the kinematics of the rotating body have been represented in the bond graph, the dynamics can be introduced simply by adding Inertial elements to the body as in figure 2.22. Note that once integral causality has been chosen for the angular velocity inertial component, the linear velocity inertial components are automatically set to have derivative causality.

2.3.2 Coupled Links.

Extending the bond graph to represent coupled links increases the complexity as the base of the second link is not fixed in space but depends on the velocity of its attachment point to the first link. As with the model of the simple body, the task is to find the velocities of the centres of mass of the links. From figure 2.23 it can be seen that

$$v_{x_1} = -\omega_1 l_{11} sin(\alpha_1) \tag{2.23}$$

$$v_{y_1} = \omega_1 l_{11} cos(\alpha_1) \tag{2.24}$$

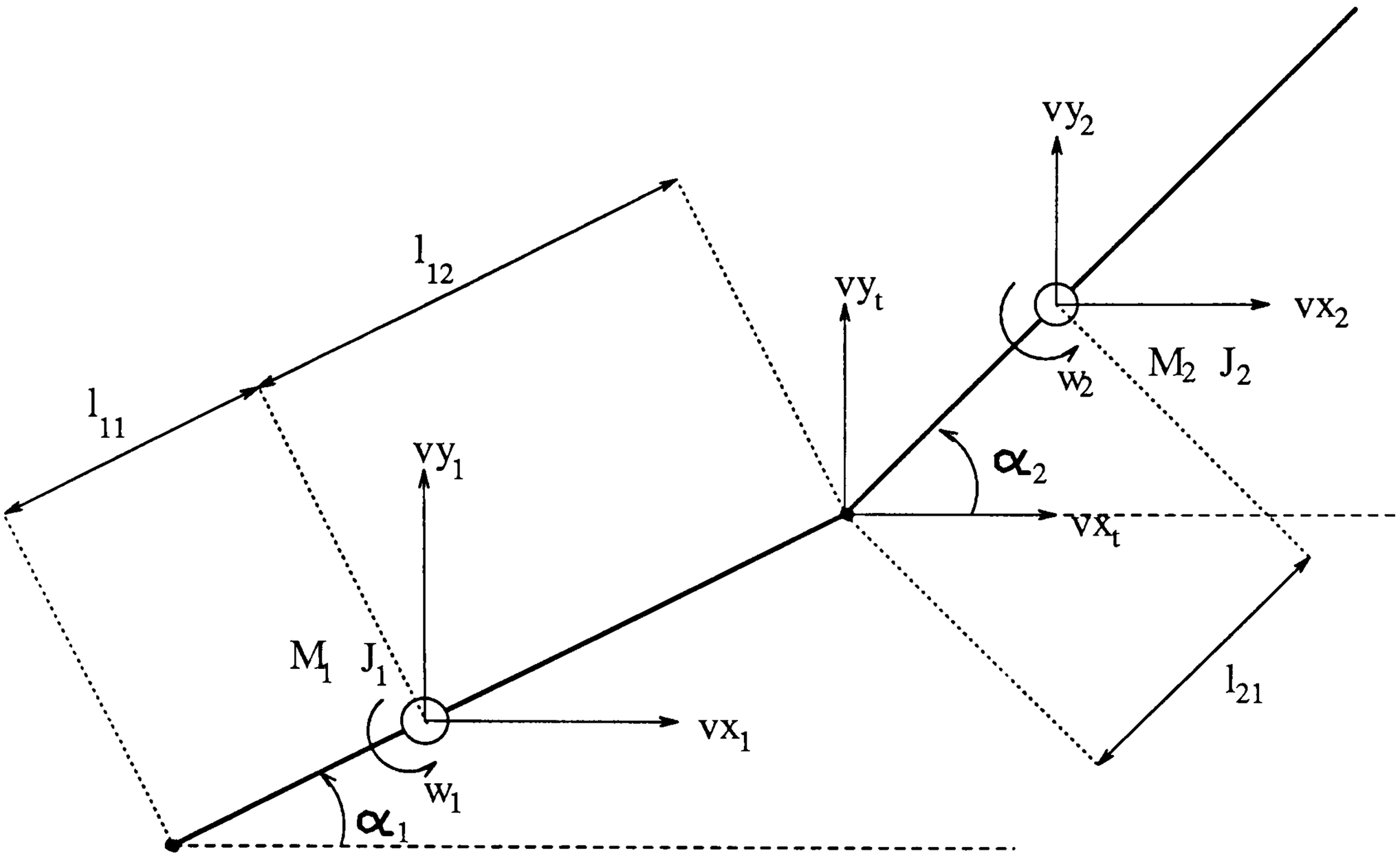


Figure 2.23: Coupled Links.

$$v_{x_t} = -\omega_1(l_{11} + l_{12})sin(\alpha_1) = v_{x_1} - \omega_1 l_{12} sin(\alpha_1) \tag{2.25}$$

$$v_{y_t} = \omega_1(l_{11} + l_{12})cos(\alpha_1) = v_{y_1} + \omega_1 l_{12} cos(\alpha_1) \tag{2.26}$$

$$v_{x_2} = v_{x_t} - \omega_2 l_{21} sin\alpha_2 = v_{x_1} - \omega_1 l_{12} sin(\alpha_1) - \omega_2 l_{21} sin(\alpha_2) \tag{2.27}$$

$$v_{y_2} = v_{y_t} + \omega_2 l_{21} cos\alpha_2 = v_{y_1} + \omega_1 l_{12} cos(\alpha_1) + \omega_2 l_{21} cos_(\alpha_2) \tag{2.28}$$

where ω_1,ω_2 are the absolute angular velocities of the first and second links.

Equations 2.27 and 2.28 hold the key to the development of the bond graph. From the graph of the single rotating body we already have the first terms of these equations v_{x_1} and v_{y_1}. The second terms of both equations may be generated from the angular velocity of the first link using modulated transformers with gains $-l_{12}sin(\alpha_1)$ and $l_{12}cos(\alpha_1)$. The third terms of the equations may be generated from the absolute angular velocity of the second link ω_2 by modulated transformers with gains $-l_{21}sin(\alpha_2)$ and $l_{21}cos(\alpha_2)$. The three terms may then be

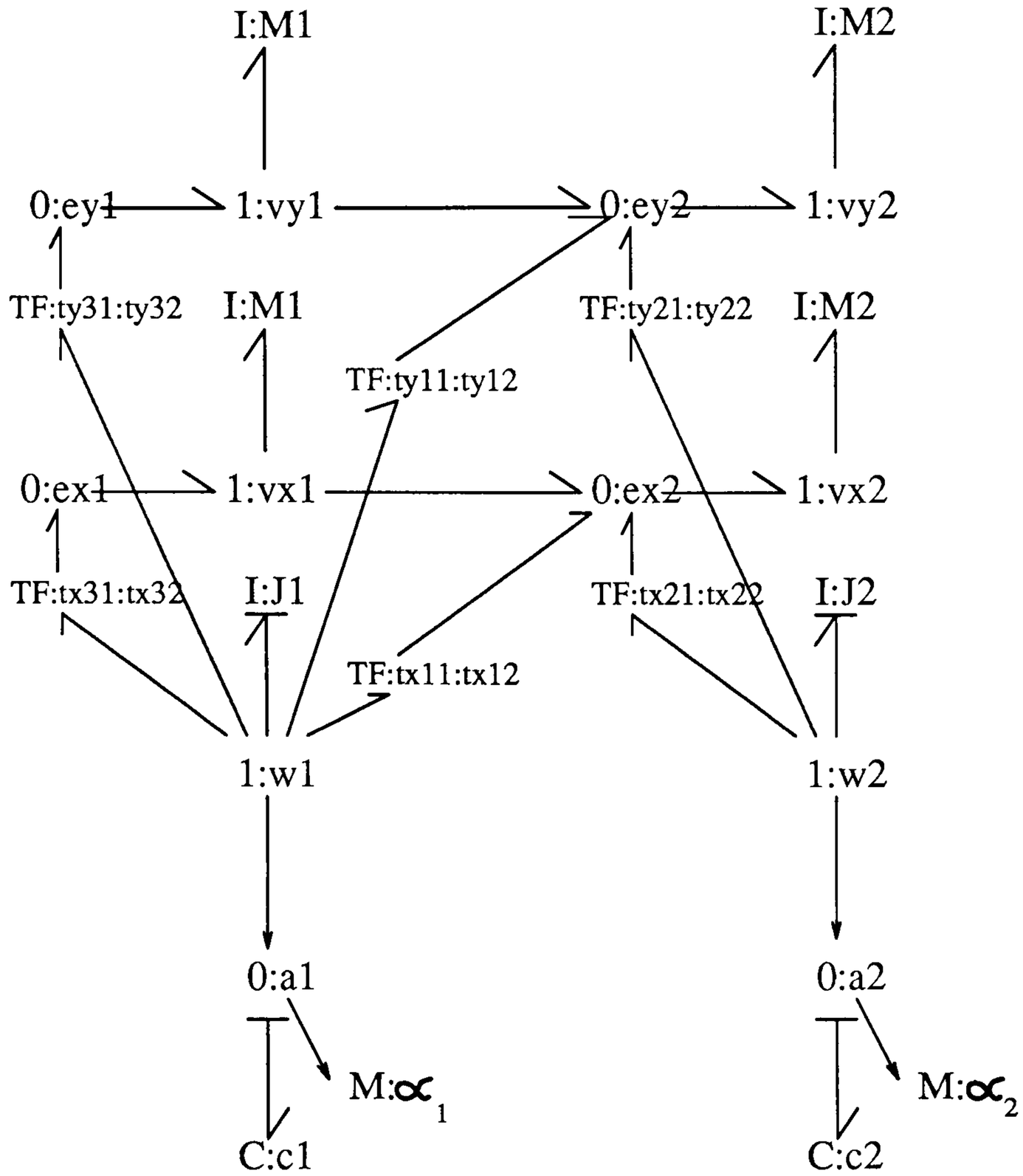


Figure 2.24: Bond Graph of Coupled Links.

added together using a **0** junction which sums flows, the result of which is passed onto a **1** junction which represents the velocities themselves. The resulting bond graph including inertial elements is shown in figure 2.24.

Figure 2.24 represents the free body version of the coupled links. To convert this into a two-link manipulator requires the addition of torques at the pivot points. The dynamics of an actuator mounted directly at the pivot would act on the relative velocities between the links and not the absolute velocities. For the first link, the relative and absolute velocities are the same but for the second link the relative velocity is the difference between the absolute velocities of the two links. This relative velocity may be incorporated into the bond graph through use of an extra **0** junction **et2** as in figure 2.25. The input torques may then be represented using effort **S**ources. This bond graph now represents the generic

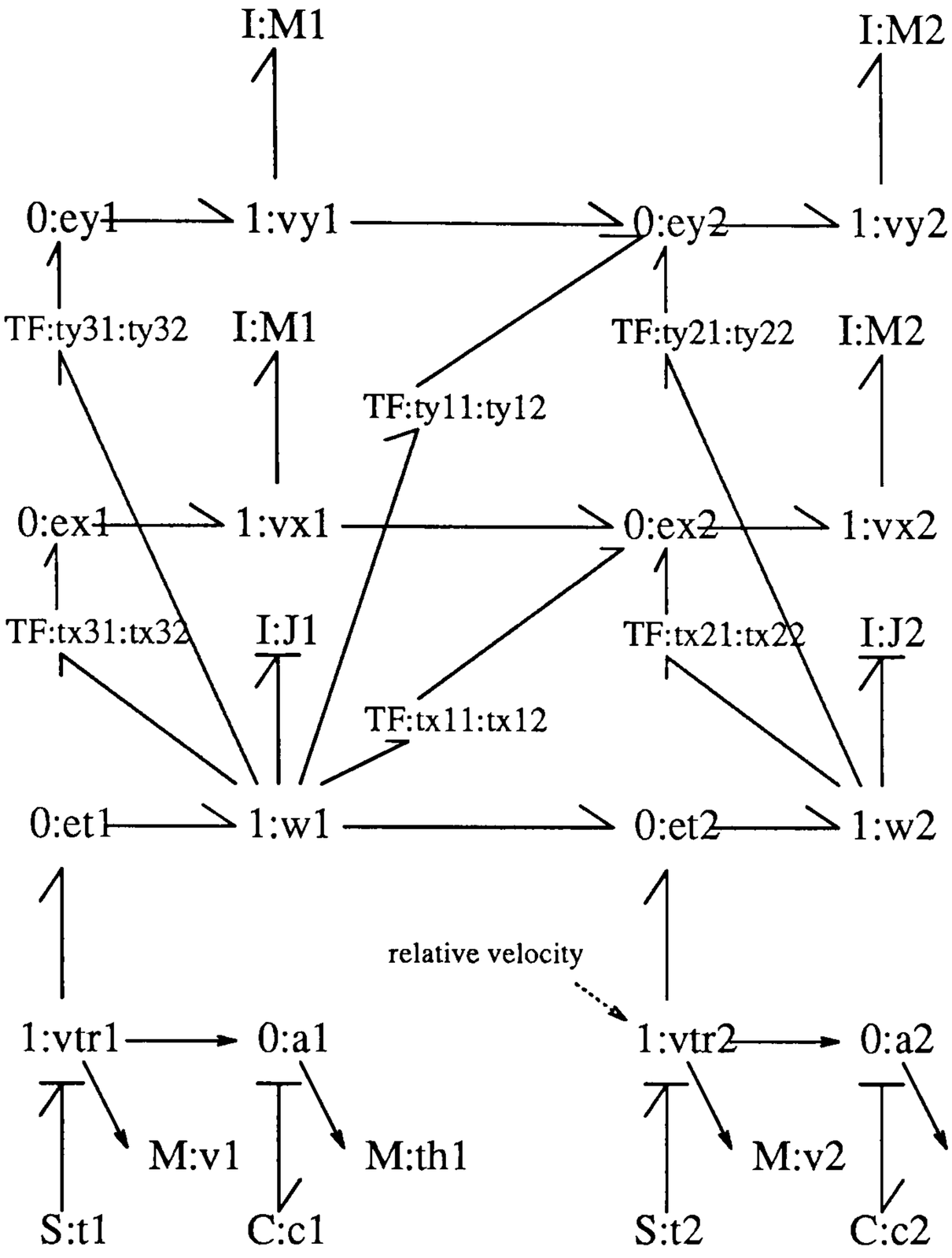


Figure 2.25: Bond Graph of Generic Two-Link Manipulator.

two-link manipulator.

2.3.3 Summary.

It is useful to note some points about the construction of the bond graph for a generic two-link manipulator.

1. It was constructed by generating the absolute velocities of key points in the system - notably the centres of mass of the links.

2. Accelerations have not been considered.

3. Inertial components are added once velocities have been represented.

It is apparent that the bond graph is constructed by considering the kinematics of the manipulator. The power of the bond graph lies in the fact that from this kinematic representation the full non-linear dynamic equations of motion may be generated automatically.

Whilst the bond graph shown is for a two-link manipulator, more links may be added simply by extending the pattern. Furthermore, the effect of gravity for a vertically oriented manipulator may be represented by attaching gravity sources to the vertical velocity elements.

2.4 Construction of the Bond Graph for the Experimental Two-Link Manipulator.

The generic bond graph for a two-link manipulator is an ideal case. To modify this graph to represent the experimental two-link manipulator(DD2lm) it is necessary to include the dynamics of the actuators; voltage controlled d.c.motors.

2.4.1 Mass of the Second Motor.

The motors drive the links directly without the use of transmission systems such as gearboxes or drive belts. Whilst this simplifies the system, it means that the motor driving the second link must be mounted at the pivotal point of the second link which is situated at the distal end of the first link. As the motor is rigidly fixed to the first link, its mass could be represented by considering it an integral part of the first link and modifying the mass, moment of inertia and the position of centre of mass of the first link accordingly. It is more general, however, to represent the presence of the second motor explicitly on the bond graph. The mass of the first motor, being rigidly fixed in space, need not be represented.

Following the prescribed procedure, the mass of the second motor is incorporated into the bond graph by calculating its absolute velocities in space.

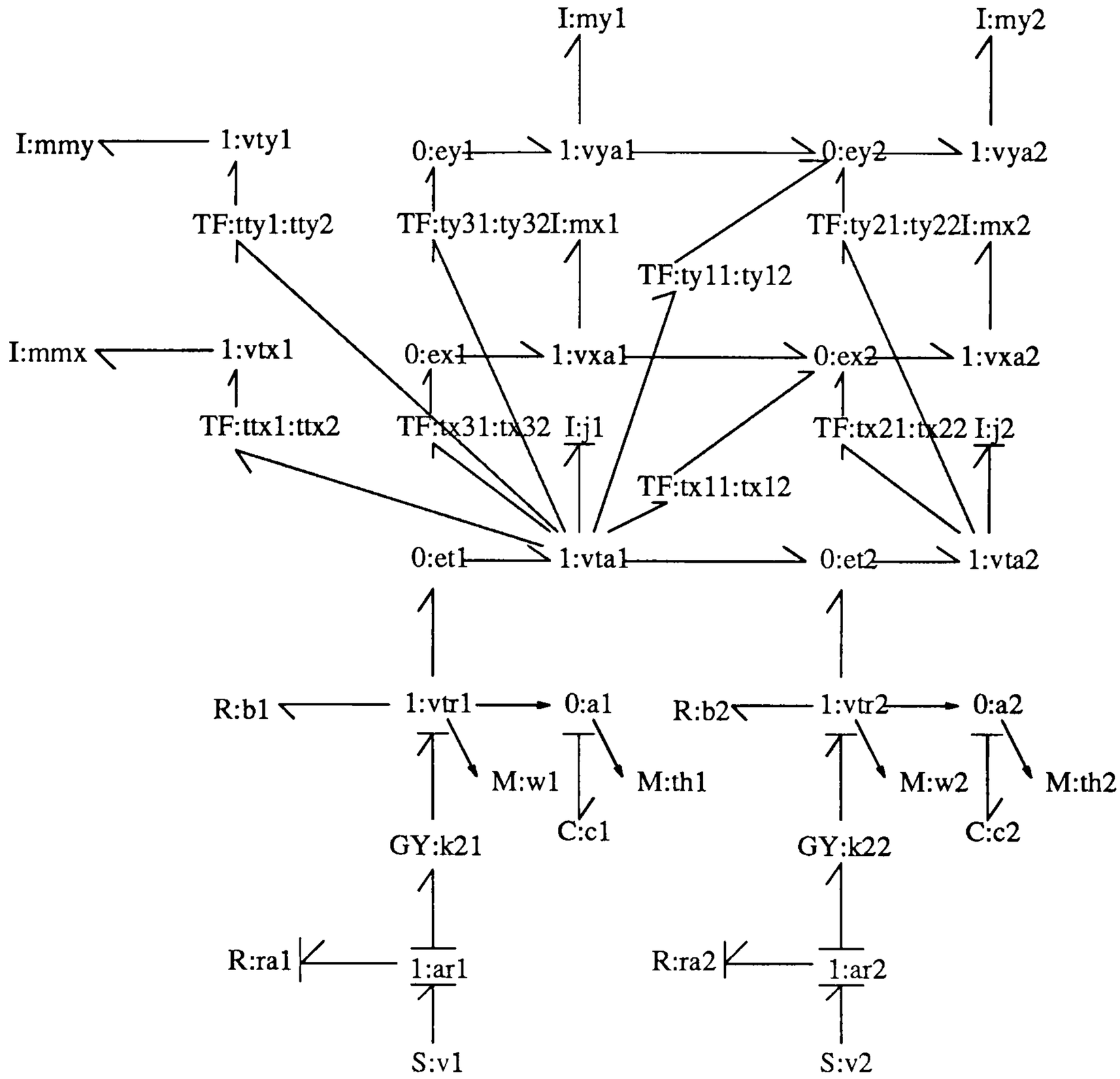


Figure 2.26: Bond Graph of Experimental Two-Link Manipulator.

This is done, as seen in figure 2.26, using modulated transformers with gains $-(l_{11}+l_{12})sin(\alpha_1)$ and $(l_{11}+l_{12})cos(\alpha_1)$ to generate the velocities **vtx1** and **vty1**, in the x and y directions respectively, from the absolute angular velocity of the first link, **vta1**, according to the relation

$$vtx_1 = -(l_{11}+l_{12})sin(\alpha_1) \times vta_1 \tag{2.29}$$

$$vty_1 = (l_{11}+l_{12})cos(\alpha_1) \times vta_1 \tag{2.30}$$

Once the velocities have been represented, the inertial elements **mmx** and **mmy** can be added to represent the actual mass of the motor.

Due to its very small moment of inertia around its own axis compared to that of the first link, the rotational motion of the second motor has been neglected.

2.4.2 Motor Dynamics.

The bond graph for a d.c.motor was developed in section 2.2.5 and for a voltage controlled d.c.motor is shown in figure 2.19. This bond graph has been incorporated into figure 2.26 by replacing the torque sources of figure 2.25. As the links are directly driven, the rotors are connected directly to the links they drive and are therefore effectively part of the link. Consequently, the effects of rotor inertia can be neglected as rotor moments of inertia are negligible compared with those of the links. Rotor inertia is therefore not represented in figure 2.26.

The important point to note about the motor dynamics is that they deal with relative velocities: the back e.m.f. of a motor, for example, is dependent on the velocity of the rotor relative to the stator which may itself be rotating in space. The bond graphs of the motors are therefore connected to the relative angular velocities **vtr1** and **vtr2** of the links.

2.4.3 Dynamics of the Two-Link Manipulator.

The dynamic equations of motion for the two-link manipulator may now be automatically generated from the bond graph in figure 2.26 using the Model Transformation Toolbox [30].

The state equations for the DD2lm are

$$x = \begin{pmatrix} h_1 \\ h_2 \\ \theta_1 \\ \theta_2 \end{pmatrix};\ z = \begin{pmatrix} m_1\dot{x}_1 \\ m_1\dot{y}_1 \\ m_2\dot{x}_2 \\ m_2\dot{y}_2 \\ m_m\dot{x}t_1 \\ m_m\dot{y}t_1 \end{pmatrix};\ y = \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \theta_1 \\ \theta_2 \end{pmatrix};\ u = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \tag{2.31}$$

$$\dot{x}_1 = \frac{1}{(2l_1^2 l_2^2 m_1 m_2 ra_2 ra_1)} \begin{pmatrix} -24(((b_2+b_1)ra_1 + k2_1^2)ra_2 + ra_1 k2_2^2)l_2^2 m_2 x_1 + \dots \\ \dots 24(ra_2 b_2 + k2_2^2)l_1^2 m_1 ra_1 x_2 + \dots \\ \dots 2(ra_2 k2_1 u_1 - ra_1 k2_2 u_2)l_1^2 l_2^2 m_1 m_2 - \dots \\ \dots (\dot{z}_1 + 2\dot{z}_3 + 2\dot{z}_5)sin(x_3)l_1^3 l_2^2 m_1 m_2 ra_2 ra_1 - \dots \\ \dots (\dot{z}_2 + 2\dot{z}_4 + 2\dot{z}_6)cos(x_3)l_1^3 l_2^2 m_1 m_2 ra_2 ra_1 \end{pmatrix} \quad (2.32)$$

$$\dot{x}_2 = \frac{1}{(2l_1^2 l_2^2 m_1 m_2 ra_2)} \begin{pmatrix} -(24(l_1^2 m_1 x_2 - l_2^2 m_2 x_1)(ra_2 b_2 + k2_2^2) + \dots \\ \dots sin(x_3 + x_4)l_1^2 l_2^3 m_1 m_2 ra_2 \dot{z}_3 + \dots \\ \dots cos(x_3 + x_4)l_1^2 l_2^3 m_1 m_2 ra_2 \dot{z}_4 - \dots \\ \dots 2l_1^2 l_2^2 m_1 m_2 k2_2 u_2) \end{pmatrix} \quad (2.33)$$

$$\dot{x}_3 = \frac{(12x_1)}{(l_1^2 m_1)} \quad (2.34)$$

$$\dot{x}_4 = \frac{(12(l_1^2 m_1 x_2 - l_2^2 m_2 x_1))}{(l_1^2 l_2^2 m_1 m_2)} \quad (2.35)$$

$$y_1 = \frac{(12x_1)}{(l_1^2 m_1)} \quad (2.36)$$

$$y_2 = \frac{(12(l_1^2 m_1 x_2 - l_2^2 m_2 x_1))}{(l_1^2 l_2^2 m_1 m_2)} \quad (2.37)$$

$$y_3 = x_3 \quad (2.38)$$

$$y_4 = x_4 \quad (2.39)$$

$$z_1 = \frac{(6sin(x_3)x_1)}{l_1} \quad (2.40)$$

$$z_2 = \frac{(6cos(x_3)x_1)}{l_1} \quad (2.41)$$

$$z_3 = \frac{(6(sin(x_3 + x_4)l_1 m_1 x_2 + 2sin(x_3)l_2 m_2 x_1))}{(l_1 l_2 m_1)} \quad (2.42)$$

$$z_4 = \frac{(6(cos(x_3 + x_4)l_1 m_1 x_2 + 2cos(x_3)l_2 m_2 x_1))}{(l_1 l_2 m_1)} \quad (2.43)$$

$$z_5 = \frac{(12sin(x_3)m_m x_1)}{(l_1 m_1)} \quad (2.44)$$

$$z_6 = \frac{(12cos(x_3)m_m x_1)}{(l_1 m_1)} \tag{2.45}$$

where

$\mathbf{x}$ = State vector.

$\mathbf{y}$ = Output vector.

$\mathbf{u}$ = Input vector.

h_i = Angular momentum of i^{th} link.

θ_i = Relative angular position of i^{th} link.

m_i = Mass of i^{th} link.

m_m = Mass of second motor.

v_i = Voltage input into i^{th} motor.

ra_i = Armature resistance of the i^{th} motor.

b_i = Viscous friction coefficient of the i^{th} motor.

$k2_i$ = Torque constant of the i^{th} motor.

In this form the equations of motion are not useful for simulation as the right hand sides of the equations contain terms in $\dot{x}$. Transferring terms involving the state derivatives to the left hand side of the equations yields the set of constrained state equations as follows.

Where

$$\dot{\chi} = E\dot{X} \tag{2.46}$$

$$\dot{\chi}_1 = \frac{1}{(l_2^2 m_2 ra_2 ra_1 l_1^2 m_1)} \begin{pmatrix} -(12(((b_2+b_1)ra_1 + k2_1^2)ra_2 + ra_1 k2_2^2)l_2^2 m_2 x_1 - \dots \\ \dots 12(ra_2 b_2 + k2_2^2)ra_1 l_1^2 m_1 x_2 - \dots \\ \dots (ra_2 k2_1 u_1 - ra_1 k2_2 u_2)l_2^2 m_2 l_1^2 m_1) \end{pmatrix} \tag{2.47}$$

$$\dot{\chi}_2 = \frac{(12(l_2^2 m_2 x_1 - l_1^2 m_1 x_2)(ra_2 b_2 + k2_2^2) + l_2^2 m_2 k2_2 l_1^2 m_1 u_2)}{(l_2^2 m_2 ra_2 l_1^2 m_1)} \tag{2.48}$$

$$\dot{\chi}_3 = \frac{(12x_1)}{(l_1^2 m_1)} \tag{2.49}$$

$$\dot{\chi}_4 = \frac{-(12(l_2^2 m_2 x_1 - l_1^2 m_1 x_2))}{(l_2^2 m_2 l_1^2 m_1)} \tag{2.50}$$

$$y_1 = \frac{(12x_1)}{(l_1^2 m_1)} \tag{2.51}$$

$$y_2 = \frac{-(12(l_2^2 m_2 x_1 - l_1^2 m_1 x_2))}{(l_2^2 m_2 l_1^2 m_1)} \tag{2.52}$$

$$y_3 = x_3 \tag{2.53}$$

$$y_4 = x_4 \tag{2.54}$$

$$\begin{aligned}
E(1,1) &= \frac{(4(3m_2 + m_1 + 3m_m))}{m_1} \\
E(1,2) &= \frac{(6cos(x_4)l_1)}{l_2} \\
E(1,3) &= \frac{-(6sin(x_4)l_1 x_2)}{l_2} \\
E(1,4) &= \frac{-(6sin(x_4)l_1 x_2)}{l_2} \\
E(2,1) &= \frac{(6cos(x_4)l_2 m_2)}{(l_1 m_1)} \\
E(2,2) &= 4 \\
E(2,3) &= \frac{(6sin(x_4)l_2 m_2 x_1)}{(l_1 m_1)} \\
E(3,3) &= 1 \\
E(4,4) &= 1
\end{aligned} \tag{2.55}$$

In this form all the state derivatives are on the left hand side of the equations and can therefore be calculated. The matrix E is non-singular and hence invertible to enable the state vector to be calculated from the vector $\dot{\chi}$.

2.5 Conclusion.

We have seen in this chapter how the generic bond graph for a rigid, planar, revolutionary two-link manipulator is constructed by considering just the kinematic relationships of the system. From this generic form, the specific bond graph for the experimental two-link manipulator was constructed by adding bond graphs

representing the actuators; in this case voltage controlled d.c.motors. The physical placement of the motors, which affects the dynamics of the system, was also taken into consideration.

From the causally complete bond graph of the DD2lm, the dynamic equations of motion were generated in two forms of which one, the set of constrained state equations, is suitable for use in a simulation. The generation of these dynamic equations of motion is done automatically by computer as the causally complete bond graph provides an unambiguous system representation from which the equations can be derived in a systematic way.

Whilst the experimental manipulator is a relatively simple system, the dynamic equations of motion incorporating actuator dynamics are complex. The ability to generate the equations of motion automatically and in a form suitable for inclusion in simulation packages and control software is therefore a significant advantage. Furthermore, as the system is represented graphically, additions and modifications to the system can be made easily and quickly with the knowledge that the mathematical implications will be handled automatically.

Chapter 3

Design and Construction of the Experimental Two-Link Manipulator.

3.1 Introduction.

The difficulty of using commercially available robotic manipulators as experimental test-beds for research is widely recognised. An, Atkeson and Hollerbach [2] outlined a typical scenario as follows:

> A new robotics research lab purchases a commercial robot to undertake experimental control studies. The lab soon realises that the programming language and host computer are too limited, and undertakes to rip out the computer system and implement its own. The only catch is that a detailed specification of the servo system is needed, and if the lab is lucky the robot manufacturer will agree to provide this information.

After 2 years the lab has finally managed to get the arm under its own computer control with a custom made planning and control system. The lab then decides that it is important to include contact sensing, but the remaining servo system and bandwidth are not conducive for incorporating such sensors. After one year, the servo system is patched and the lab is finally able to conduct some experiments in robot control. Ultimately, the manipulators design is seen to prohibit meaningful control experiments, due to the reasons discussed above. The lab is stymied in its quest to study robot control, and must resort to simulation.

To avoid these pitfalls, it was decided that an experimental two-link manipulator be constructed to our own design. This allowed us to achieve the following:

- direct drive capability.
- low expense.
- easily modified links.
- simplicity of design.

Direct drive robots are becoming more commercially available (the AdeptOne Direct-Drive robot for example) but are very expensive due to the high specification required of the motors.

Keeping the design of the manipulator as simple as possible helps in the creation of an accurate mathematical model through having to consider only the ideal rigid body dynamics of the system. Furthermore, link masses and moments of inertia may be easily calculated or measured as the links are straightforward beams. The two-link two-dimensional SCARA type robot (see figure 3.1) is the simplest type of manipulator which demonstrates the non-linear characteristics

of robotic manipulators such as Coriolis and centrifugal forces and was therefore chosen as the configuration for the experimental manipulator.

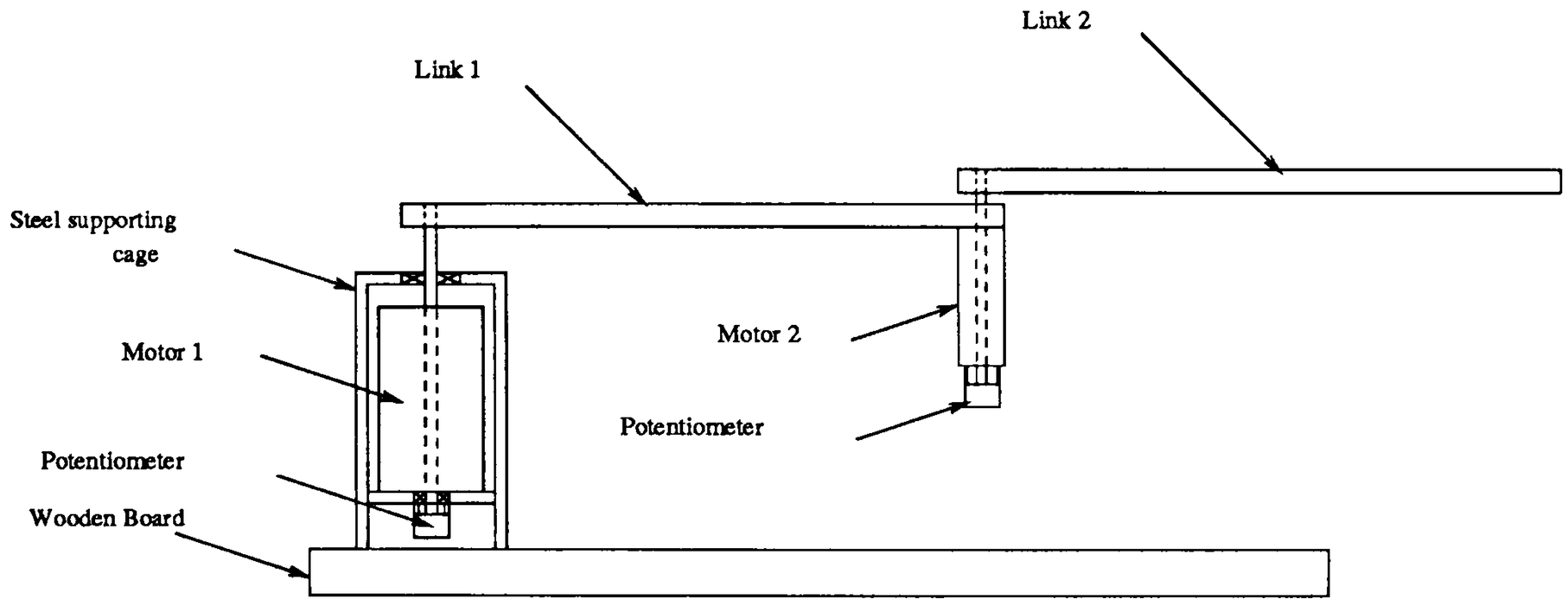


Figure 3.1: Two-Link SCARA Type Robotic Manipulator.

3.2 Specifications.

3.2.1 Direct Drive.

A directly driven robot is a robot in which the links are attached directly to the actuators without the use of gears or drive-belts. In the case of motor driven rotational joints, for example, the link is attached directly to the rotor shaft via an inflexible shaft extension if necessary.

The advantages of using direct drive are

- low friction.
- no backlash.
- high slew rates/accelerations.
- accurate torque control.

Most advanced robotic control schemes such as computed torque [3] or Resolved Motion Rate Control [35] rely on the ability of the actuator to control

joint torques accurately. The presence of a gearbox, however, increases friction significantly to the extent that, for the example of the PUMA robot, up to 50% of the actuator torque may be lost to friction [2]. This friction is difficult to model accurately as it is dependent on manipulator configuration and load and is therefore dificult to counteract. Joint torque/force sensors are available but are expensive and difficult to use in practice. Consequently, to research and implement model based control it is easier to specify directly driven links. The need to model gearboxes is therefore dispensed with.

Whilst direct drive offers many advantages, some disadvantages must be taken into consideration.

- low torque capability.
- high power consumption.

The low torque capability is due to the lack of torque amplification available using gearboxes. With no torque amplification the motors must supply higher torques which, for a given motor, requires higher currents. I^2R losses in the armature circuit are consequently higher leading to overheating of the motors. These problems may be reduced by designing the manipulator in such a way that the motors need never hold the links against gravity. The horizontal SCARA type robot achieves this by having vertical link rotational axes.

3.2.2 Motors/Amplifiers.

One of the purposes of the experimental two-link manipulator is to test how effective the bond graph model is at predicting the motion of the system. Robotic manipulators are characterised as being highly non-linear, multiple-input multiple-output systems so it is desirable that the experimental manipulator also demonstrates these characteristics.

The non-linearities of the system are caused by the presence of Coriolis and centrifugal forces acting on the links and link interactions caused by moments and forces acting through the link joints. These forces increase quadratically with link angular velocities so to maximise their effect the motors must be capable of accelerating the links to high velocities. High torque motors are required.

The final choice of motors was a compromise between performance and cost. A motor works in conjunction with a power amplifier and it is the combination of the two which defines the performance of the motor. The type of amplifier chosen for the DD2lm was the EM100B Servo Control amplifier supplied by McLennan. The same model of amplifier was chosen to power both the first and second motors in order to simplify the operation of the manipulator.

The EM100B amplifier has a peak current output of 10 Amps so to maximise the use of this available current, high torque constant motors were chosen: the S19-1B/T and the S372-1A/T, both iron cored permanent magnet d.c.servo motors supplied by Electrocraft.

The first motor, the S19-1B/T, has a torque constant of 0.23Nm/A and a peak torque capability of 3.35Nm but due to the terminal resistance of the motor and the maximum available amplifier voltage of $\pm 24V$, this peak torque is reduced to 1.62Nm when used in conjunction with the EM100B.

The second motor, the S372-1A/T, was chosen for its low mass of 0.44kg and relatively high torque constant of 0.044Nm/A. Low mass is important as the second motor is mounted at the end of the first link and therefore increases its moment of inertia. With the EM100B, the peak torque capability of the second motor is 0.23Nm.

3.2.3 Amplifiers.

The EM100B is a D.C.Servo system consisting of a power amplifier controlled by an analogue Servo Control Module. For our application, motor control is

handled using computers so the Servo Control Module was modified to configure the amplifier as a constant gain voltage amplifier: the voltage supplied to the motor is directly proportional to the signal voltage from the control computer effectively giving a voltage controlled d.c. motor. The amplifier therefore just supplies the current, and hence power, necessary to run the motor.

Specifications for the motors and amplifiers are given in Appendix A.

3.3 Instrumentation.

The state of the two-link manipulator is determined by the angular positions and velocities of the links so these must either be measured or derived.

Angular position is measured using 10 turn potentiometers energised using a zener stabilised voltage source to minimise drift. The potentiometer wiper shafts are connected directly to the motor shafts as shown in figures 3.2 and 3.3. The angle of the motor shaft from a specified datum point can then be ascertained by measuring the voltage of the potentiometer wiper and converting this to radians using a pre-calibrated linear transformation equation.

For each motor, motor angular velocity is measured using a tachometer mounted integrally within the motor housing. As the motor shaft revolves the tachometer produces a voltage proportional to the angular velocity which can then be measured. The signals from the tachometers are very poorly conditioned, however, and must be filtered to remove high frequency noise. This is done digitally in software using second order low pass Butterworth filters with an empirically determined cut-off frequency of 10Hz.

3.4 Construction.

3.4.1 First Joint.

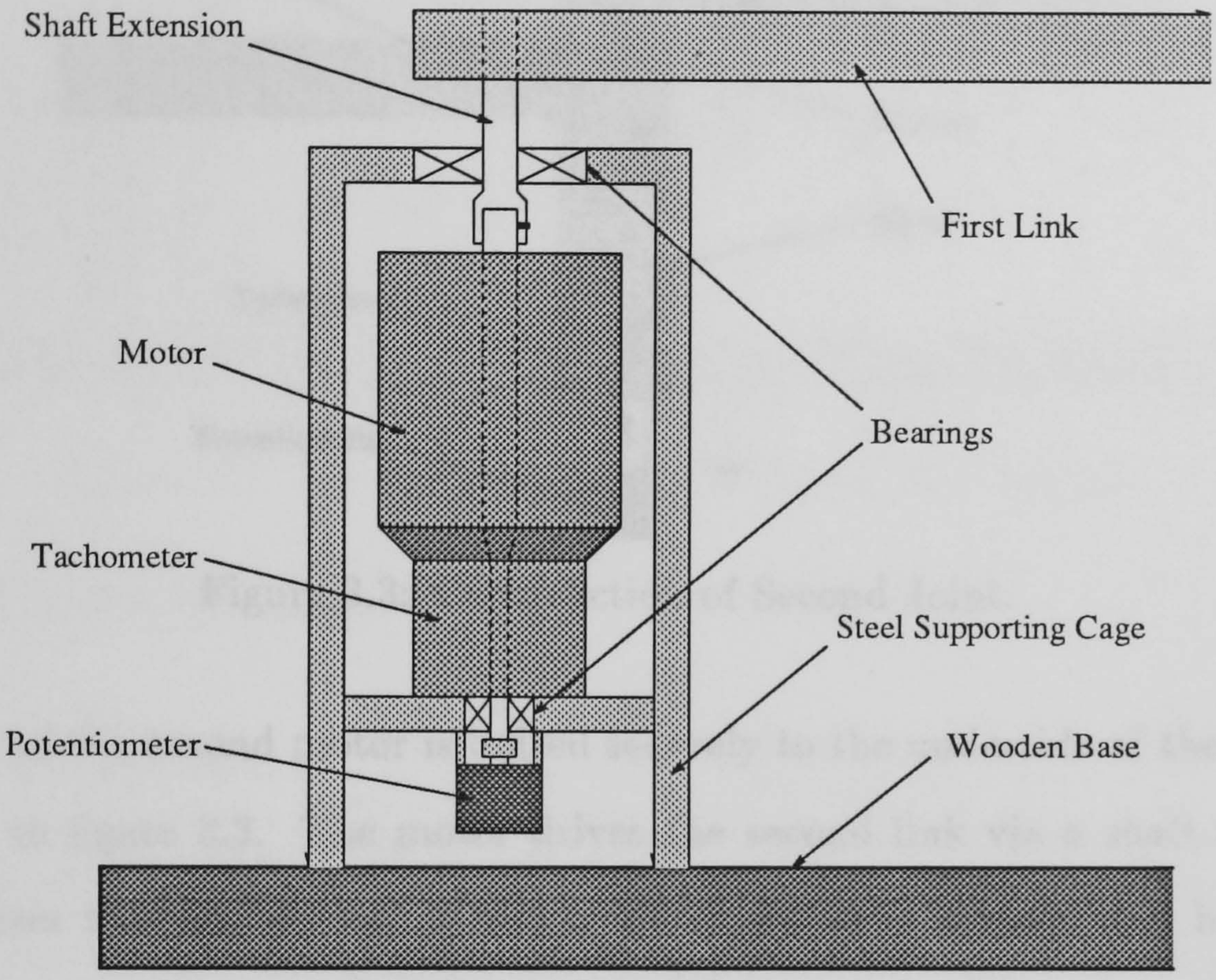


Figure 3.2: Construction of First Joint.

The construction of the first joint is shown in figure 3.2. The first motor is held rigidly in a steel cage mounted on a wooden base. The axial load on the motor shaft, caused by the weight of the links and second motor, is well within the maximum load of 4.54Kg specified for the motor bearings. Radial and torsional loads from the links are borne by roller bearings mounted within the steel cage above and below the motor.

The motor shaft is connected to the first link via a steel shaft extension attached using grub screws. The shaft extension passes through the end of the steel box section link and is clamped securely in place.

The potentiometer is sited below the motor. Its body is attached firmly to the steel supporting cage whilst the potentiometer shaft is connected to the lower end of the motor shaft using a steel collar fastened with grub screws.

3.4.2 Second Joint.

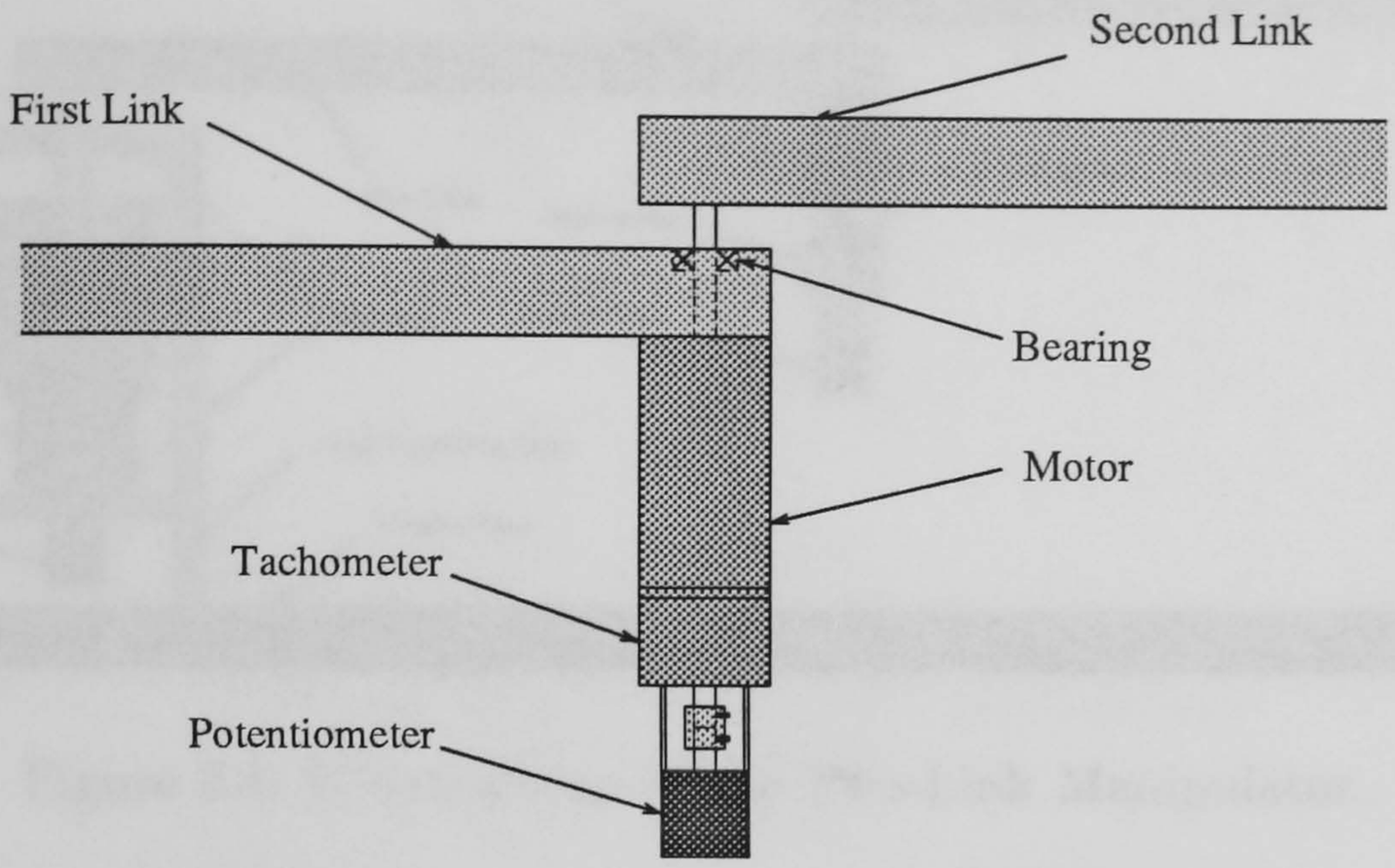


Figure 3.3: Construction of Second Joint.

The body of the second motor is bolted securely to the underside of the first link as shown in figure 3.3. The motor drives the second link via a shaft extension which passes through the end of the first link where it is supported by a roller bearing to reduce the radial and torsional load on the motor bearings. Again, the shaft extension is connected to the second link using a clamp.

The body of the potentiometer is clamped to the casing of the second motor whilst the potentiometer shaft is connected to the bottom of the motor shaft in the same way as for the first motor. In this way, the potentiometer measures the relative angular displacement between the first and second links.

3.4.3 Links.

The links are made from hollow square uniform box-section steel beams with a mass length density of 0.95Kg/m. The stiffness of the links is such that they may be considered rigid and the dynamics of flexible links may be ignored. Figure 3.4 shows the complete two-link manipulator.

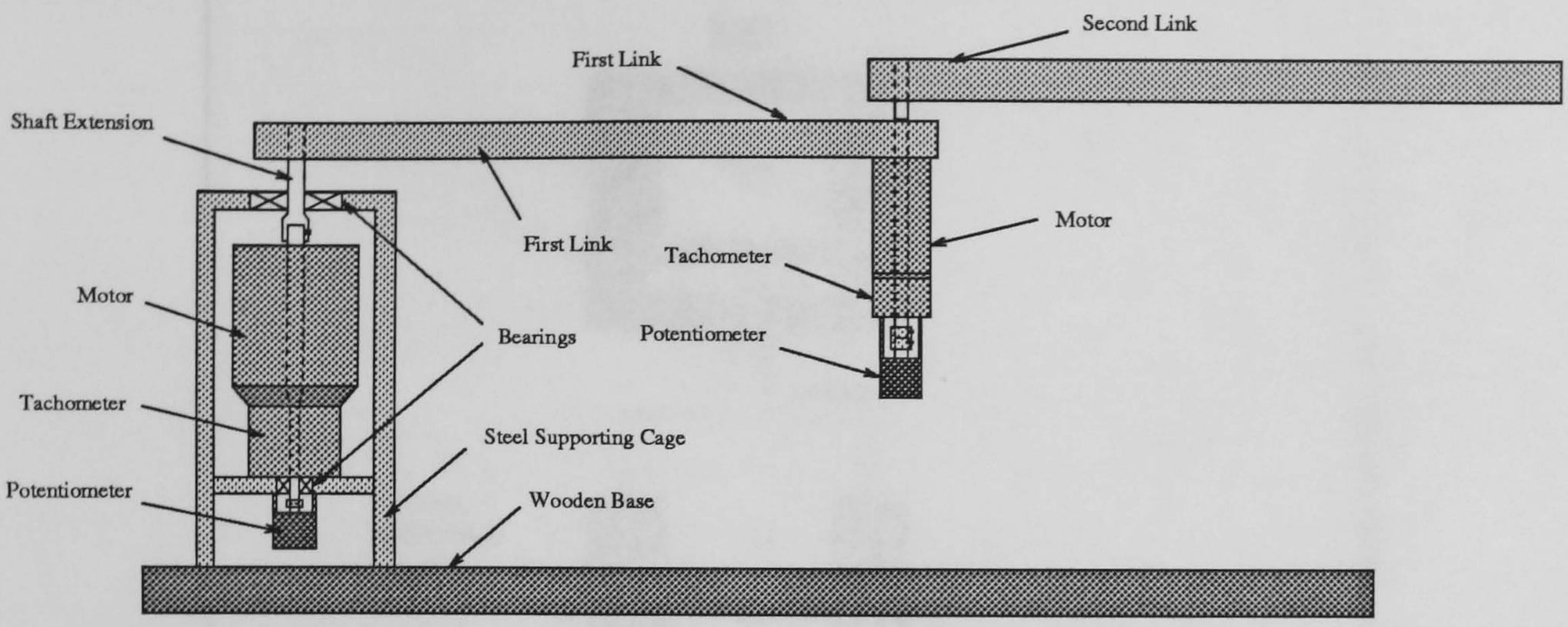


Figure 3.4: Construction of the Two-Link Manipulator.

3.5 Computing.

3.5.1 Hardware.

The computing hardware used for this project consists of a Sun3 workstation, two Motorola MC68020 series computers and two Burr Brown MPV901A data acquisition boards configured together as shown in figure 3.5.

Data Acquisition Boards.

Two data acquisition boards are used: one for each motor/amplifier combination. The Burr Brown MPV901A [36] is an analog I/O board with 32 single end (or 16 differential) input channels and two output channels. The input channels are connected to a 12bit analogue to digital converter with a sample/hold amplifier whilst the output channels have 12bit digital to analogue converters.

The input voltage range of the A/D converter is selectable. For the first motor the maximum range of $\pm 10V$ is used but the second motor requires the smaller range of $\pm 2.5V$ to maintain the velocity resolution as the second tachometer gives only a quarter of the output voltage of the first tachometer for a given angular velocity. With these ranges, the resolutions available for angular positions and

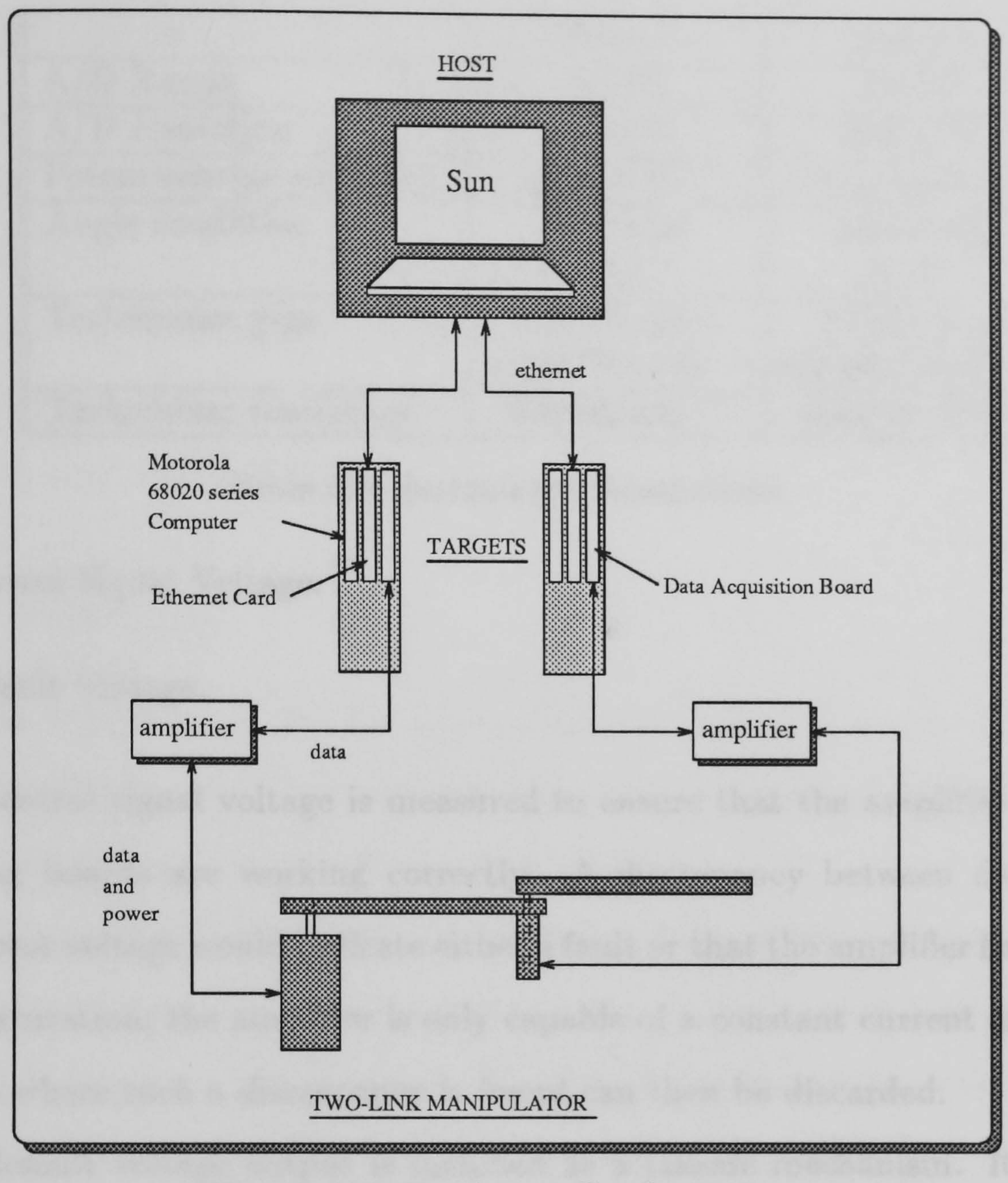


Figure 3.5: Hardware Configuration.

velocities are shown in table 3.5.1.

The following measurements are taken from each motor:

- Motor Voltage.
- Control Signal voltage.
- Tachometer Voltage.
- Potentiometer Voltage.

As the motor voltage may exceed the range of the A/D converter it is scaled down using a potential divider. The voltage is rescaled in the control software.

The output from the d2a converter to each amplifier/motor combination are:

	Motor 1	Motor 2
A/D Range	$\pm 10V$	$\pm 2.5V$
A/D resolution	0.0049V	0.0012V
Potentiometer resolution	$2.618radV^{-1}$	$2.618radV^{-1}$
Angle resolution	0.0128rad 0.72^o	0.0031rad 0.18^o
Tachometer gain	$14V/Kr.p.m.$ $0.1337V/rads^{-1}$	$3V/Kr.p.m.$ $0.0286V/rads^{-1}$
Tachometer resolution	$0.0366rads^{-1}$	$0.0420rads^{-1}$

Table 3.1: Instrument Resolutions.

- Control Signal Voltage.
- Default Voltage.

The control signal voltage is measured to ensure that the amplifier and data acquisition boards are working correctly. A discrepancy between desired and actual motor voltage would indicate either a fault or that the amplifier has reached current saturation; the amplifier is only capable of a constant current of 2 Amps. Test runs where such a discrepancy is found can then be discarded.

The default voltage output is included as a failsafe mechanism. It provides one of the differential inputs to the amplifier, the other being the control signal voltage, and is set to $0V$ at the beginning of each test. If the computer fails the D/A converter outputs a default voltage of $+5V$ to both output channels but , as the amplifier is a differential amplifier, the resultant motor voltage is zero. Without this protection a computer failure would cause the motor to accelerate out of control at full power.

Computers.

Two Motorola MC68020 series computers are used; one for each motor/amplifier combination. These computers are dedicated to the system but can communicate with each other and a Sun3 workstation via ethernet. The MC68020 computer card, ethernet card and MPV901A data acquisition board are housed in a VMEbus

based racking system.

The Sun3 computer is a general access computer which is used to develop and compile software and to store and analyse results. For some applications it is also used to run part of the control algorithm where it communicates in real time with the dedicated MC68020s over ethernet. This is far from ideal as ethernet is a public access communication network without the guaranteed immediate access required for real time control purposes.

3.5.2 Software.

The software used to control the manipulator is CONIC: a large and powerful system for the design and development of concurrent and distributed applications [37] [38] [39] developed at Imperial college, London.

CONIC allows software systems to be designed in a modular format. Modules may be compiled for, and run concurrently on, multiple computers of different types. It is based on the language PASCAL but with added communication protocols to allow messages to be passed between modules which may be running on different computers.

The modular approach to programming is well suited to our application. Many of the tasks that must be executed by the control software fall into natural sub-groups:

- collection and filtering of data.
- the control algorithm.
- data file handling.
- user interface.

These separate tasks must all be co-ordinated in real time in order for the task to run successfully.

The modular approach has two significant benefits:

1. It allows the tasks to be programmed separately in self contained programs (modules). These programs may be altered without affecting the rest of the system as long as the module interface remains unchanged.

2. The programs may run in parallel on several different computers thus reducing the execution time. Modules only stop when waiting for data from other modules.

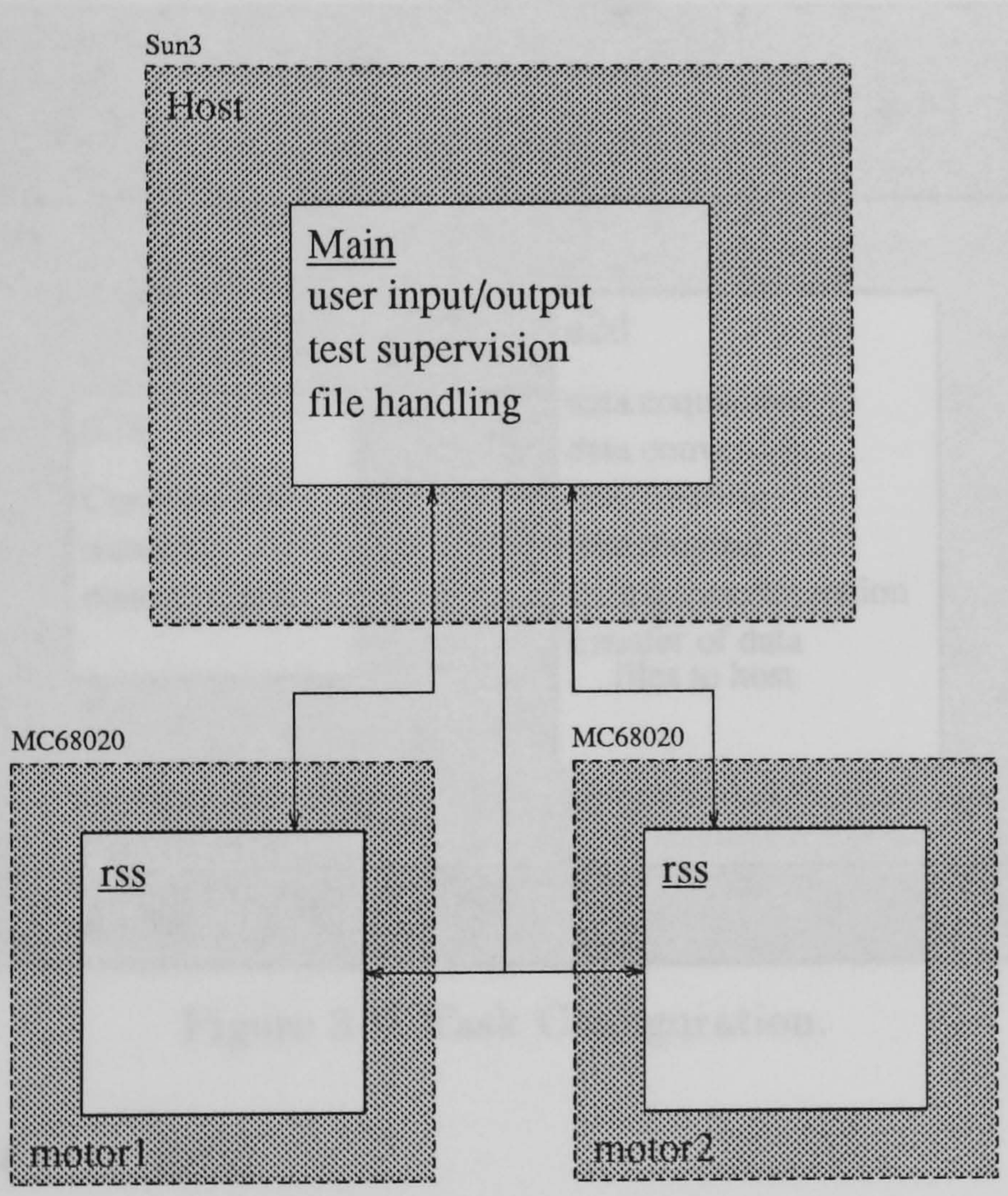


Figure 3.6: Software Configuration.

An example of the modular approach for the control of the two link manipulator is given diagrammatically in figures 3.6 and 3.7. The modules **Main** and **rss** are software modules written in CONIC.

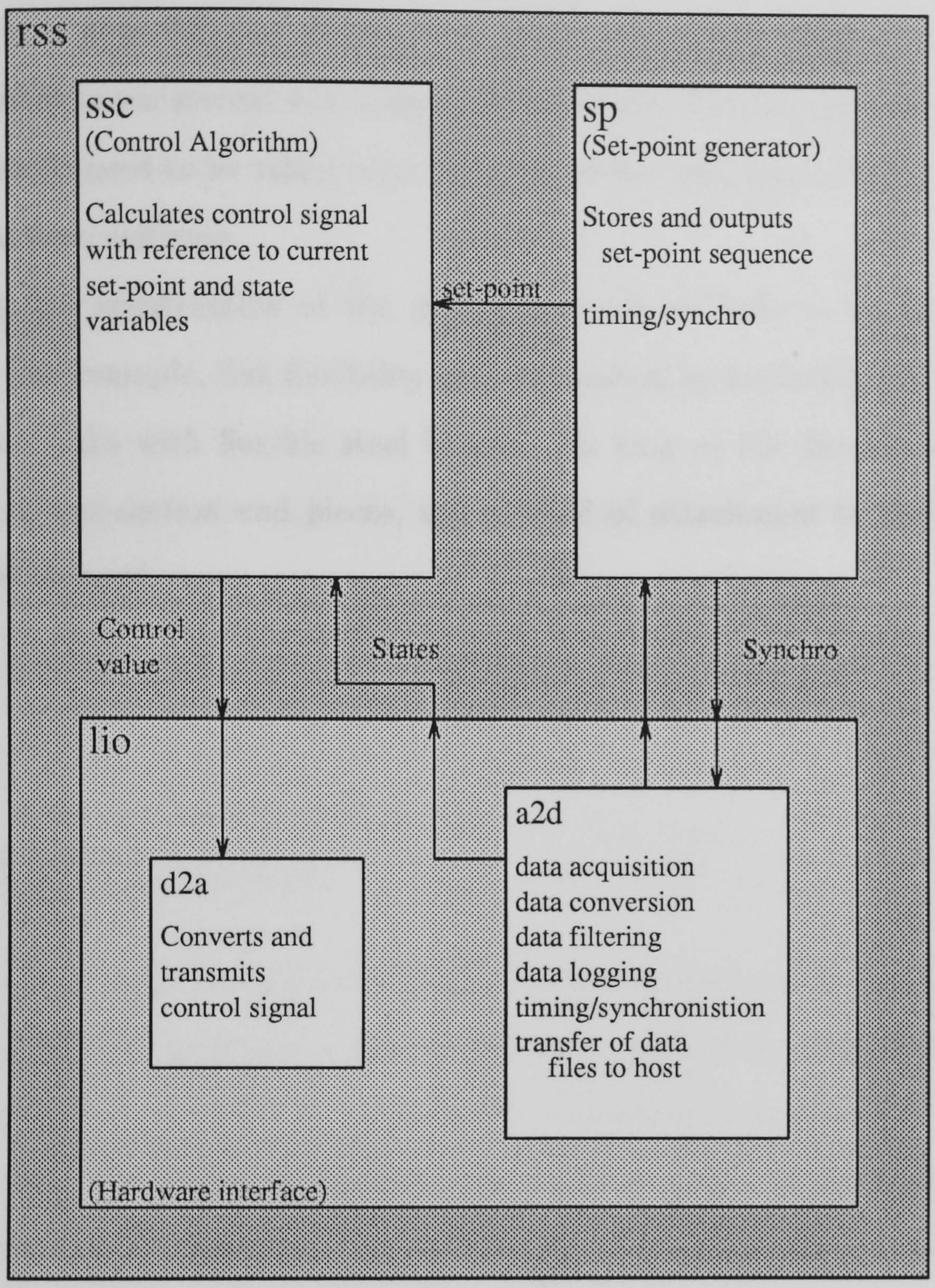


Figure 3.7: Task Configuration.

3.6 Conclusion.

The experimental two-link manipulator is a powerful and accessible system on which to perform experimental control studies. Its simplicity allows us to create an accurate mathematical model by considering the ideal rigid body dynamics of the system.

As the manipulator is custom-made, the instrumentation and control is completely accessible and can thus be easily modified. The software language,

CONIC, is a powerful and flexible tool which allows the control tasks to be distributed between several computers. Furthermore, if extra instrumentation or measurements need to be taken from the system they may easily be incorporated into the system software.

Lastly, the construction of the manipulator allows links to be easily interchanged. For example, link flexibility may be studied by replacing the rigid steel box-section links with flexible steel beams. As long as the flexible beams are attached to box-section end pieces, the method of attachment to the actuators remains unchanged.

Chapter 4

Validation of the Bond Graph Model.

4.1 Introduction.

We now have an experimental two-link manipulator and a bond graph model from which the dynamic equations of motion are easily obtainable. To have confidence in the model, it is necessary to validate these equations of motion.

Any model of a system can only include a subset of the full system dynamics; for example, the model of the two-link manipulator (see chapter 2) only includes the ideal rigid body dynamics. The task of model validation is to determine whether the model is sufficiently accurate for the purposes for which it was constructed. If it is not, it might be necessary to refine the model to incorporate more of the dynamics of the system. Model construction is therefore an iterative process with model validation providing the means to gauge whether to refine the model or not.

For the case of the two-link manipulator, the model is validated by comparing data collected from the experimental apparatus with simulations obtained using the dynamic equations of motion derived from the bond graph. Any mathematical

model consists of a structure together with parameters. Most of the parameters for the DD2lm, such as link masses or motor torque constants, can either be measured or are given in the specifications for the motors. Joint friction, however, is extremely difficult to measure accurately so the first part of this chapter deals with how friction may be identified.

4.2 System Identification.

To identify the friction parameters for the joints we consider the links separately and therefore need only use the mathematical model of a d.c. motor and not of the whole manipulator. Whilst it is the friction of the joints we wish to identify, the direct drive configuration allows us to treat each joint as if it consists solely of a d.c. motor; the additional friction of the joint bearings simply adds to the friction of the motor bearings.

Two forms of friction are to be identified:

- linear viscous friction,
- non-linear stiction.

Both these forms of friction are shown diagrammatically in figure 4.1.

The linear friction characteristic requires only one parameter, the gradient of the line, B, to be identified but the stiction characteristic, proposed by Canudas de Wit [40] and Li [41], requires the identification of the two extra stiction parameters f_1 and f_2. We wish to discover whether the additional complexity of incorporating stiction into the model is justified by improved performance over linear friction.

Two identification routines are used to identify the parameters,

- a routine based on the use of a state-variable filter (s.v.f) and
- a routine based on the use of the singular value decomposition (s.v.d).

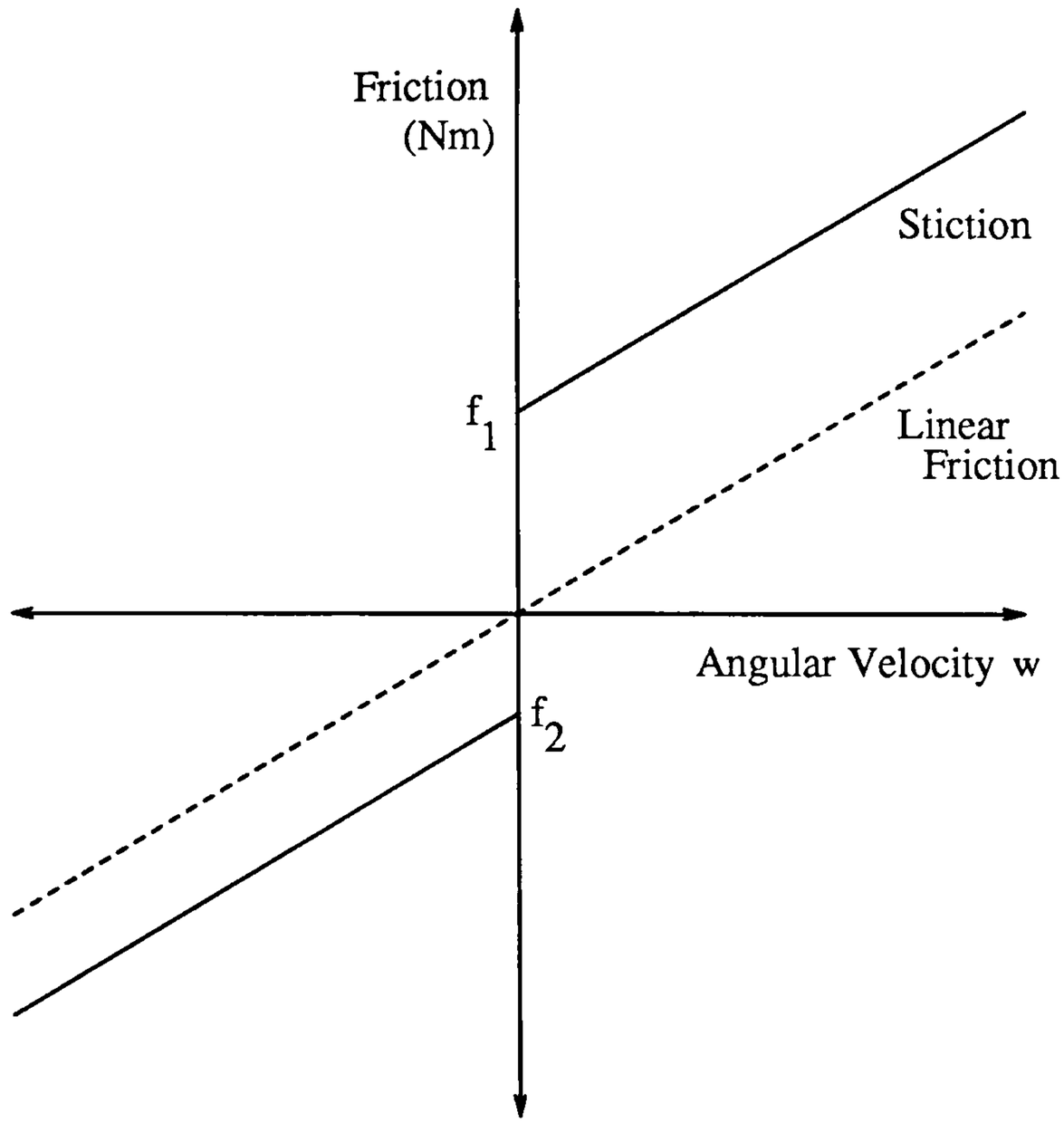


Figure 4.1: Friction Characteristics of a D.C.Motor

In addition to the friction parameters, the identification routines are used to identify the motor inertia. As the tests are carried out with the links attached to the motors, it is effectively the link inertias that are being identified. These can be easily calculated but their identification provides an extra means of assessing the accuracy of the identification routines by comparing the calculated and identified values.

4.2.1 Development of model for a D.C. motor.

As the motors are voltage controlled, the identification routines require a model that relates motor angular velocity to motor input voltage.

The torque for an armature controlled d.c. motor is proportional to the armature current, i_a. If the flux density is ϕ then the torque is given by

$$T(t) = k_1 \phi i_a(t) = k_2 i_a(t) \tag{4.1}$$

where k_2 is the torque-constant of the motor. The back e.m.f., v_b, is proportional

to the motor angular velocity $d\theta/dt$ so, with reference to the motor equivalent circuit (see figure 4.2), the armature voltage is given by

$$v_a(t) = R_a i_a(t) + k_3 \frac{d\theta(t)}{dt} + L_a \frac{di_a(t)}{dt} \tag{4.2}$$

where R_a is the armature resistance and L_a is the armature inductance.

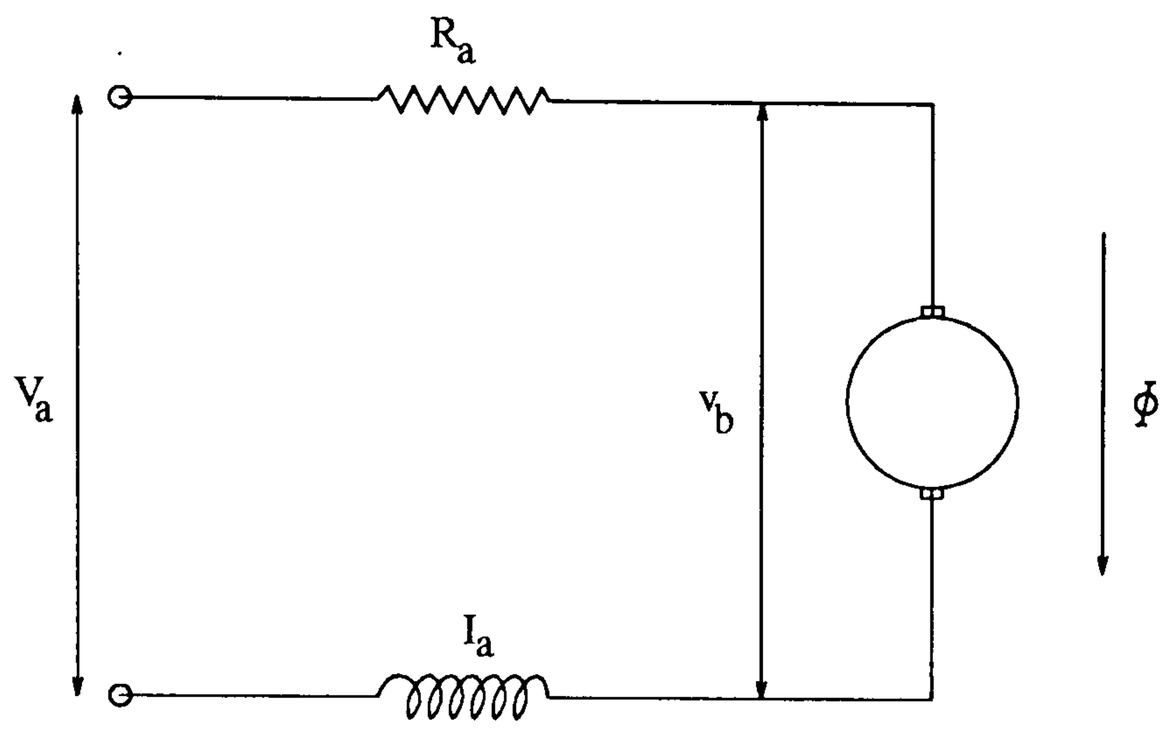


Figure 4.2: D. C. Motor Equivalent Circuit

The armature voltage can now be related to the angular velocity by considering the mechanical properties of the motor.

We shall assume the non-linear stiction-friction model proposed by Canudas de Wit [40] and Li [41], and combine static and viscous friction as shown in figure 4.1. If we assume that the motor inertia (or the motor+load inertia) is I then the torque is

$$T(t) = I \frac{d^2\theta(t)}{dt^2} + B \frac{d\theta(t)}{dt} + f_1 sign(\omega + |\omega|) + f_2 sign(\omega - |\omega|) \tag{4.3}$$

where B is the slope of the viscous friction in $\mathrm{Nm}/rads^{-1}$ and ω is the angular velocity of the motor. The linear friction model would only include the first two of the four right hand terms of equation 4.3.

Using the Laplace s variable as a convenient form of representation, equations 4.1 to 4.3 become

$$T(s) = k_2 I_a(s) \tag{4.4}$$

$$V_a(s) = (R_a + sL_a)I_a(s) + sk_3\theta(s) \tag{4.5}$$

$$T(s) = s(Is + B)\theta(s) + f_1 sign(\omega + |\omega|) + f_2 sign(\omega - |\omega|) \tag{4.6}$$

Combining equations 4.4 and 4.5 and eliminating T(s) in 4.6 gives

$$s\theta(s) = \frac{k_2 V_a(s) - (R_a + sL_a)(f_1 sign(\omega + |\omega|) + f_2 sign(\omega - |\omega|))}{((Is + B)(R_a + sL_a) + k_3 k_2)} \tag{4.7}$$

This is a second order system but in most d.c. motors the armature inductance is very small compared with the armature resistance and can be neglected. The resulting first order system is thus

$$s\theta(s) = \frac{(k_2/IR_a)V_a - (1/I)(f_1 sign(\omega + |\omega|) + f_2 sign(\omega - |\omega|))}{s + (1/IR_a)(BR_a + k_3 k_2)} \tag{4.8}$$

4.2.2 System Identification of Motor.

State Variable Filter Method.

The motors parameters are identified using a least-squares identification method detailed in Gawthrop [42]. This reference lists the identification routines as a 'toolbox' of matlab M.files, but to make sense of these routines it is necessary to rewrite the model in a form compatible for system identification.

The standard form for system identification is

$$y(s) = \frac{B(s)}{A(s)}u(s)$$

where y(s) is the system output, u(s) the input.Writing equation 4.8 in this form gives

$$s\theta(s) = \frac{1}{s + a_1}(b_0 V_a(s) - (1/I)(f_1 sign(\omega + |\omega|) + f_2 sign(\omega - |\omega|)))$$

where $a_1 = (1/IR_a)(BR_a + k_3k_2)$,$b_0 = \frac{k_2}{IR_a}$.

Re-arranging this equation and introducing a state-variable filter C(s) gives

$$\frac{s^2\theta(s)}{C(s)} = \frac{b_0V_a(s)}{C(s)} - \frac{a_1s\theta(s)}{C(s)} - \frac{(f_1/I)sign(\omega + |\omega|)}{C(s)} - \frac{(f_2/I)sign(\omega - |\omega|)}{C(s)} \quad (4.9)$$

C(s) is a polynomial of the same degree as A(s). Its function is to convert the LHS of equation 4.9 into a proper function thus avoiding the noise amplification effects of differentiation. As it stands, the left hand side of equation 4.9 is improper with respect to motor position but proper with respect to motor angular velocity, $s\theta(s)$.

From this representation it can be seen that the non-linearities can be thought of as extra inputs to the system and treated accordingly. Converting 4.9 to the time domain and writing in the standard form for least-squares identification

$$\Phi(t) = X^T(t)Q$$

where

$$\Phi(s) = \frac{s^2\theta(s)}{C(s)} = \frac{s}{C(s)}(s\theta s)$$

$$X(s) = \left[\frac{V_a(s)}{C(s)} \; \frac{s\theta(s)}{C(s)} \; \frac{sign(\omega + |\omega|)}{C(s)} \; \frac{sign(\omega - |\omega|)}{C(s)} \right]^T$$

Parameter vector Q:

$$Q = \left[b_0 \; -a_1 \; -\frac{f_1}{I} \; -\frac{f_2}{I} \right]^T$$

$$= \left[\frac{k_2}{IR_a} \; -\frac{1}{IR_a}(BR_a + k_3k_2) \; -\frac{f_1}{I} \; -\frac{f_2}{I} \right]^T$$

The least-squares identification routine returns the parameter vector Q as output having been given, as input to the routine, the motor input voltage and the resulting angular velocity variation with time as well as the augmented 'inputs' $sign(\omega - |\omega|)$ and $sign(\omega + |\omega|)$, constructed from the angular velocity output.

The parameter vector Q includes 7 unknowns in 4 equations but as k_2, k_3 and R_a can be either measured or read off data sheets the moment of inertia I,

viscous friction B, and stiction parameters f_1 and f_2 can all be calculated. The identification of a linear model can be achieved by ignoring the last two stiction terms of equation 4.9 in which case only the inertia and the viscous friction will be identified.

Singular Value Decomposition Method.

This method is based on Swevers [43].

The singular value decomposition routine uses the pseudo inverse A^+ to solve the set of over-determined equations

$$Ax = b$$

$$x = A^+ b$$

If n is the number of data points and p the number of unknown parameters then A is an $(n \times p)$ matrix of measured data, x is a column vector of the p unknown parameters and b a column vector of measured inputs. This solution is equivalent to the least squares solution

$$min_x \|Ax - b\|_2^2$$

The pseudo inverse A^+ is formed from the singular value decomposition. S is the $(n \times p)$ matrix formed by having the singular values of A on its main diagonal in decreasing order, all other terms being zero. U and V^t are orthonormal matrices formed from the left and right singular vectors respectively. These vectors are related to A by

$$A = USV^t$$

As the matrices U and V^t are orthonormal the pseudo inverse of A can be formed by

$$A^+ = VS^+U^t$$

where S^+ is the $(n \times p)$ matrix formed with the inverses of the singular values on its main diagonal.

The pseudo inverse of a matrix can be obtained in practice by using Matlabs `pinv` function.

It is now necessary to manipulate our model of the d.c motor into the form $b = Ax$. Re-arranging equation 4.8 and converting to the time domain gives

$$V_a = \frac{IR_a}{k_2}\frac{d^2\theta(t)}{dt^2} + (k_3 + \frac{BR_a}{k_2}\frac{d\theta(t)}{dt}) + \frac{R_a f_1}{k_2} sign(\omega + |\omega|) + \frac{R_a f_2}{k_2} sign(\omega - |\omega|) \quad (4.10)$$

$$A = \left[\frac{d^2\theta(t)}{dt^2} \quad \frac{d\theta(t)}{dt} \quad sign(\omega + |\omega|) \quad sign(\omega - |\omega|) \right]$$

$$x = \left[\frac{IR_a}{k_2} \quad (k_3 + \frac{BR_a}{k_2}) \quad \frac{R_a f_1}{k_2} \quad \frac{R_a f_2}{k_2} \right]^T$$

$$b = V_a$$

It would be possible to augment the data matrix A as above but it includes the angular acceleration of the motor, a variable which is not measured. Although the angular acceleration could be formed by differentiating the angular velocity vector this would result in noise amplification problems. A better solution is to integrate the whole equation. The A and b matrices then become

$$A = \left[\frac{d\theta(t)}{dt} \quad \theta(t) \quad \int^t sign(\omega + |\omega|)dt \quad \int^t sign(\omega - |\omega|)dt \right]$$

$$b = \int^t V_a dt$$

The parameter vector x can then be identified from the augmented data vector A and input vector b using $x = A^+ b$. Before this is done however, the position, angular velocity and motor voltage input vectors are filtered using a first order Butterworth filter with a frequency cut-off of 20Hz to avoid colouration of the broadbanded measurement noise by the least-squares algorithm.

4.2.3 Tests.

Tests were carried out with the motors in the following configurations:

- First motor with first link attached.
- First motor with first link and second motor attached.
- Second motor with second link attached.

For each test, the motor was made to follow the set-point trajectory shown in figure 4.3 with the use of a simple proportional and velocity feed-back controller tuned to give an approximately critically damped response [1].

This particular set-point trajectory was chosen as, with different starting positions, the motor could be made to move rapidly with high input motor voltages to the set point at the beginning of the test and then be made to follow the smooth trajectory at lower velocities. The high accelerations at the start of the test help in the identification of inertia and viscous friction whilst the lower velocities during the smooth portion of the trajectory help in the identification of the stiction parameters f_1 and f_2.

Filters.

For both identification methods, first order Butterworth filters are used to filter out high frequency noise in both the motor voltage vector and output angular velocity vector. The filters used were low-pass zero phase shift filters with a cut off frequency of 20Hz, chosen with reference to the power spectral densities of input and output vectors. It can be seen from figure 4.4 that above 20Hz noise is the dominant signal.

The effect of the filters can be seen in figure 4.5. The filters were implemented using Matlabs `filtfilt` function [44] which, by filtering the data forwards and backwards, causes no phase shift.

[1] The performance of the controller is not critical for the system identification tests.

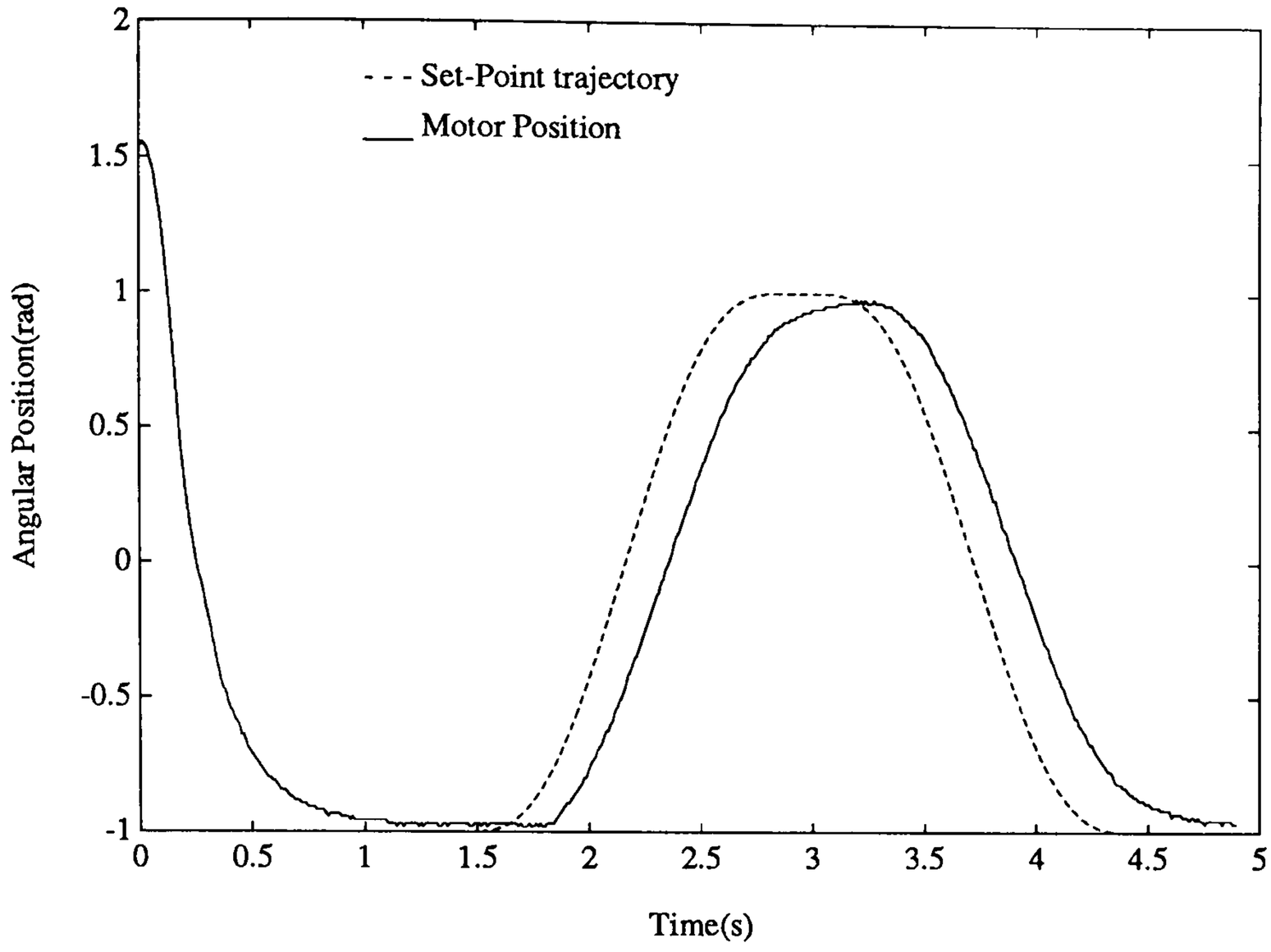


Figure 4.3: Set Point Trajectory for Identification Tests.

Simulations.

Simulations were obtained for a given identified parameter vector Q by calculating the motor angular acceleration for each time point in the motor voltage input vector and then integrating using a simple Euler algorithm to give motor velocity and position.

4.2.4 Results.

Six tests were carried out, two for each of the configurations outlined in section 4.2.3. For each test, sampled at 100Hz, inertia and friction parameters were identified in three ways:

1. State variable filter identified stiction, viscous friction and inertia.

2. State variable filter identified linear viscous friction and inertia.

3. Singular value decomposition identified stiction, viscous friction and inertia.

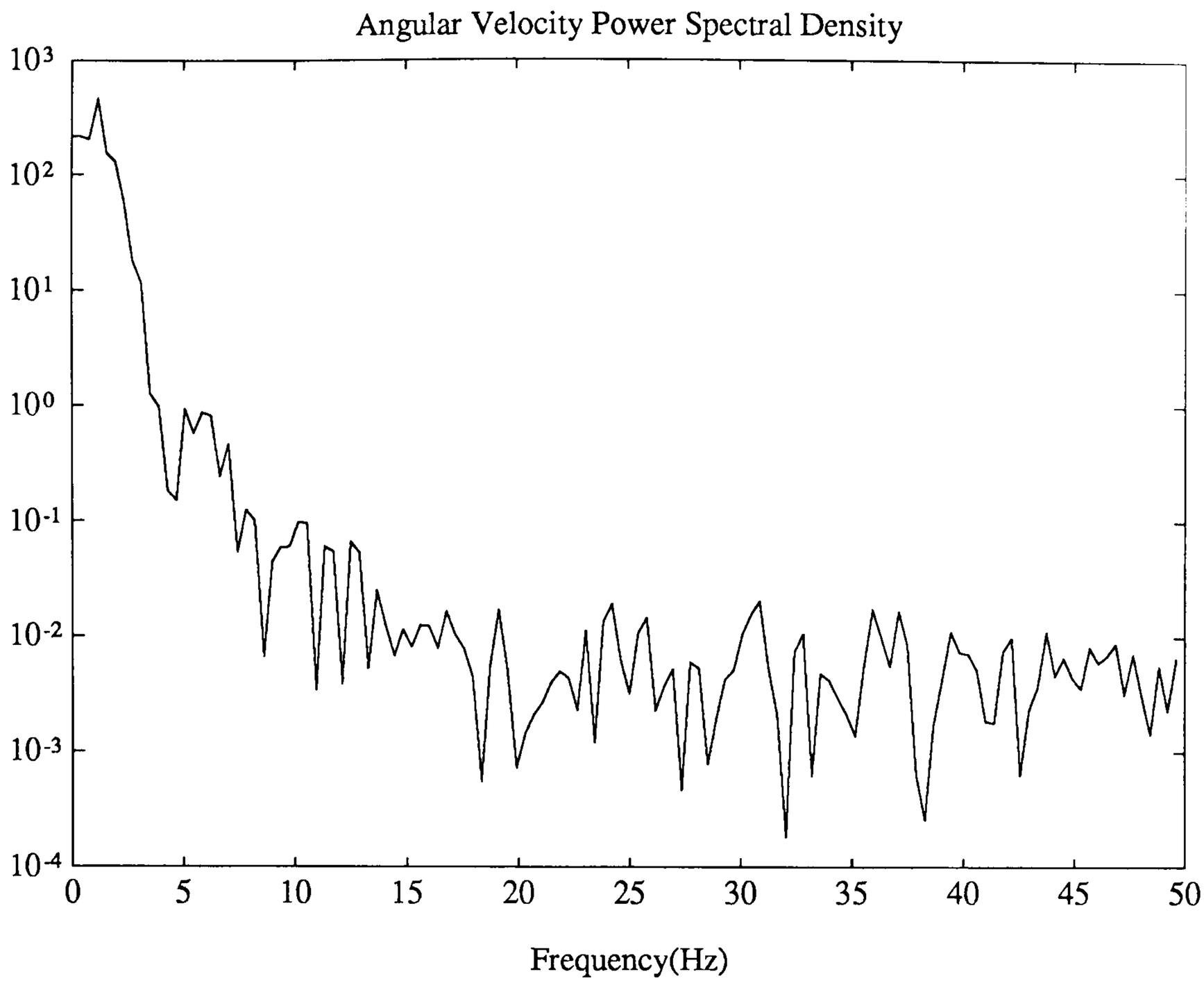


Figure 4.4: Power Spectral Densities of Input and Output Data.

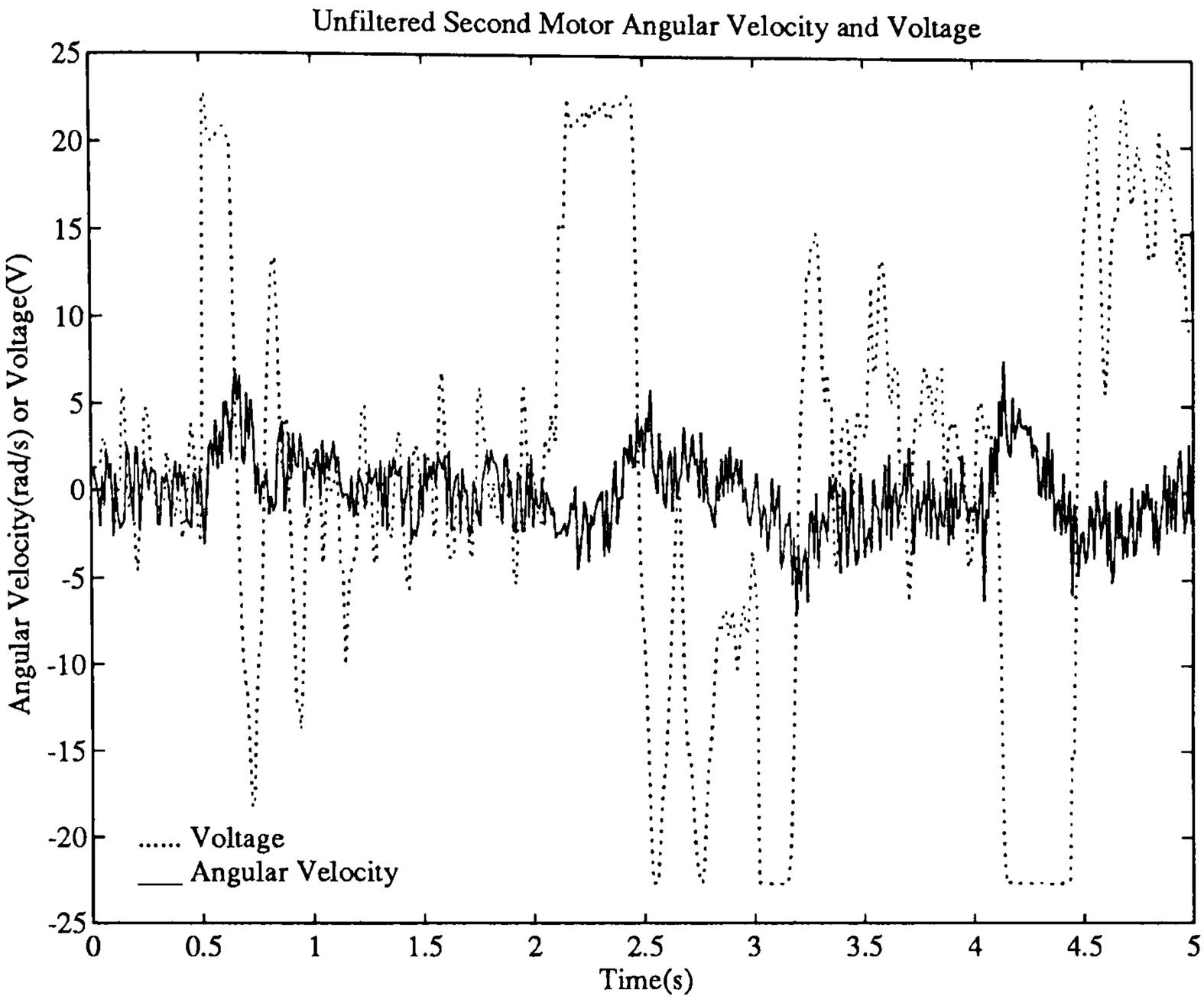


Figure 4.5: Filtered Input and Output Data.

Open loop simulations of link position and angular velocity were then obtained for each set of parameters and plotted against actual position and velocity in figures 4.6 to 4.11. The values of the identified parameters are given in tables 4.1 to 4.3 together with calculated values of inertia and quoted values of viscous friction and stiction (for the motors only) from the motor specifications.

	Test One			Test Two			Quoted
	s.v.f	s.v.d.	linear	s.v.f	s.v.d.	linear	
I (kgm^2)	0.0157	0.0155	0.0209	0.0157	0.0158	0.0178	0.0153
B ($\frac{Nm}{rads^{-1}}$)	-0.0075	-0.0090	0.0283	0.0029	-0.0000	0.0127	0.0001
f_1 (Nm)	0.0613	0.0632	-	0.0493	0.0554	-	0.0570
f_2 (Nm)	0.0722	0.0736	-	0.0497	0.0534	-	0.0570
Cond.No.	189	19.5	20.5	122	14.9	10.0	

Table 4.1: Identified Parameters for First Joint: First Link only.

	Test Three			Test Four			Quoted
	s.v.f	s.v.d.	linear	s.v.f	s.v.d.	linear	
I (kgm^2)	0.0701	0.0696	0.0753	0.0697	0.0691	0.0755	0.0800
B ($\frac{Nm}{rads^{-1}}$)	0.0077	0.0032	0.0307	-0.0008	-0.0042	0.0157	0.0001
f_1 (Nm)	0.0310	0.0356	-	0.0496	0.0531	-	0.0570
f_2 (Nm)	0.0504	0.0553	-	0.0680	0.0726	-	0.0570
Cond.No.	161	13.5	5.88	168	12.8	3.82	

Table 4.2: Identified Parameters for First Joint: Second Motor attached.

Discussion.

Motor 1.

It can be seen from the first four tests that the modelling and identification of non-linear stiction considerably improves the simulation of link position and angular velocity over the simple linear model. The identified joint parameters agree closely with the quoted parameters for motor friction and stiction which gives confidence in the use of motor parameters for joint friction. This is not surprising as the joint bearings and construction are of high quality and would add little friction

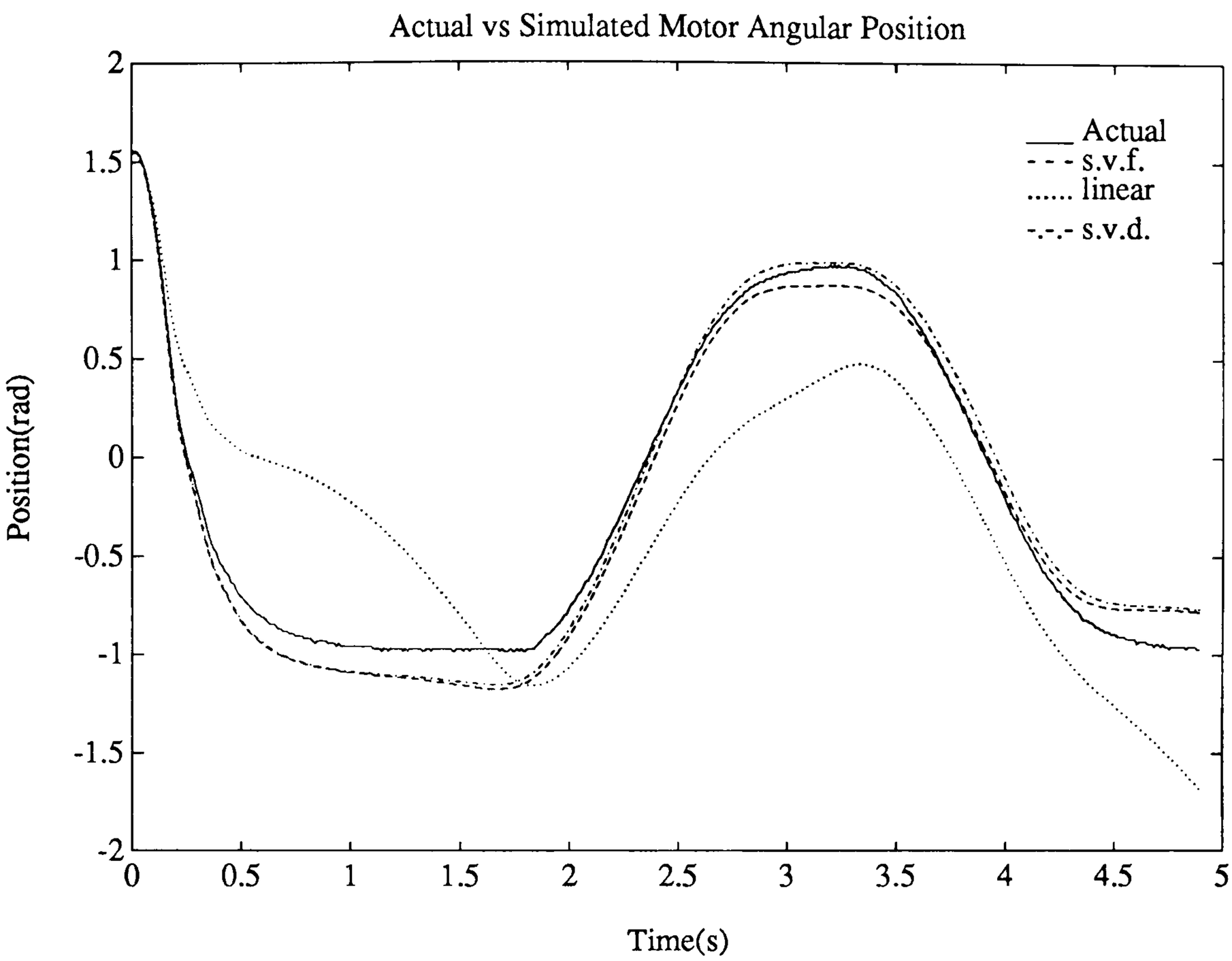


Figure 4.6: Test 1: First motor - first link only attached.

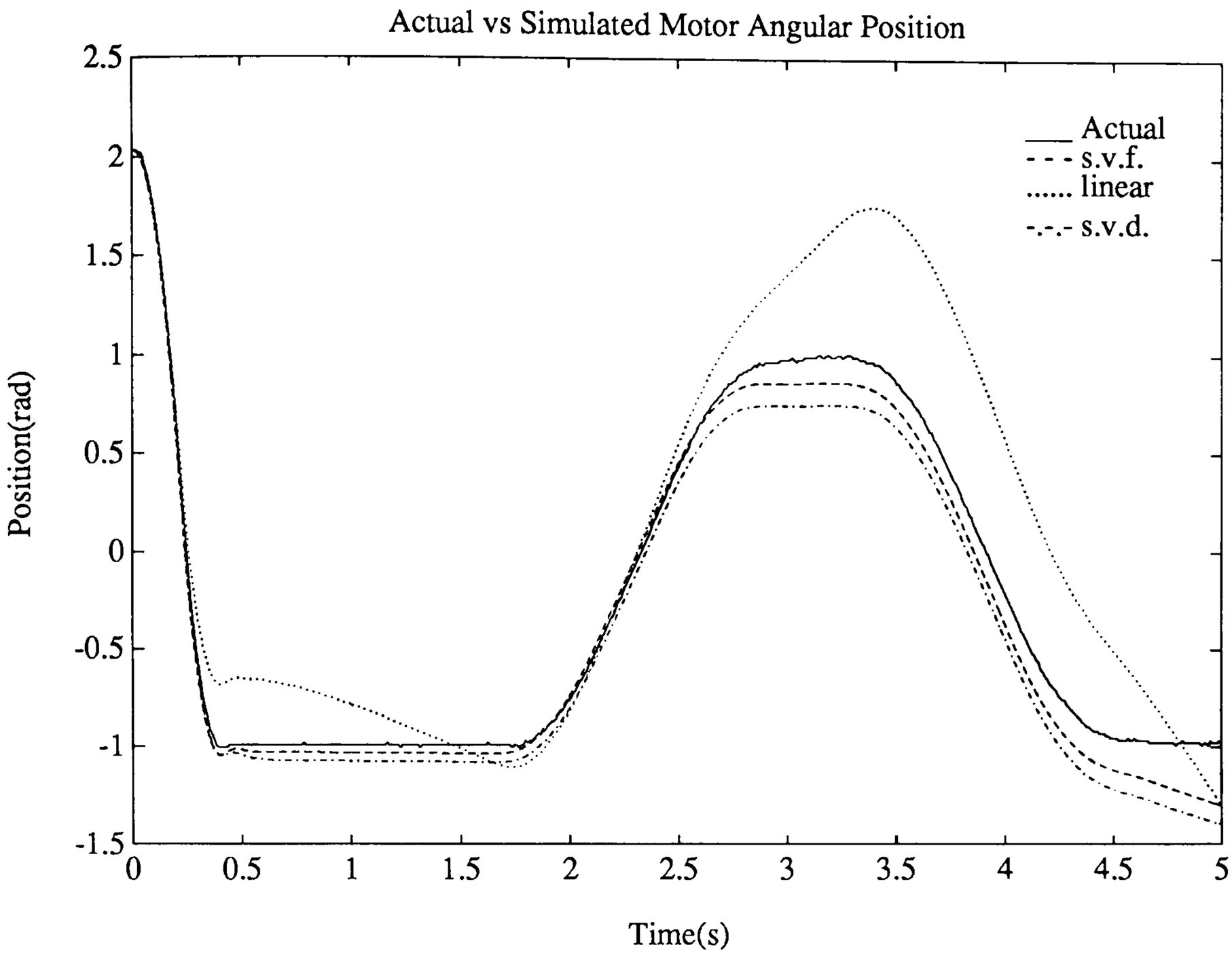
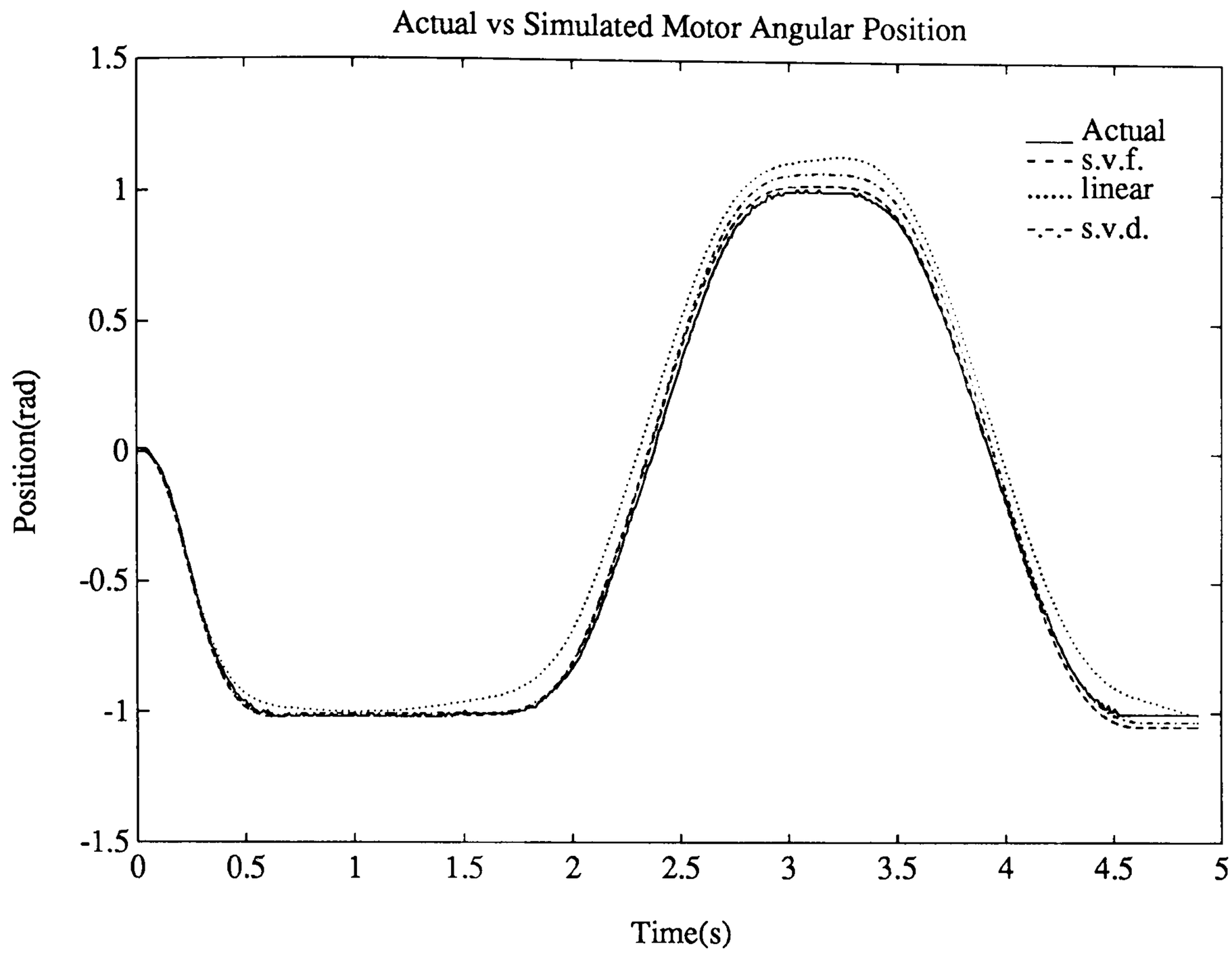


Figure 4.7: Test 2: First motor - first link only attached.



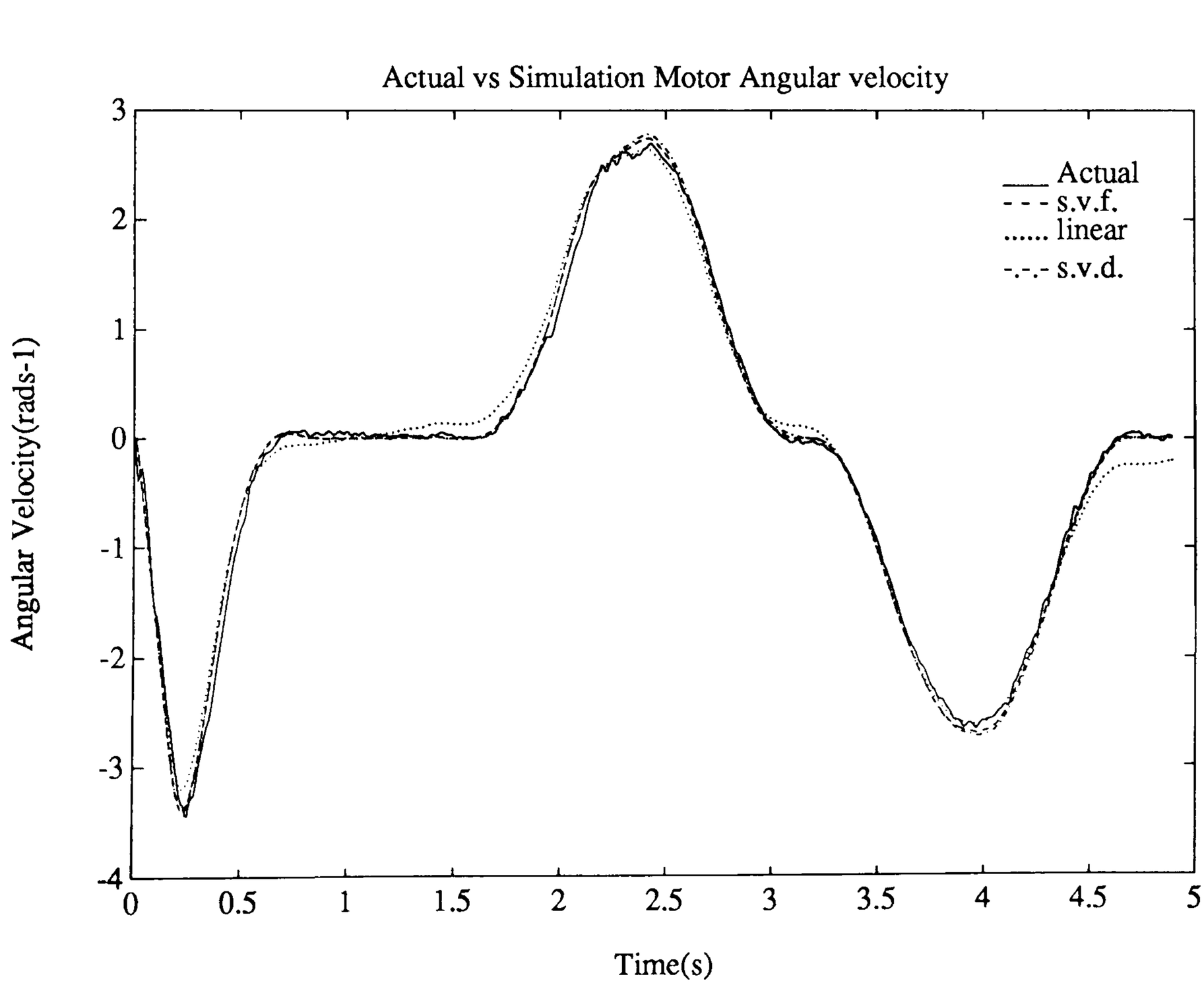
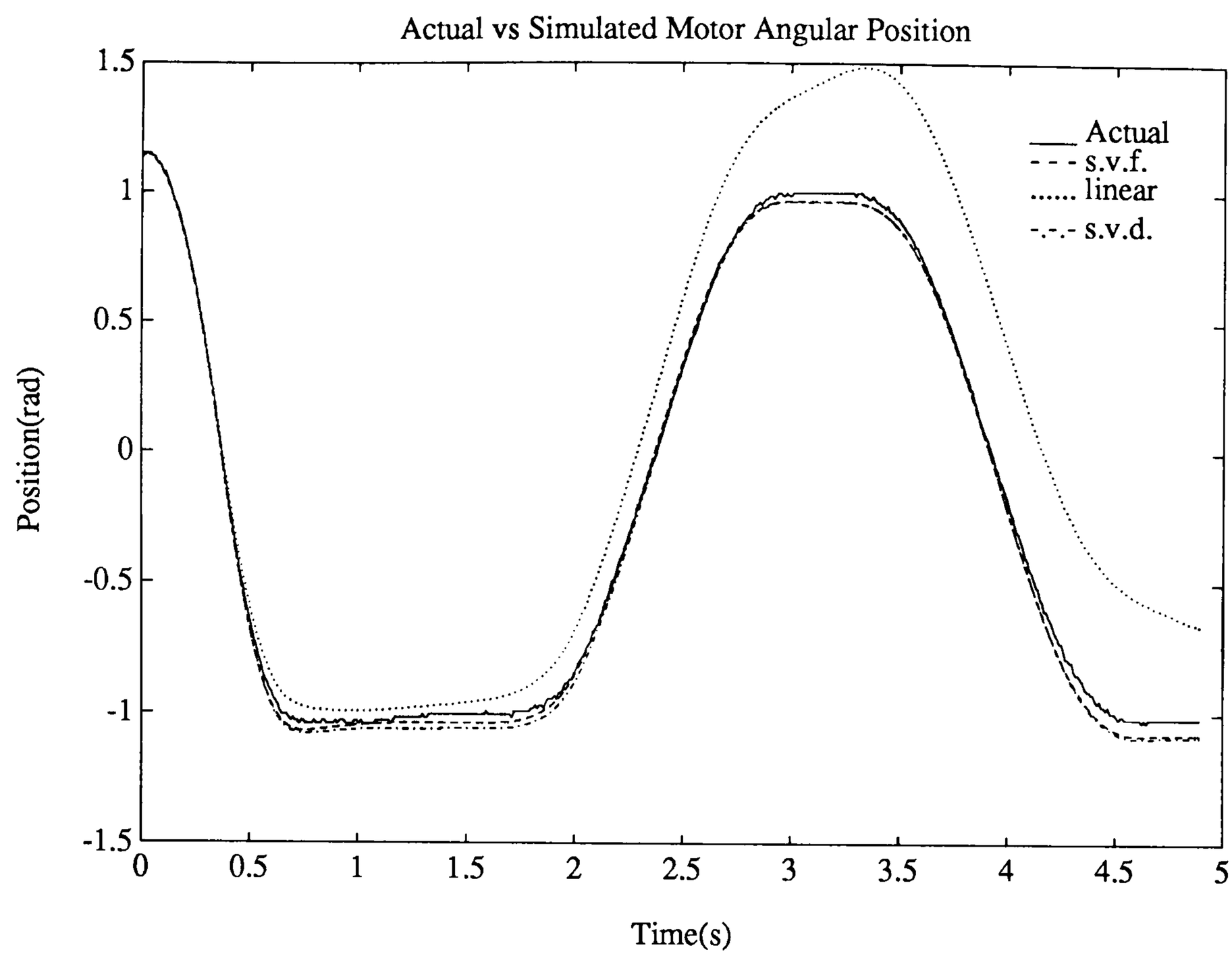


Figure 4.8: Test 3: First motor - first link with second motor attached.



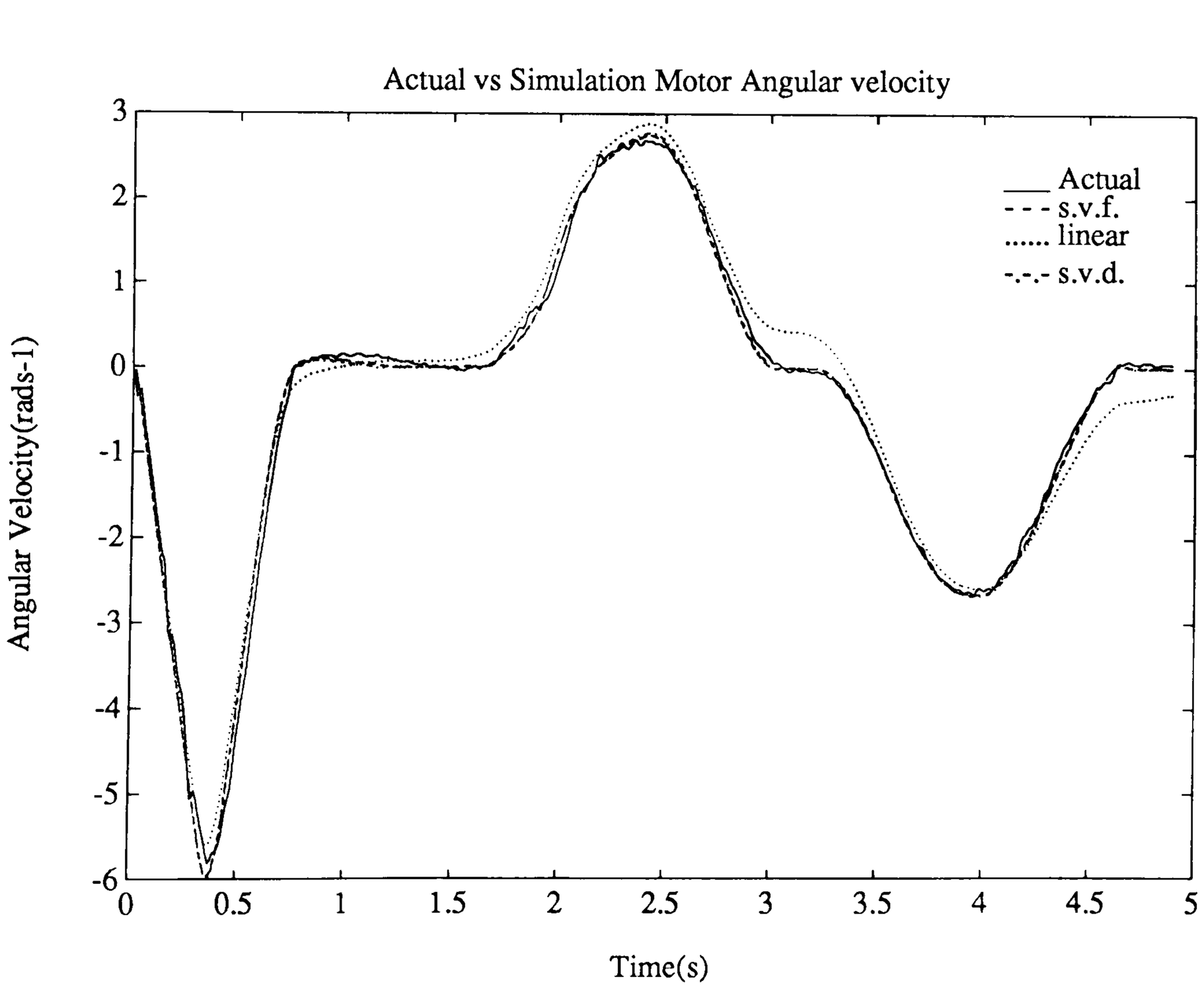
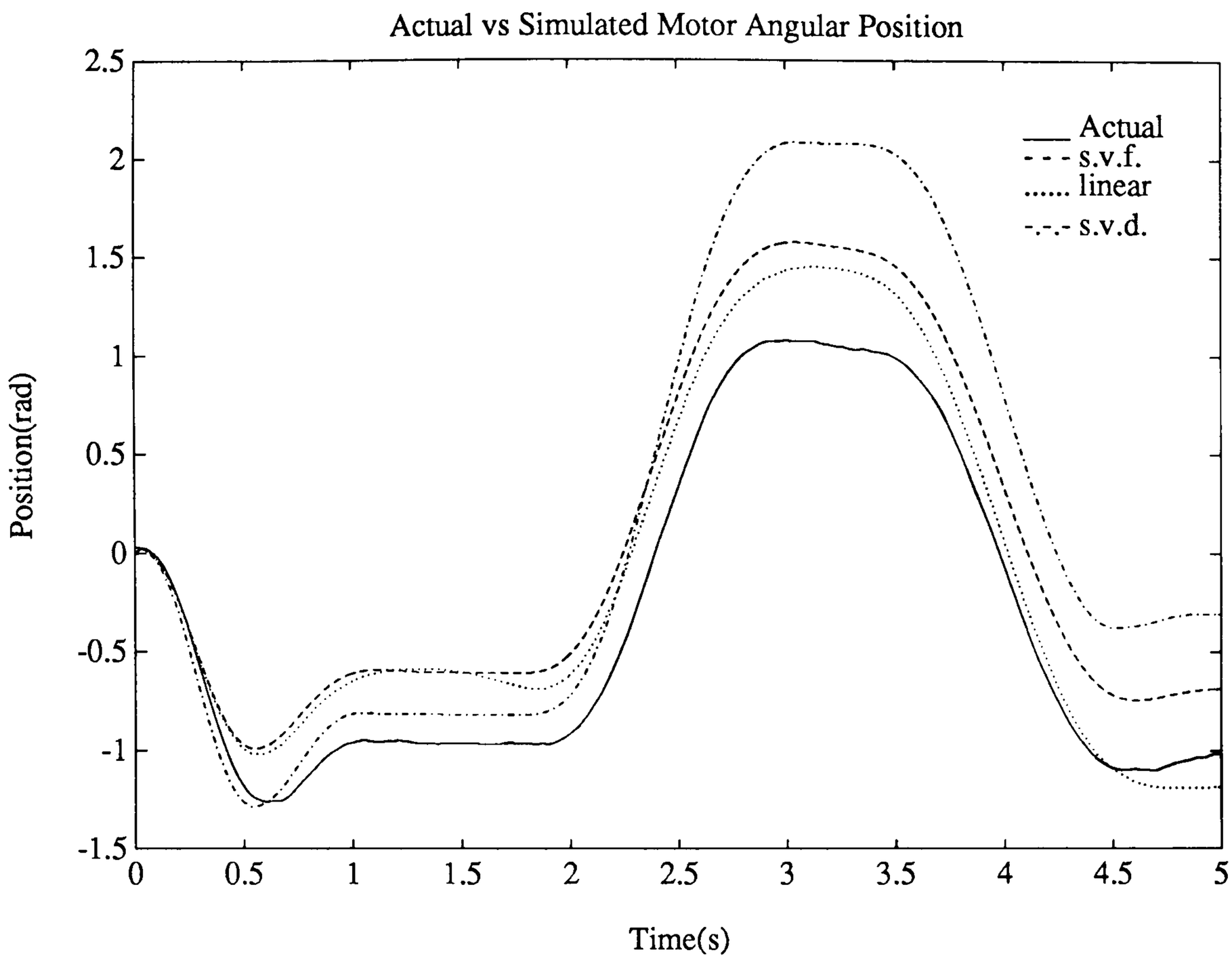


Figure 4.9: Test 4: First motor - first link with second motor attached.



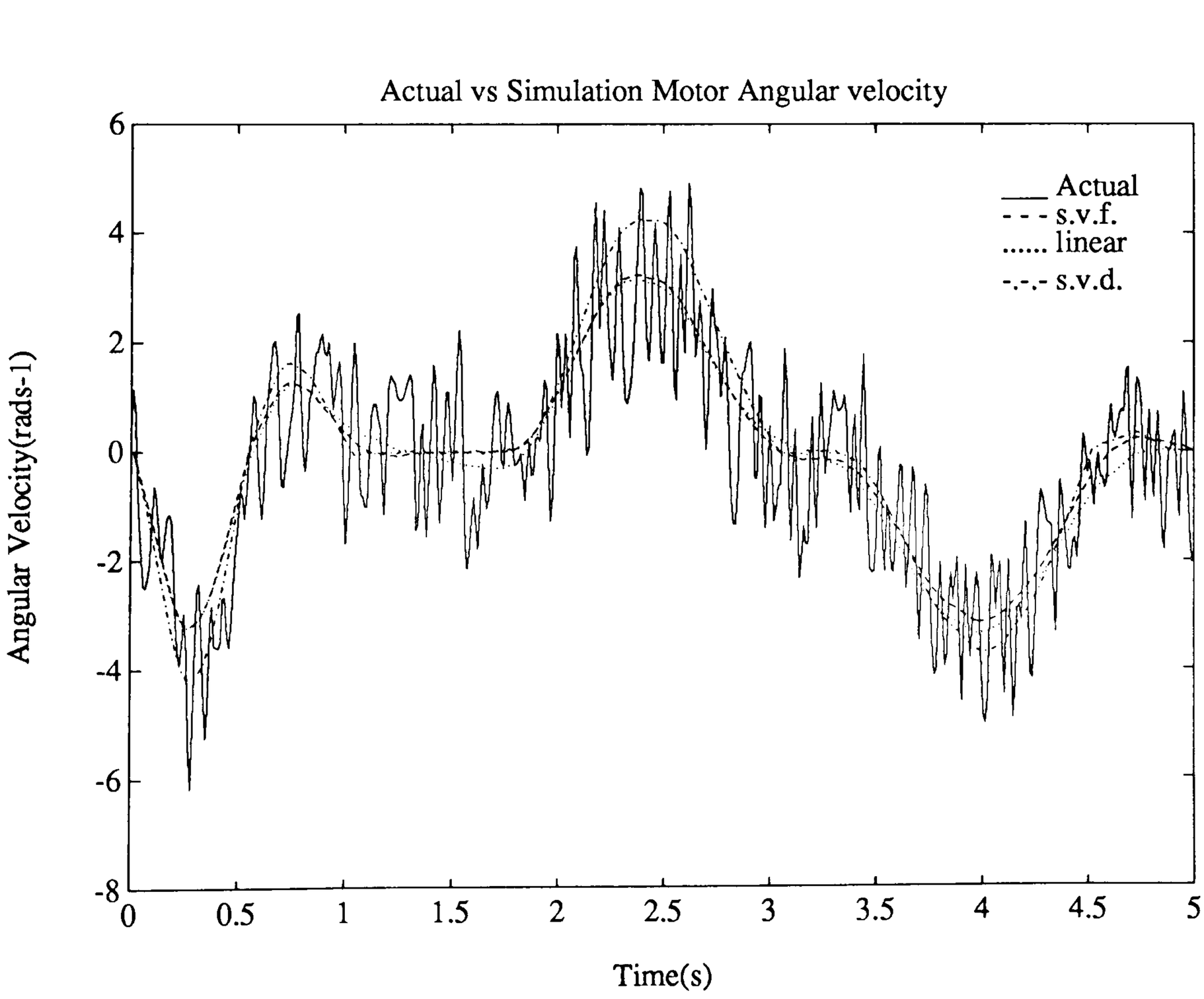


Figure 4.10: Test 5: Second motor - second link attached.

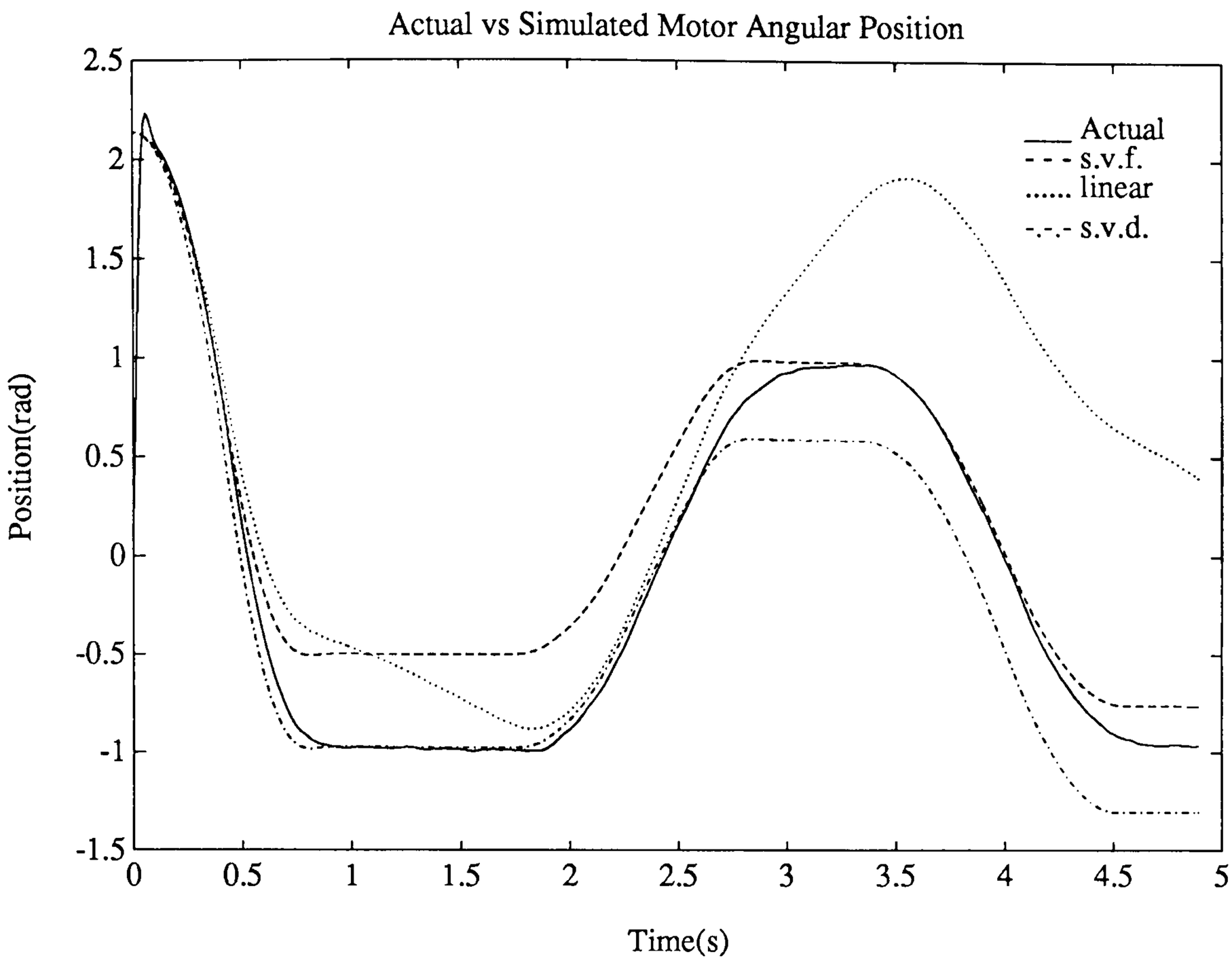


Figure 4.11: Test 6: Second motor - second link attached.

to the system. There is no significant difference between the state variable filter identified parameters and the singular value decomposition identified parameters.

The identification of inertia was reasonably accurate for all the identification routines but all found it difficult to identify the linear viscous coefficient, B, probably due to it being so low and dominated by stiction at the low angular velocities attained by the two link manipulator. The linear identification routine seems to confuse non-linear stiction for viscous friction leading to gross over-estimates of this parameter.

The condition number quoted for each identification routine in each test gives a measure of how easily the parameters were identified; the lower the condition number the better. It is the ratio of the highest singular value of the data matrix to the lowest. The large differences in condition number between identification routines for a given test are explained by the following:

- The state variable filter routine squares the data matrix thus squaring the condition number as well.

- There are only two singular values in the data matrix for the linear routine compared to four for the non-linear identification routines.

- The MATLAB routine which calculates the singular values ignores values which fall below a threshold value as it considers that these are caused by round-off errors. True, but small, singular values may therefore be lost.

For these reasons, condition numbers are not a reliable indication of good parameter estimates. It is better to rely on comparison of simulations using the identified parameters against data obtained from the actual system. When this is done it can be seen that the estimates from test 3 seem to provide the best simulation of the first link.

Motor 2.

The most striking difference between the first and second joints is the quality of the signal from the second tachometer. Figures 4.10 and 4.12 show how poorly conditioned the angular velocity measurement is even after filtering.

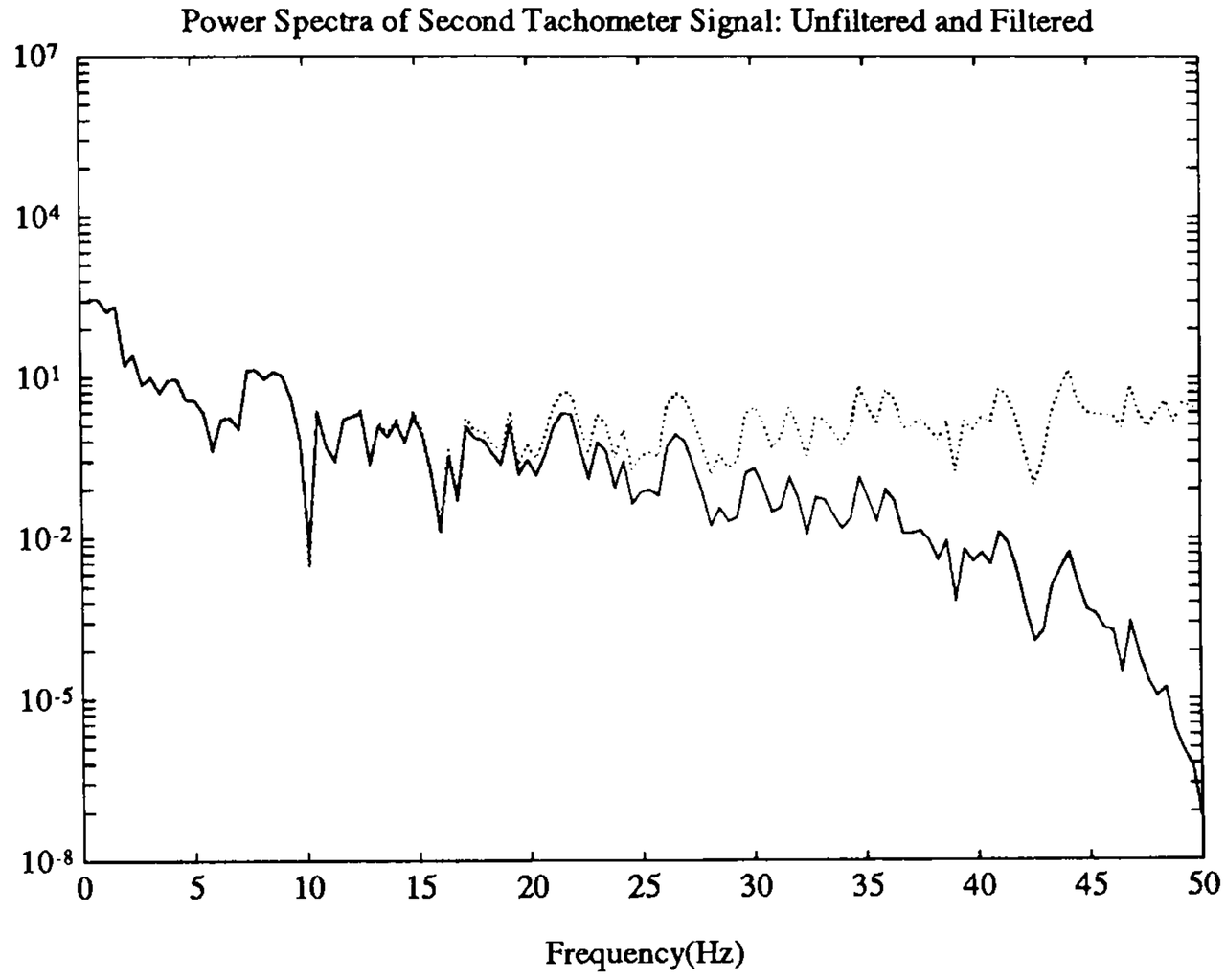


Figure 4.12: Spectral Density of Second Tachometer Signal before and after Filtering.

The poor tachometer signal makes it difficult for the identification routines to estimate the parameters as demonstrated by the high condition numbers in table 4.3. The state variable filter derived values for test five show remarkable agreement with the friction parameters quoted for the second motor and produce a good simulation of angular velocity. As the inertial parameter is also very close to the calculated inertia, these parameters are adopted for the second link.

4.3 Model Validation.

Validation of the bond-graph model was achieved by comparing data obtained from the DD2lm with simulations from the mathematical model using the parameters identified in the previous section.

	Test Five			Test Six			Quoted
	s.v.f	s.v.d.	linear	s.v.f	s.v.d.	linear	
I (kgm^2)	0.0132	0.0098	0.0135	0.0116	0.0098	0.0135	0.0130
B ($\frac{Nm}{rads^{-1}}$)	0.0000	-0.0062	0.0060	-0.0026	-0.0031	0.0015	0.0000
f_1 (Nm)	0.0152	0.0290	-	0.0323	0.0352	-	0.015
f_2 (Nm)	0.0158	0.0317	-	0.0253	0.0275	-	0.015
Cond.No.	3403	28.8	3.84	586	22.9	3.11	

Table 4.3: Identified Parameters for Second Joint.

4.3.1 Experimental Tests.

Actual data from the fully assembled experimental manipulator was obtained using a simple independent joint proportional and derivative (or velocity) controller tuned to give an approximately critically damped response for each joint. Again, the controller is of little importance in the model validation tests as we merely wish to gather data with the links interacting dynamically. Each test lasted for five seconds at a sample rate of 100Hz.

Four tests were carried out with the links following four different trajectories as shown in figure 4.13. The trajectories for tests 7,8 and 9 were chosen to induce high link interactions between the links whilst the trajectory for test 10 was chosen as a more typical manipulator move.

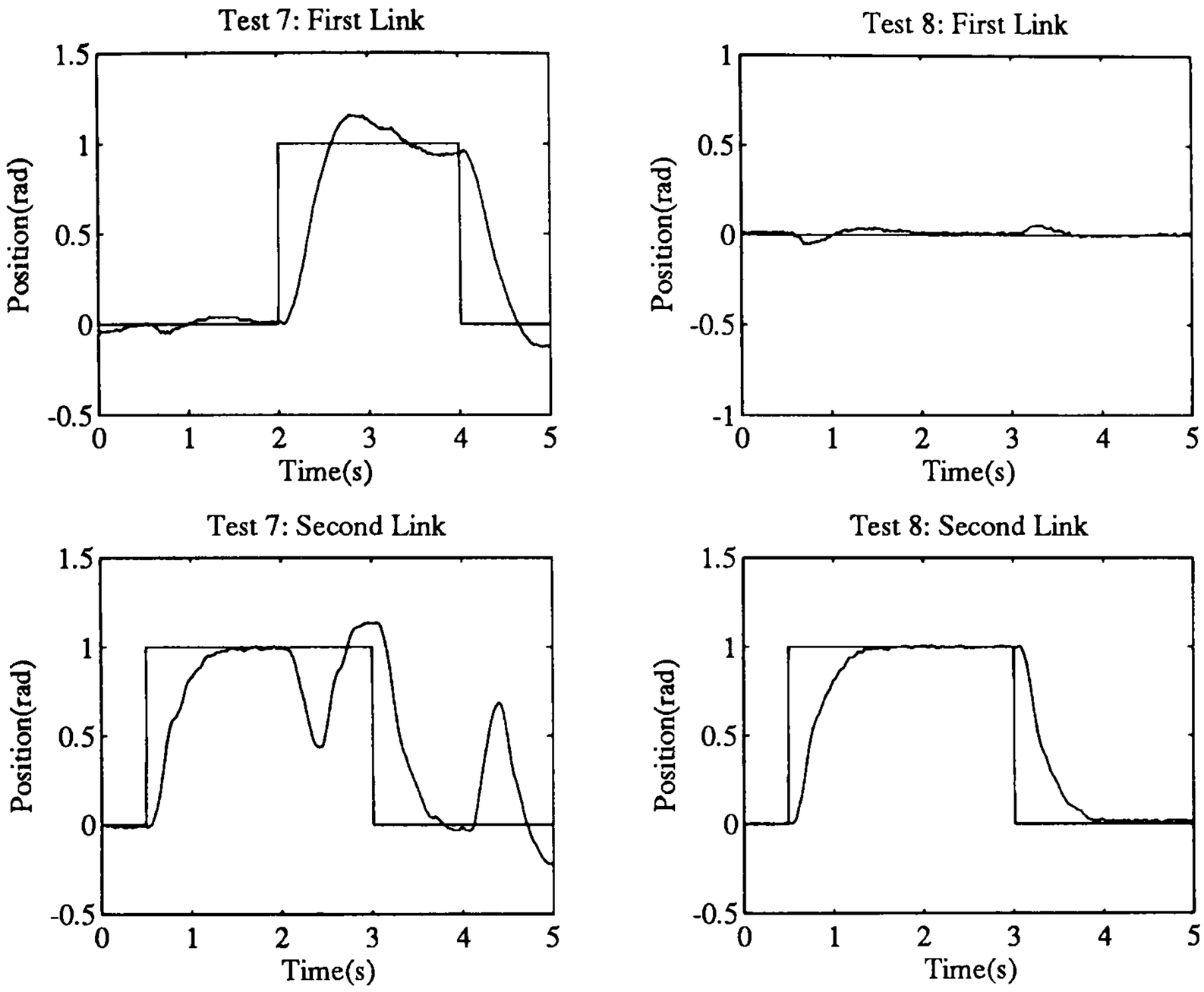


Figure 4.13: Test Set-Point and Attained Trajectories.

Simulations.

Simulations were carried out in two ways:

- Closed loop simulations.
- Open loop simulations.

The closed loop simulations used the automatically derived equations of motion from the bond graph in the simulation package SIMULAB [21] with the same controller as used for the experimental tests modelled as shown in figure 4.14. The response of each of the three friction models to the set-point trajectories used for the experimental tests could then be plotted against the response of the DD2lm.

The three friction models are

- no joint friction.
- linear viscous friction.
- stiction.

Variable time-step simulations with a fifth order Runge-Kutta integration algorithm were used throughout. The parameters used in each model are given in table 4.4.

Open loop simulations were obtained by using the input motor voltages, measured during the experimental tests, as inputs to each of the three models. The outputs from the models were then compared with the output from the DD2lm. The SIMULAB configuration for the open loop simulations is shown in figure 4.15.

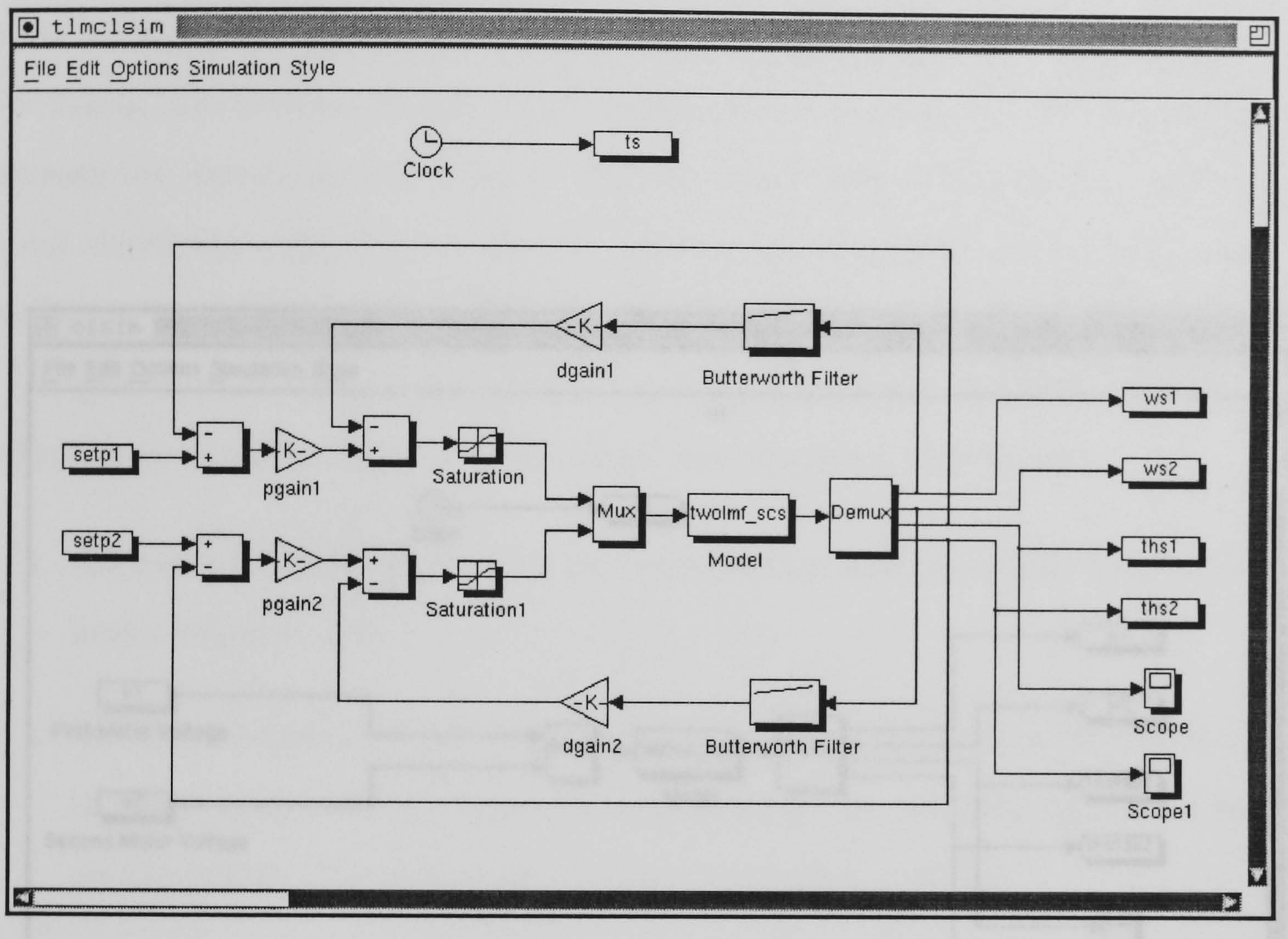


Figure 4.14: SIMULAB configuration for closed loop simulations.

	First Link		Second Link	
	Linear Friction	Stiction	Linear Friction	Stiction
B ($\frac{Nm}{rads^{-1}}$)	0.03	0.005	0.003	0.0000
f_1 (Nm)	-	0.04	-	0.015
f_2 (Nm)	-	0.055	-	0.015

Table 4.4: Friction Parameters used for Simulations.

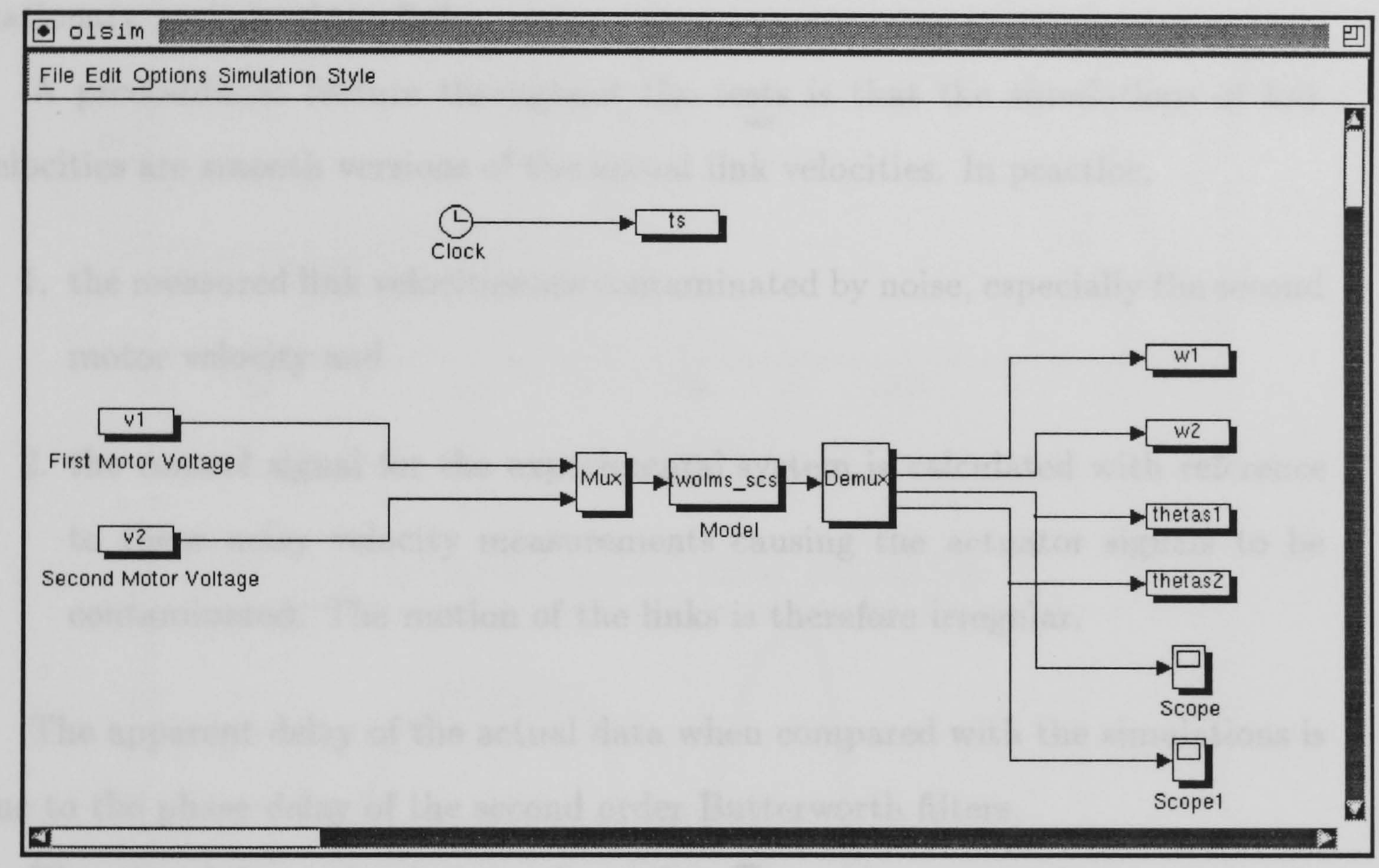


Figure 4.15: SIMULAB configuration for open loop simulations.

4.3.2 Results and Discussion.

Closed Loop Simulations.

It can be seen from figures 4.16 to 4.23 that the mathematical model is capable of simulating the dynamics of the two-link manipulator accurately. The difference between the models incorporating no friction, linear viscous friction and stiction is not significant although the stiction model is more capable of predicting the stationary periods of the links.

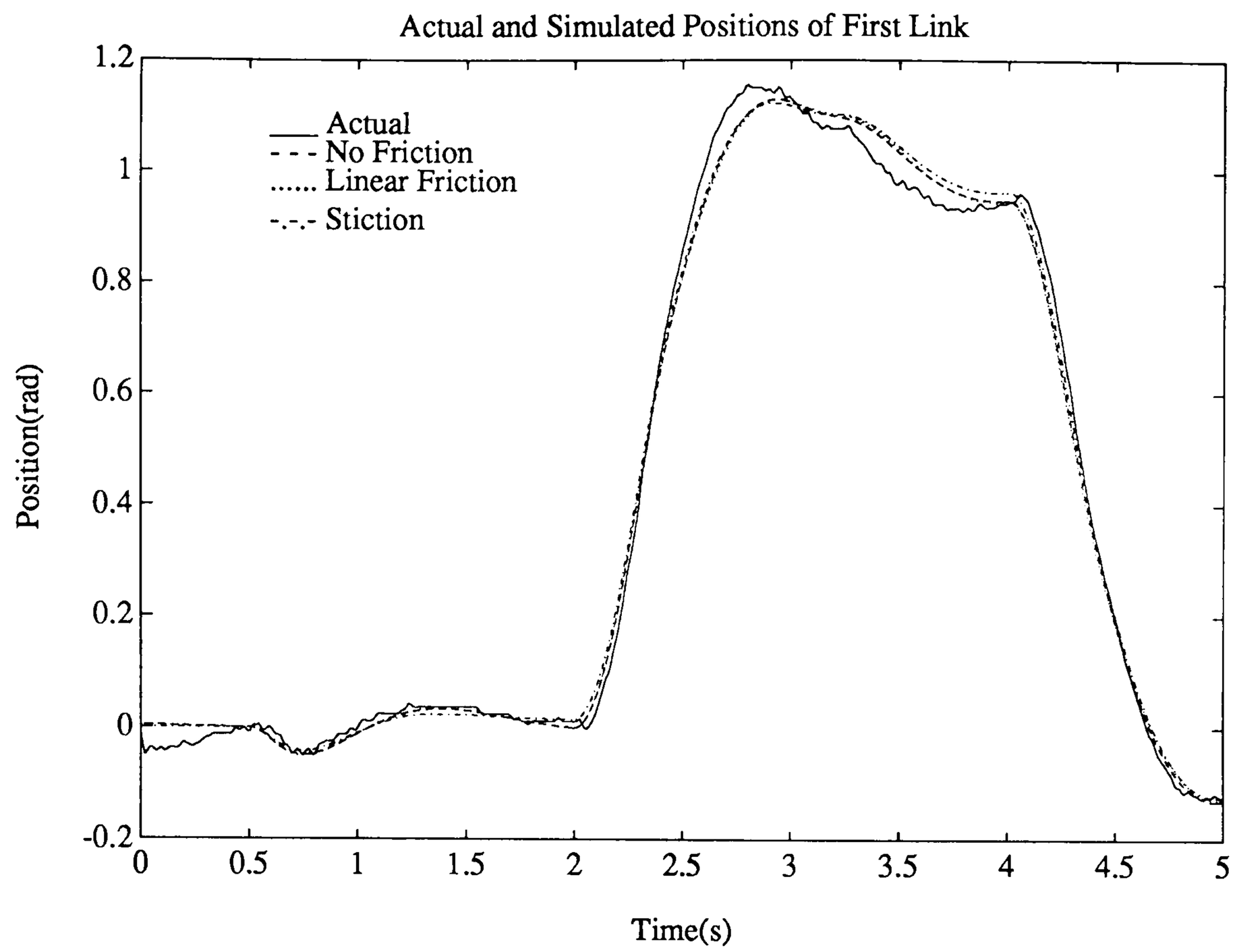
A predominant feature throughout the tests is that the simulations of link velocities are smooth versions of the actual link velocities. In practice,

1. the measured link velocities are contaminated by noise, especially the second motor velocity and

2. the control signal for the experimental system is calculated with reference to these noisy velocity measurements causing the actuator signals to be contaminated. The motion of the links is therefore irregular.

The apparent delay of the actual data when compared with the simulations is due to the phase delay of the second order Butterworth filters.

The closed loop simulations show that there is no great improvement in model accuracy between the no friction model and the stiction model despite the considerable increase in model complexity.



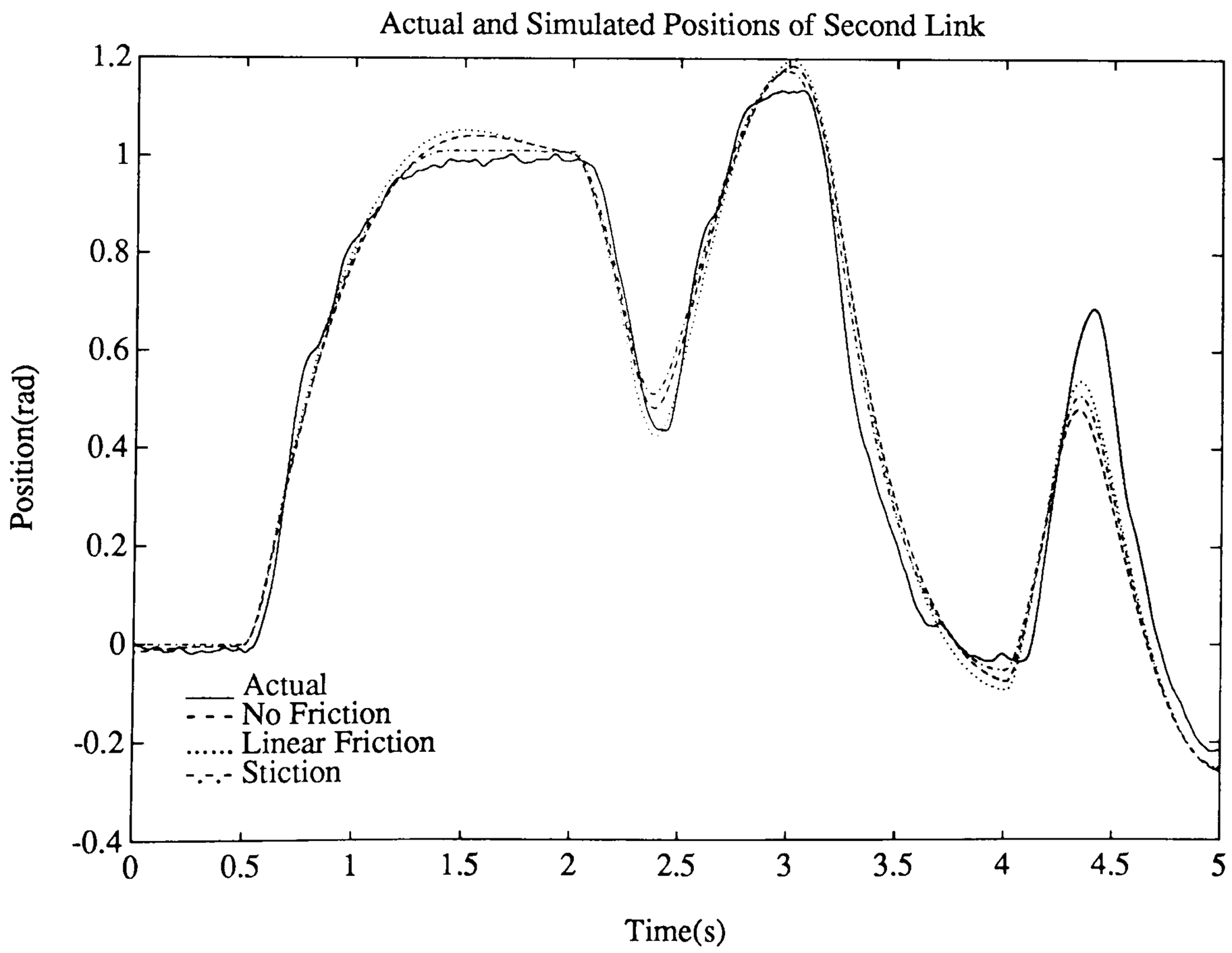


Figure 4.16: Test 7: Actual and Simulated Link Positions.

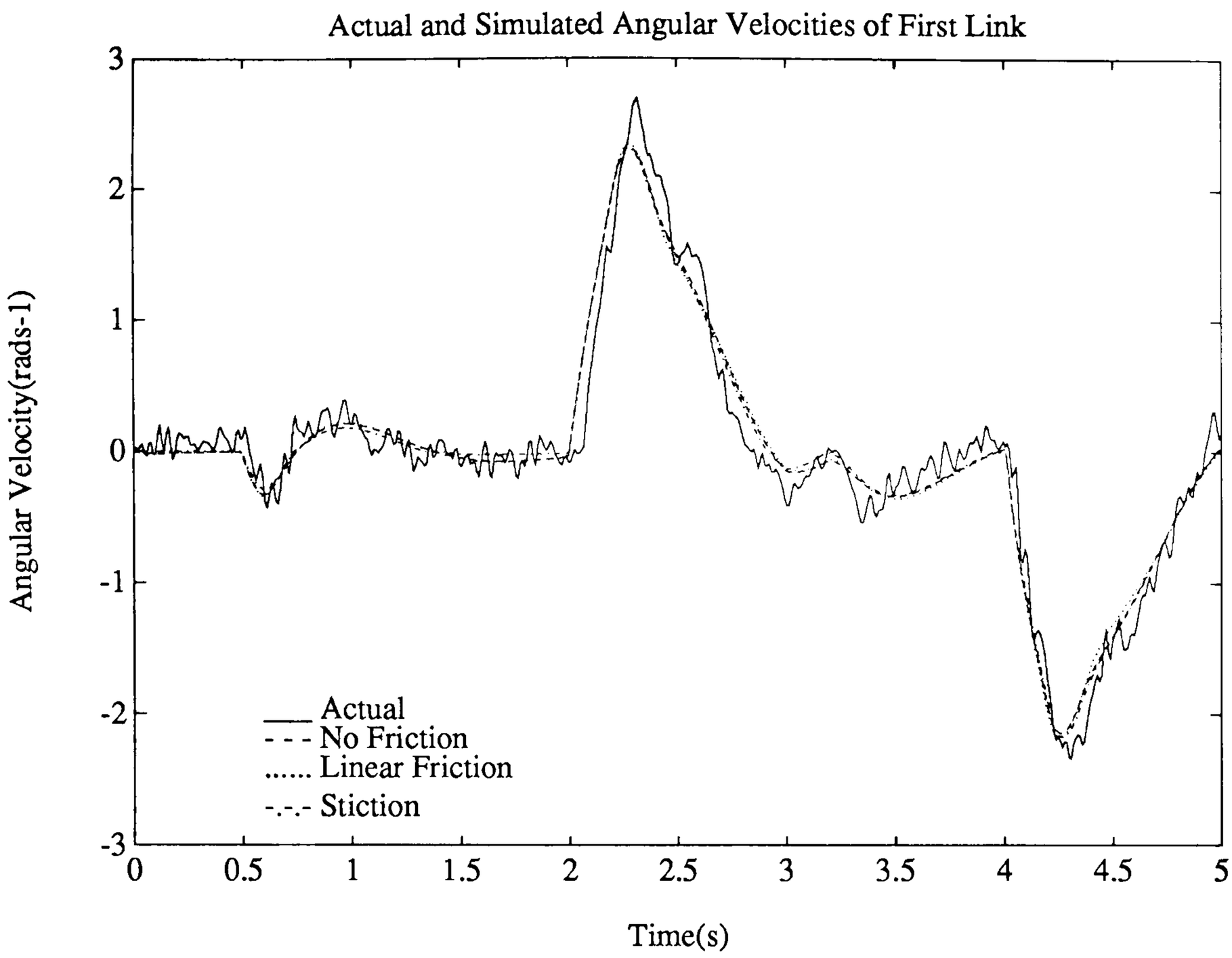
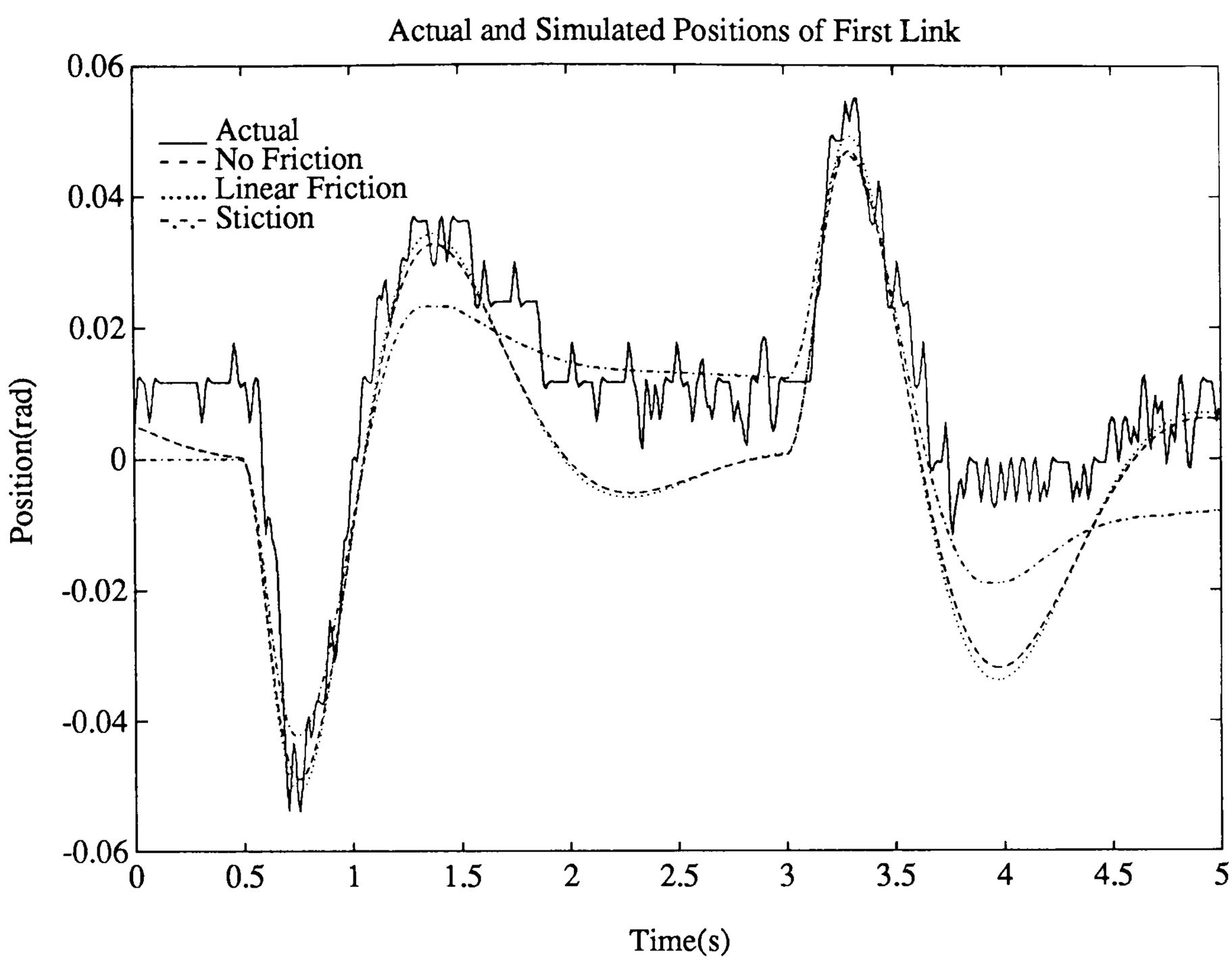


Figure 4.17: Test 7: Actual and Simulated Link Velocities.



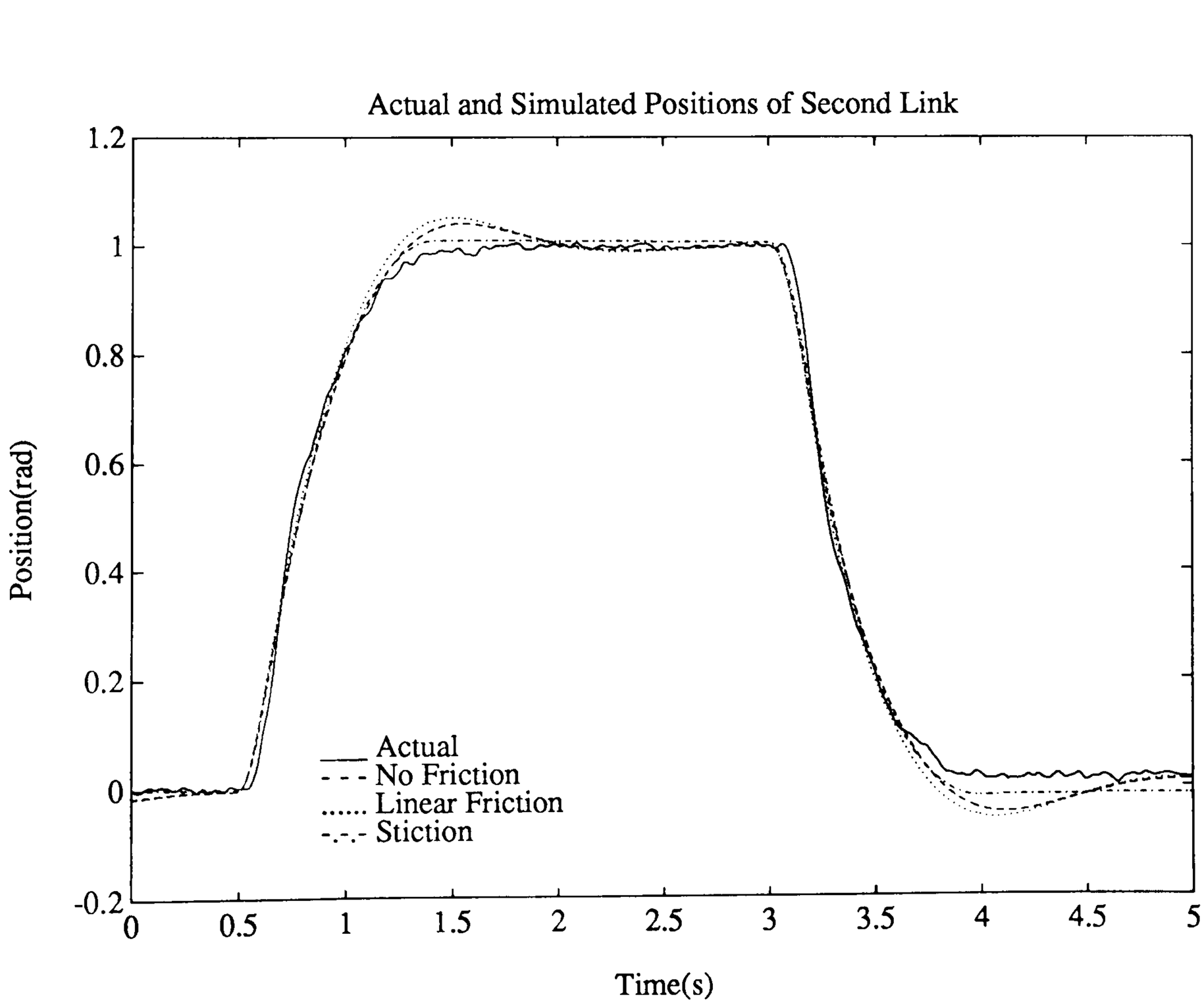


Figure 4.18: Test 8: Actual and Simulated Link Positions.

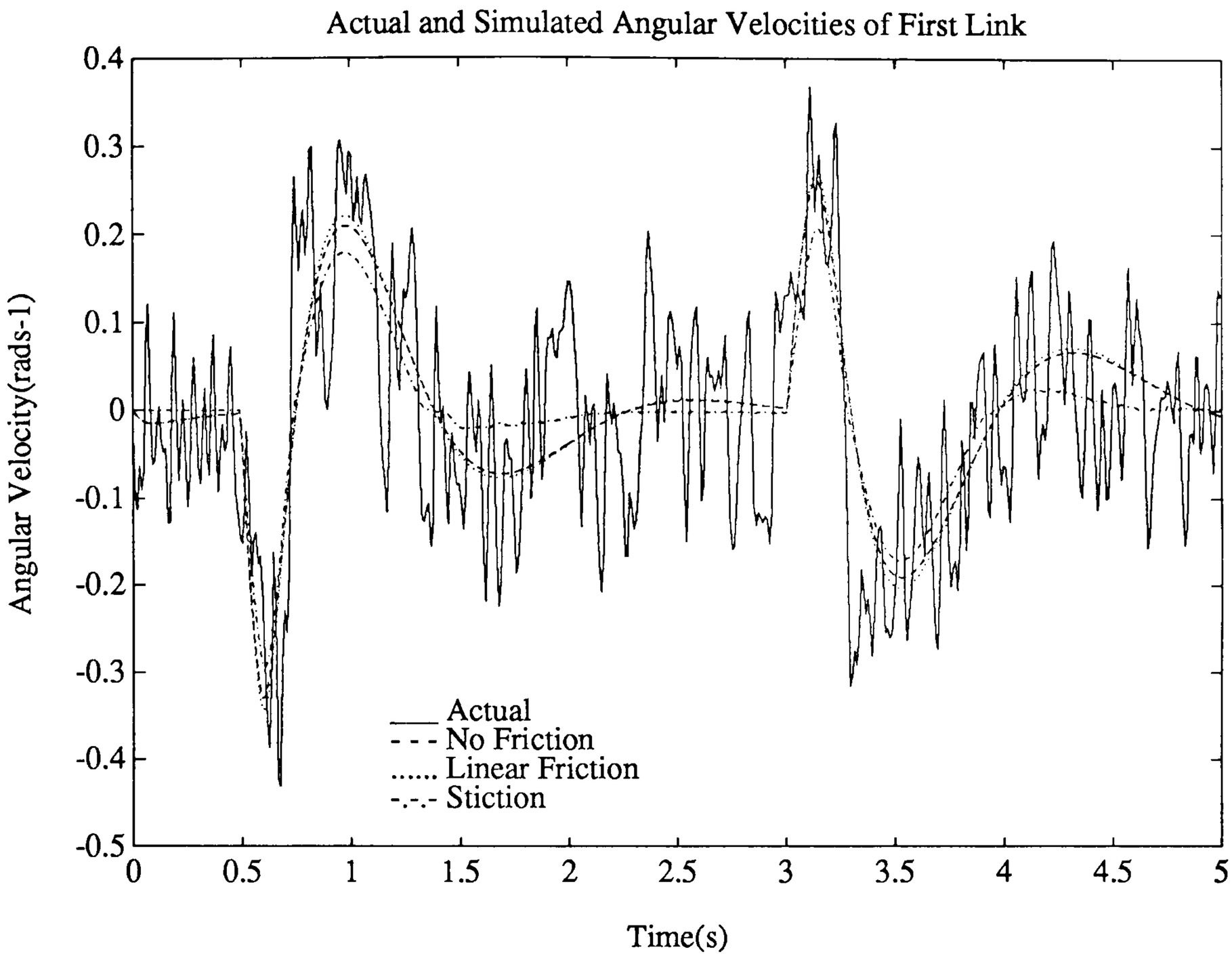
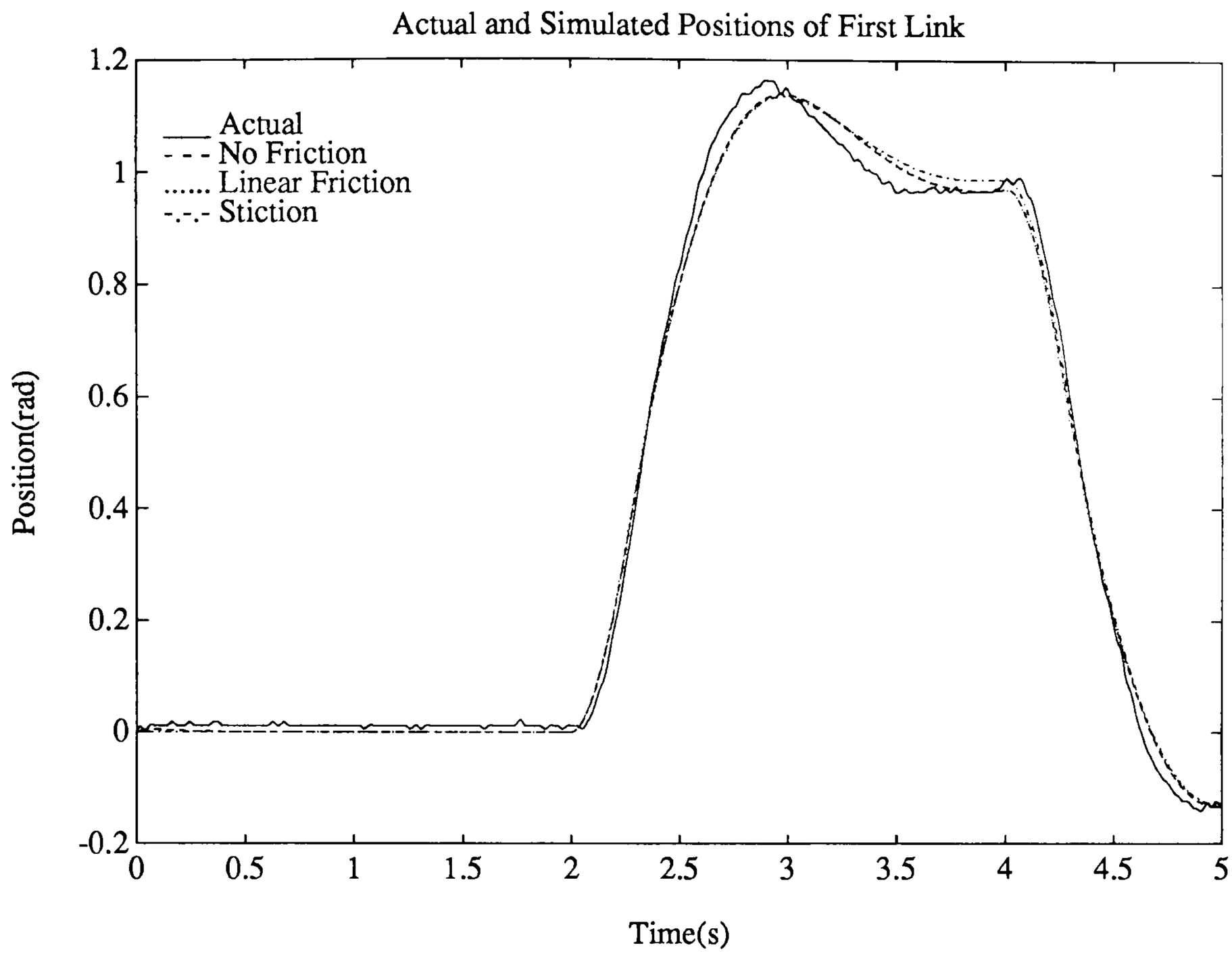


Figure 4.19: Test 8: Actual and Simulated Link Velocities.



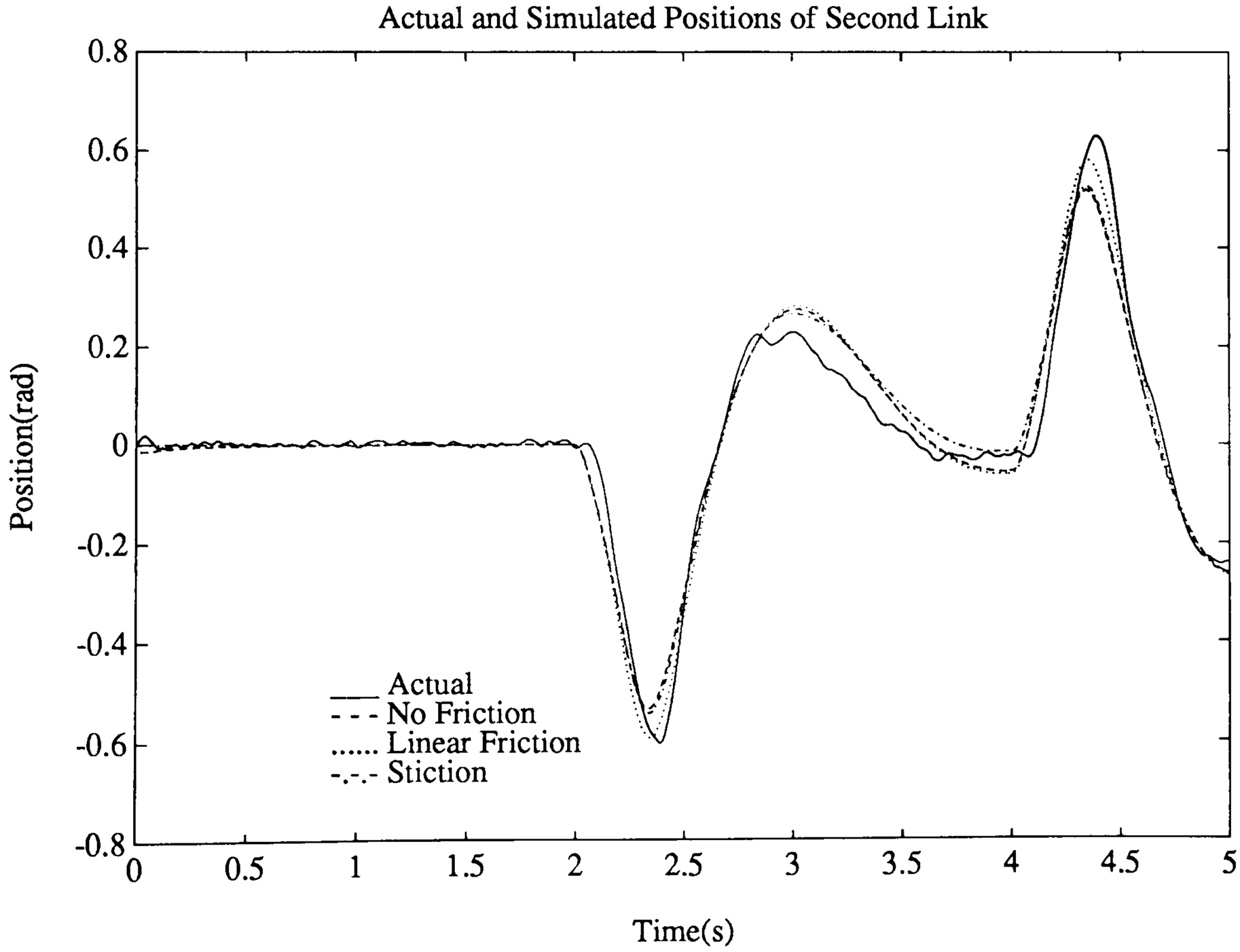


Figure 4.20: Test 9: Actual and Simulated Link Positions.

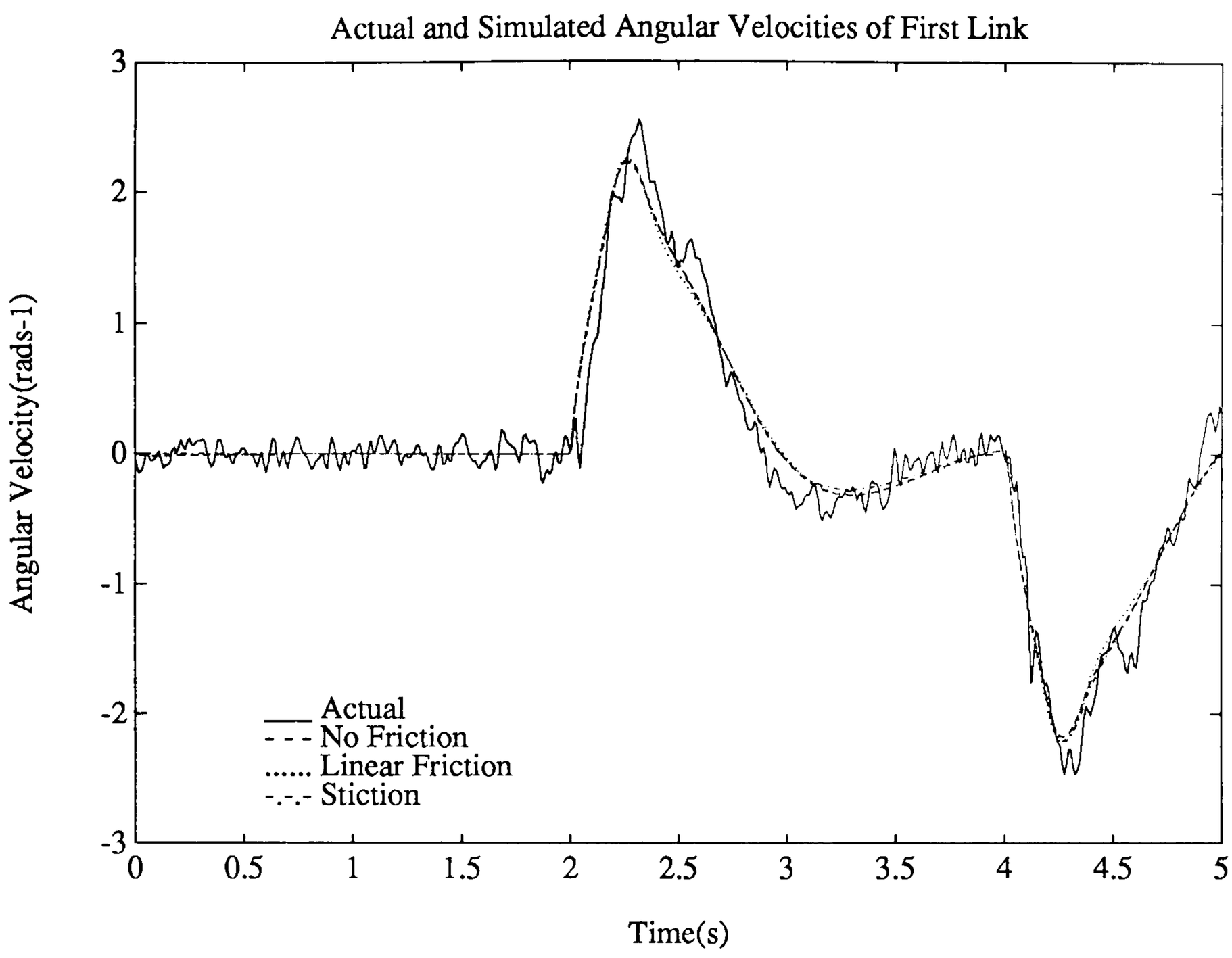
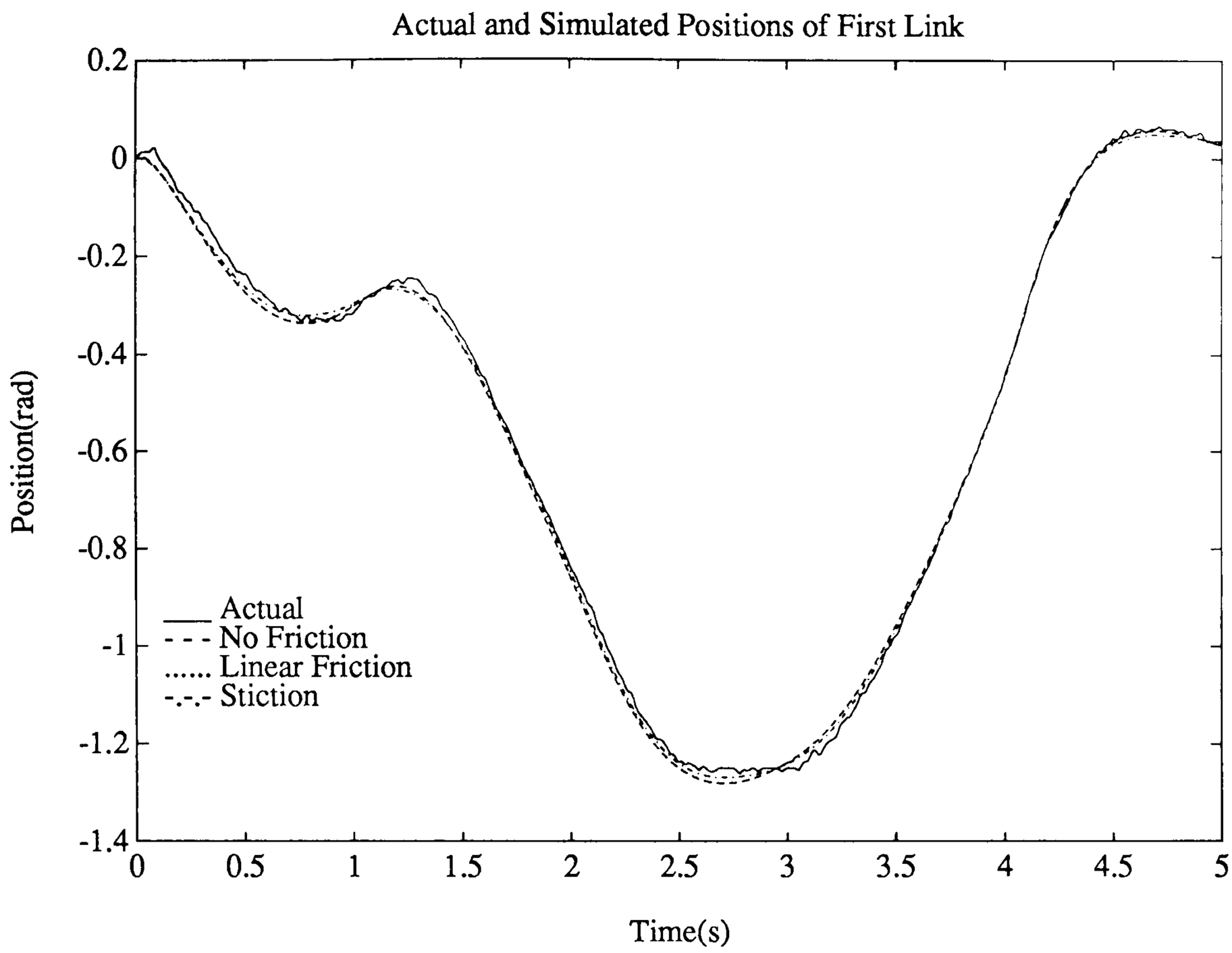


Figure 4.21: Test 9: Actual and Simulated Link Velocities.



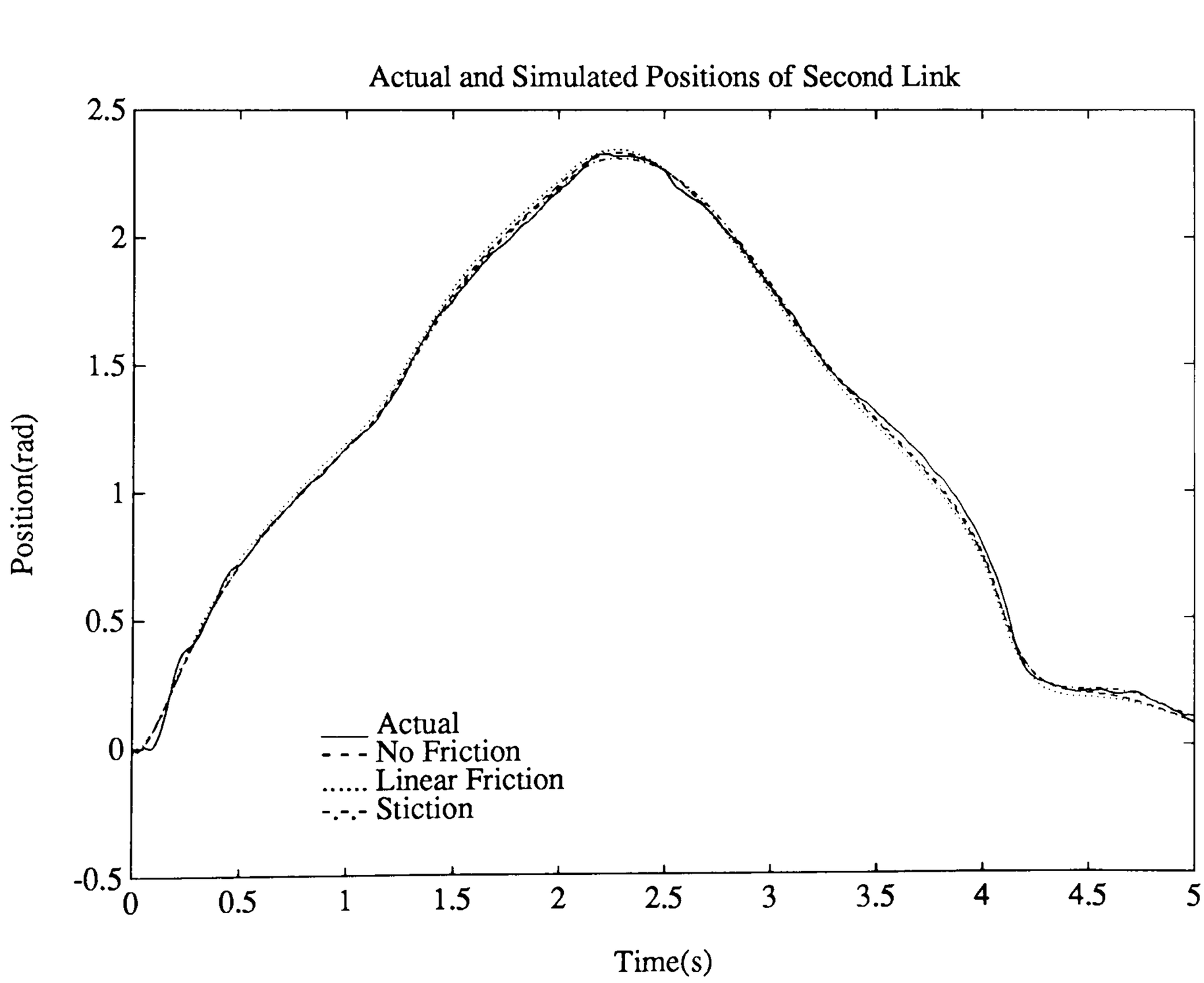
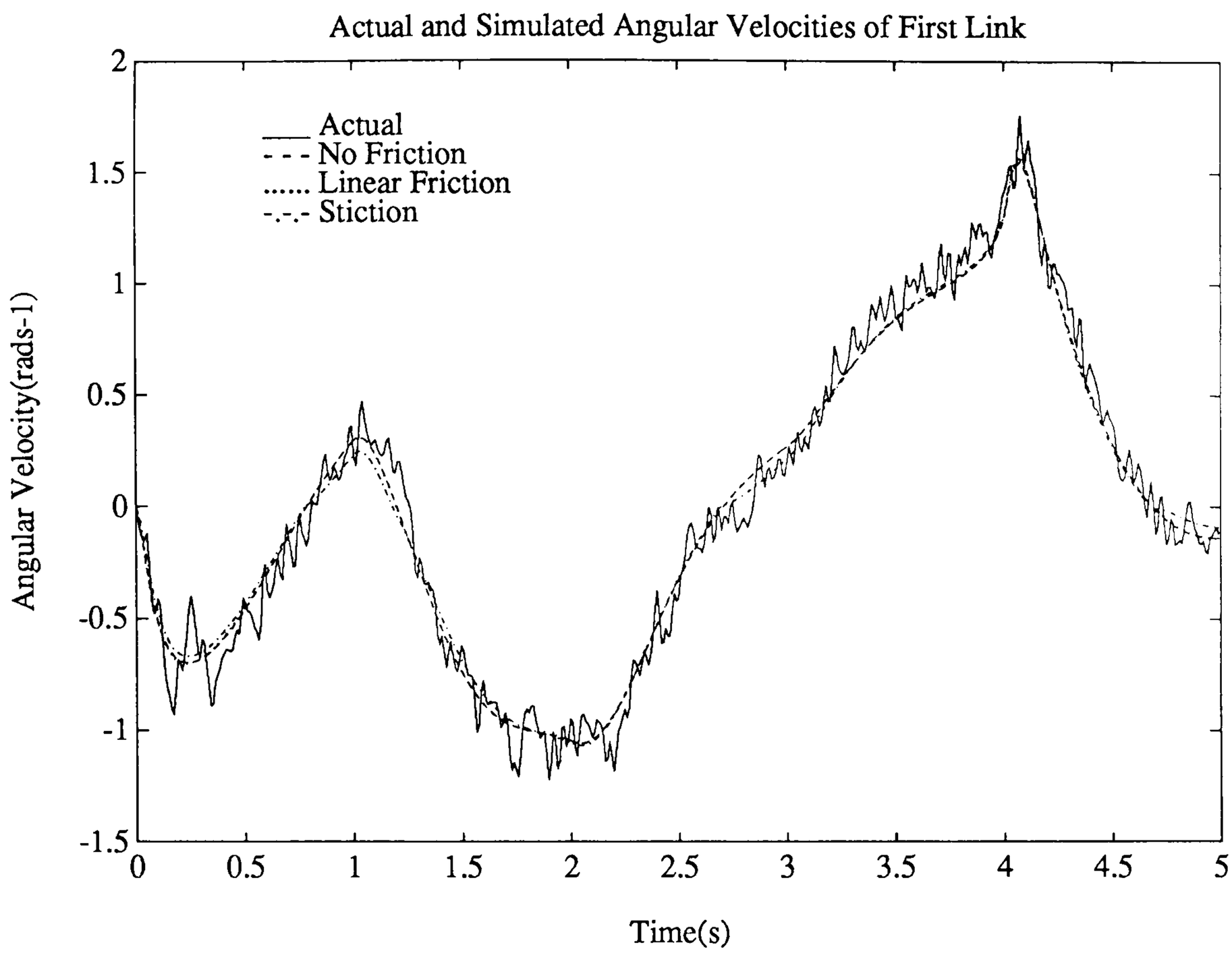


Figure 4.22: Test 10: Actual and Simulated Link Positions.



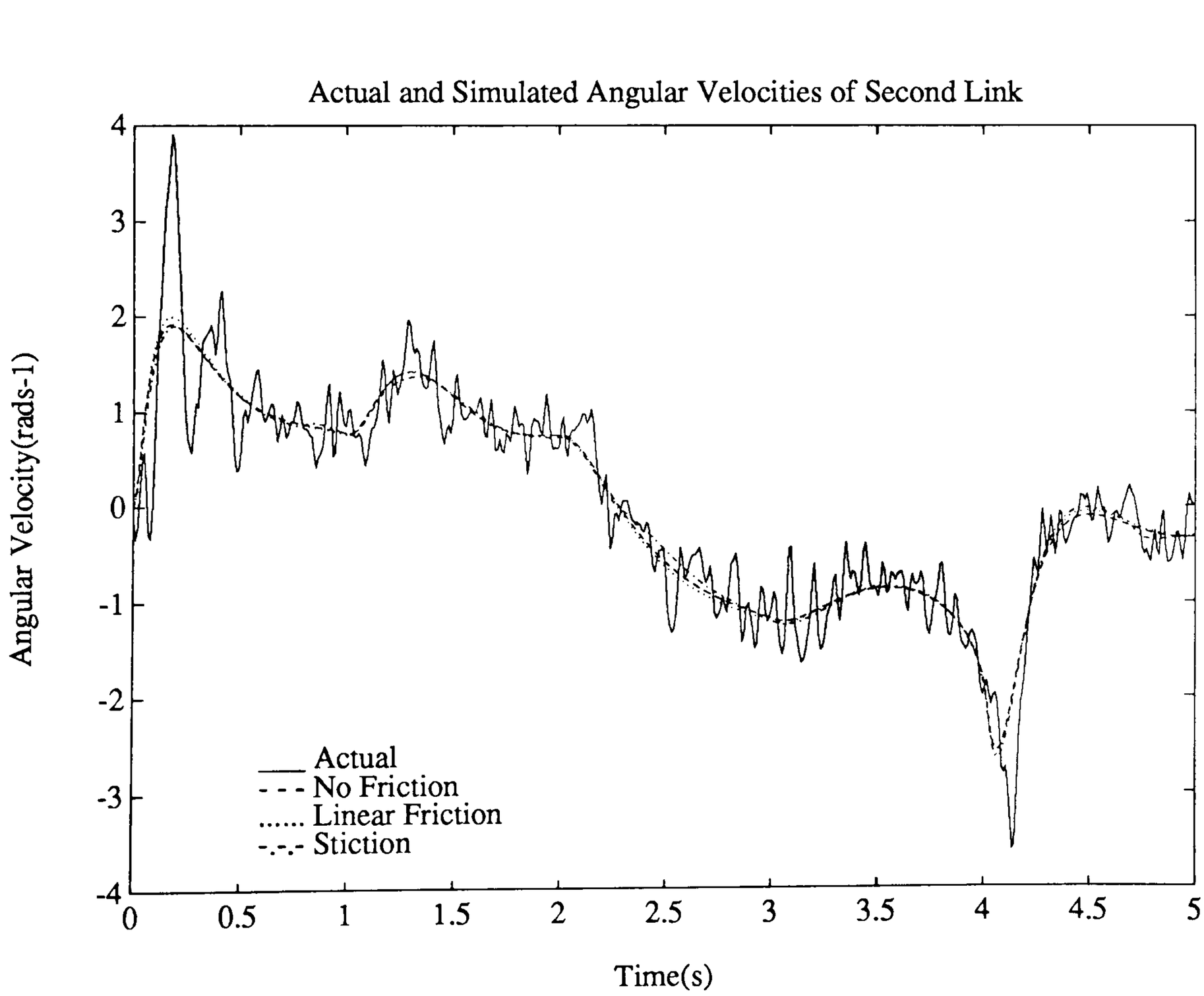


Figure 4.23: Test 10: Actual and Simulated Link Velocities.

Open Loop Simulations.

The difference between the friction models becomes more marked when open loop simulations are performed as shown in figure 4.24 which uses test 7 as an example.

None of the models provides an accurate simulation of the manipulator but the linear friction model is particularly poor due to the inaccurately identified viscous friction parameter.

These results show that if the model were to be used in an open loop format the friction model would need to be refined.

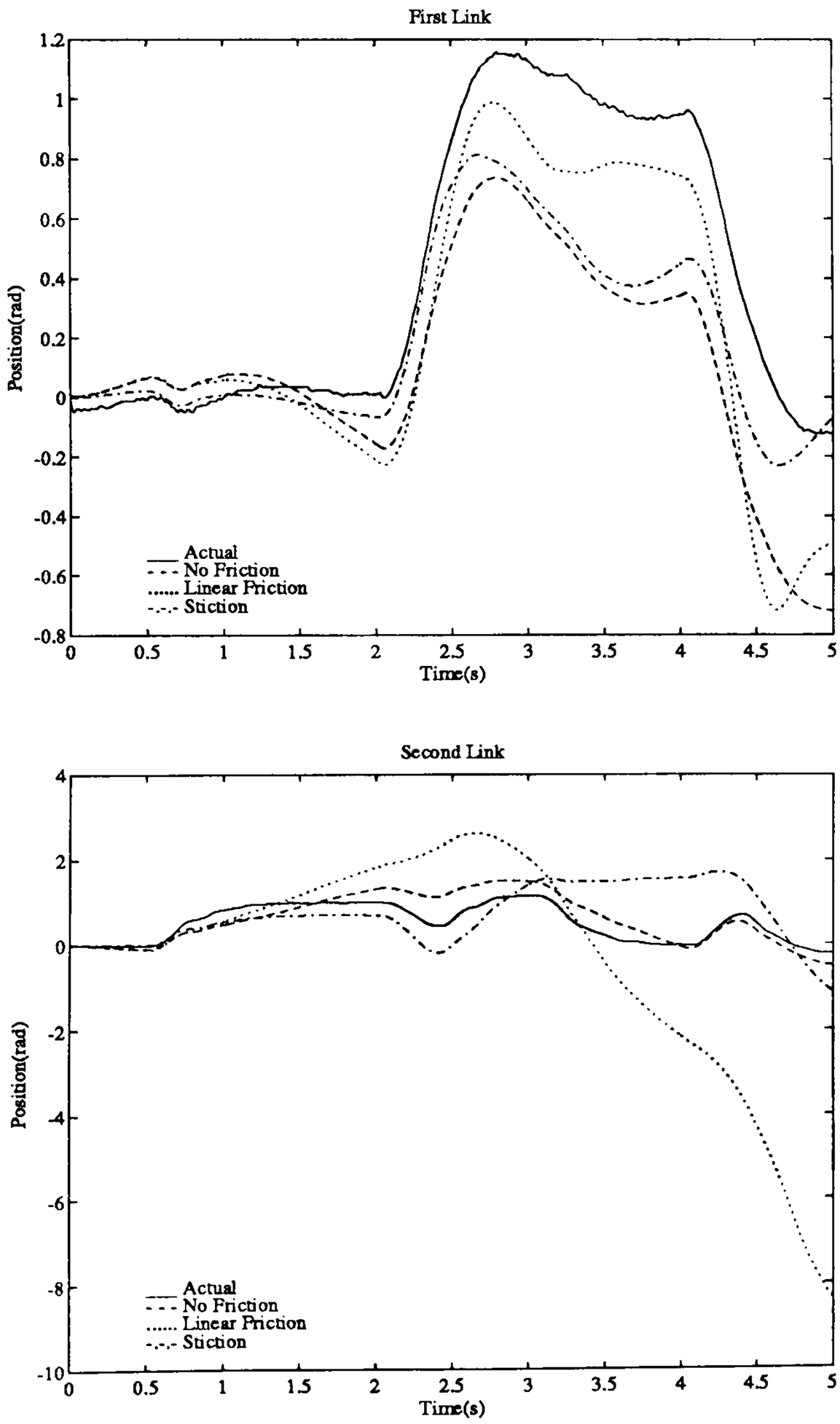


Figure 4.24: Test 7: Open Loop Simulations.

4.4 Robustness.

To gauge the robustness of the mathematical model to parameter variations, a number of closed loop simulations were performed where a particular parameter was altered from its measured or identified value.

The following parameters were tested in this way.

- Mass of second motor (affects the inertia of the first link).
- Mass of the second link.
- Torque constant of the first motor.
- Torque constant of the second motor.
- Stiction parameters of the first joint.
- Stiction parameters of the second joint.

The results of the robustness tests are shown in figures 4.25 to 4.26. Each graph shows how doubling or halving the value of the named parameter affects the closed loop simulation for the trajectory used in test 7. The experimental results for test 7 are shown for comparison. All other model parameters remain unchanged from their nominal values used to obtain the simulation in figure 4.16.

It is apparent from the results that the model is most sensitive to changes in the second motor torque constant and the inertia of the second link. The model is less sensitive to first link parameters. The model is robust to changes in the stiction parameters.

These results are encouraging as they demonstrate that the model is most robust to those parameters that are the most difficult to identify; namely stiction. The parameters which the model is sensitive to, such as torque constants and link inertias, are either accurately quoted in the motor specifications or can be easily calculated.

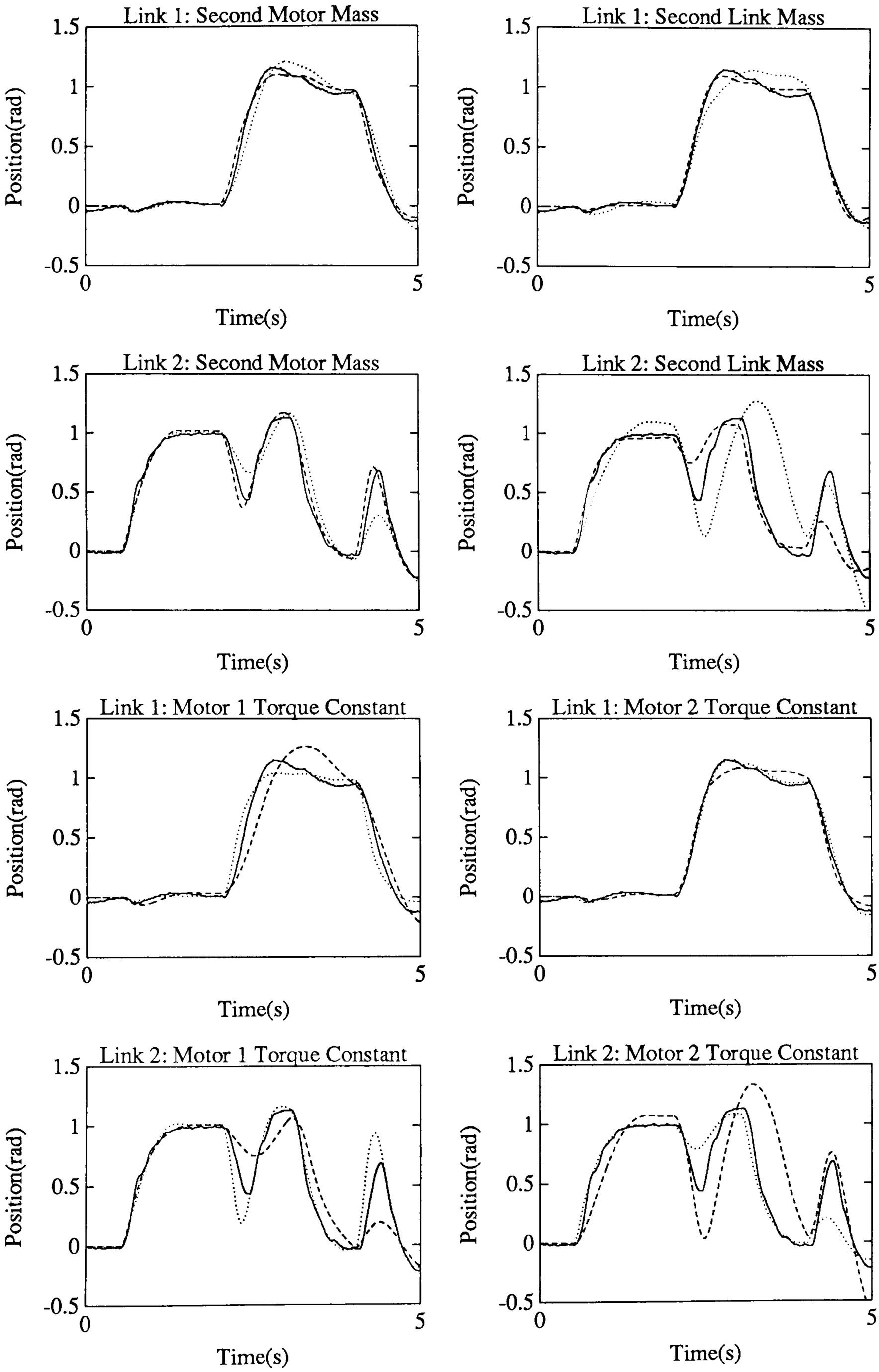


Figure 4.25: Robustness Tests.

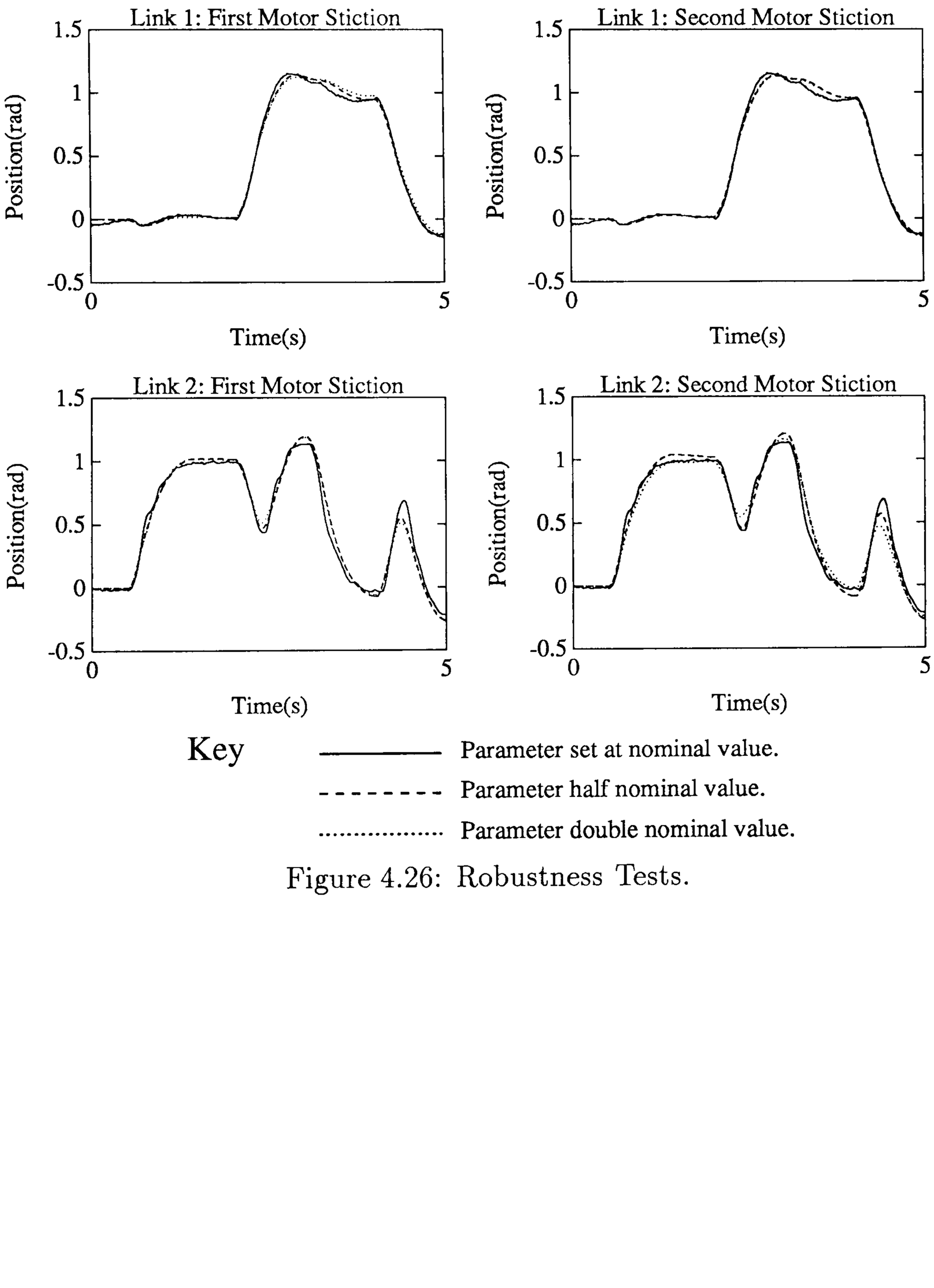


Figure 4.26: Robustness Tests.

4.5 Conclusions.

1. The bond graph model is capable of describing the dynamics of the experimental two link manipulator accurately.

2. For closed loop simulations, the modelling of joint friction is not critical but modelling stiction improves the accuracy.

3. For open loop simulations, refinement of the friction model would be required.

4. The model is robust to parameter variations.

5. The parameters that the model is most sensitive to are those which are most easily measured, such as link inertias, or are accurately quoted in the motor specifications.

6. The model is more sensitive to the parameters of the second link.

It has been shown that for the use of the model in a closed loop format, modelling of joint friction is not necessary to achieve accurate simulations of manipulator movements. If joint friction is to be modelled it is important that linear viscous friction is not identified in isolation as the presence of stiction severely degrades the identification.

A major ability of the model is that it is capable of predicting the motion of the experimental manipulator free of noise. It is this ability that will be exploited in the next chapter.

Chapter 5

Bond Graph Model Based Observer.

5.1 Introduction.

The control of the experimental two-link manipulator is limited by the contamination on measurements of link angular velocities. Noise contamination of tachometer signals and its restriction on the attainable closed-loop bandwidth of controllers is a generally recognised problem associated with robotic control [26] [45] [46] [47] [2] [48] [49] [50]. In this chapter we explore how the bond graph may be used to create a model based observer to improve the control of the manipulator by using estimated link angular velocities in the feedback controller.

The use of observers in this area is not new. Canudas de Wit and Slotine [51] used a sliding observer (a class of variable structure non-linear system) to estimate the joint speeds of rigid robots whilst Nicosia, Tomei and Tornambe [52] used a pseudo linearisation technique involving a dynamic state-space change of co-ordinates to linearise the non-linear dynamics of a robotic manipulator from which a linear Luenberger observer could be implemented. Canudas de Wit, Åstrom and Fixot [26] modified the sliding observer approach in 1990 to produce

a smooth non-linear observer (an observer with smooth, differentiable gains) in which the switching gains of the sliding observer were replaced with differentiable non-linear functions to yield an exponentially stable observer. The use of observers for flexible joint robots has also been proposed by Nicosia , Tornei and Tornambe [53] [54] [55], Tomei [56] and Hun *et al* [57].

Whilst the above approaches are highly mathematical in nature, the bond graph observer provides a graphical method to obtain the algorithm and software required to implement a non-linear model-based observer. This chapter deals first with the modifications required to turn the bond graph for the DD2lm into a form suitable for the creation of the observer using techniques developed by Karnopp [22] and Gawthrop [58]. It then outlines the method for selecting the observer feedback gain matrix required to make the states of the observer track the states of the system. Finally, the observer is implemented practically to gauge how effective it is at improving the performance of the experimental manipulator.

The experimental results presented in this chapter were obtained without the use of filters to prevent aliasing of the sampled signals. Appendix C repeats the tests presented in this chapter but employs anti-aliasing filters to prevent aliasing of the measured positions and angular velocities of the manipulator. It is seen that the resulting performance of the manipulator is degraded by the use of the anti-aliasing filters so they are not used in the collection of the main body of results.

5.2 Creation of the Observer.

To understand the requirements of the bond graph observer fully, it is instructive to review the standard form of state-space observer. The standard form for state space equations [59] is:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \tag{5.1}$$

$$\boldsymbol{y} = C\boldsymbol{x} + D\boldsymbol{u} \tag{5.2}$$

where

$\boldsymbol{x}$ = n×1 vector of states

$\boldsymbol{u}$ = m×1 vector of inputs

$\boldsymbol{y}$ = p×1 vector of outputs

A = n×n matrix

B = n×m matrix

C = p×n matrix

D = p×m matrix

An ideal observer has the form

$$\dot{\hat{\boldsymbol{x}}} = A\hat{\boldsymbol{x}} + B\boldsymbol{u} + T(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{5.3}$$

$$\hat{\boldsymbol{y}} = C\hat{\boldsymbol{x}} + D\boldsymbol{u} \tag{5.4}$$

where

$\hat{\boldsymbol{x}}$ = n×1 vector of state estimates

$\hat{\boldsymbol{y}}$ = p×1 vector of estimated outputs

T = n×p matrix of feedback gains

We can generate an error vector **e** from

$$\mathbf{e} = \boldsymbol{x} - \hat{\boldsymbol{x}} \tag{5.5}$$

Differentiating 5.5 gives

$$\begin{aligned}
\dot{\mathbf{e}} &= \dot{\boldsymbol{x}} - \dot{\hat{\boldsymbol{x}}} \\
&= A\boldsymbol{x} + B\boldsymbol{u} - A\hat{\boldsymbol{x}} - B\boldsymbol{u} - T(\boldsymbol{y} - \hat{\boldsymbol{y}}) \\
&= A(\boldsymbol{x} - \hat{\boldsymbol{x}}) - TC(\boldsymbol{x} - \hat{\boldsymbol{x}}) \\
&= (A - TC)\mathbf{e} \qquad (5.6)
\end{aligned}$$

Equation 5.6 is the equation which governs the dynamics of the error vector **e** . If the A,C matrix pair is observable it is theoretically possible to place the eigenvalues of the matrix (A-TC) to any desired position on the complex plane through selection of the feedback gain matrix T. The poles of equation 5.6 may therefore be arbitrarily fast causing the observer states to converge to system states arbitrarily quickly after which they will track system states perfectly. In practice, the speed of the poles is limited by observer model inaccuracies and numerical stability considerations.

5.2.1 Bond Graph Observers.

The method for augmenting observers in a bond graph format was developed by Karnopp [22] and is outlined in this section.

It can be seen from equation 5.4 that the feedback loop consists of taking the difference between the outputs of the system and observer and feeding these errors via the feedback gain matrix T into the estimated state derivatives $\hat{\boldsymbol{x}}$. The feedback loop adds n additional inputs to the observer.

To see how these additional inputs may be augmented it must be noted that the states of a system are represented on a bond graph by any Compliance or Inertia elements which have integral causality. These elements may be connected to either a **0** or **1** junction leading to the four different combinations shown in figure 5.1.

The elements (e_1,f_1) in the diagrams represent the connections of the junctions

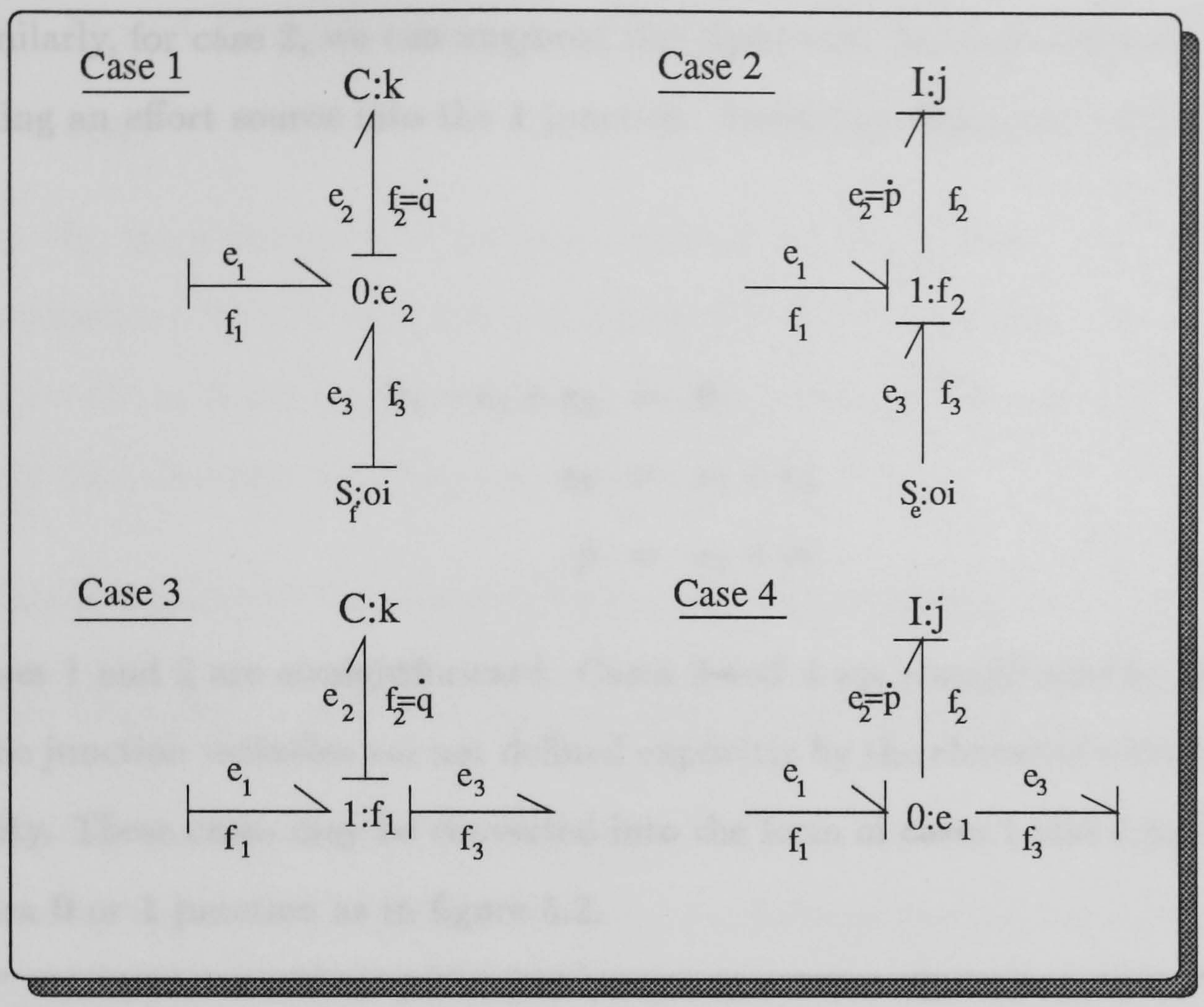


Figure 5.1: Observer Input Combinations.

to the rest of the system bond graph; there may be more than one in reality but one is sufficient to demonstrate how the observer inputs are augmented.

For case 1, the state is represented by q and the state derivative by $\dot{q}$. Summing the flows into the **0** junction gives

$$\begin{aligned} f_1 - f_2 + f_3 &= 0 \\ f_2 &= f_1 + f_3 \end{aligned} \tag{5.7}$$

but $f_2 = \dot{q}$ hence

$$\dot{q} = f_1 + f_3 \tag{5.8}$$

To augment the input into this state derivative we need to impose a *flow* source S_f into the junction. If we label the source input oi (observer input) we get

$$\dot{q} = f_1 + oi \tag{5.9}$$

where f_1 will provide the rest of the dynamics of the system which affects $\dot{q}$.

Similarly, for case 2, we can augment the input into the state derivative $\dot{p}$ by providing an effort source into the **1** junction. Summing efforts into the junction yields

$$\begin{aligned} e_1 - e_2 + e_3 &= 0 \\ e_2 &= e_1 + e_3 \\ \dot{p} &= e_1 + oi \end{aligned} \tag{5.10}$$

Cases 1 and 2 are straightforward. Cases 3 and 4 are complicated by the fact that the junction variables are not defined explicitly by the elements with integral causality. These cases may be converted into the form of cases 1 and 2 by adding an extra **0** or **1** junction as in figure 5.2.

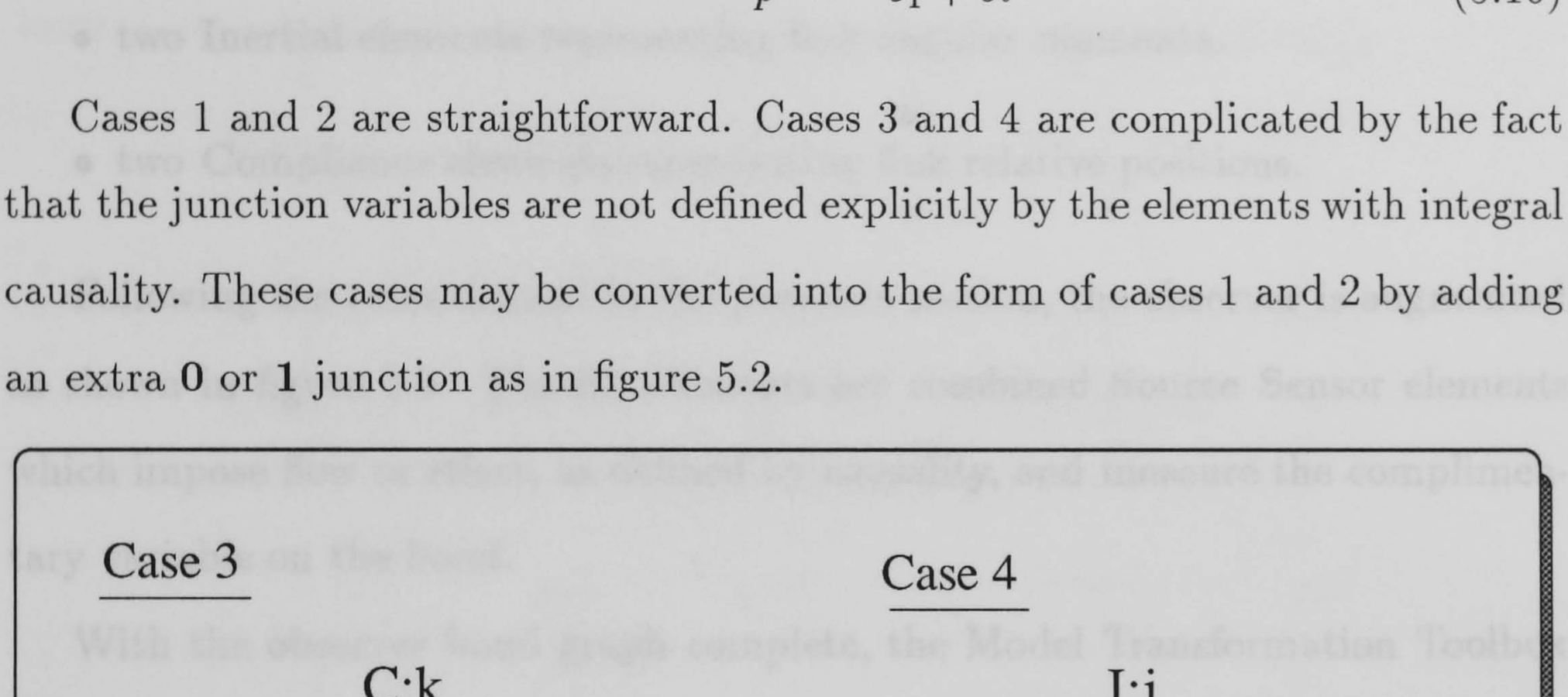


Figure 5.2: Observer Input Combinations.

5.2.2 Bond Graph Observer for the Two-Link Manipulator.

Fortunately, the elements with integral causality in the bond graph for the two link manipulator all fall into cases 1 and 2 of the previous section and hence the conversion of the bond graph into observer form is straightforward. There are in fact only four elements with integral causality:

- two **I**nertial elements representing link angular momenta.
- two **C**ompliance elements representing link relative positions.

Following the rules defined in the previous section, the observer is augmented as shown in figure 5.3. The **SS** elements are combined **S**ource **S**ensor elements which impose flow or effort, as defined by causality, and measure the complimentary variable on the bond.

With the observer bond graph complete, the Model Transformation Toolbox (MTT) is able to derive the observer equations automatically in a number of formats. One format is human readable form and this is shown in appendix B. It is also possible to create the software required to implement the observer in CONIC and the routine which accomplishes this, together with the actual software, is given in appendix D.

5.3 Selecting the Feedback Gain Matrix, T.

5.3.1 Theory.

The selection of the feedback gain matrix T is complicated by the fact that the observer is non-linear as it reflects the highly non-linear nature of the two-link manipulator. A solution is to design T using a linearised model which allows

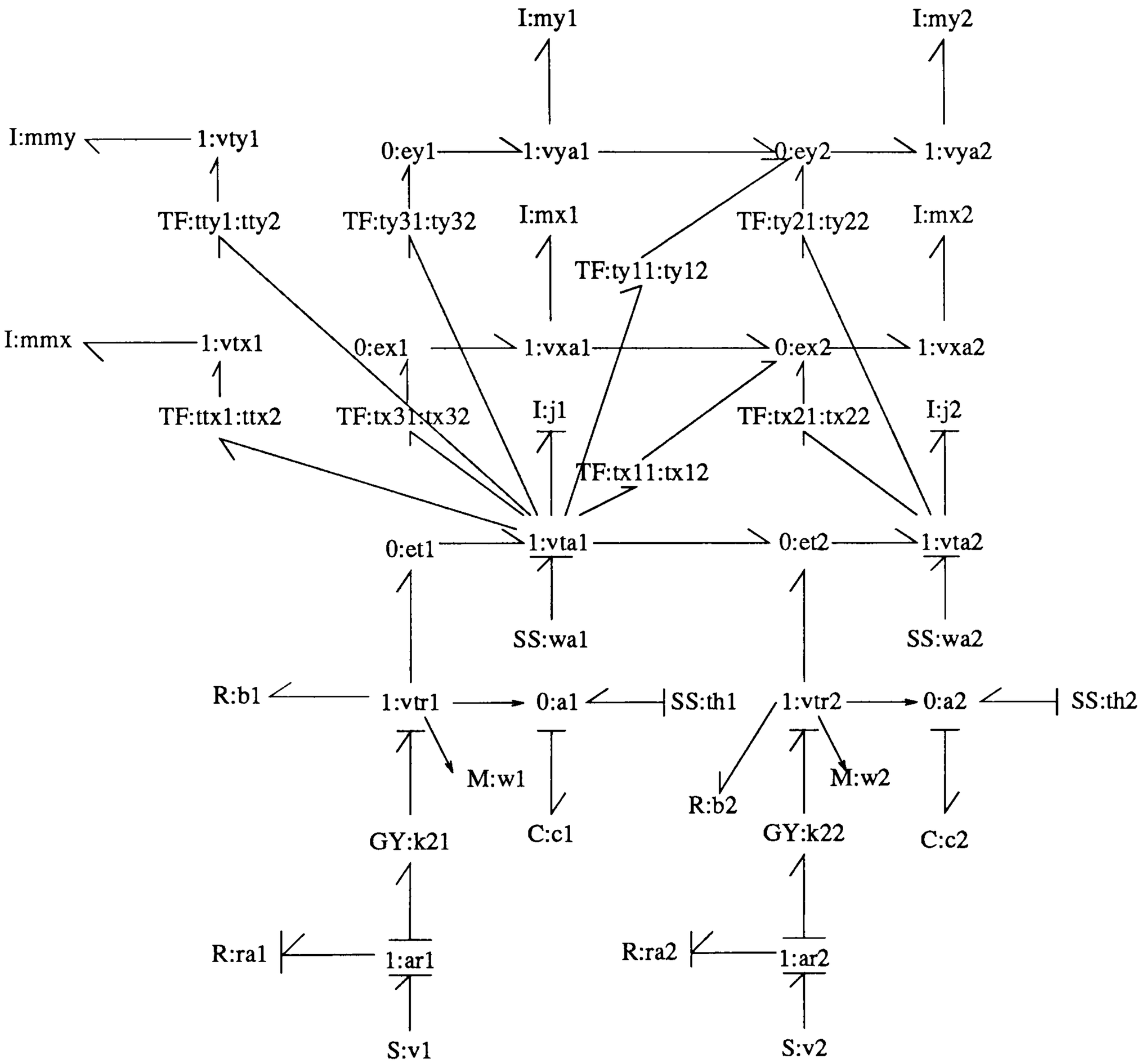


Figure 5.3: Bond Graph Observer for the Two-Link Manpulator.

the use of standard observer feedback gain matrix design techniques. The non-linear observer can therefore be used with a linear feedback loop which forces the observer states to track the system states. Although the poles of the complete observer are assigned for the model linearised around a particular operational point, it is assumed that once the observer states are tracking, the errors between system and observer outputs will be small causing the feedback loop to inject only small corrective inputs to the observer.

To summarise, if the feedback loop is capable of initially forcing observer states to converge to system states, it should be capable of tracking them.

Linearising the Robot Equations.

The equations of motion for robotic manipulators are not of the form of equations 5.1 and 5.2 but have instead the differential algebraic equation (DAE) form [17]

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} + F\dot{\mathbf{z}} \tag{5.11}$$

where $\mathbf{z}$ is a vector of constrained states: states of the system which are determined through algebraic combinations of the true system states $\boldsymbol{x}$. Hence

$$\mathbf{z} = Z(\boldsymbol{x}) \tag{5.12}$$

Differentiating with respect to time

$$\frac{\partial \mathbf{z}}{\partial t} = \frac{\partial \mathbf{z}}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial t} \tag{5.13}$$

$$\dot{\mathbf{z}} = G\dot{\boldsymbol{x}} \tag{5.14}$$

Hence, in 5.11

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} + FG\dot{\boldsymbol{x}} \\ (I - FG)\dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} \\ E\dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} \end{aligned} \tag{5.15}$$

To obtain the linearised state space equations from the set of constrained state equations $\dot{\boldsymbol{\chi}}$ where $\dot{\boldsymbol{\chi}} = E\dot{\boldsymbol{x}}$, MTT performs the following operations:

$$A = \frac{\partial \dot{\boldsymbol{\chi}}}{\partial \boldsymbol{x}} \tag{5.16}$$

$$B = \frac{\partial \dot{\boldsymbol{\chi}}}{\partial \boldsymbol{u}} \tag{5.17}$$

$$C = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \tag{5.18}$$

$$D = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}} \tag{5.19}$$

$$E = (I - FG) \tag{5.20}$$

The matrices are now linearised apart from the E matrix which is a function of the state vector. The system may be linearised around a specified point by supplying the numerical values of the states at this point. In the case of the two-link manipulator we wished to linearise around the state in which the manipulator is stationary with the second link extended. The state vector corresponding to this state is $\boldsymbol{x} = [0, 0, 0, 0]^T$.

We now have the linearised system equations

$$E\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \tag{5.21}$$

$$\boldsymbol{y} = C\boldsymbol{x} + D\boldsymbol{u} \tag{5.22}$$

and we wish to design the feedback gain matrix, T. The observer equations become

$$E\dot{\hat{\boldsymbol{x}}} = A\hat{\boldsymbol{x}} + B\boldsymbol{u} + T(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{5.23}$$

$$\hat{\boldsymbol{y}} = C\hat{\boldsymbol{x}} + D\boldsymbol{u} \tag{5.24}$$

The matrix E is non-singular and hence invertible. Equation 5.23 may therefore be written

$$\dot{\hat{\boldsymbol{x}}} = E^{-1}A\hat{\boldsymbol{x}} + E^{-1}B\boldsymbol{u} + E^{-1}T(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{5.25}$$

Defining an error vector $\mathbf{e} = \boldsymbol{x} - \hat{\boldsymbol{x}}$, we get

$$\begin{aligned}
\dot{\mathbf{e}} &= \dot{\boldsymbol{x}} - \dot{\hat{\boldsymbol{x}}} \\
&= E^{-1}A\boldsymbol{x} + E^{-1}B\boldsymbol{u} - E^{-1}A\hat{\boldsymbol{x}} - E^{-1}B\boldsymbol{u} - E^{-1}T(\boldsymbol{y} - \hat{\boldsymbol{y}}) \\
&= E^{-1}A(\boldsymbol{x} - \hat{\boldsymbol{x}}) - E^{-1}TC(\boldsymbol{x} - \hat{\boldsymbol{x}}) \\
&= (E^{-1}A - E^{-1}TC)\mathbf{e}
\end{aligned} \tag{5.26}$$

Through selection of the matrix $E^{-1}T$ the eigenvalues of the system $\dot{\mathbf{e}} = (E^{-1}A - E^{-1}TC)\mathbf{e}$ may be assigned to any desired positions on the complex plane as long as the $(E^{-1}A, C)$ pair is observable. To obtain the feedback gain matrix T however, the matrix $E^{-1}T$ must be pre-multiplied by the matrix E.

5.3.2 Practice.

To aid in the design of the feedback gain matrix T, the observer is simulated using the package SIMULAB [21] to observe the states of the system model as shown in figure 5.4. This simulation configuration allows us to gauge the effectiveness of the observer with different feedback gain matrices and different starting conditions easily and quickly. It is, of course, an idealised configuration as the observer and system models are almost identical and noise free although white noise may be injected into the system measurements.

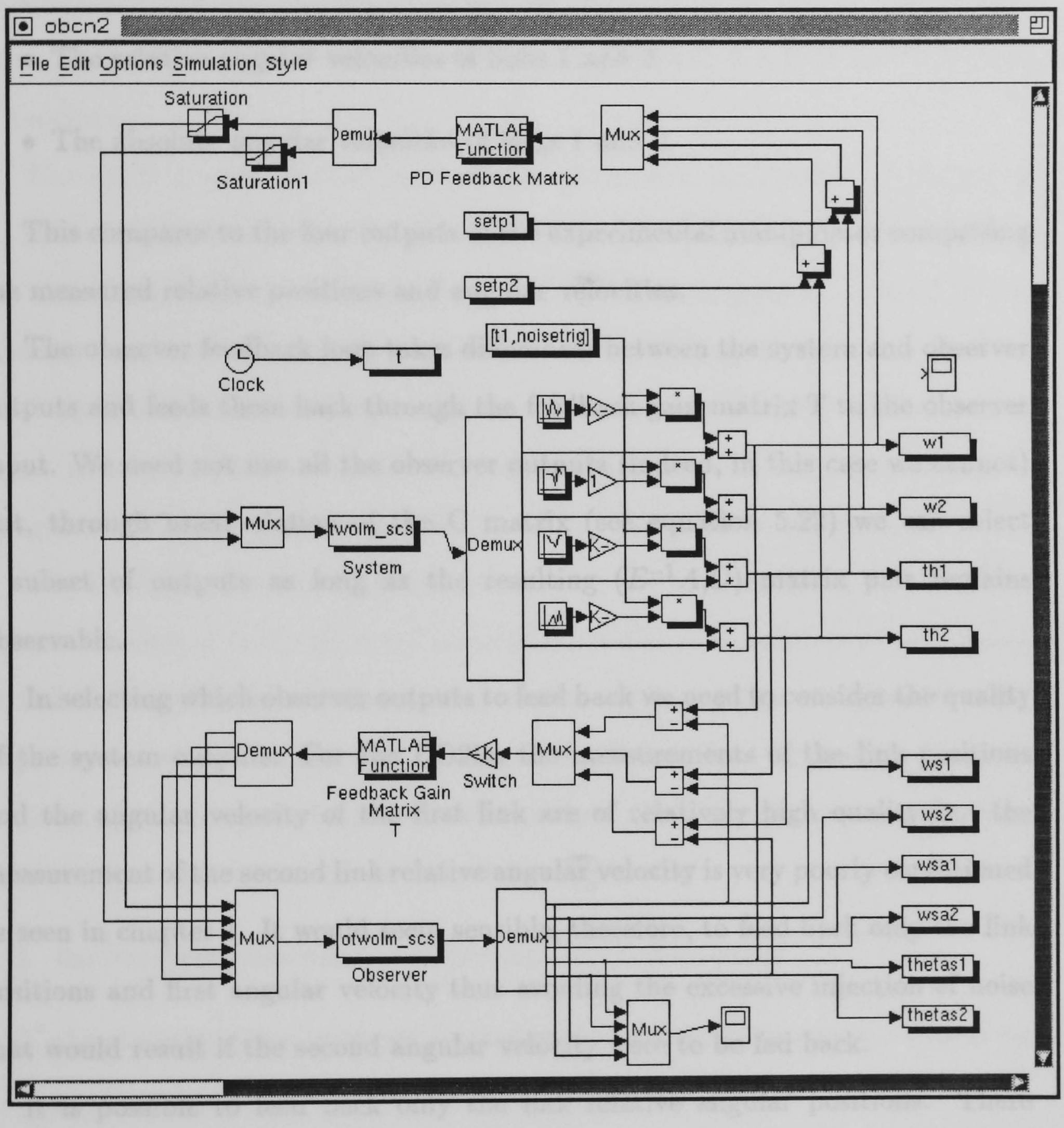


Figure 5.4: SIMULAB Observer Simulation Configuration.

Selection of the Output Space.

The observer has six outputs:

- The relative angular positions of links 1 and 2.
- The relative angular velocities of links 1 and 2.
- The absolute angular velocities of links 1 and 2.

This compares to the four outputs of the experimental manipulator comprising the measured relative positions and angular velocities.

The observer feedback loop takes differences between the system and observer outputs and feeds these back through the feedback gain matrix T to the observer input. We need not use all the observer outputs (indeed, in this case we cannot) but, through manipulation of the C matrix (see equation 5.22) we can select a subset of outputs as long as the resulting $(E^{-1}A,C)$ matrix pair remains observable.

In selecting which observer outputs to feed back we need to consider the quality of the system outputs. For the DD2lm the measurements of the link positions and the angular velocity of the first link are of relatively high quality but the measurement of the second link relative angular velocity is very poorly conditioned as seen in chapter 4. It would seem sensible, therefore, to feed back only the link positions and first angular velocity thus avoiding the excessive injection of noise that would result if the second angular velocity were to be fed back.

It is possible to feed back only the link relative angular positions. There is, however, a slight benefit in using the measured first link angular velocity as it allows tighter tracking of the state involving first link angular momentum as there is a direct relationship between the two. The measurement is of sufficiently high quality not to inject excessive noise into the observer so there is little or no penalty in using it.

Pole Placement.

The most straightforward method of designing the feedback gain matrix T is by pole placement [59] [60] executed using MATLABs' `place` routine [44]. This routine allows the four poles of the system to be placed anywhere on the complex plane by specifying their desired positions in a vector together with the $E^{-1}A$ and C matrices.

The small matlab routine `getgains.m` automates the process of designing T. It first defines the numerical values of the model parameters before creating the numerical linearised matrices E, A, B, C, D. The outputs to be used in the observer feedback loop are then selected by manipulating the C matrix and the feedback gain matrix is designed using the `place` routine. Finally, the T matrix is augmented by premultiplying the designed feedback gain matrix by E. The routine `getgains.m` together with its called subroutines are given in appendix E.

The selection of where to place the poles on the complex plane is, to a large extent, a matter of trial and error. To achieve a stable, non-oscillatory convergence of the estimated states to the system states, all four poles are placed on the negative real axis. The speed of convergence is determined by how far from the origin the poles are placed; the further to the left, the faster the convergence. The limits on the allowable range of the poles are such that

1. the observer poles must be faster than the system poles.

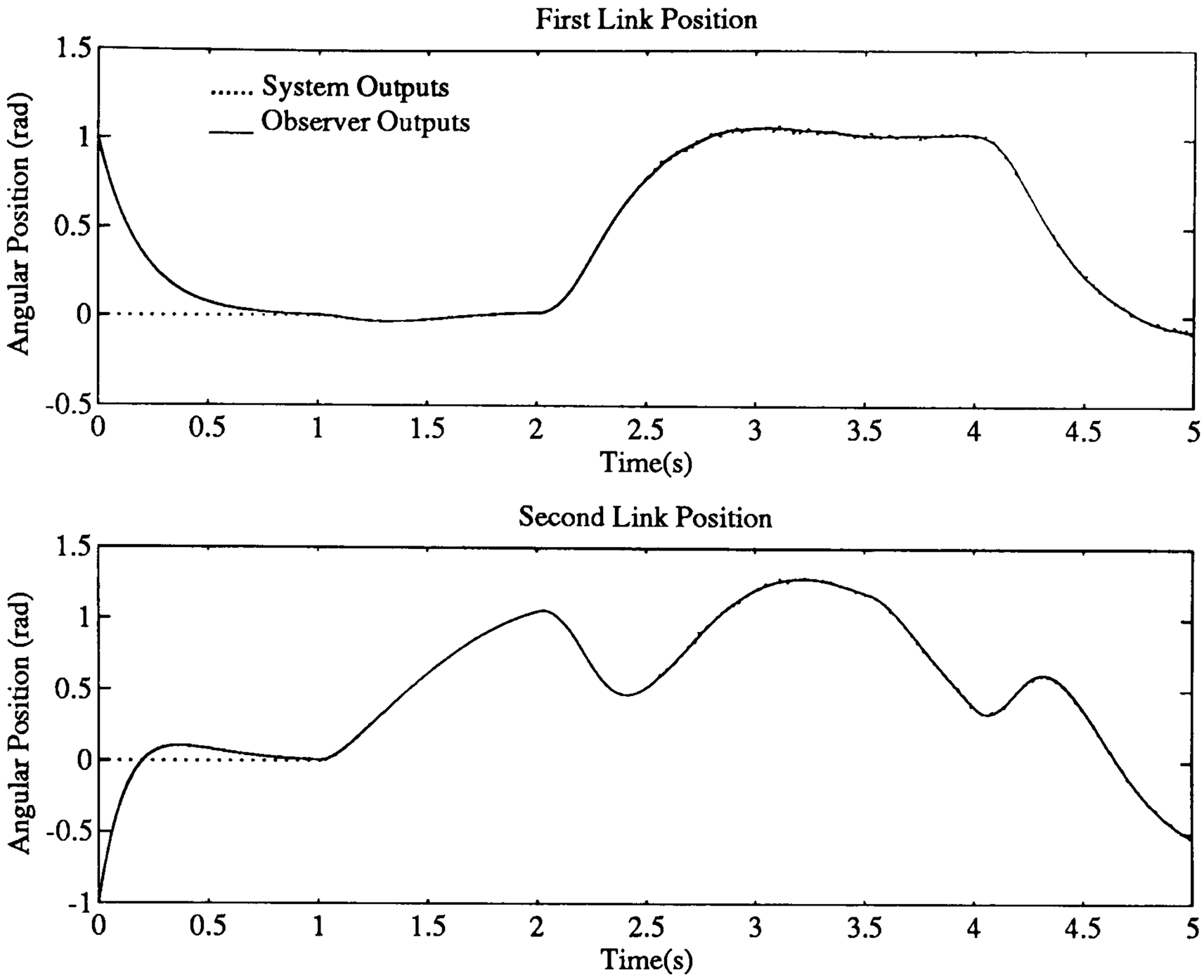
2. the observer poles must not be so fast that numerical errors are caused on implementation of the observer at a finite sample rate.

Condition 1 is easily met as the eigenvalues of the $E^{-1}A$ matrix at the point of linearisation are [0,0,-0.2067,-0.0239]. Condition 2 limits the practicable speed of the poles but there is a third limitation. The feedback loop tries to force the observer to track the system outputs but if the speed of the observer poles is too high, the observer will track the high frequency measurement noise. The choice

of poles is therefore a compromise between fast convergence of estimated states to system states and rejection of measurement noise.

Figures 5.5 to 5.7 demonstrate the effect of pole placement using the simulation configuration shown in figure 5.4. The states of the observer are initialised at $[0, 0, 1, -1]^T$ which corresponds to a stationary manipulator with link 1 at 1 radian and link 2 at a realtive angle of -1 radian. The system is initialised with state vector $[0, 0, 0, 0]^T$.

As the poles get faster, the convergence of the observer outputs to system outputs also gets faster implying faster convergence of estimated states to system states. White noise is injected into the system outputs at t=2.5s in similar amounts as seen on the experimental measurements. It can be seen that the observer is able to filter out the noise effectively but, when the observer poles are placed at [-40,-40,-40,-40] the observer begins to track the noise causing the observer outputs to become noisy aswell. Placing the observer poles at [-10,-10,-10,-10] seems to be a reasonable compromise between speed of convergence and the quality of observer outputs.



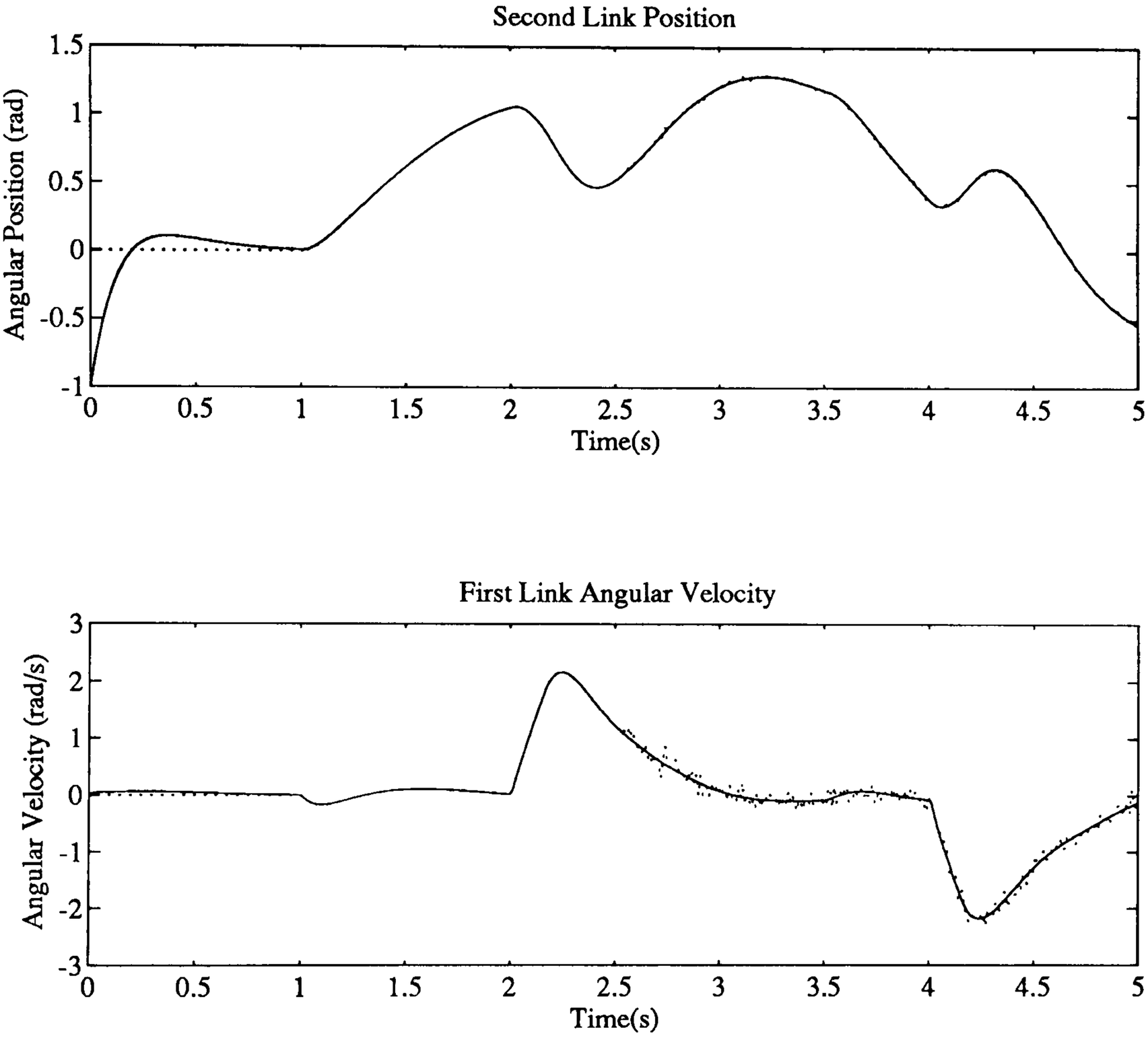


Figure 5.5: Simulated Observer with poles at [-5.-5.-5.-5]

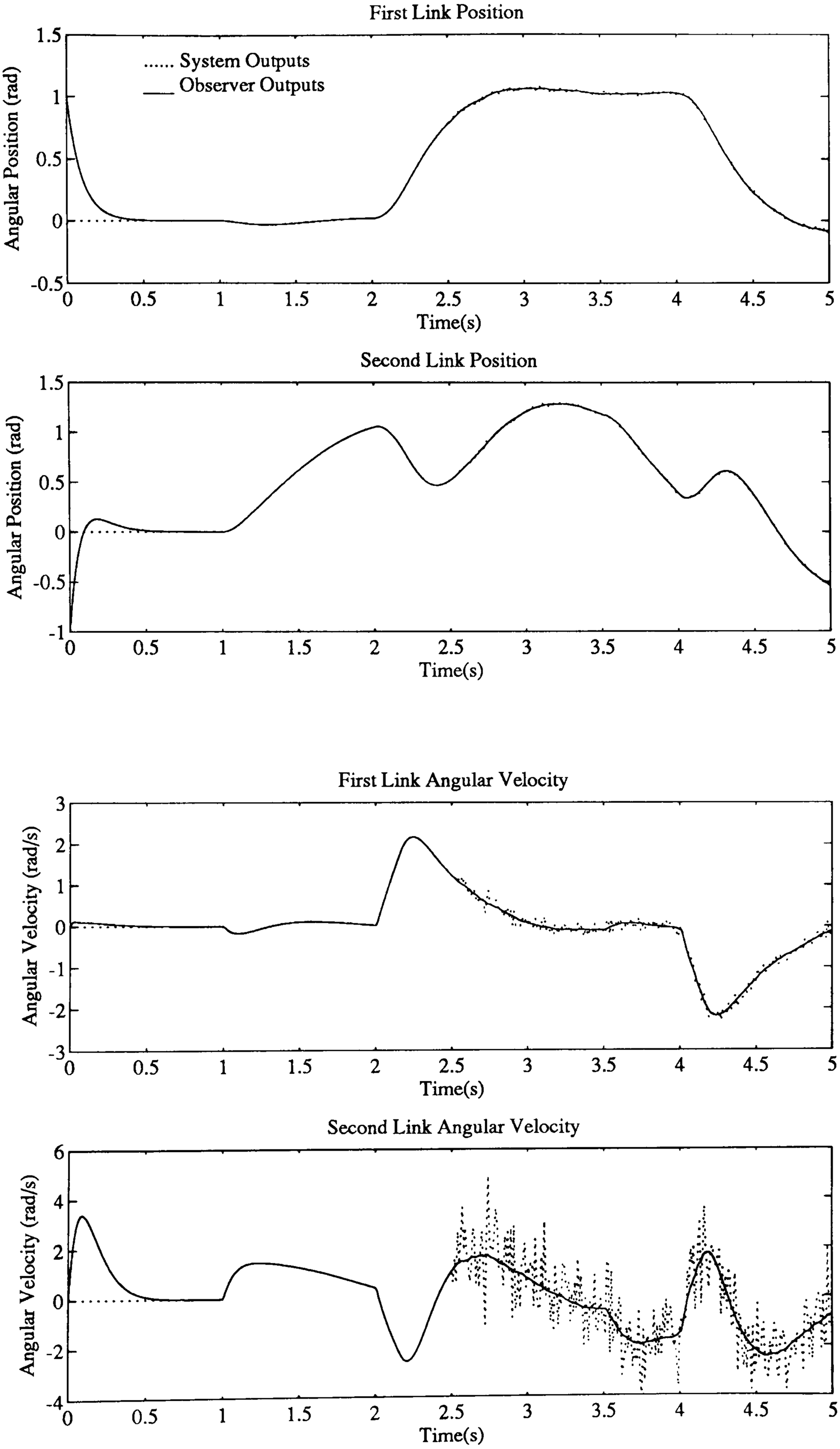
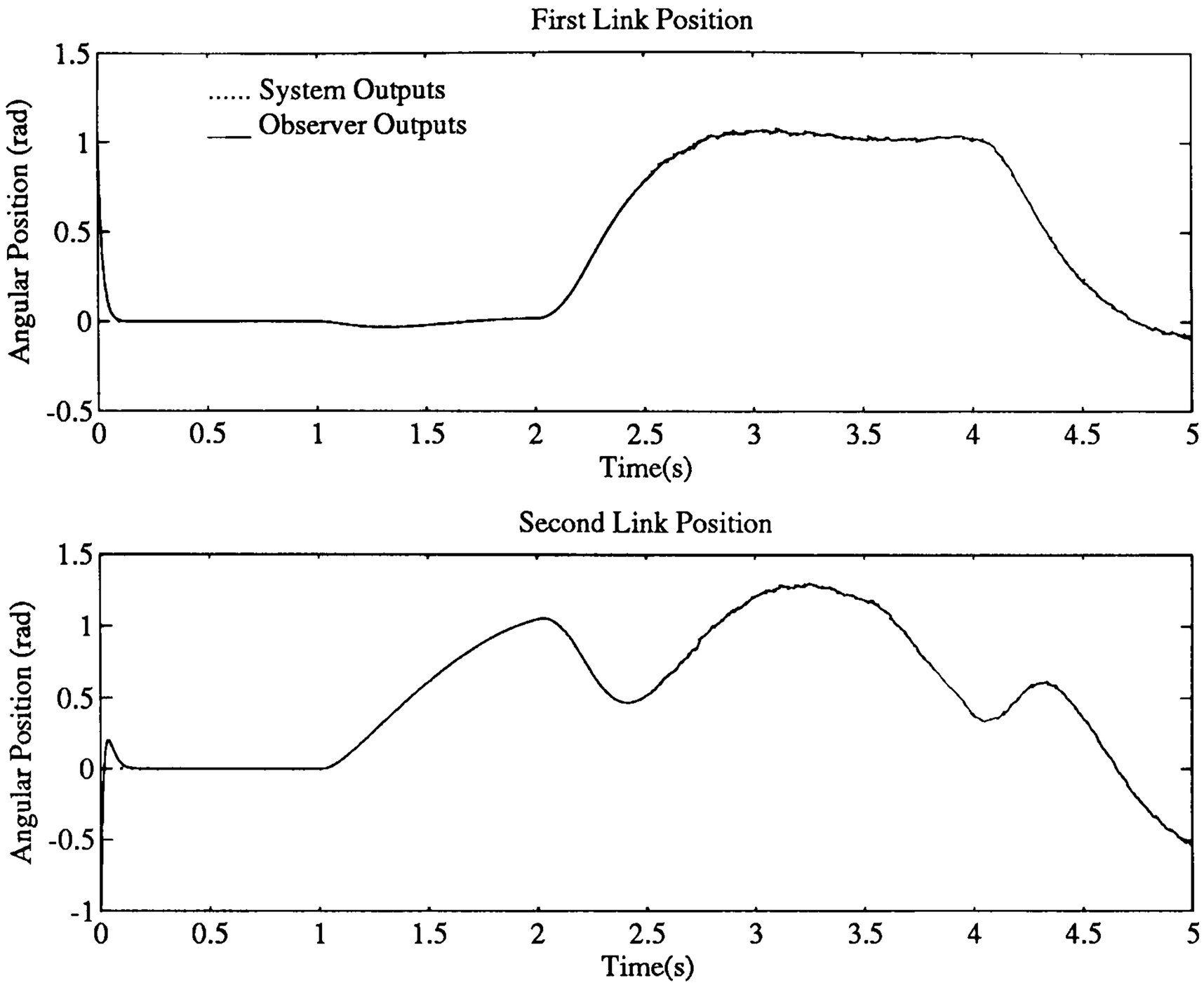


Figure 5.6: Simulated Observer with poles at [-10.-10,-10,-10]



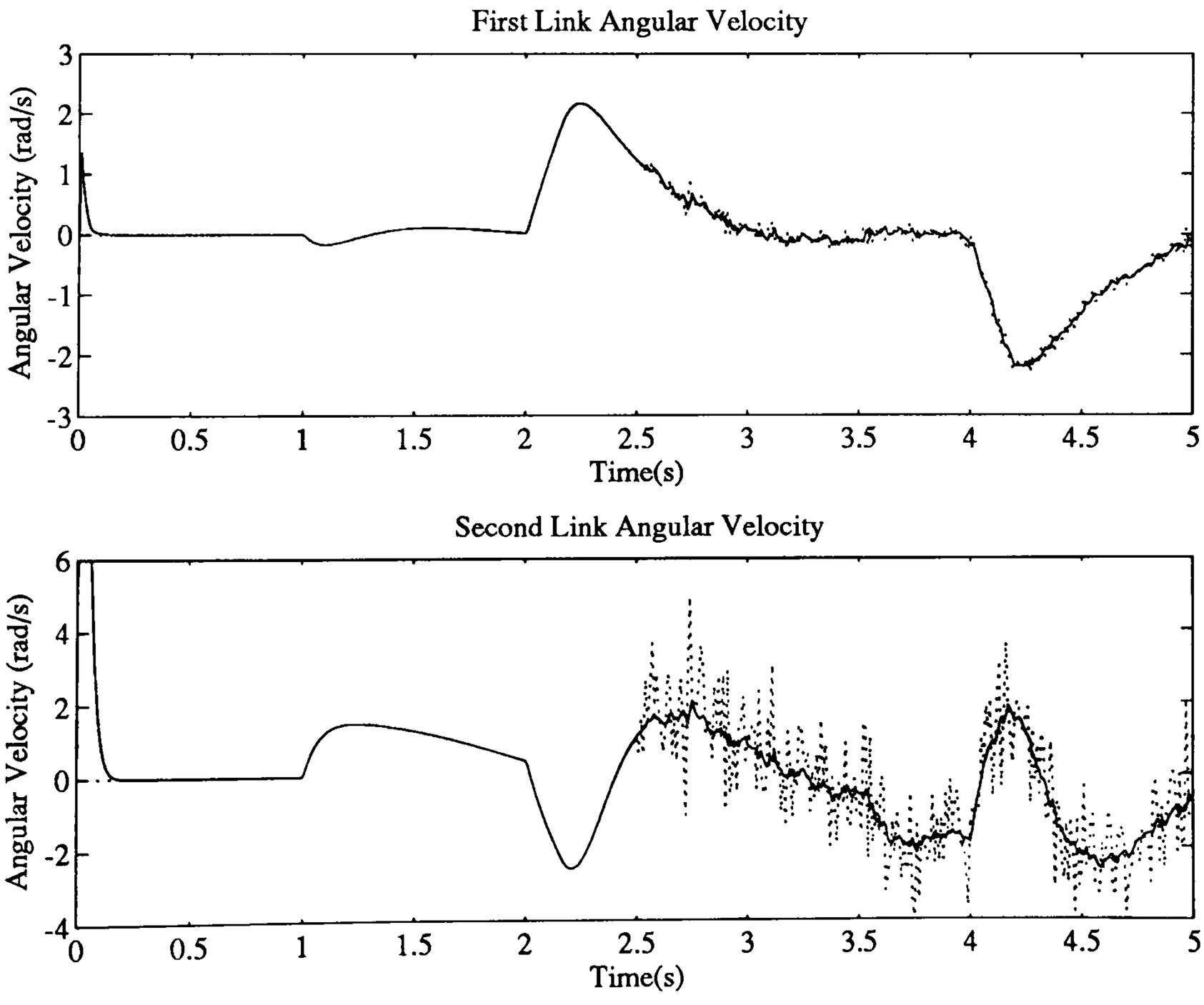


Figure 5.7: Simulated Observer with poles at [-40,-40,-40,-40]

Observer with Real Data.

To see how the observer copes with real data from the two-link manipulator, the SIMULAB configuration shown in figure 5.8 can be used. The data collected for test 7 of the previous chapter was fed through the observer from data files under a range of conditions including

- different sample rates.
- different integration algorithms.
- different pole placements.
- different friction models.

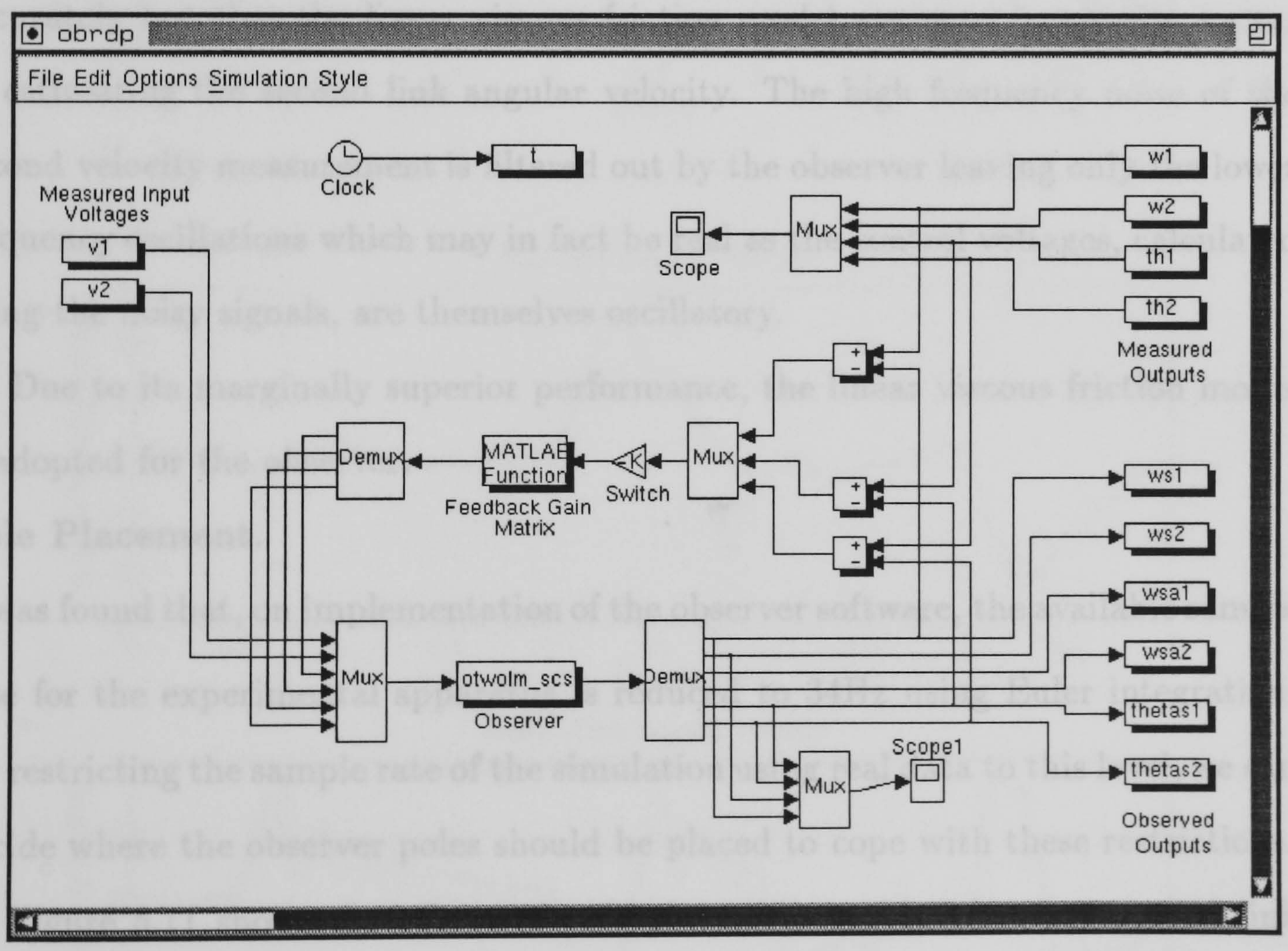


Figure 5.8: Simulated Observer but using Real Data.

For brevity, only the results for trying different friction models and pole placements are given here.

Friction Models.

Two types of model are tested:

- No modelled joint friction.
- Modelled joint linear viscous friction.

The resulting graphs of observed outputs compared with actual outputs are shown in figures 5.9 and 5.10. For both friction models the poles are placed at [-10,-10,-10,-10] and the simulation sample rate is 100Hz using Euler integration. The actual angular velocity of the second link shown on the graphs is the differentiated position signal as it is better conditioned than the filtered tachometer signal.

It can be seen that both models are capable of observing the system outputs accurately but that the linear viscous friction model seems to be slightly better at estimating the second link angular velocity. The high frequency noise of the second velocity measurement is filtered out by the observer leaving only the lower frequency oscillations which may in fact be real as the control voltages, calculated using the noisy signals, are themselves oscillatory.

Due to its marginally superior performance, the linear viscous friction model is adopted for the observer.

Pole Placement.

It was found that, on implementation of the observer software, the available sample rate for the experimental apparatus is reduced to 34Hz using Euler integration. By restricting the sample rate of the simulation using real data to this level, we can decide where the observer poles should be placed to cope with these restrictions.

Figure 5.11 shows the effects of pole placement on the observed second link angular velocity. It is apparent that the best results are obtained when the poles are placed at [-20,-20,-20,-20]. At [-40,-40,-40,-40] the numerical integrations starts to disintegrate leading to poorly conditioned observations.

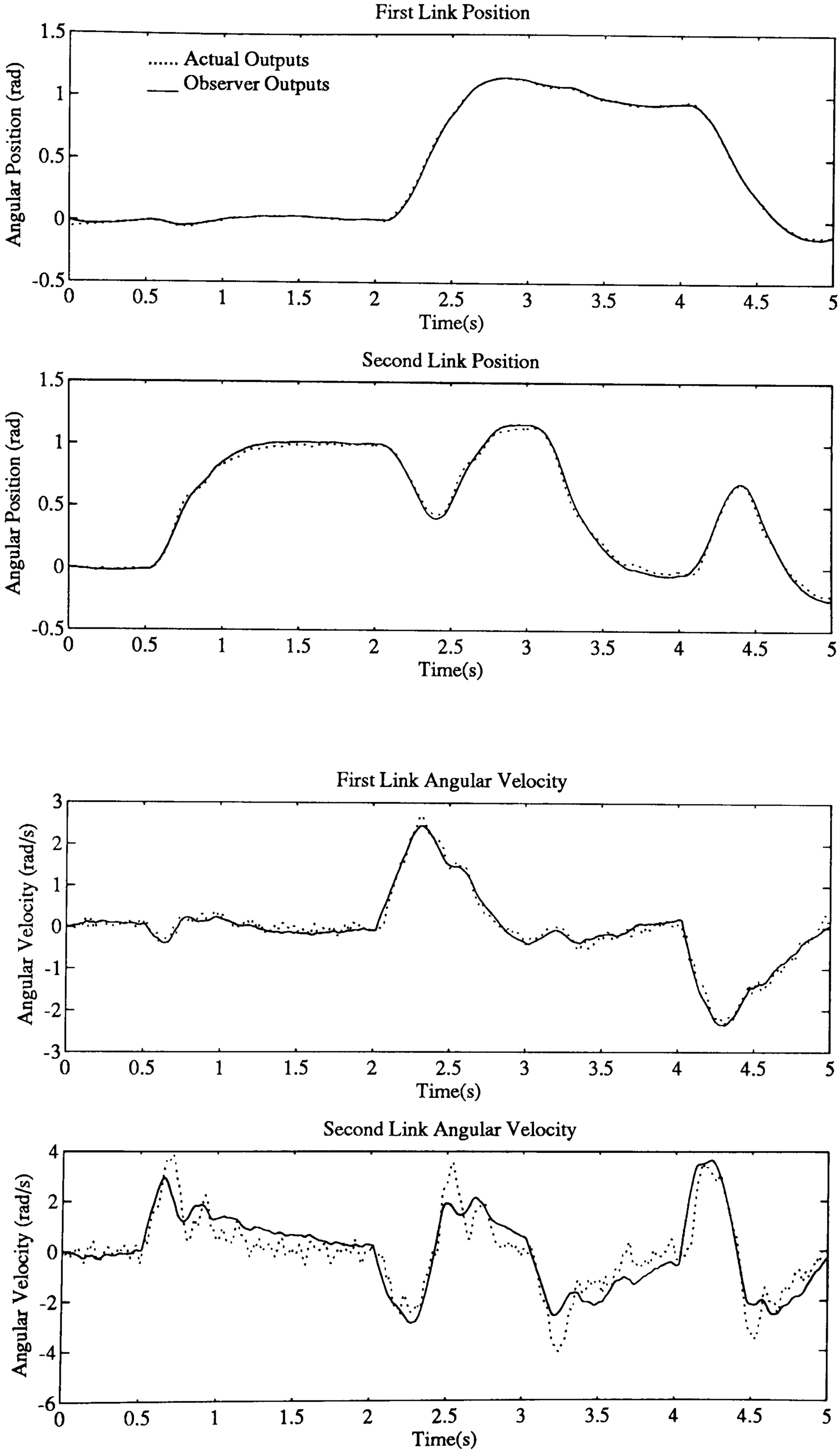


Figure 5.9: Observer with no friction.

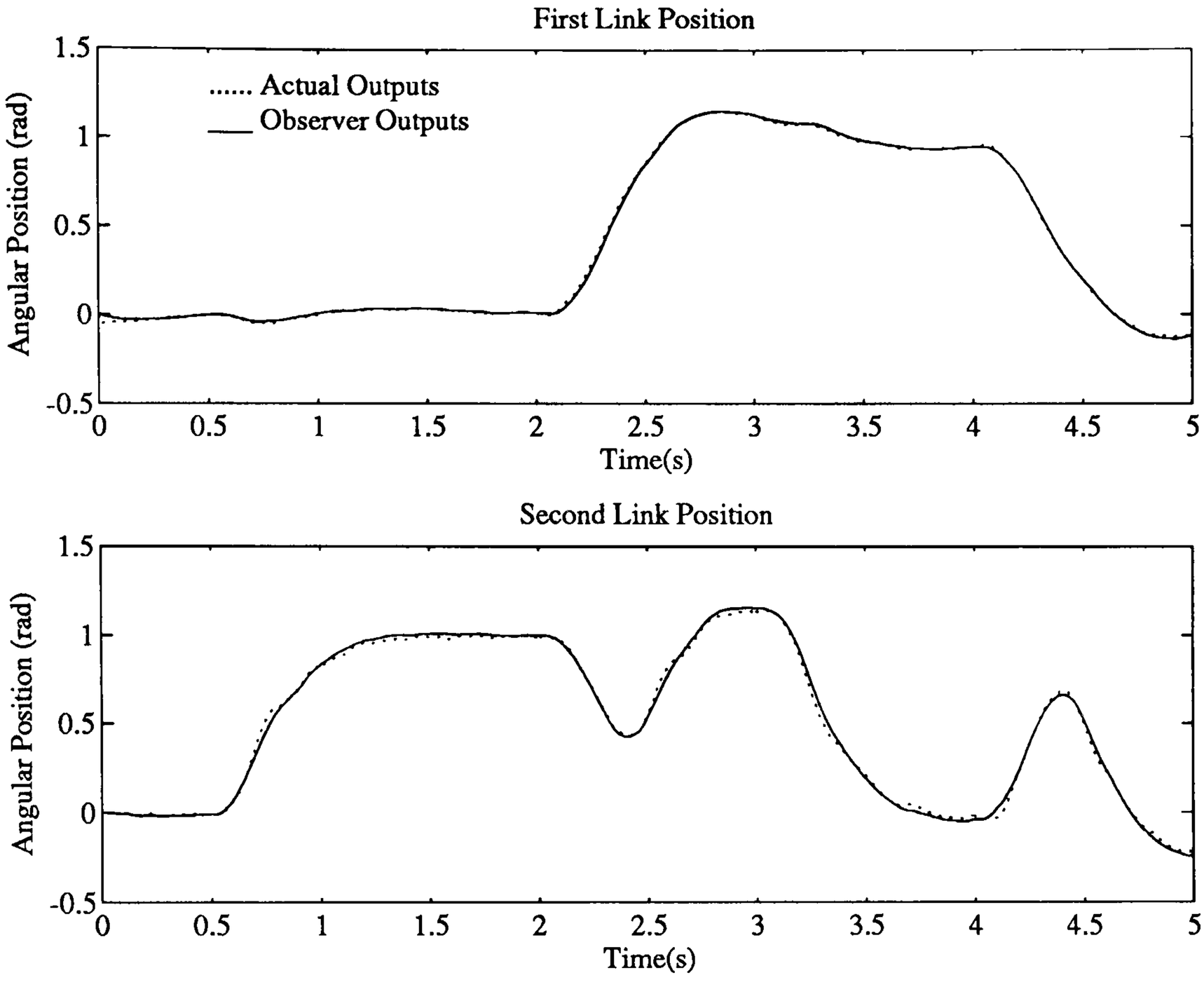


Figure 5.10: Observer with linear viscous friction

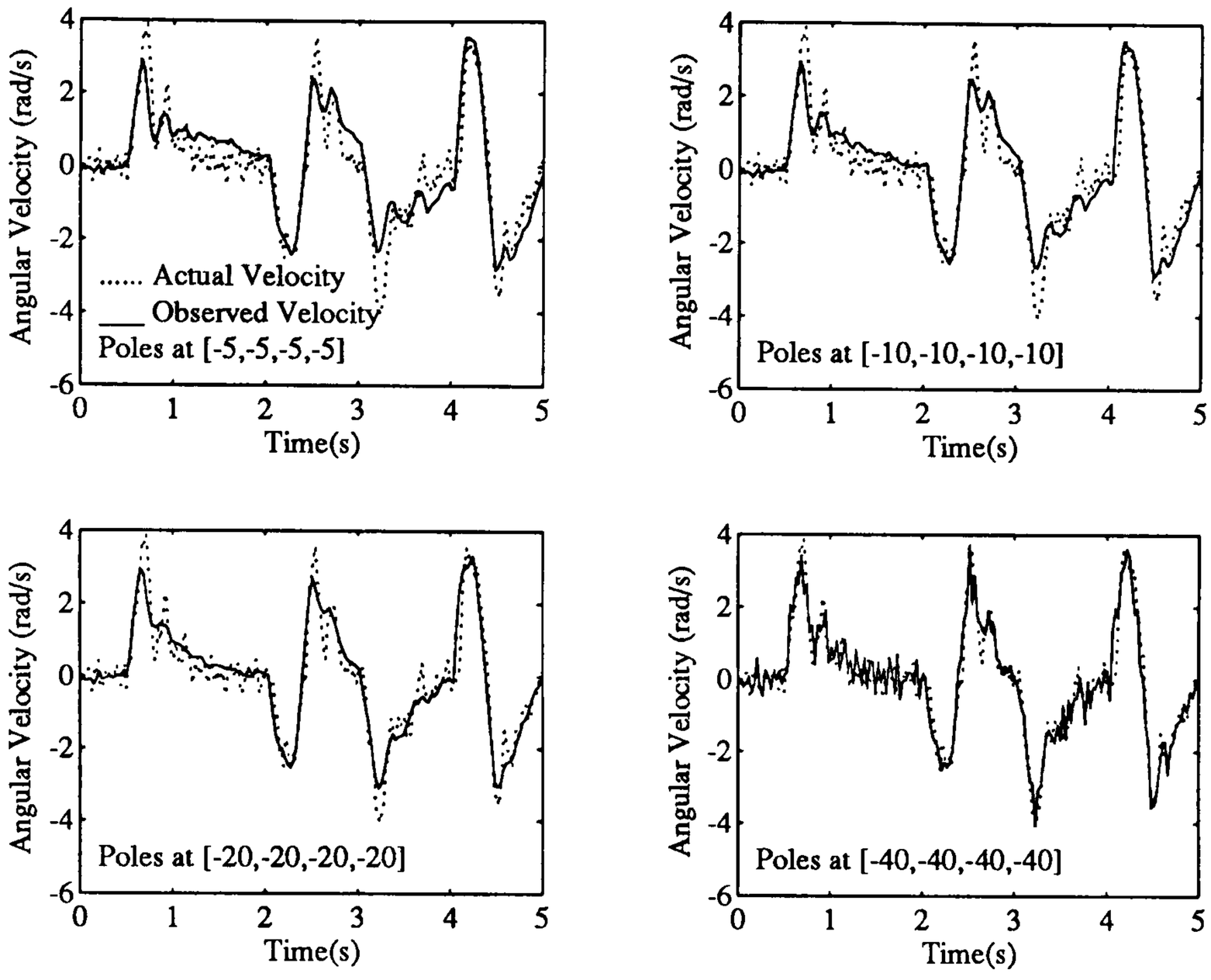


Figure 5.11: Effect of Pole Placement on Observed Second Link Angular Velocity.

5.4 Implementation of the Observer.

5.4.1 CONIC Configuration.

Implementation of the observer for use in the practical control of the experimental two-link manipulator essentially involves implementing the simulation configuration of figure 5.8 in CONIC. The practical configuration is shown in figure 5.12

The observer is run in real time on a Sun3 workstation which communicates with the target motorola MC68020 computers via ethernet. Ethernet is, however, a general communications network without guaranteed access to a particular computer at a given time. The average time for the passing of a piece of data from targets to the Sun, and vice versa, is approximately $\frac{1}{400}$second. As 8 parameters are passed in each time step, this limits the maximum sample rate to approximately 50Hz. Computation time further reduces this to 34Hz. This is acceptable for the control of the DD2lm, as the dynamics of the manipulator are slow due to its

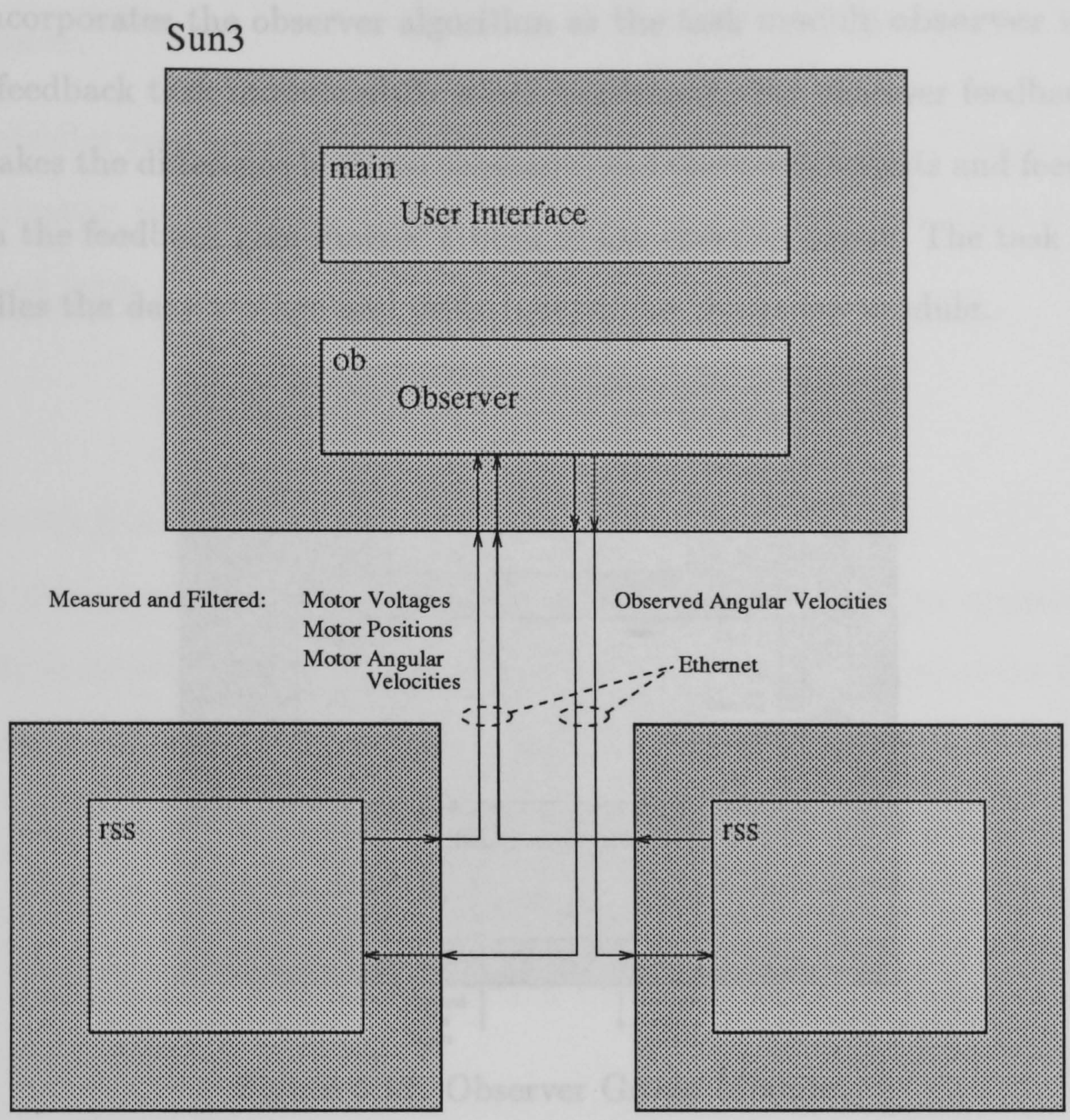


Figure 5.12: CONIC Configuration for the Observer.

rigid construction, but it is far from ideal to use a general access communications network for implementing real time control.

In SIMULAB a range of sophisticated, variable step size integration algorithms are available which require the model to be run several times in each time step. For the real time application this is too costly in terms of reduction in the available sample rate; there is a trade-off between accuracy, speed and stability. Consequently, simple Euler integration is used to integrate the state derivatives to produce the estimated states in the observer.

The inputs to the observer are the measured angular positions and velocities of the links filtered using second order low pass Butterworth filters with a cut-off frequency of 15Hz to prevent aliasing. The observer group module **ob** (see figure

5.13) incorporates the observer algorithm as the task module **observer** together with a feedback task module **obfb** which implements the observer feedback loop. **Obfb** takes the difference between measured and observed outputs and feeds these through the feedback gain matrix T back to the observer input. The task module **io** handles the data storage and input/output for the group module.

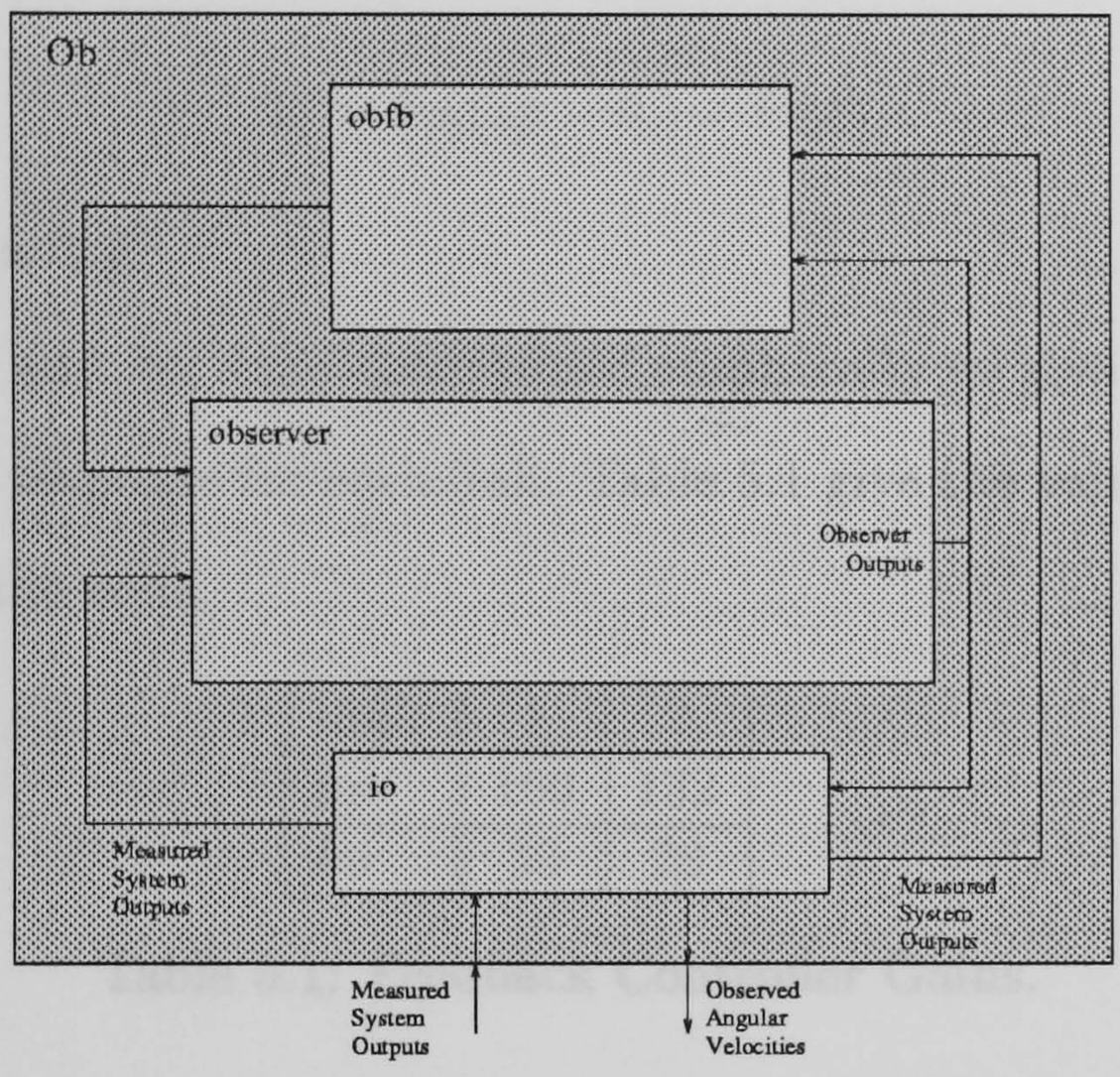


Figure 5.13: Observer Group Module.

The outputs of the observer are the observed angular positions and velocities of the links. To reduce the number of parameters passed over ethernet, only the observed angular velocities are sent back to the independent joint pd controllers running on the targets. This is acceptable as the position measurements are of relatively high quality when compared with the angular velocity measurements.

The full set of software required to implement the observer is given in appendix D.

5.4.2 Results.

On implementation of the observer it was found that the best results were obtained with the observer poles placed at [-10,-10,-10,-10] giving an acceptable compromise between state tracking capability and numerical integration errors.

Controller.

The controller used in the following tests is a simple independent joint proportional and derivative feedback controller in which the control voltage for each link is calculated from

$$V_i = kp_i(sp - \theta_i) - kv_i\dot{\theta}_i \tag{5.27}$$

where i=ith link = 1,2.

The gains kp_i and kv_i were determined empirically to give an approximately critically damped response for each link. Table 5.1 give the values of the feedback gains used in all the tests.

Link	1	2
kp	100	132
kv	30	28

Table 5.1: Feedback Controller Gains.

Smooth Set-Point Trajectory.

The first set of results, shown in figures 5.14 to 5.19 uses a smooth fourth - order set-point position trajectory for links 1 and 2. Resultant positions, angular velocities and motor actuation voltages are given for two separate tests:

- Test 1. Observed angular velocities used in the control algorithm.
- Test 2. Tachometer measured angular velocity used for the first link. Differentiated position measurement used for the second link.

It can be seen from the graphs that when the observer is used in test 1 the response of the manipulator is smooth and stable compared with the highly oscillatory response of the manipulator when the measured velocities are used in the feedback controller. To achieve a comparably smooth trajectory from

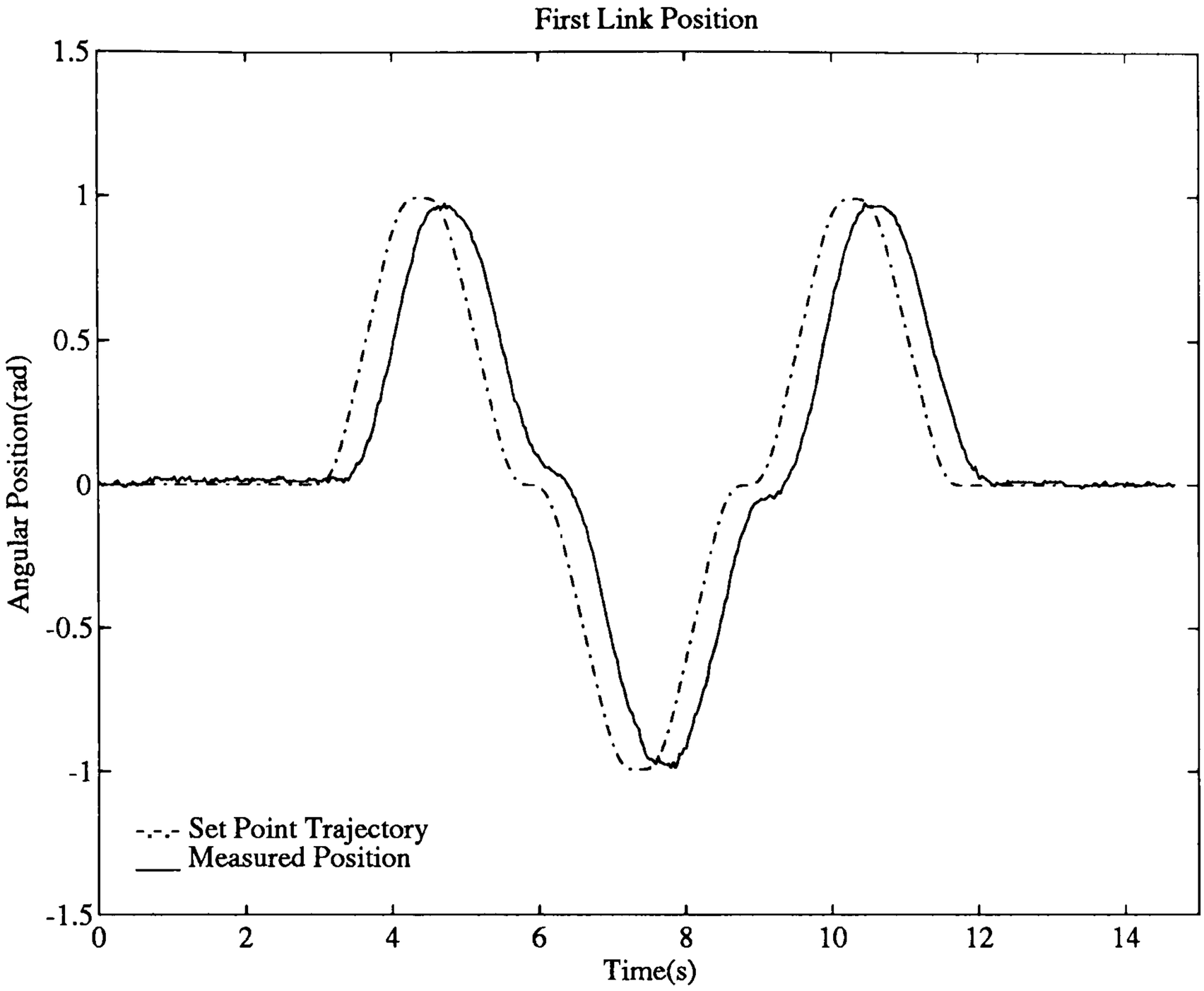


Figure 5.14: Test 1. Link Positions and Set Points using Observed Angular Velocities in the Controller.

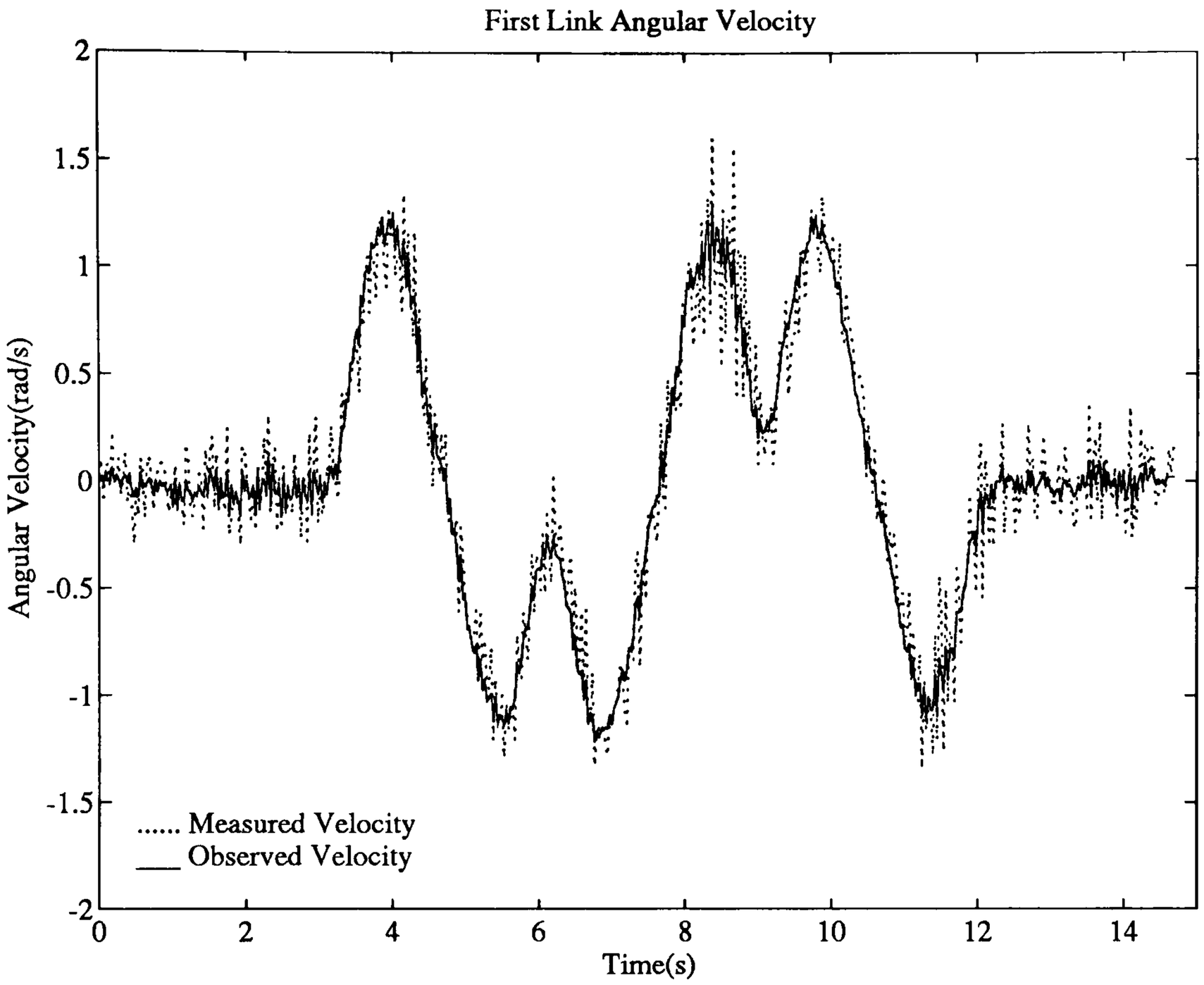
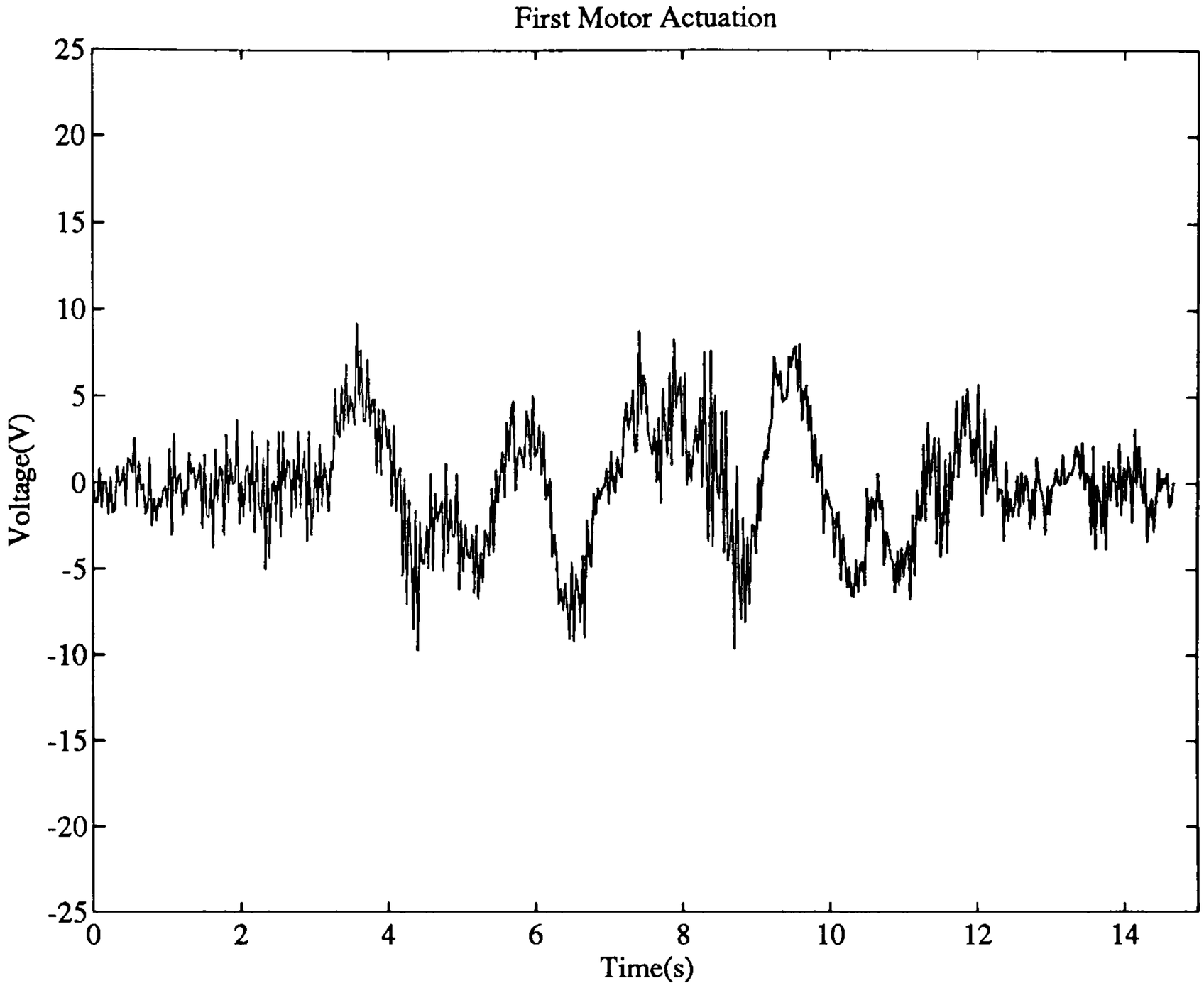


Figure 5.15: Test 1. Link Angular Velocities using Observed Angular Velocities in the Controller.



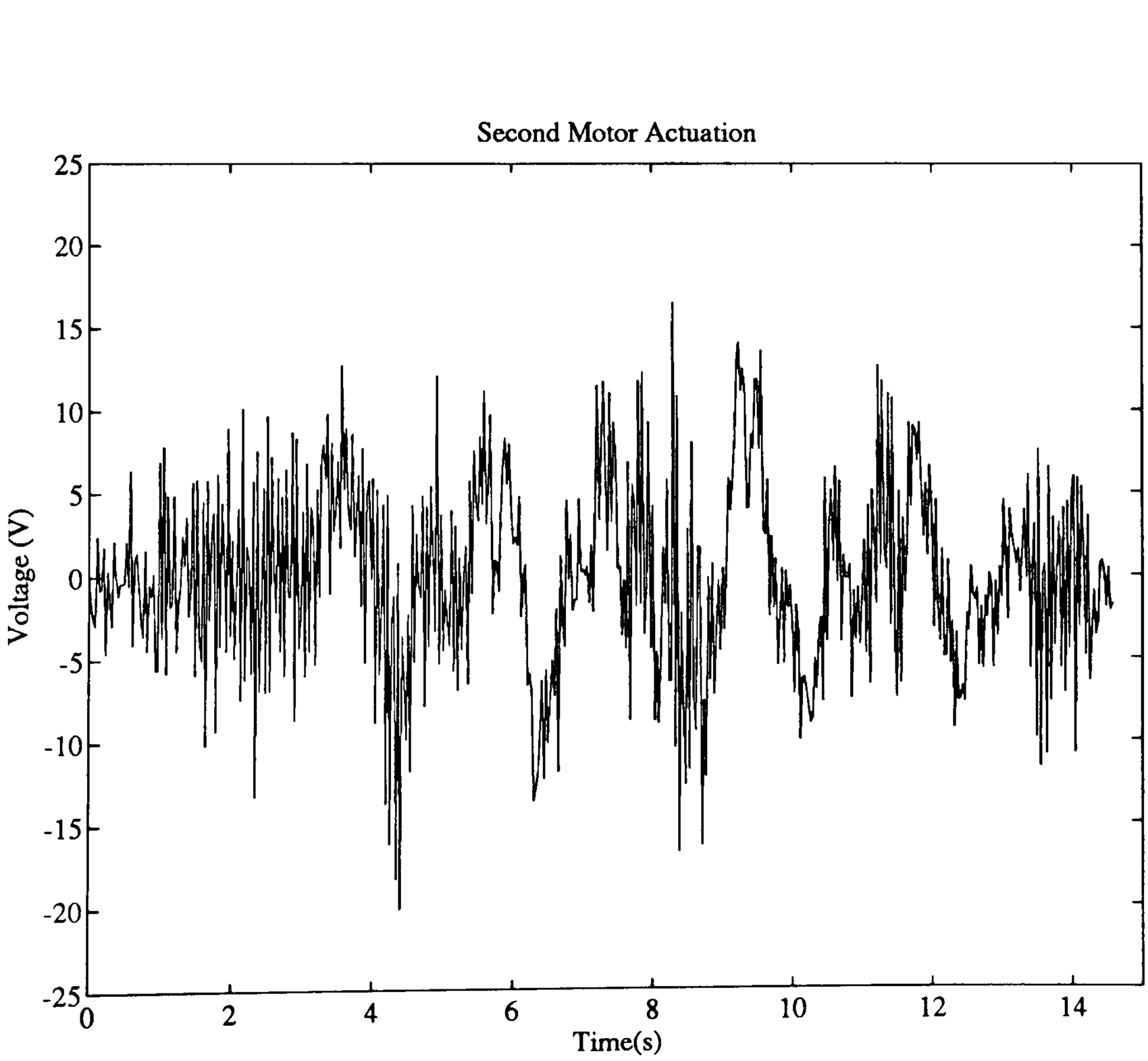


Figure 5.16: Test 1. Motor Actuation Voltages using Observed Angular Velocities in the Controller.

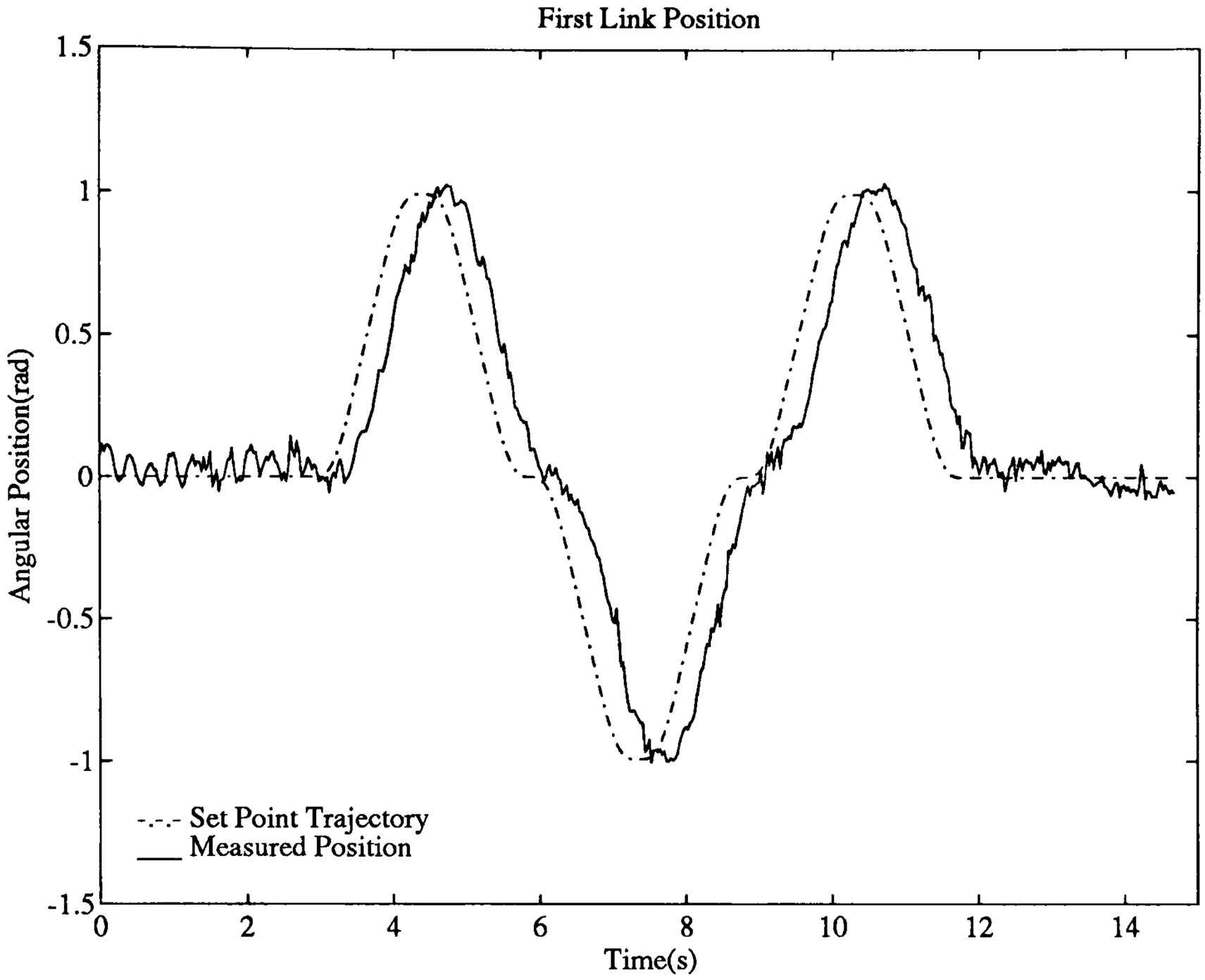


Figure 5.17: Test 2. Link Positions and Set Points using Measured Angular Velocities in the Controller.

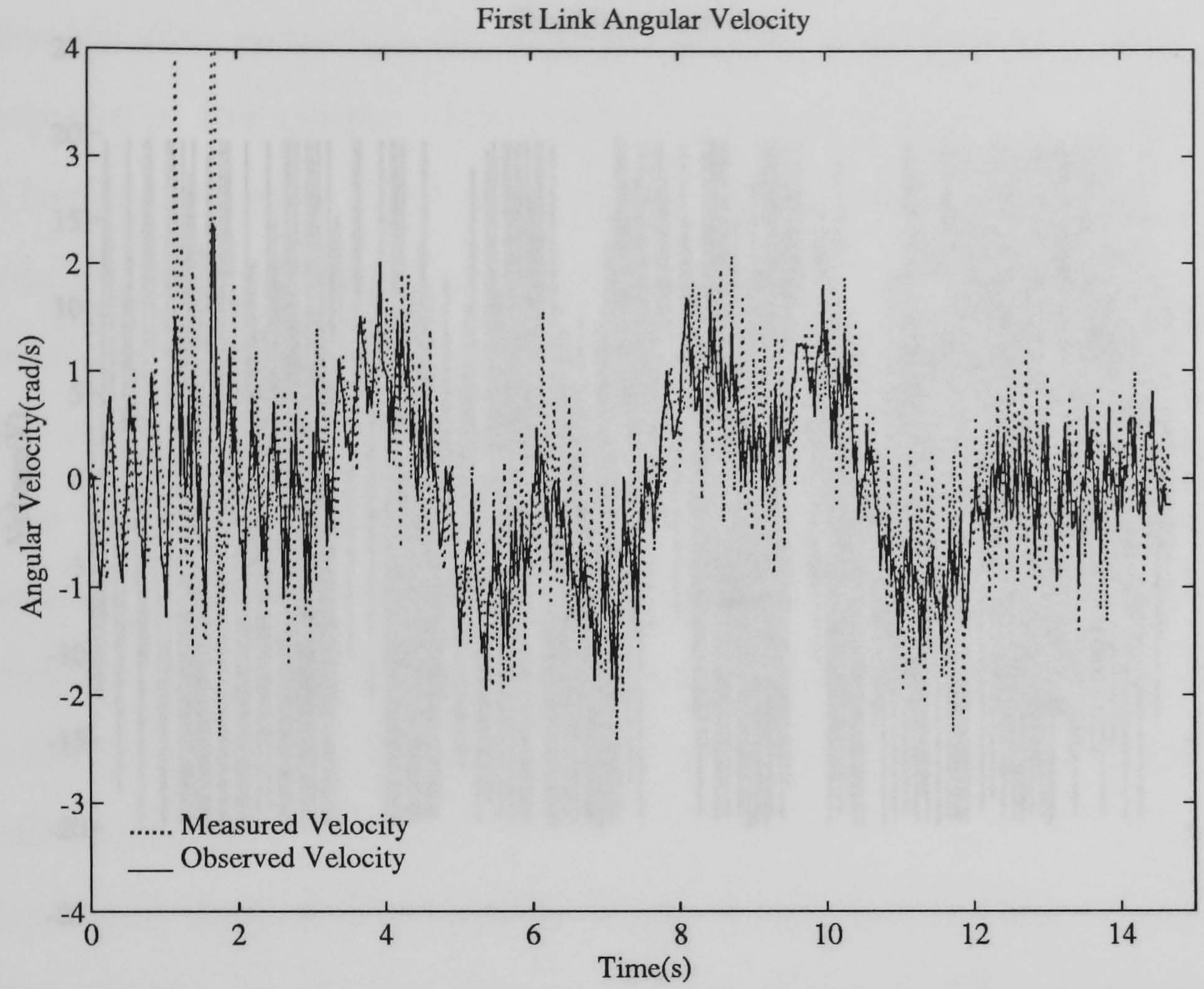


Figure 5.18: Test 2. Link Angular Velocities using Measured Angular Velocities in the Controller.

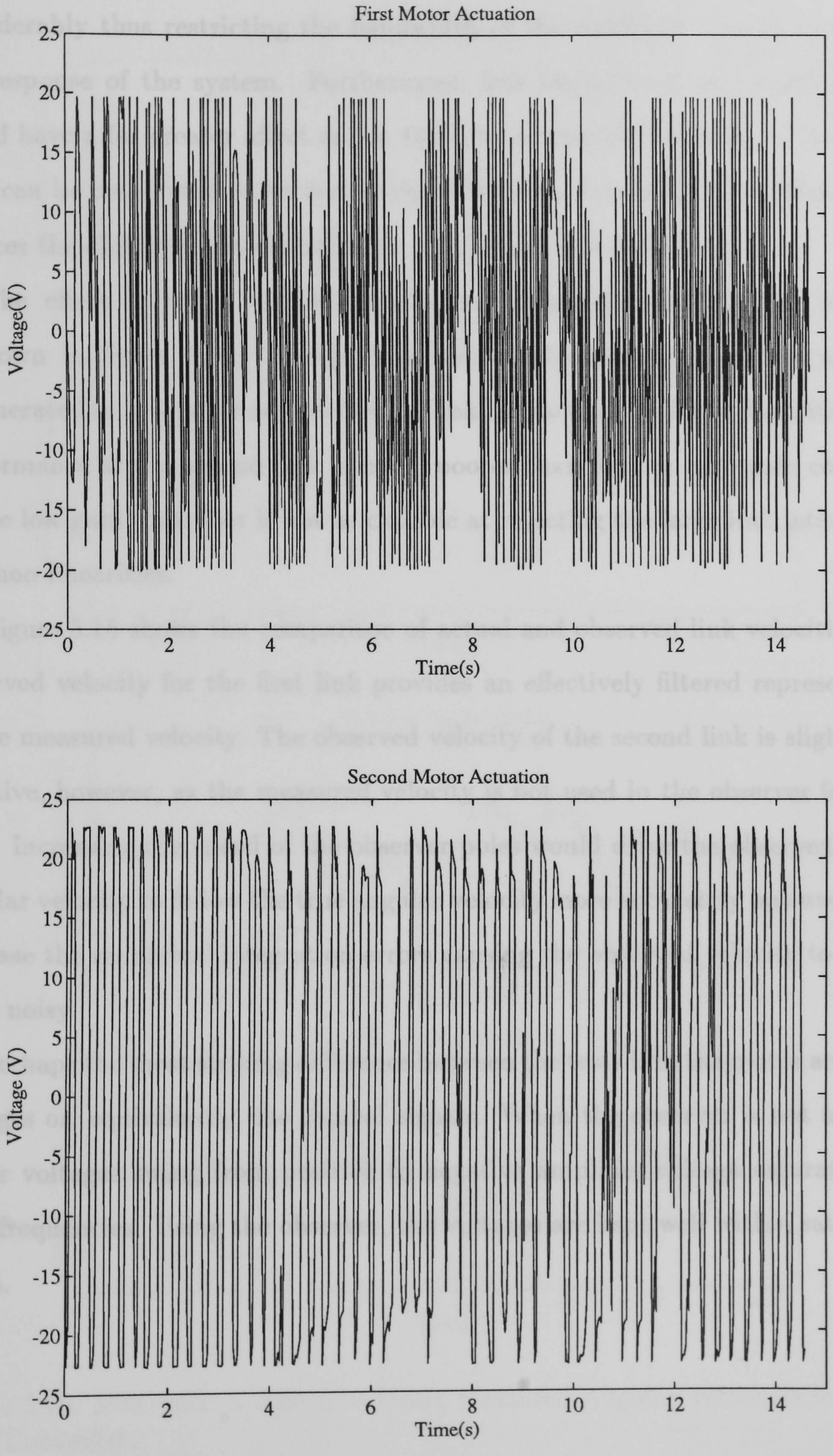


Figure 5.19: Test 2. Motor Actuation Voltages using Measured Angular Velocities in the Controller.

the non-observed system the gains of the feedback controller have to be reduced considerably thus restricting the bandwidth of the controller and slowing down the response of the system. Furthermore, link interactions and non-linearities would have a far greater affect as, for the simple independent joint pd controller, they can be considered to be system disturbances . Lowering the feedback gains reduces the ability of the controller to reject these disturbances.

The effect of using low feedback gains with measured angular velocities is shown in figure 5.20. The performance of the first link has not seriously degenerated as it is not much affected by link interactions and non-linearities. The performance of the second link is much poorer than for the high gain controller as the low gain controller is not as capable at rejecting the large link interactions and non-linearities.

Figure 5.15 shows the comparison of actual and observed link velocities. The observed velocity for the first link provides an effectively filtered representation of the measured velocity. The observed velocity of the second link is slightly less effective, however, as the measured velocity is not used in the observer feedback loop. Increasing the speed of the observer poles would drive the observed second angular velocity to follow the true angular velocity more accurately but would also increase the numerical integration errors causing the observed velocity to appear more noisy.

Perhaps the most striking difference between the tests is in the motor actuation voltages or, equivalently, the control signals. When the observer is not used the motor voltages swing from positive to negative amplifier voltage saturations at high frequencies. Using the observer, the voltages are kept well within saturation levels.

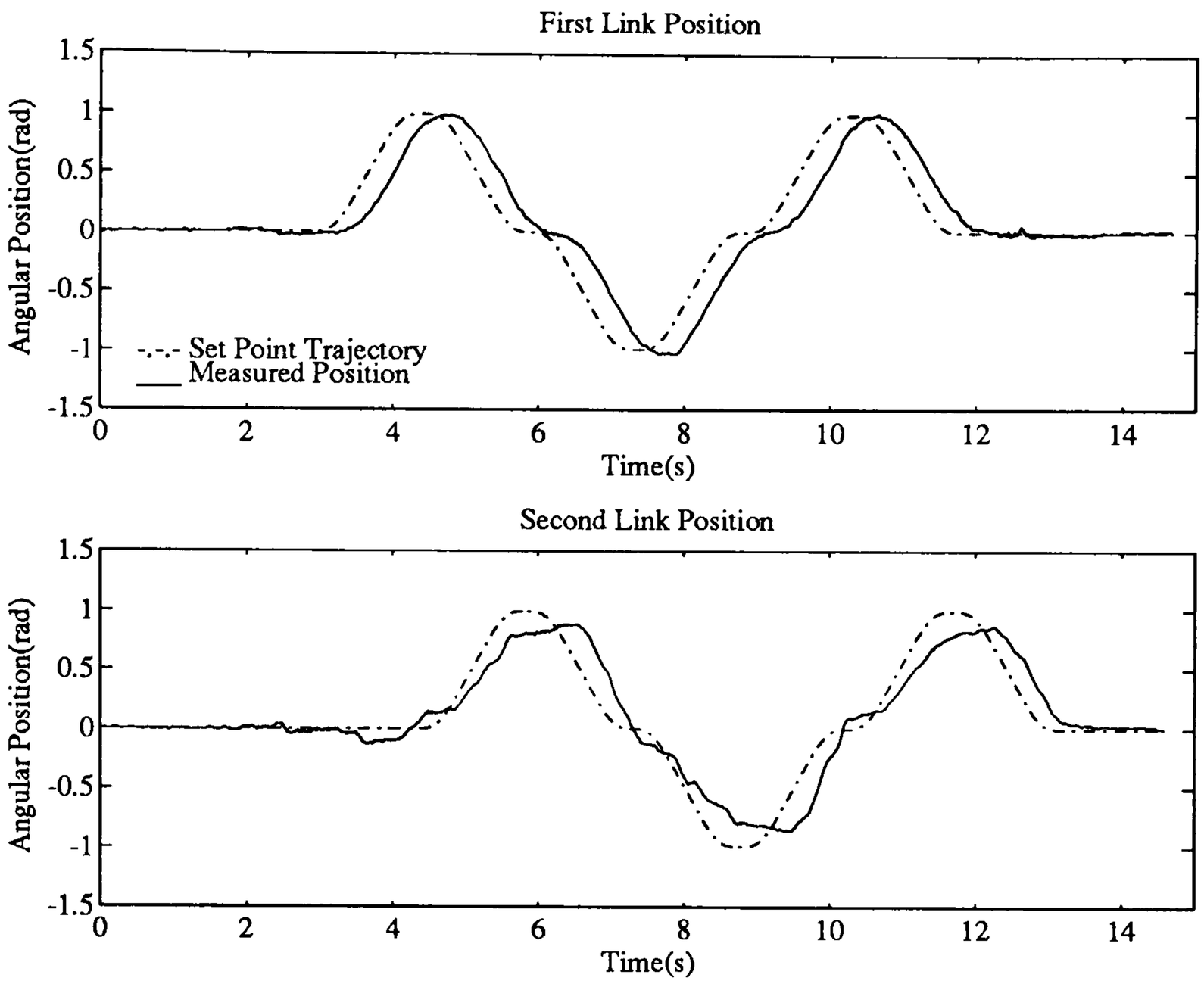


Figure 5.20: Manipulator Response Using Measured Angular Velocities in a Low-Gain Controller.

Square Wave Set Point Trajectories.

To discover how the observer copes under more testing conditions, the links were made to follow a square set-point trajectory which induces high voltages and velocities in the system. Again, two comparative tests were carried out:

- Test 3. Observed angular velocities used.

- Test 4. Measured/differentiated-position angular velocities used.

The results of these test are shown in figures 5.21 to 5.26. Again, it can be seen that the response using the observer is smooth and stable as opposed to the highly oscillatory response of the non-observed test.

There are, however, some apparent deficiencies in the control of the manipulator when the observer is used. With reference to the position measurement of the second link in figure 5.21 it can be seen that, at point A, there is a larger interaction between the links than that seen for the non-observed case in figure 5.24. Additionally, at the points labelled B on figure 5.21, the links do not reach their set points.

The large link interaction is caused not by poorer control of the second link but by the faster response of the first link inducing higher velocities and hence larger interactions. The controller tries to reject this disturbance but is constrained by amplifier saturation (see figure 5.23) for which the only cure is a more capable amplifier. The interaction is smaller when the observer is not used because the first link only attains a velocity of approximately -3radians/sec rather than the -4radians/sec when the observer is used. The first link angular accelerations will also be much lower.

This slower response is due to the continual voltage saturation evident in figure 5.26 which depletes the capacitor reserves of the amplifier which is capable of providing currents over 2 Amps only for short periods. When the amplifier is required to provide a sustained voltage to the first motor it fails and reverts to its

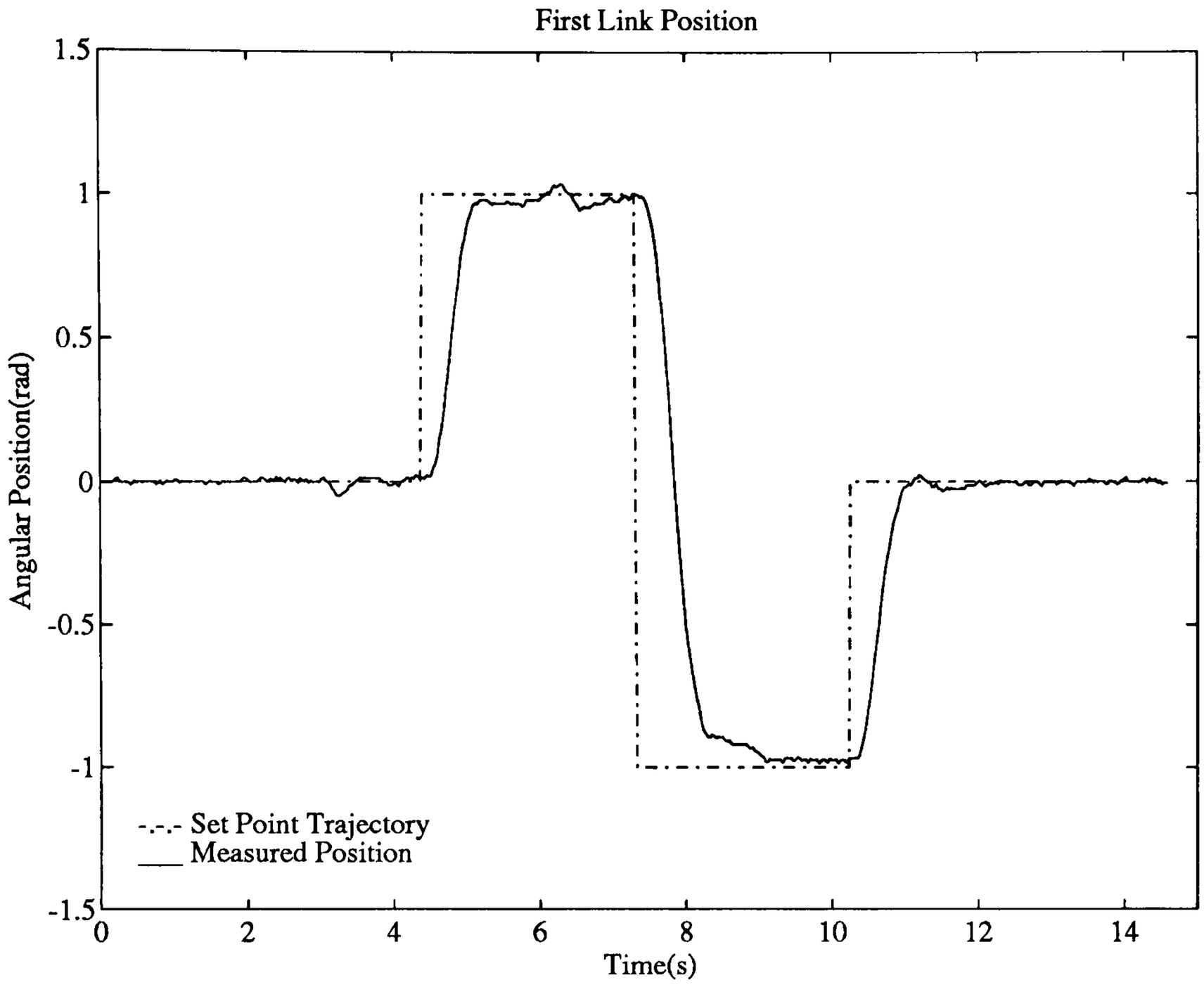


Figure 5.21: Test 3. Link Positions and Set Points using Observed Angular Velocities in the Controller.

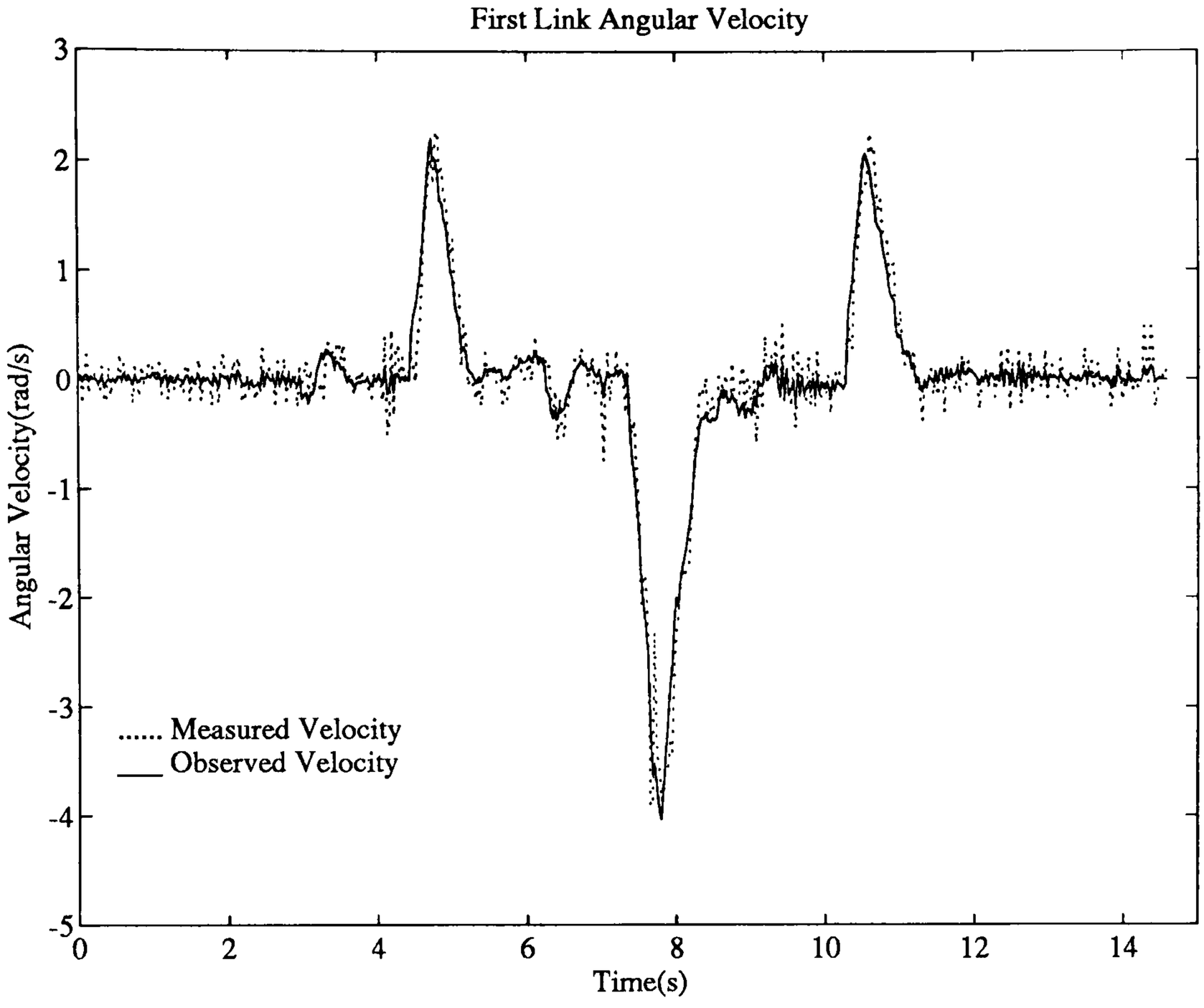


Figure 5.22: Test 3. Link Angular Velocities using Observed Angular Velocities in the Controller.

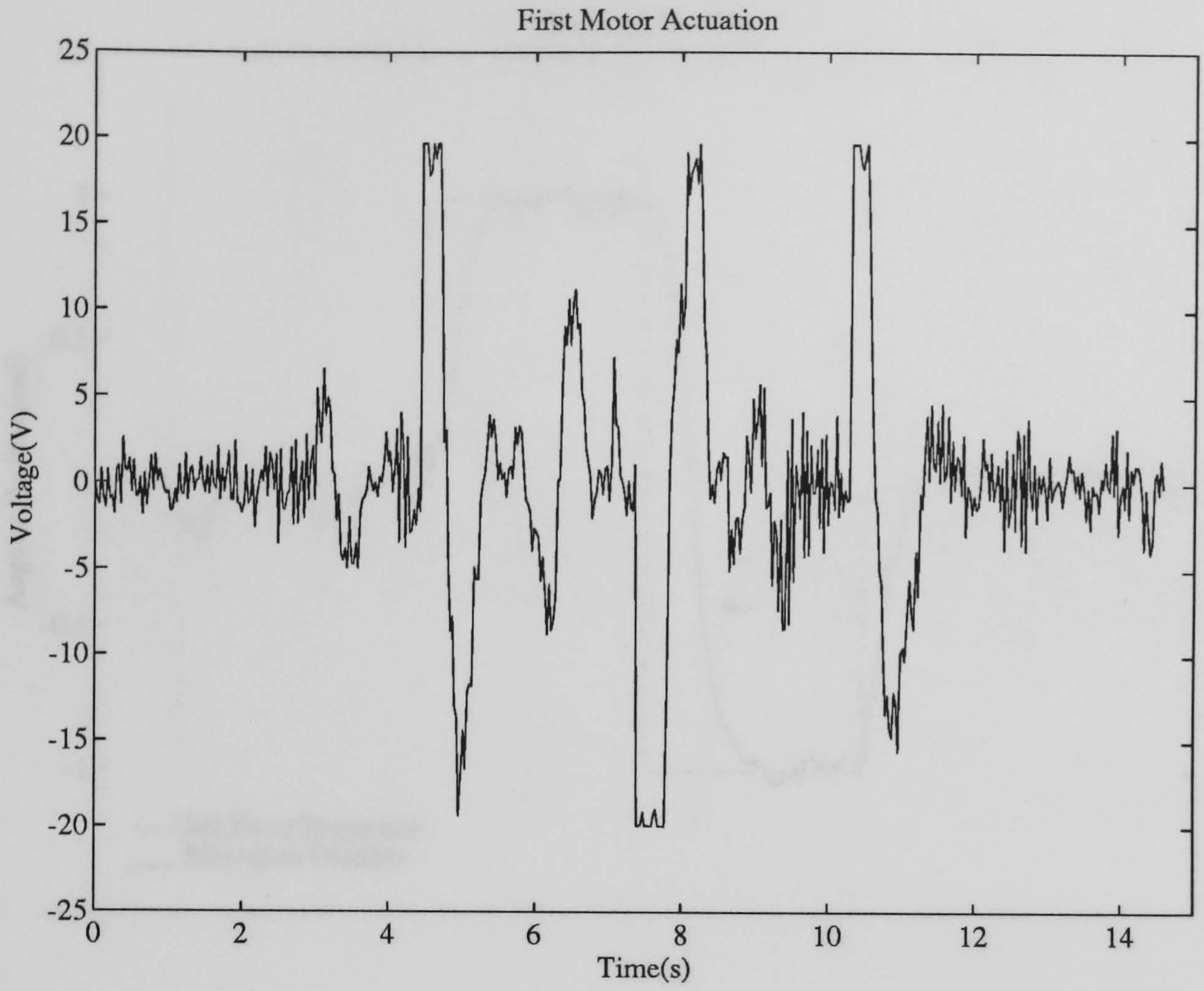


Figure 5.23: Test 3. Motor Actuation Voltages using Observed Angular Velocities in the Controller.

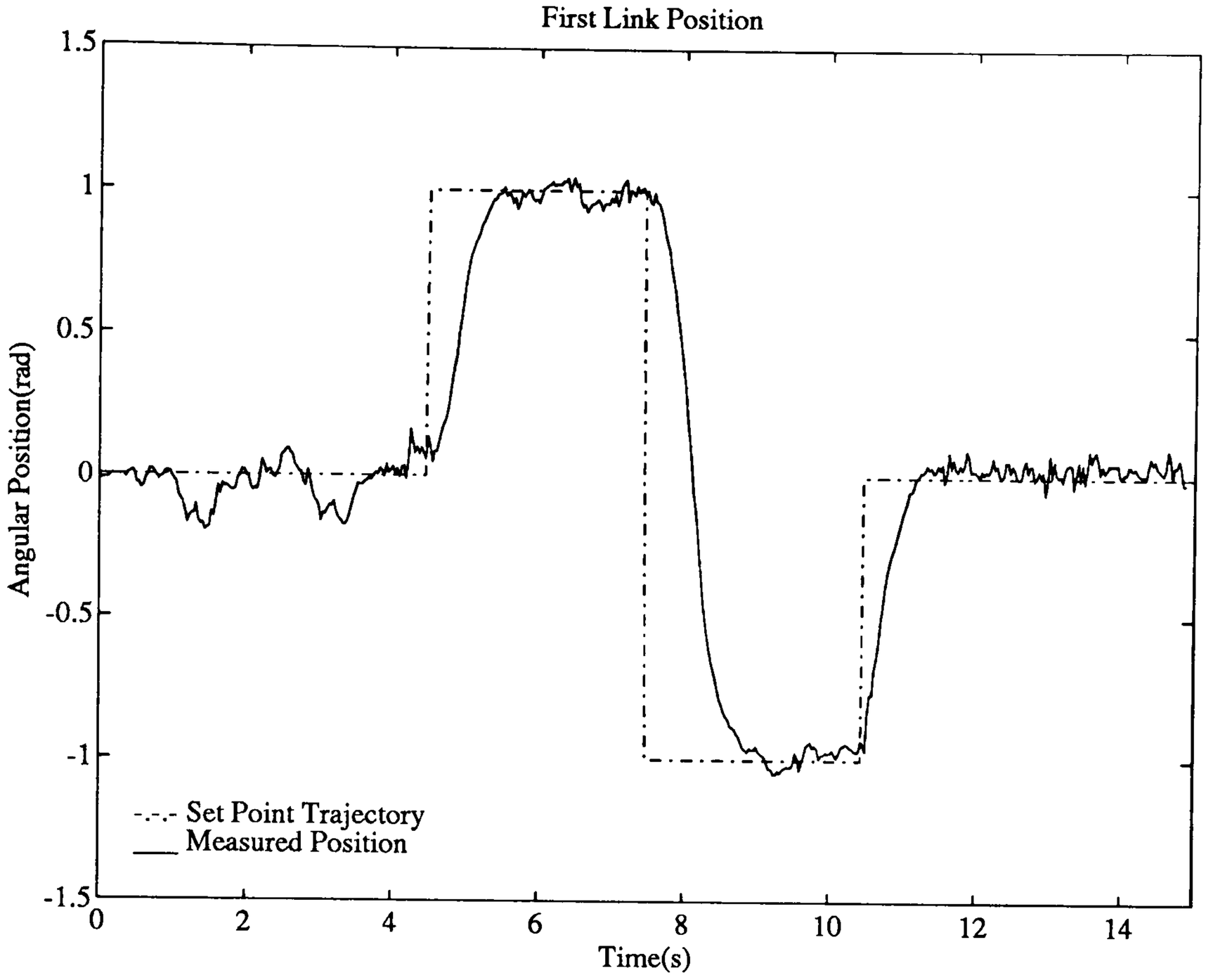


Figure 5.24: Test 4. Link Positions and Set Points using Measured Angular Velocities in the Controller.

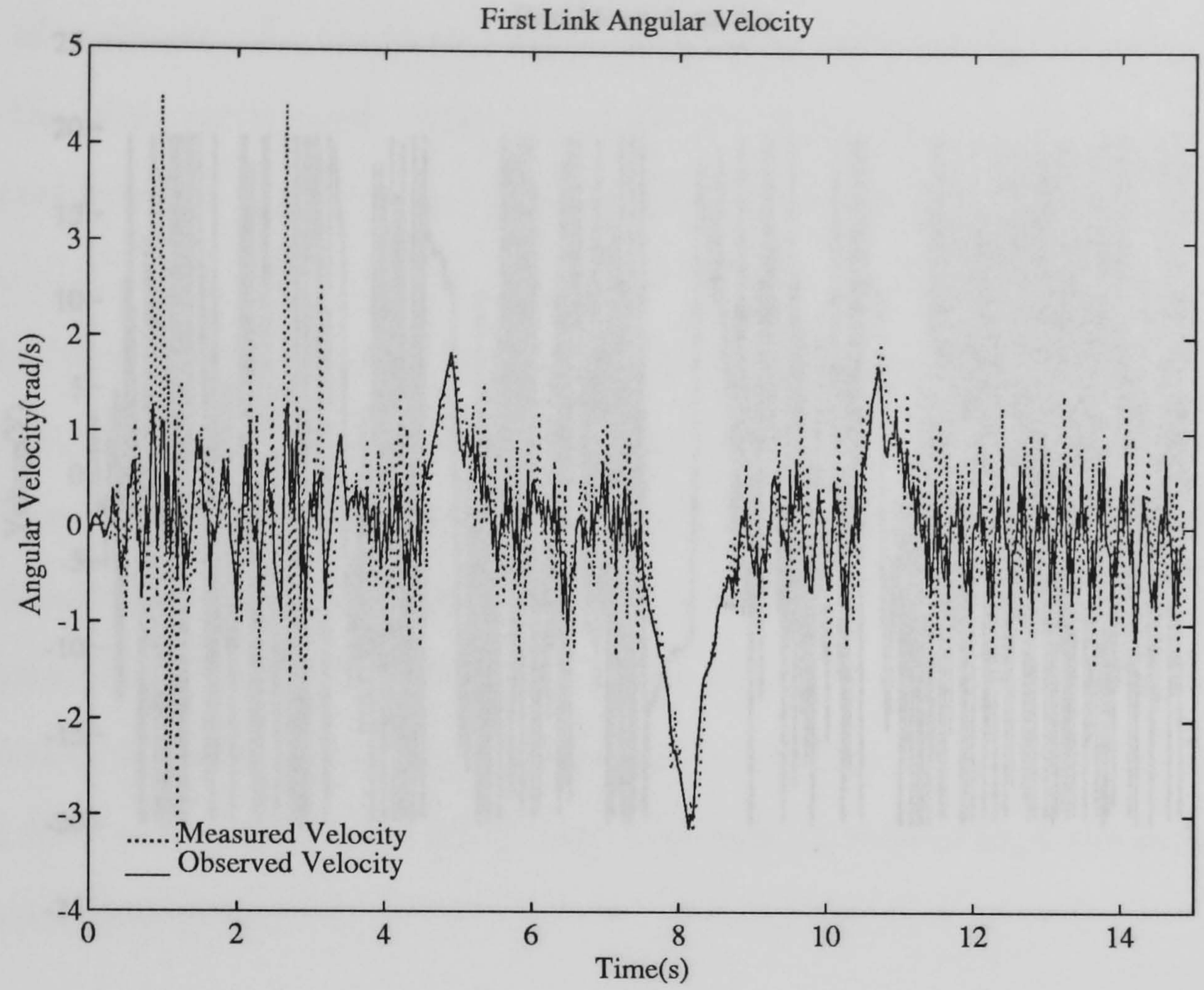
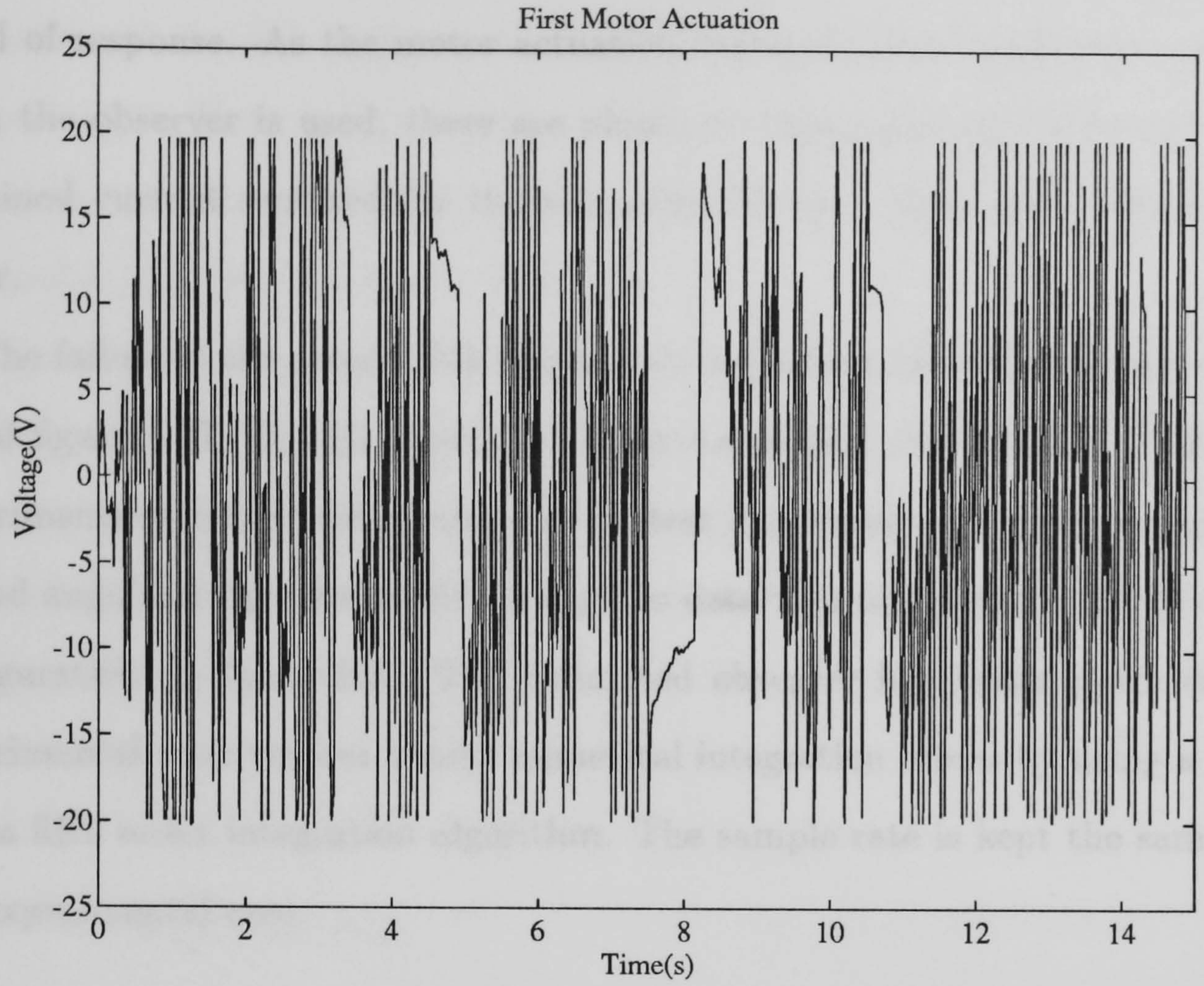


Figure 5.25: Test 4. Link Angular Velocities using Measured Angular Velocities in the Controller.



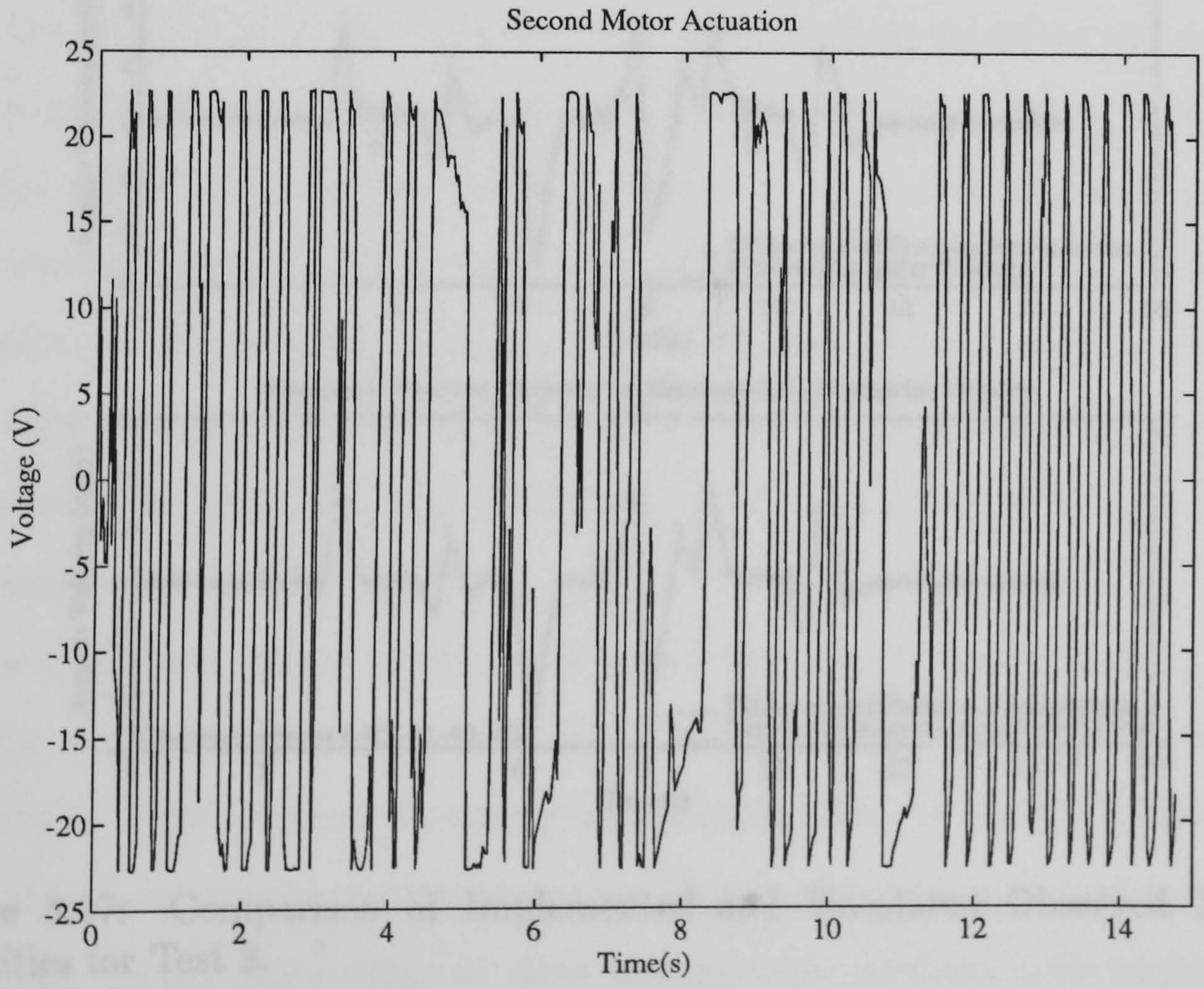


Figure 5.26: Test 4. Motor Actuation Voltages using Measured Angular Velocities in the Controller.

continual current rating of 2 Amps thus reducing the available torque and thus the speed of response. As the motor actuation voltages do not approach saturation when the observer is used, there are plenty of capacitor reserves to provide the sustained current required for the set point change. The response is therefore faster.

The failure of the second link to reach its set-points can be explained with the use of figure 5.27. In this figure, the observed second angular velocity from the experimentally implemented observer in test 3 is compared with the simulated second angular velocity obtained using the data files for test 3 in the SIMULAB configuration of figure 5.8. The simulated observer has faster poles than the experimental observer but avoids numerical integration errors by using a Runge-Kutta fifth order integration algorithm. The sample rate is kept the same as for the experimental case.

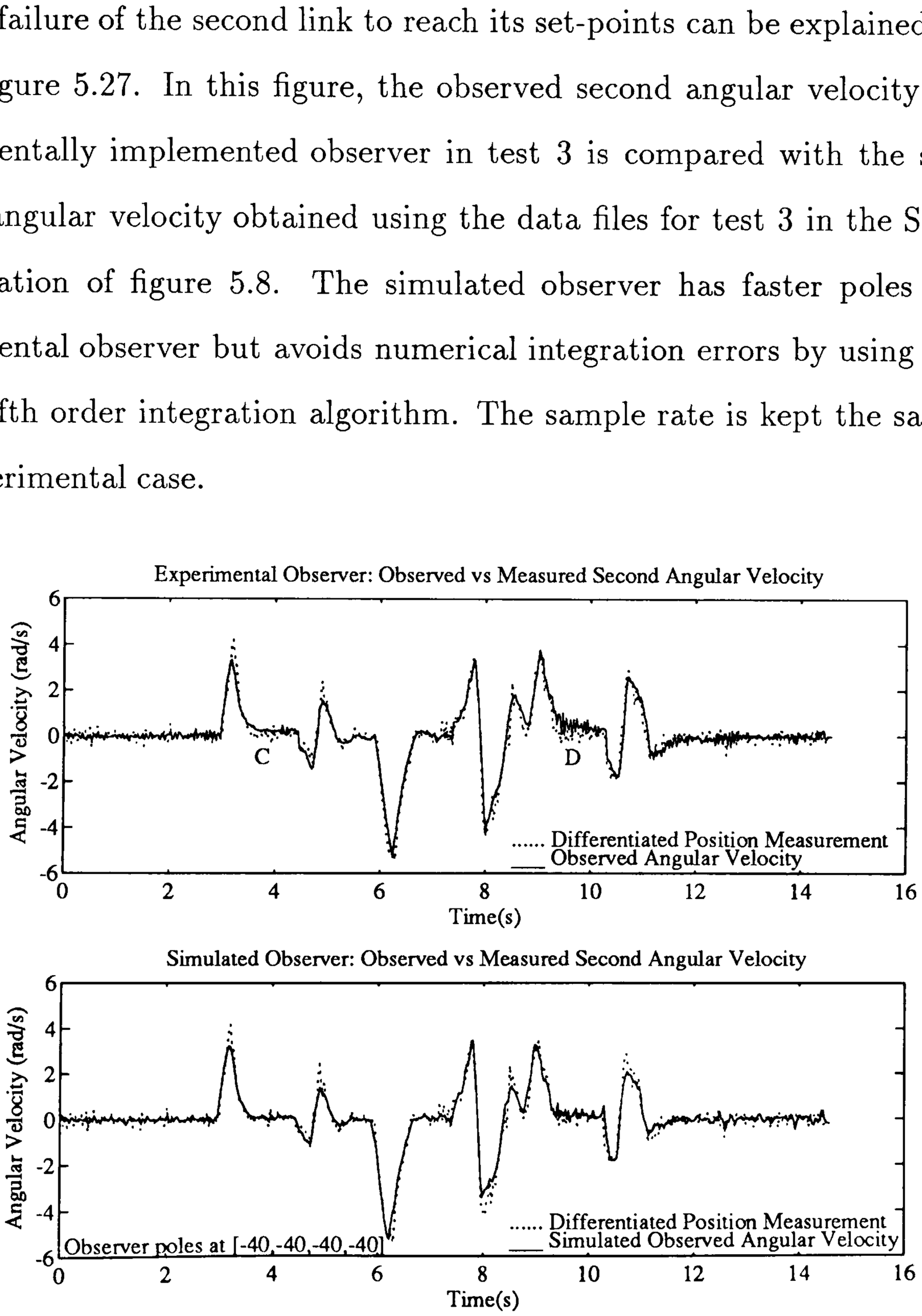


Figure 5.27: Comparison of Implemented and Simulated Observed Angular Velocities for Test 3.

The reason for the second link failing to reach its set point is the over-estimation of the link angular velocity at points C and D on the upper graph of figure 5.27. This over-estimation causes an offset voltage from the derivative part of the pd controller which is then balanced by a proportional voltage offset resulting in a steady state positional error.

The over-estimation of the angular velocity is caused by slow observer poles which do not allow the rejection of the disturbance caused by un-modelled stiction. Speeding up the poles to [-40,-40,-40,-40] significantly improves the estimation of angular velocity and would virtually cure the problem but, due to the lack of computational speed, the Runge-Kutta integration algorithm could not be implemented.

5.5 Conclusion.

It has been shown that a non-linear bond graph derived observer may be used to improve the control of an experimental manipulator by replacing the measured link angular velocities (or differentiated position measurements) with estimated angular velocities generated by an observer. Using a linear feedback loop, the observer can be made to track the states of the experimental manipulator accurately and provide relatively well-conditioned observed angular velocities. This allows the gains of a simple independent joint pd controller to be significantly increased thus extending the bandwidth of the controller and improving the performance of the manipulator.

An important aspect of the bond graph observer is that it may be created quickly and easily from the bond graph of the normal system. The software required to implement the observer practically may then be created automatically from this bond graph. Furthermore, the linearised state-space matrices can be produced for any state-point to allow the observer feedback gain matrix to be

designed using standard linear observer theory.

Whilst the computational system used to implement the observer is very crude and lacking in speed of computation, it has been shown, with the use of simulations, that the implementation of the observer may be improved considerably with the use of more powerful computers with dedicated communications systems. Additional computing power would allow the use of more sophisticated integration algorithms thus enabling faster observer poles to be implemented without encountering numerical integration problems. Dedicated communication links would allow for higher sample rates. These computing facilities are available but pressure of time disallowed their use in this research.

Finally, it must be stated that the techniques presented in this chapter are generic and could be easily used for more complex systems than the two-link manipulator. The relative simplicity of the two link manipulator allowed the results of the research to be implemented practically to demonstrate that the techniques would work for real systems.

Chapter 6

Inverse-Model Based Control.

6.1 Introduction

The core bond graph of a system defines how the states of the system interact in an unforced manner without inputs and outputs. It is similar to the 'A' matrix of the standard linear state-space representation. This core is a fundamental representation of the physics of the system and through judicious choice of **S**ource and **M**easurement elements, the inputs and outputs of the system may be altered without having to modify the core.

The ability to modify the input/output space of a model easily is useful in robotic control as it allows inverse-model based control schemes such as feed-forward control [5] or computed torque control [8] [9] [3] to be implemented. These control schemes, outlined in chapter 1, calculate the input torques required to force the manipulator to follow a predetermined trajectory $(\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d)$, where $\boldsymbol{\theta}_d$ is a vector of desired joint angles. Hence the input to the model is the trajectory and the output is the torque vector. The model is therefore run in an 'inverse' way to the physical system.

This chapter explores how the bond graph for the experimental two-link manipulator may be modified to create the software required to implement a

computed torque controller. The controller is first created in a simulated form to control a simulated two-link manipulator. It is then created in CONIC to control the actual DD2lm. Comparisons are given between the performance of standard independent joint controllers and the inverse model-based control scheme.

As in chapter 5, the experimental results presented in this chapter were obtained without the use of filters to prevent aliasing of the sampled signals due to the excessive phase delay introduced into the measurements by the anti-aliasing filters (see appendix C).

6.2 Computed Torque Control Scheme.

To understand the requirements of the controller we shall review the computed torque control scheme, first outlined in chapter 1. The computed torque technique [8] [14] is also known as the 'inverse problem technique' [3] [9]. It is similar to the feed-forward controller, in that it calculates the torques required to drive the manipulator to follow a desired trajectory but differs in that the inverse model is incorporated into the feedback loop in order to decouple the dynamics of the links.

The general form of the dynamic equations of motion for a robotic manipulator is

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{\theta})(\ddot{\boldsymbol{\theta}}) + \mathbf{V}(\dot{\boldsymbol{\theta}}) + \mathbf{f}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta}; i, j = 1, 2...n) + \boldsymbol{g}(\boldsymbol{\theta}) \tag{6.1}$$

where

n	=	number of degrees of freedom of the manipulator
$\boldsymbol{\tau}$	=	n $\times$ 1 vector of generalised torques/forces
$\boldsymbol{J}(\boldsymbol{\theta})$	=	n $\times$ n inertia matrix
$\mathbf{V}$	=	n $\times$ n viscous friction matrix
$\mathbf{f}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta})$	=	n $\times$ 1 vector of Coriolis and centrifugal terms
$\boldsymbol{g}(\boldsymbol{\theta})$	=	n $\times$ 1 vector of gravity terms

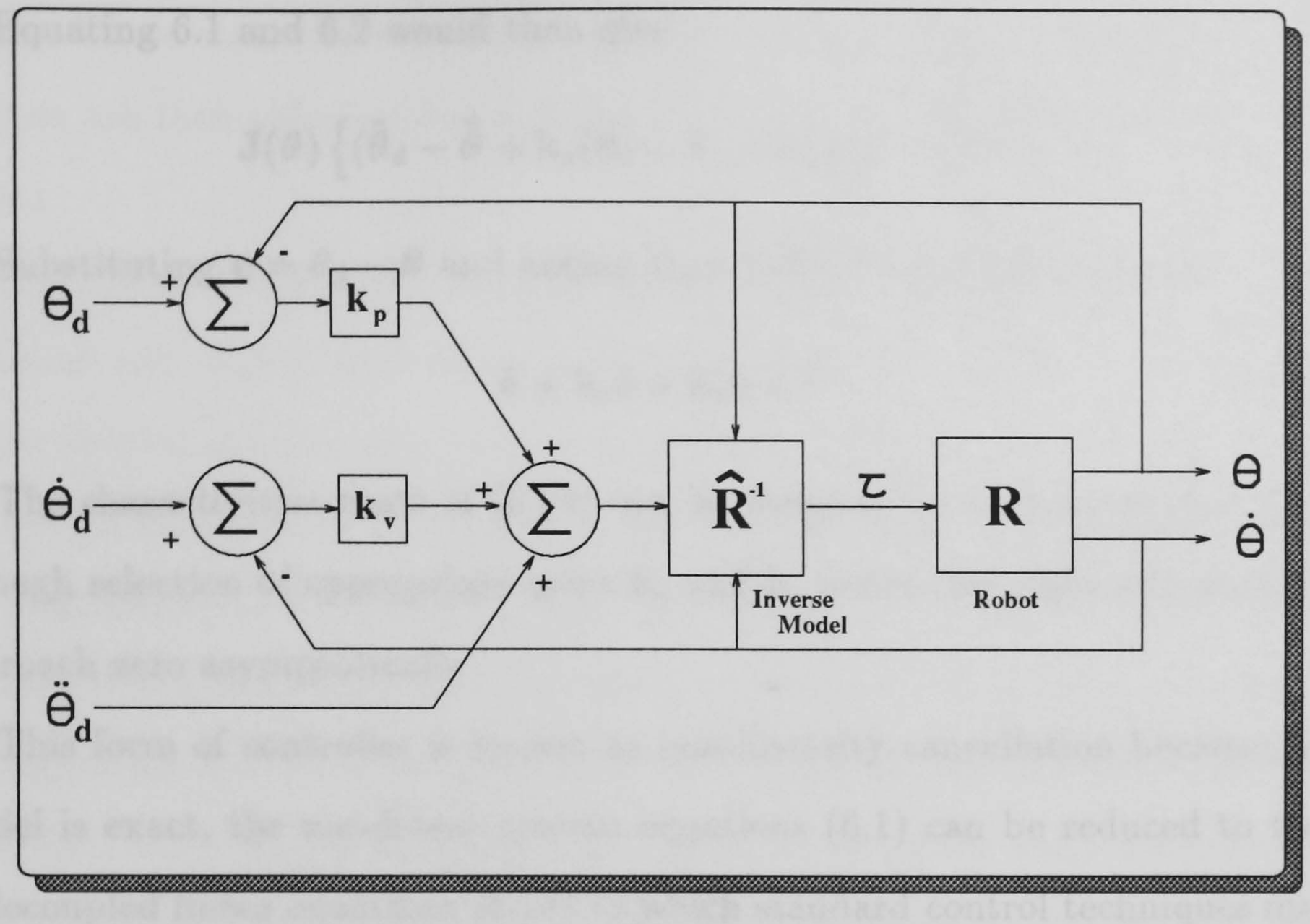


Figure 6.1: Schematic of Computed Torque Controller.

For the computed torque technique, the desired input torque is given by

$$\boldsymbol{\tau} = \hat{\mathbf{J}}(\boldsymbol{\theta})\left\{(\ddot{\boldsymbol{\theta}}_d + \mathbf{k}_v(\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}) + \mathbf{k}_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}))\right\} + \hat{\mathbf{V}}(\dot{\boldsymbol{\theta}}) + \hat{\mathbf{f}}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta}; i, j = 1, 2...n) + \hat{\mathbf{g}}(\boldsymbol{\theta}) \tag{6.2}$$

or, more concisely

$$\boldsymbol{\tau} = \hat{R}^{-1}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}^*) \tag{6.3}$$

where the augmented acceleration vector $\ddot{\boldsymbol{\theta}}^*$ is given by

$$\ddot{\boldsymbol{\theta}}^* = \ddot{\boldsymbol{\theta}}_d + \mathbf{k}_v(\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}) + \mathbf{k}_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}) \tag{6.4}$$

If the model is exact then

$$\hat{\mathbf{J}}\boldsymbol{\theta} = \mathbf{J}\boldsymbol{\theta} \tag{6.5}$$

$$\hat{\mathbf{V}}(\dot{\boldsymbol{\theta}}) = \mathbf{V}(\dot{\boldsymbol{\theta}}) \tag{6.6}$$

$$\hat{\mathbf{f}}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta}; i, j = 1, 2...n) = \mathbf{f}(\dot{\theta}_i\dot{\theta}_j, \boldsymbol{\theta}; i, j = 1, 2...n) \tag{6.7}$$

$$\hat{\mathbf{g}}(\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}) \tag{6.8}$$

Equating 6.1 and 6.2 would then give

$$\mathbf{J}(\boldsymbol{\theta})\left\{(\ddot{\boldsymbol{\theta}}_d - \ddot{\boldsymbol{\theta}} + \mathrm{k}_v(\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}) + \mathrm{k}_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}))\right\} = 0 \tag{6.9}$$

Substituting $\mathbf{e} = \boldsymbol{\theta}_d - \boldsymbol{\theta}$ and noting that $\mathbf{J}(\boldsymbol{\theta})$ is nonsingular yields

$$\ddot{\mathbf{e}} + \mathrm{k}_v\dot{\mathbf{e}} + \mathrm{k}_p\mathbf{e} = 0 \tag{6.10}$$

The characteristic roots of (6.10) can be assigned to have negative real parts through selection of appropriate gains $\mathbf{k}_v$ and $\mathbf{k}_p$ hence the trajectory error $\mathbf{e}$ will approach zero asymptotically.

This form of controller is known as non-linearity cancellation because, if the model is exact, the non-linear system equations (6.1) can be reduced to the set of decoupled linear equations (6.10) to which standard control techniques may be applied.

A disadvantage of computed torque is that the input torque $\boldsymbol{\tau}$ is calculated by the model using inputs based on the actual trajectory and not just the pre-planned trajectory. Consequently, the model must be run on-line thus reducing the sample rate for a given control computer. As the speed and capability of computing hardware and software increase, however, this drawback should become less significant.

An important aspect to note is that the inverse model $\hat{R}^{-1}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}^*)$ is a *function* of the link positions, velocities and augmented link accelerations, and not a dynamic system in itself; i.e. the inverse model does not possess states as such, they must be passed to the model as parameters.

6.3 Creation of the Inverse Model Bond Graph.

Bond graphs containing elements with integral causality are system models rather than functions as they contain states. In the system dynamic equations of motion. the state derivatives are calculated with reference to the system states and system

inputs. These state derivatives are then integrated to update the states. System outputs are then calculated as a function of the states and, possibly, the system inputs.

The states of the model for the two-link manipulator are defined by the link positions and angular velocities. In the inverse model required by the computed torque control scheme these states are passed to the model as parameters. It is the output equations of the model that are required with the model inputs being the augmented accelerations $\ddot{\theta}^*$. We therefore need to augment the bond graph for the DD2lm with angular accelerations as inputs and link torques as outputs.

Figure 6.2 shows the inverse model bond graph for the two-link manipulator. The input accelerations **a1** and **a2** are integrated using the unity **I** elements **int1** and **int2** to give angular velocities which are then imposed onto the link angular velocity junctions **vtr1** and **vtr2** through the **0:t1** and **0:t2** junctions. The output torques are obtained using **M**easurement elements from these junctions.

The dynamic equations of motion for this model are:

$$x = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \omega_1 \\ \omega_2 \end{pmatrix} ; \; z = \begin{pmatrix} h_1 \\ m_1\dot{x}_1 \\ m_1\dot{y}_1 \\ h_2 \\ m_2\dot{x}_2 \\ m_2\dot{y}_2 \\ m_m \\ m_m \end{pmatrix} ; \; y = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} ; \; u = \begin{pmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \end{pmatrix} \tag{6.11}$$

$$\dot{x}_1 = x_3 \tag{6.12}$$

$$\dot{x}_2 = x_4 \tag{6.13}$$

$$\dot{x}_3 = u_1 \tag{6.14}$$

$$\dot{x}_4 = u_2 \tag{6.15}$$

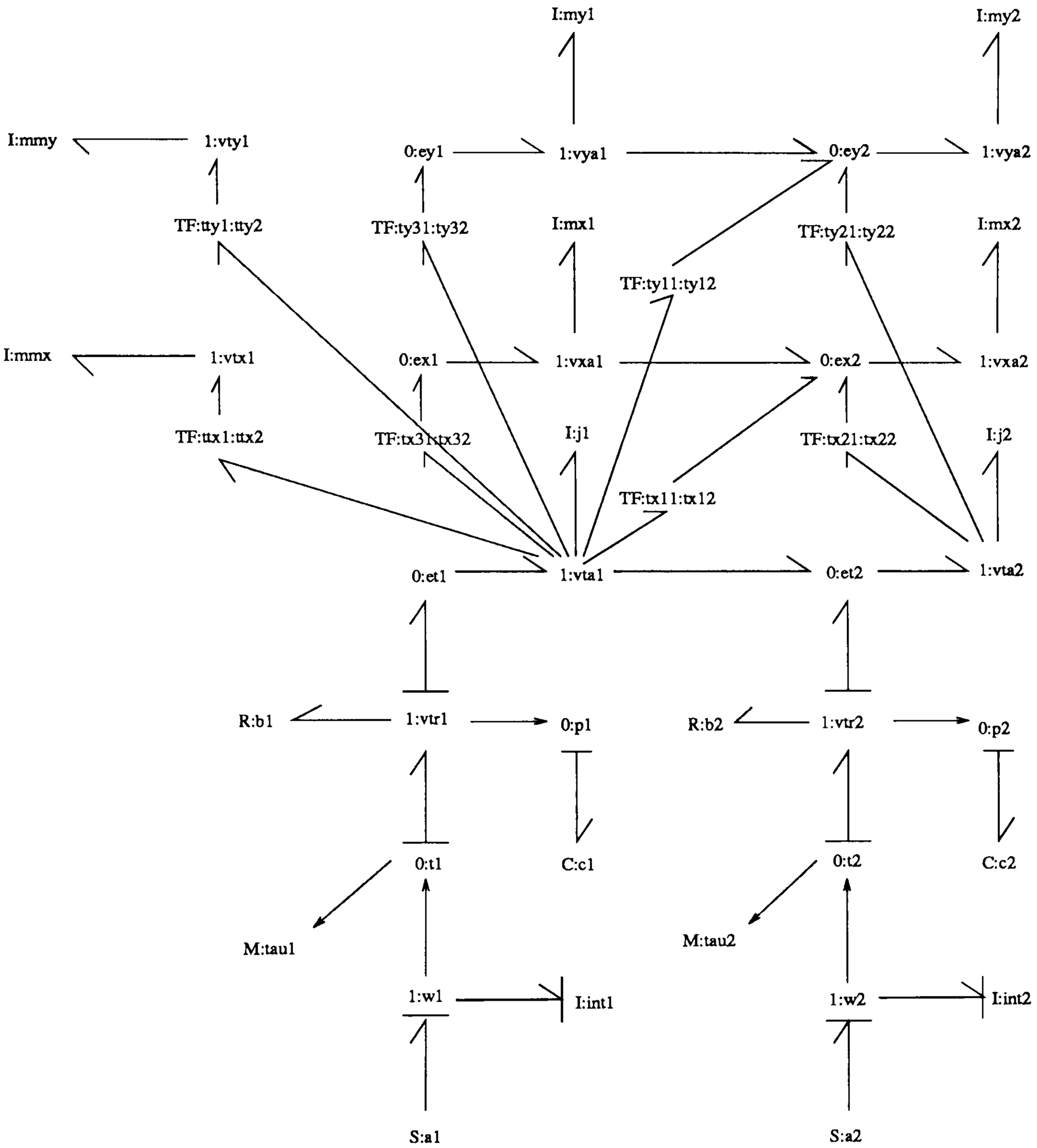


Figure 6.2: Inverse-Model Bond Graph.

$$y_1 = \frac{1}{2}\begin{pmatrix} (\dot{z}_2 + 2\dot{z}_5 + 2\dot{z}_7)sin(x_1)l_1 + \dots \\ \dots(\dot{z}_3 + 2\dot{z}_6 + 2\dot{z}_8)cos(x_1)l_1 + \dots \\ \dots sin(x_1 + x_2)l_2\dot{z}_5 + cos(x_1 + x_2)l_2\dot{z}_6 + \dots \\ \dots 2b_1x_3 + 2\dot{z}_1 + 2\dot{z}_4 \end{pmatrix} \tag{6.16}$$

$$y_2 = \frac{(sin(x_1 + x_2)l_2\dot{z}_5 + cos(x_1 + x_2)l_2\dot{z}_6 + 2b_2x_4 + 2\dot{z}_4)}{2} \tag{6.17}$$

$$z_1 = \frac{(l_1^2 m_1 x_3)}{12} \tag{6.18}$$

$$z_2 = \frac{(sin(x_1)l_1 m_1 x_3)}{2} \tag{6.19}$$

$$z_3 = \frac{(cos(x_1)l_1 m_1 x_3)}{2} \tag{6.20}$$

$$z_4 = \frac{((x_3 + x_4)l_2^2 m_2)}{12} \tag{6.21}$$

$$z_5 = \frac{(((x_3 + x_4)sin(x_1 + x_2)l_2 + 2sin(x_1)l_1 x_3)m_2)}{2} \tag{6.22}$$

$$z_6 = \frac{(((x_3 + x_4)cos(x_1 + x_2)l_2 + 2cos(x_1)l_1 x_3)m_2)}{2} \tag{6.23}$$

$$z_7 = sin(x_1)l_1 m_m x_3 \tag{6.24}$$

$$z_8 = cos(x_1)l_1 m_m x_3 \tag{6.25}$$

It can be seen that the states of the model are $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$. For the computed torque control scheme, these states are passed to the model as parameters hence only the output equations of the above model need be used. There is, however, a problem in using the standard Model Transformation Toolbox, MTT [30], as the output equations contain terms in the constrained state derivatives $\dot{\mathbf{z}}$. The standard form of MTT assumes that constrained state derivatives will not appear in the output equations and deletes them if they do. Consequently, the MTT files must be modified to retain them and to differentiate the constrained state equations so the derivatives may be replaced with equations in terms of the true states. The modified MTT files are given in appendix F.

When this is done for the inverse model, the constrained state derivatives and output equations become:

$$\dot{z}_1 = \frac{(l_1^2 m_1 u_1)}{12} \tag{6.26}$$

$$\dot{z}_2 = \frac{((sin(x_1)u_1 + cos(x_1)x_3^2)l_1 m_1)}{2} \tag{6.27}$$

$$\dot{z}_3 = \frac{(-(sin(x_1)x_3^2 - cos(x_1)u_1)l_1 m_1)}{2} \tag{6.28}$$

$$\dot{z}_4 = \frac{((u_1 + u_2)l_2^2 m_2)}{12} \tag{6.29}$$

$$\dot{z}_5 = \frac{m_2}{2}\begin{pmatrix} (x_3 + x_4)^2 cos(x_1 + x_2)l_2 + 2sin(x_1)l_1 u_1 + \dots \\ \dots (u_1 + u_2)sin(x_1 + x_2)l_2 + 2cos(x_1)l_1 x_3^2 \end{pmatrix} \tag{6.30}$$

$$\dot{z}_6 = \frac{-m_2}{2}\begin{pmatrix} (x_3 + x_4)^2 sin(x_1 + x_2)l_2 - 2cos(x_1)l_1 u_1 + \dots \\ \dots 2sin(x_1)l_1 x_3^2 - (u_1 + u_2)cos(x_1 + x_2)l_2 \end{pmatrix} \tag{6.31}$$

$$\dot{z}_7 = (sin(x_1)u_1 + cos(x_1)x_3^2)l_1 m_m \tag{6.32}$$

$$\dot{z}_8 = -(sin(x_1)x_3^2 - cos(x_1)u_1)l_1 m_m \tag{6.33}$$

$$y_1 = \frac{1}{6}\begin{pmatrix} 2((m_1 + 3m_2 + 3m_m)l_1^2 + l_2^2 m_2)u_1 - \dots \\ \dots 3((2x_3 + x_4)cos(x_1)x_4 - \dots \\ \dots (2u_1 + u_2)sin(x_1))sin(x_1 + x_2)l_1 l_2 m_2 + \dots \\ \dots 3(2x_3 + x_4)sin(x_1)cos(x_1 + x_2)l_1 l_2 m_2 x_4 + \dots \\ \dots 3(2u_1 + u_2)cos(x_1 + x_2)cos(x_1)l_1 l_2 m_2 + \dots \\ \dots 2l_2^2 m_2 u_2 + 6b_1 x_3 \end{pmatrix} \tag{6.34}$$

$$y_2 = \frac{1}{6}\begin{pmatrix} 3(sin(x_1)u_1 + cos(x_1)x_3^2)sin(x_1 + x_2)l_1 l_2 m_2 + \dots \\ \dots 2(u_1 + u_2)l_2^2 m_2 - 3sin(x_1)cos(x_1 + x_2)l_1 l_2 m_2 x_3^2 + \dots \\ \dots 3cos(x_1 + x_2)cos(x_1)l_1 l_2 m_2 u_1 + 6b_2 x_4 \end{pmatrix} \tag{6.35}$$

These are the equations to be used in the computed torque controller. The inputs to the experimental manipulator are voltages however, not torques. The required input voltage for the ith link may be calculated from

$$v_i = \frac{r_{a_i}}{k_{2_i}}\tau_i + k_{2_i}\dot{\theta}_i \tag{6.36}$$

where

r_{a_i} = armature resistance of ith motor

k_{2_i} = torque constant of ith motor

τ_i = desired input torque to ith link

$\dot{\theta}_i$ = angular velocity of ith link

6.4 Results.

6.4.1 Simulation of the Computed Torque Control Scheme.

To find out how best to implement the computed torque controller it is useful to first simulate it using SIMULAB. This allows us to check that the inverse model equations work correctly in that they calculate the torques/voltages required to force a mathematical model of the manipulator to follow the desired trajectory.

The SIMULAB configuration equivalent to figure 6.1 is shown in figure 6.3. The inverse model function is constructed from the output equations derived from the inverse model bond-graph. The robot model is created from the forward bond graph model used in previous chapters. Equation 6.4 is implemented using the feedback loops to give the augmented acceleration inputs to the inverse model. Measured link velocities and positions from the robot are fed directly to the inverse model.

Desired Trajectory Generation.

The computed torque control scheme requires the desired trajectory to be given in the form of link positions, velocities and accelerations $(\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d)$ for each sample

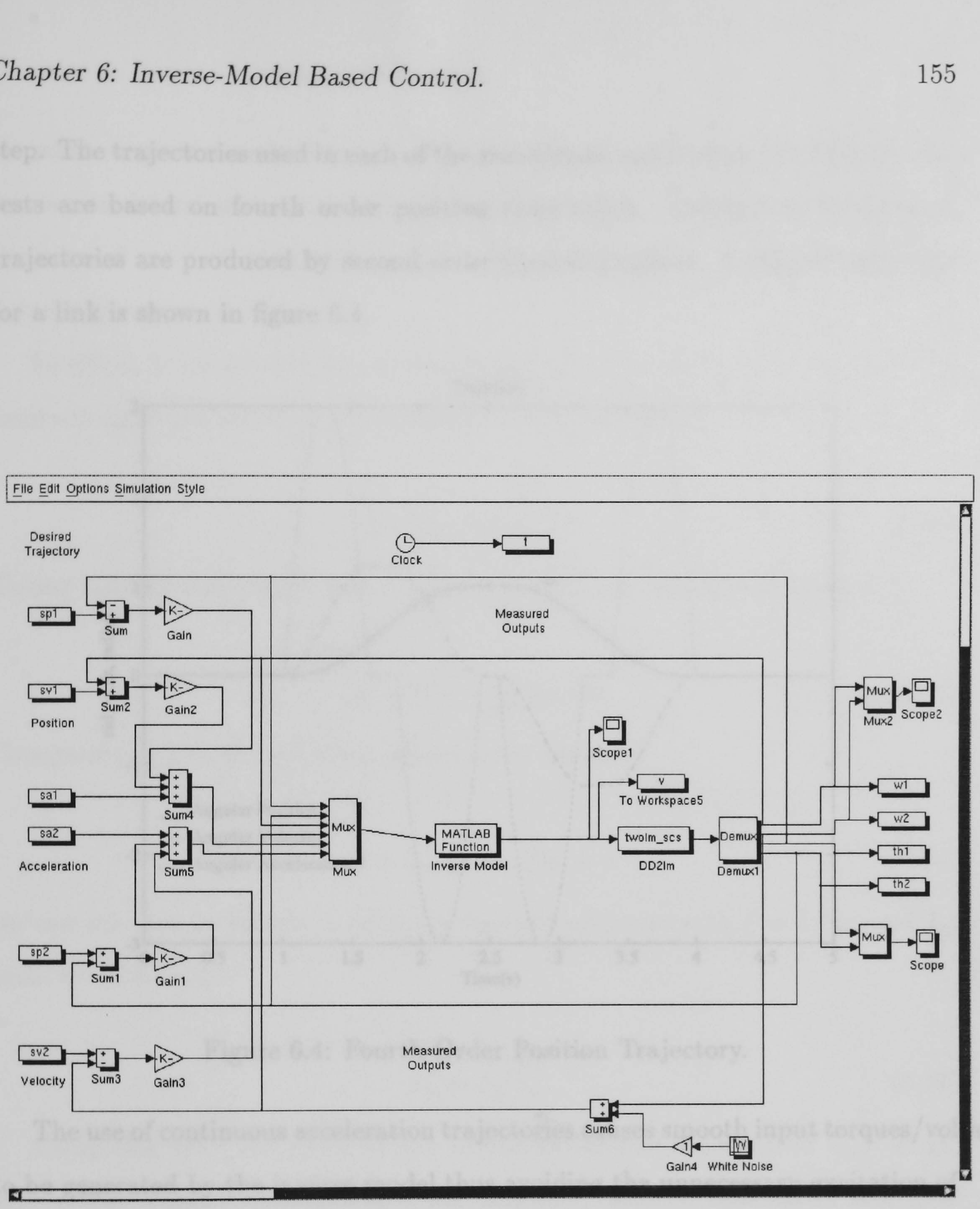


Figure 6.3: SIMULAB configuration for Computed Torque.

step. The trajectories used in each of the simulations and each of the experimental tests are based on fourth order position trajectories. Continuous acceleration trajectories are produced by second order (quartic) splines. A typical trajectory for a link is shown in figure 6.4.

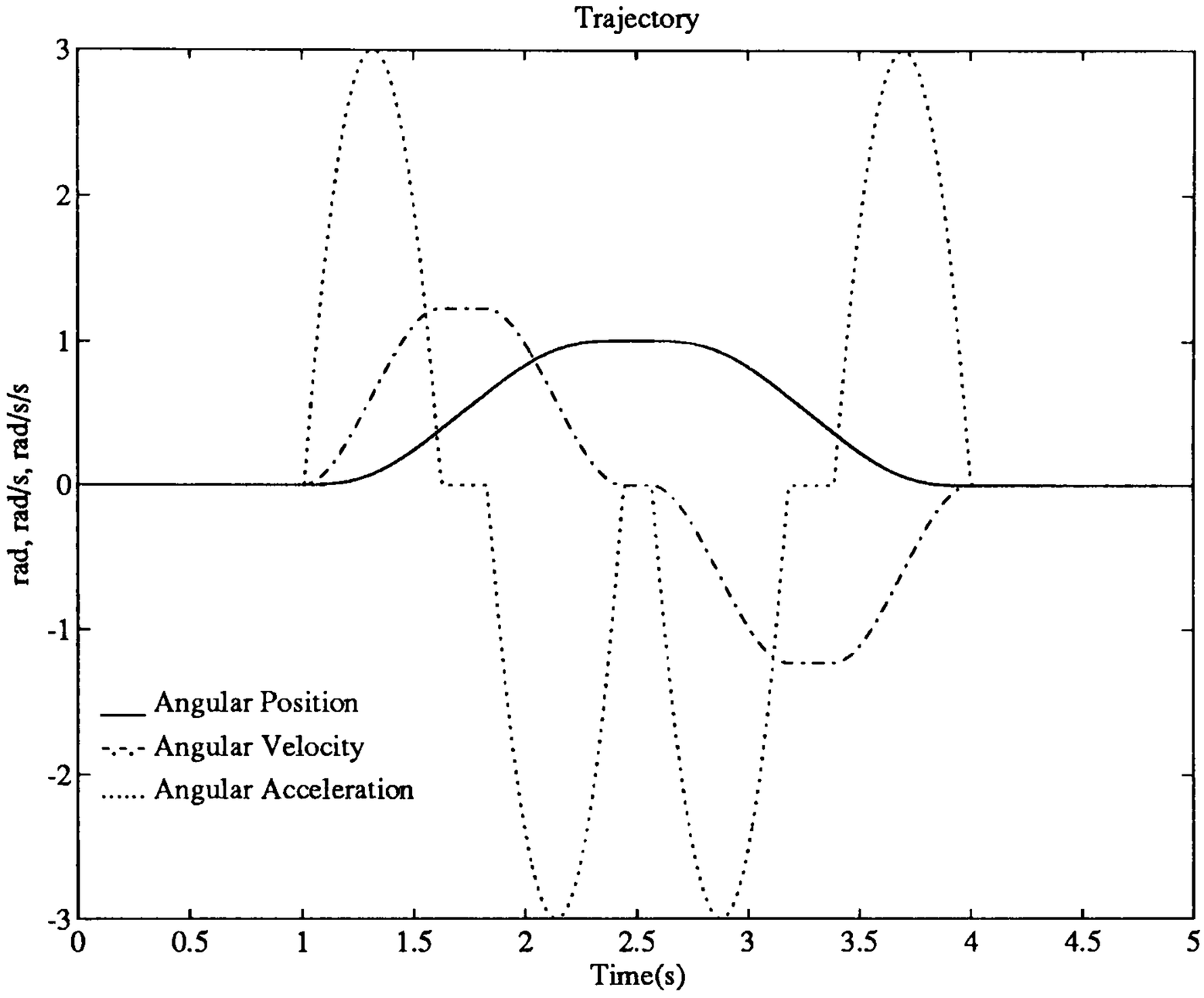


Figure 6.4: Fourth Order Position Trajectory.

The use of continuous acceleration trajectories causes smooth input torques/voltages to be generated by the inverse model thus avoiding the unnecessary excitation of higher order manipulator dynamics.

The actual trajectory used in the tests consists of a number of the above ramps linked together. To provide comparisons with earlier tests, the trajectory is similar to that used in tests 1 and 2 of the previous chapter.

6.4.2 Simulation Results.

Simulation results are shown in figures 6.5 to 6.9. Figure 6.5 shows the effect of using no feedback (i.e. zero feedback gains). The model is started with an initial

set-point error of 1 radian for the first link and -1 radian for the second. The inverse model calculates the voltage required to drive the manipulator around the form of the desired trajectory but with no feedback the initial errors are not reduced to zero.

Feedback is introduced for the simulation shown in figures 6.6 and 6.7. The feedback gains are selected with reference to equation 6.10.

$$\ddot{\mathbf{e}} + \mathbf{k}_v\dot{\mathbf{e}} + \mathbf{k}_p\mathbf{e} = 0 \tag{6.37}$$

Taking Laplace transforms with zero initial conditions for the ith link yields

$$(s^2 + k_{v_i}s + k_{p_i})e_i = 0 \tag{6.38}$$

Comparing this to the standard second order equation

$$(s^2 + 2\zeta\omega_n s + \omega_n^2)e_i = 0 \tag{6.39}$$

we can see that to achieve a critically damped response with $\zeta = 1$, k_{v_i} and k_{p_i} must be related by

$$k_{v_i} = 2\sqrt{k_{p_i}} \tag{6.40}$$

Using this rule, the gains of the feedback observer were set to $k_{p_{1,2}} = 25$, $k_{v_{1,2}} = 10$. Figure 6.6 shows how the trajectory error is reduced to zero in a critically damped way. Once on trajectory, the manipulator tracks the desired position and velocity trajectories accurately.

For the ideal system represented in the above case, the feedback loop is barely required once the manipulator is on trajectory as the feedforward part of the controller is capable of keeping the manipulator on track. For a real system modelling errors, outside disturbances and measurement noise will cause tracking errors. Figures 6.8 and 6.9 show how the presence of measurement noise on the second link velocity measurement can degrade the performance of the manipulator.

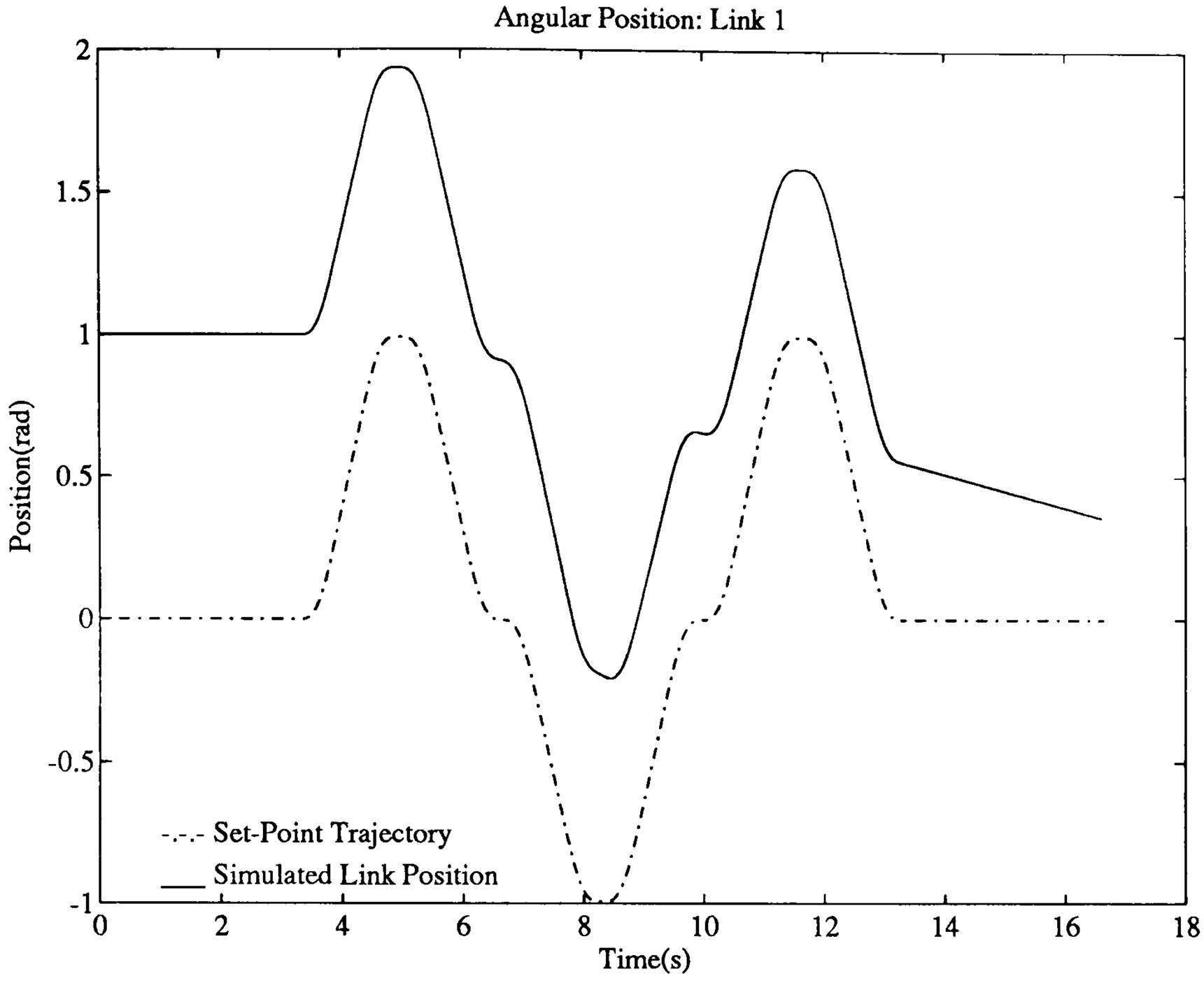


Figure 6.5: Simulation of Computed Torque Controller with no Feed-Back.

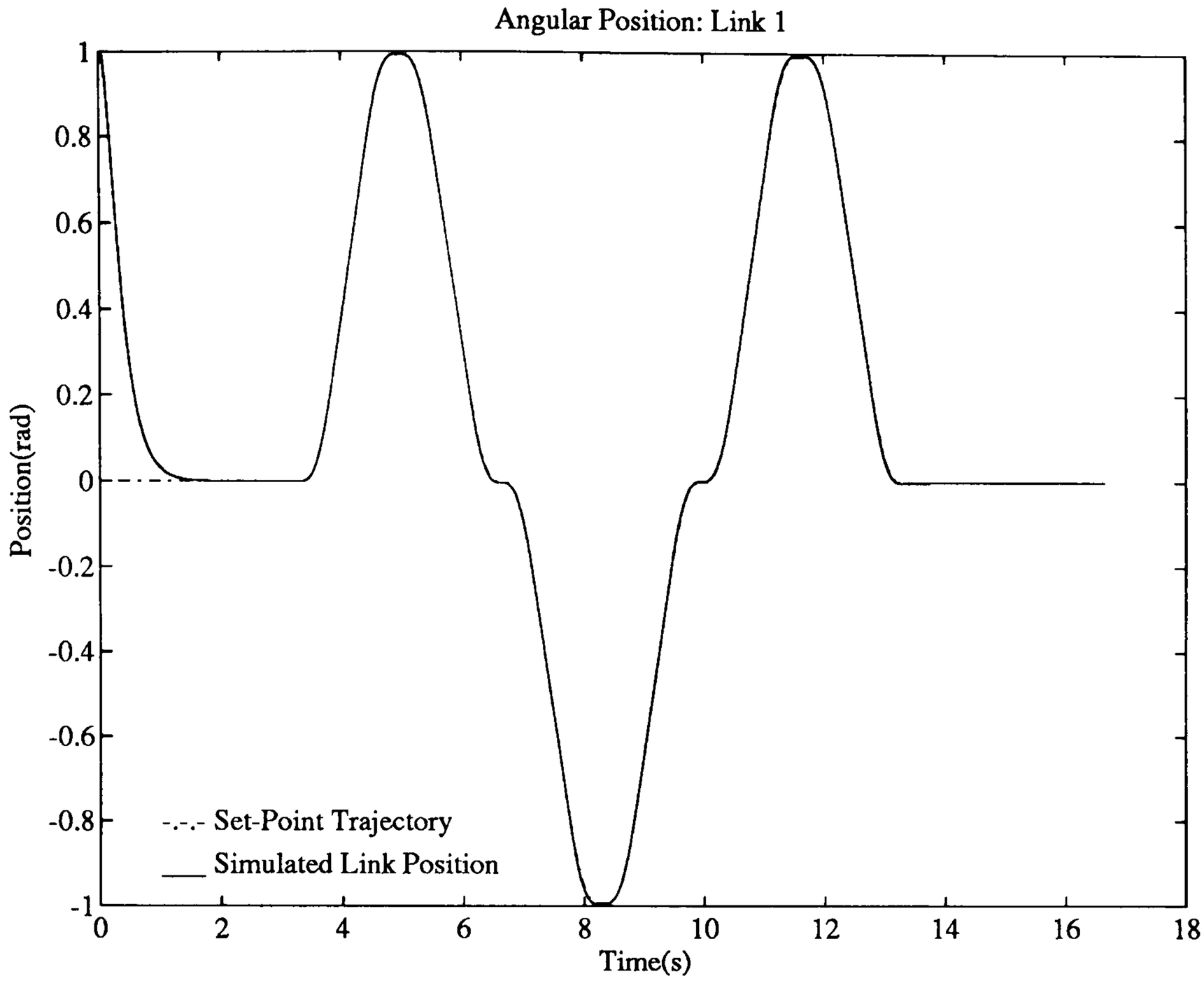


Figure 6.6: Simulation of Computed Torque Controller.

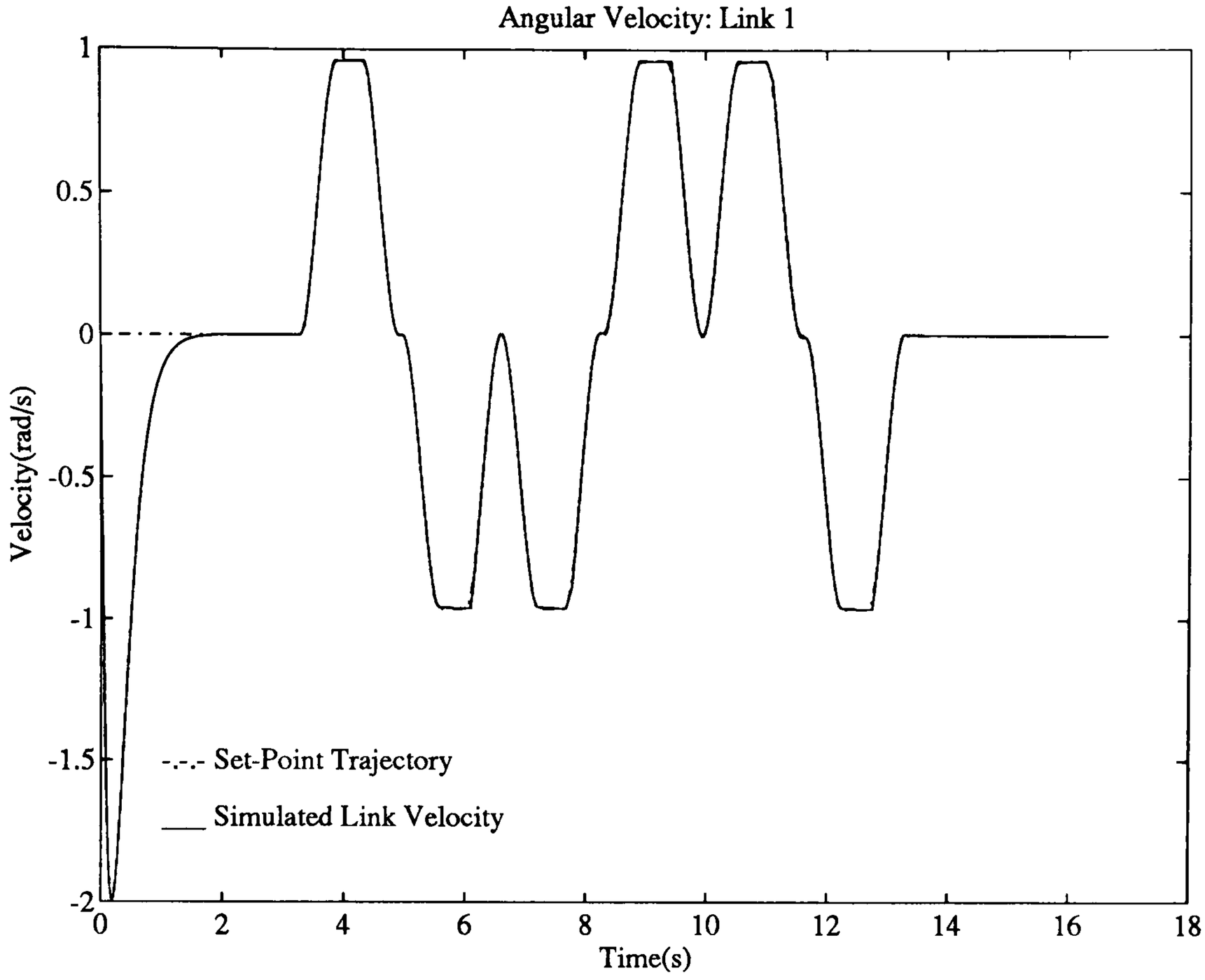


Figure 6.7: Simulation of Computed Torque Controller.

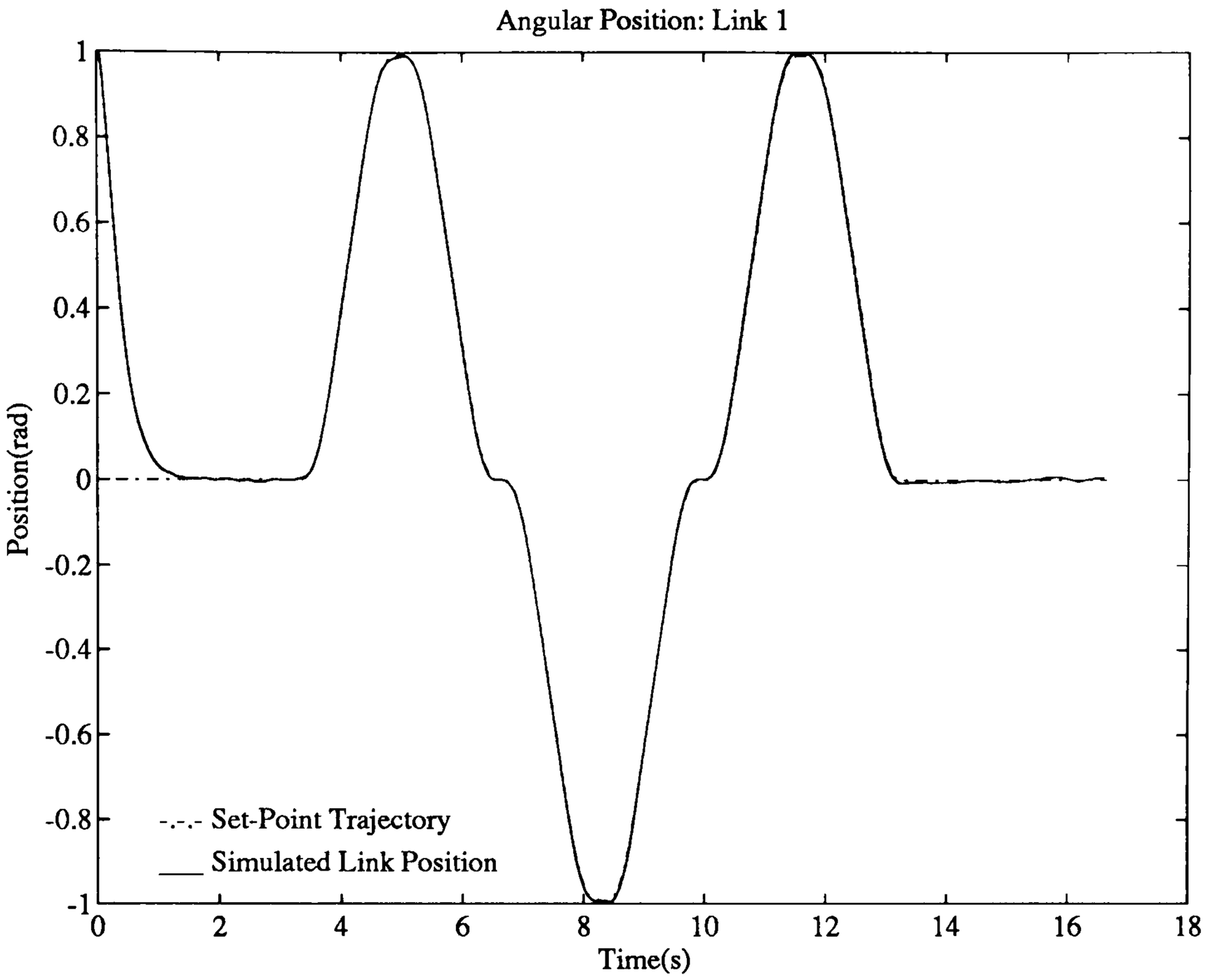


Figure 6.8: Simulation of Computed Torque Controller with Noise on the Second Velocity Measurement.

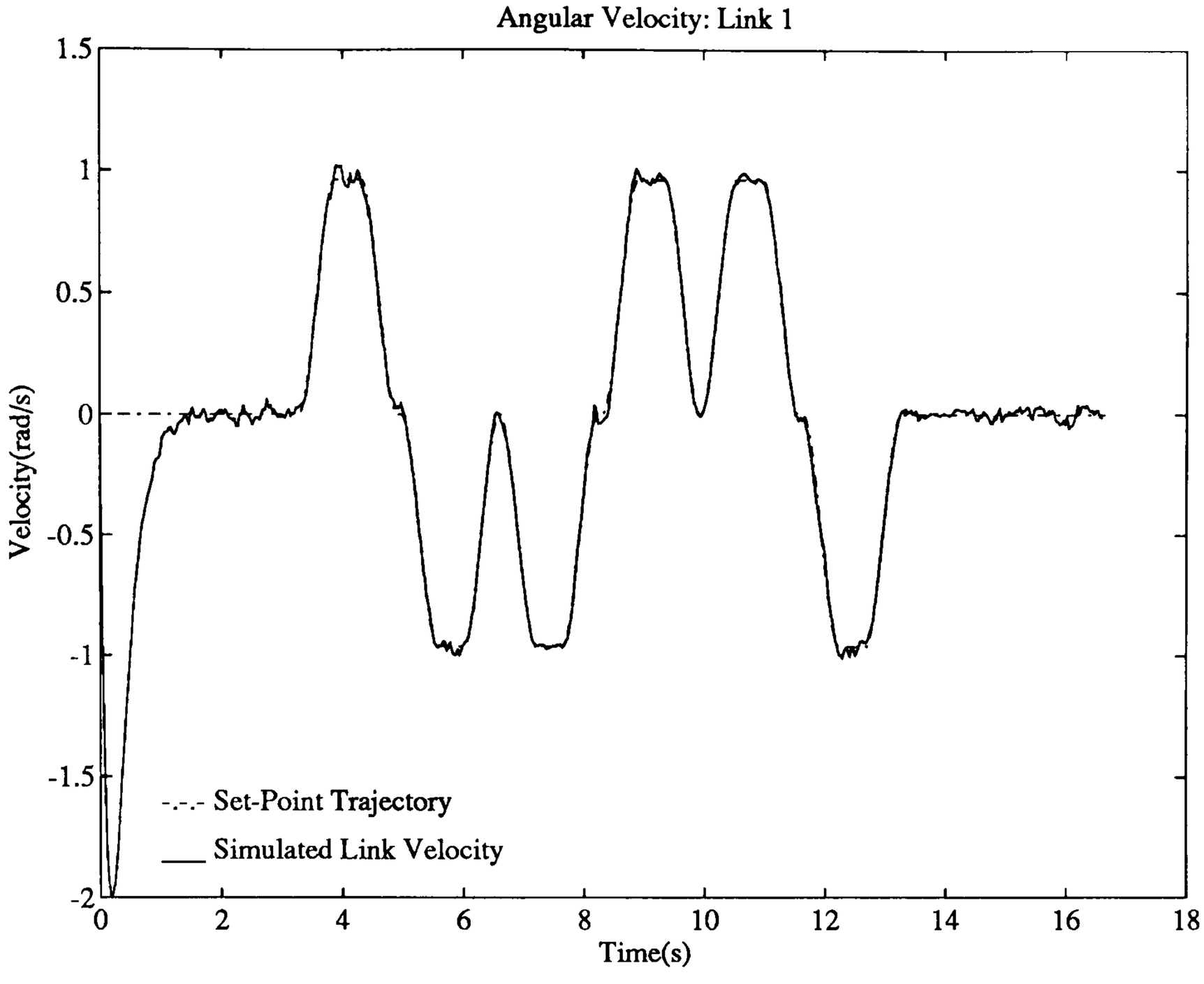


Figure 6.9: Simulation of Computed Torque Controller with Noise on the Second Velocity Measurement.

6.4.3 Experimental Results.

The implementation of the computed torque control scheme to control the experimental manipulator involved the simple translation of the SIMULAB format inverse model into CONIC. This model was then run in real time on a Sun3 computer communicating with the target motorolas via ethernet as shown in figure 6.10.

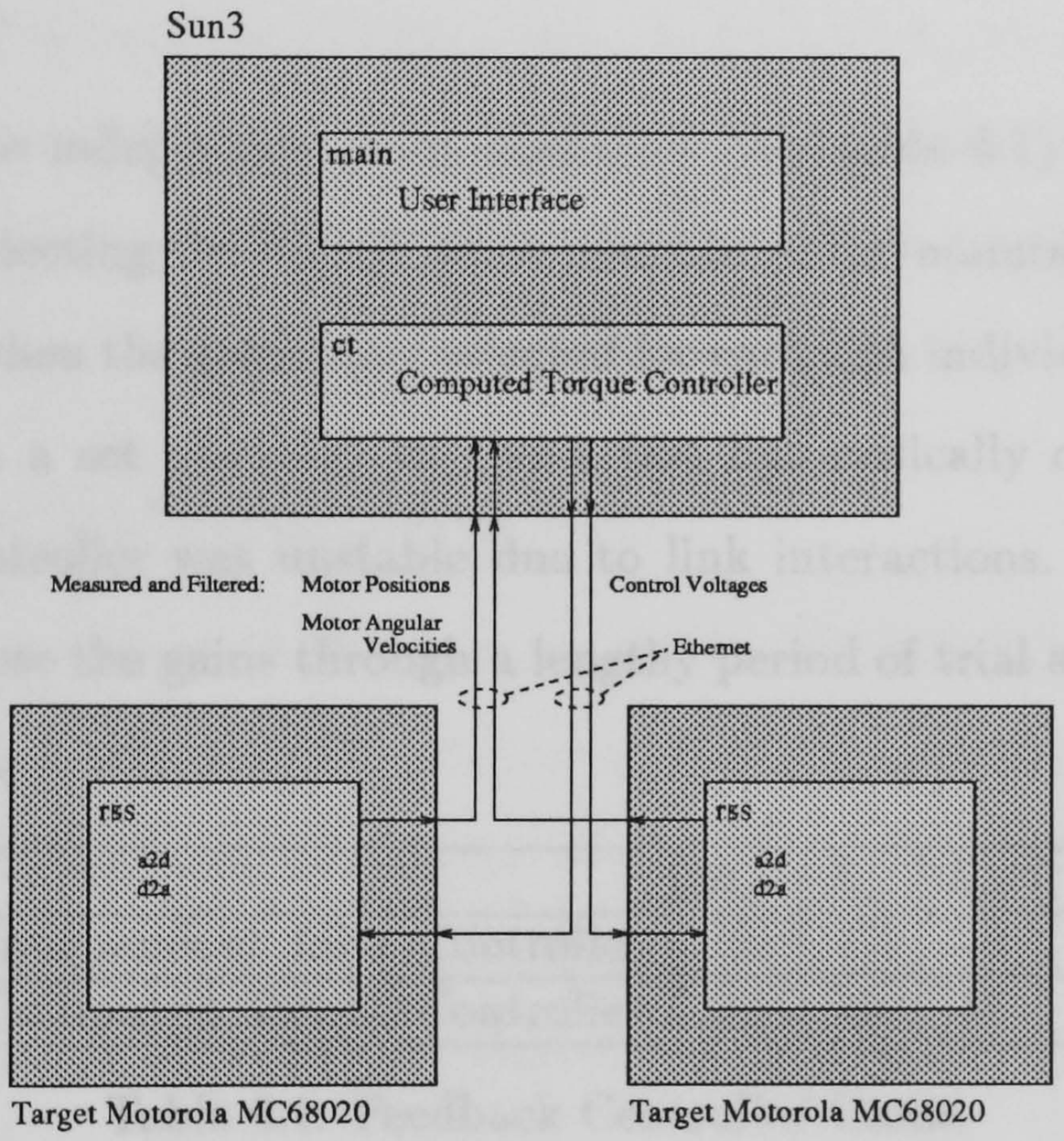


Figure 6.10: Software Configuration for Computed Torque.

As the control voltages are calculated by the computed torque software running on the Sun3, the task of the target Motorolas is reduced to data capture, conditioning and application of the control voltages. The sample rate is still constrained by the speed of ethernet communication to approximately 28Hz. As this is near the lower end of acceptable sample rates, it was not possible to run the model-based observer alongside the computed torque controller and hence the poorly conditioned derivative of the second motor position had to be used for the second link velocity feedback loop.

The results from the following controllers are compared here:

- independent joint controller.
- independent joint controller with velocity feed-forward.
- computed torque.

Gain Selection.

The gains for the independent joint controller (see table 6.1) were determined empirically by selecting the largest gains possible whilst maintaining stability. It was found that when the gains were selected for each link individually (by locking the other link in a set position) to give a fast but critically damped response, the resulting controller was unstable due to link interactions. It was therefore necessary to choose the gains through a lengthy period of trial and error with the complete system.

	k_{p_1}	k_{v_1}	k_{p_2}	k_{v_2}
Independent Joint Controller	35	17	20	4
Computed Torque Controller	25	10	40	6

Table 6.1: Feedback Controller Gains.

The selection of gains for the computed torque controller was much simpler. The gains for the first motor were chosen using equation 6.40 with a reasonably large proportional gain to provide good disturbance rejection. The relatively high quality of the first tachometer output allowed the velocity feedback gain to be selected using equation 6.40. This was not the case for the second velocity gain, however, which had to be reduced below the desired level for critical damping. This shortfall was partially offset by increased damping due to unmodelled stiction, however.

Results: Independent Joint Controllers.

The independent joint controller (see figures 6.11 and 6.12) shows poor trajectory tracking. The first link position lags the set-point trajectory as trajectory errors must be large to generate the voltages required to move the link. There is a large amount of link interaction with the second link being particularly affected by the motion of the first. Velocity trajectory tracking is also poor.

Incorporating velocity feed-forward into the independent joint controller improves trajectory tracking, especially for the first link (figure 6.13). Link interaction is still high, however, and undermines the tracking performance of the second link.

Results: Computed Torque Controller.

It can be seen from figures 6.15 and 6.16 that the implementation of computed torque considerably improves the performance of the manipulator. Both links track the desired trajectory accurately with little evidence of link interaction. The performance of the second link is degraded by the large amount of noise contamination on the derived link velocity. The response is slightly oscillatory due to the low velocity feedback gain.

Figure 6.17 and 6.18 show how the computed torque scheme copes with a far more demanding trajectory. Whilst the tracking performance is partially degraded for both links, with evidence of link interaction, the manipulator is still capable of tracking the desired trajectory whilst the performance of the independent joint controller with velocity feed-forward completely breaks down (figure 6.19).

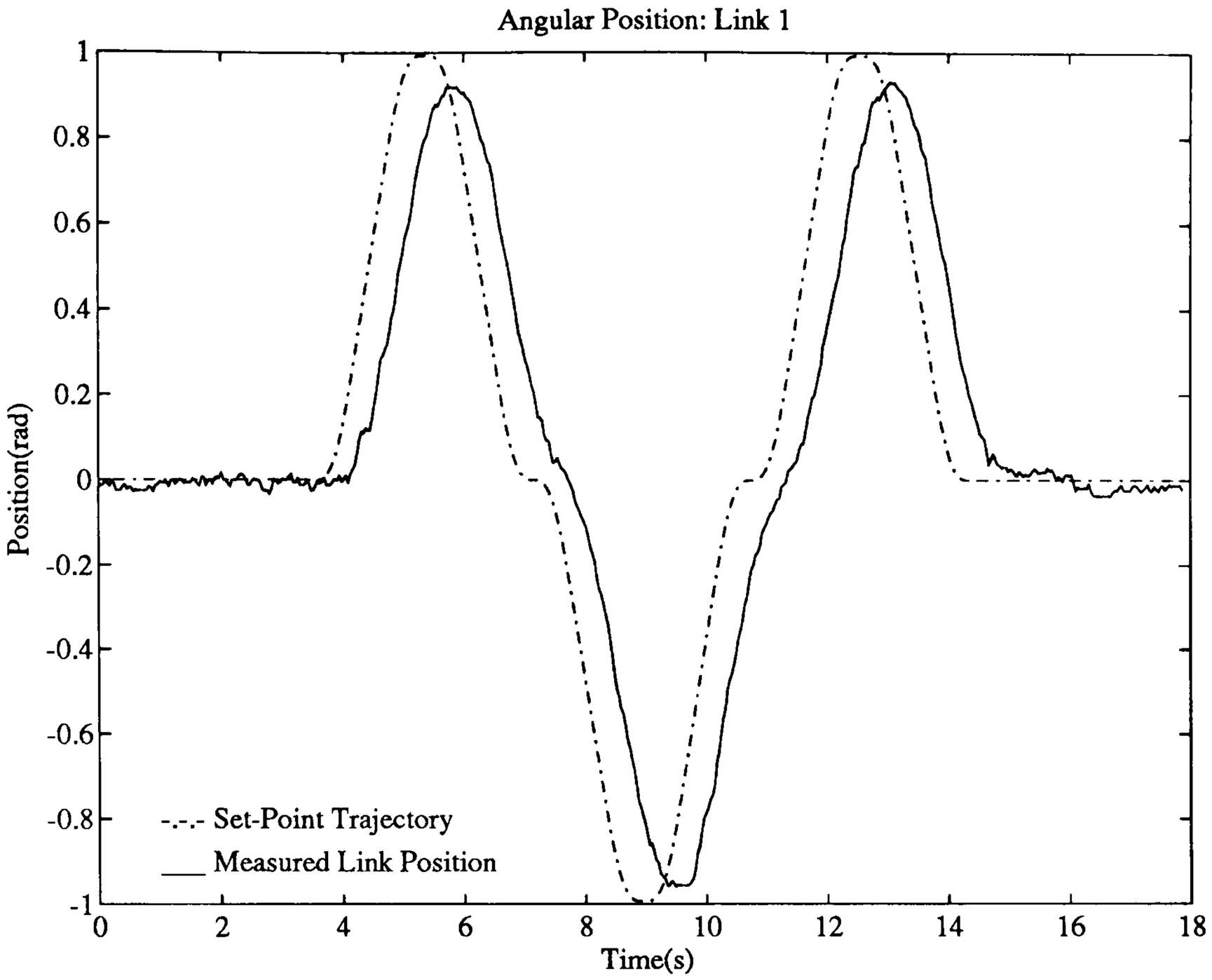


Figure 6.11: Link Positions using Independent Joint Controller.

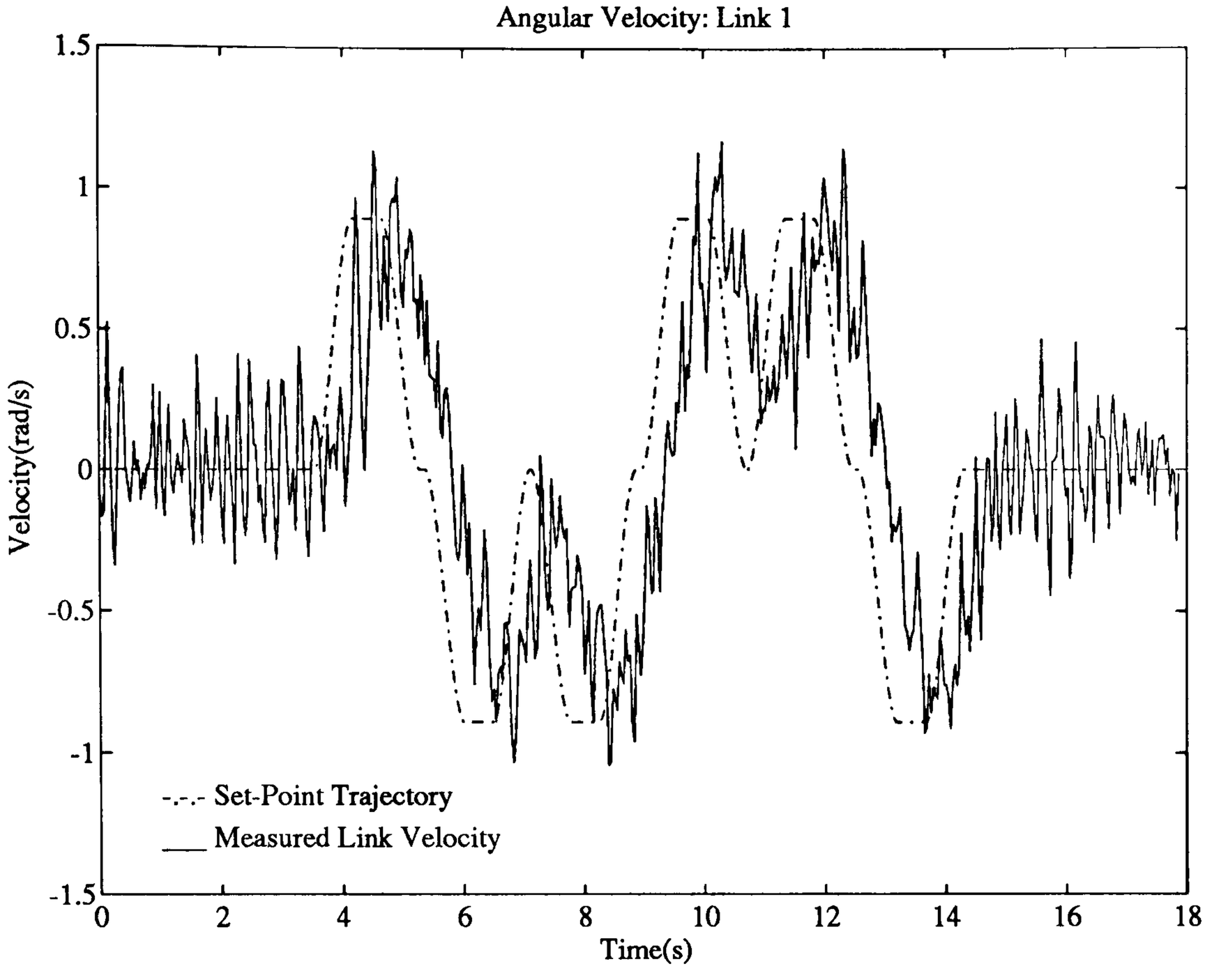


Figure 6.12: Link Velocities using Independent Joint Controller.

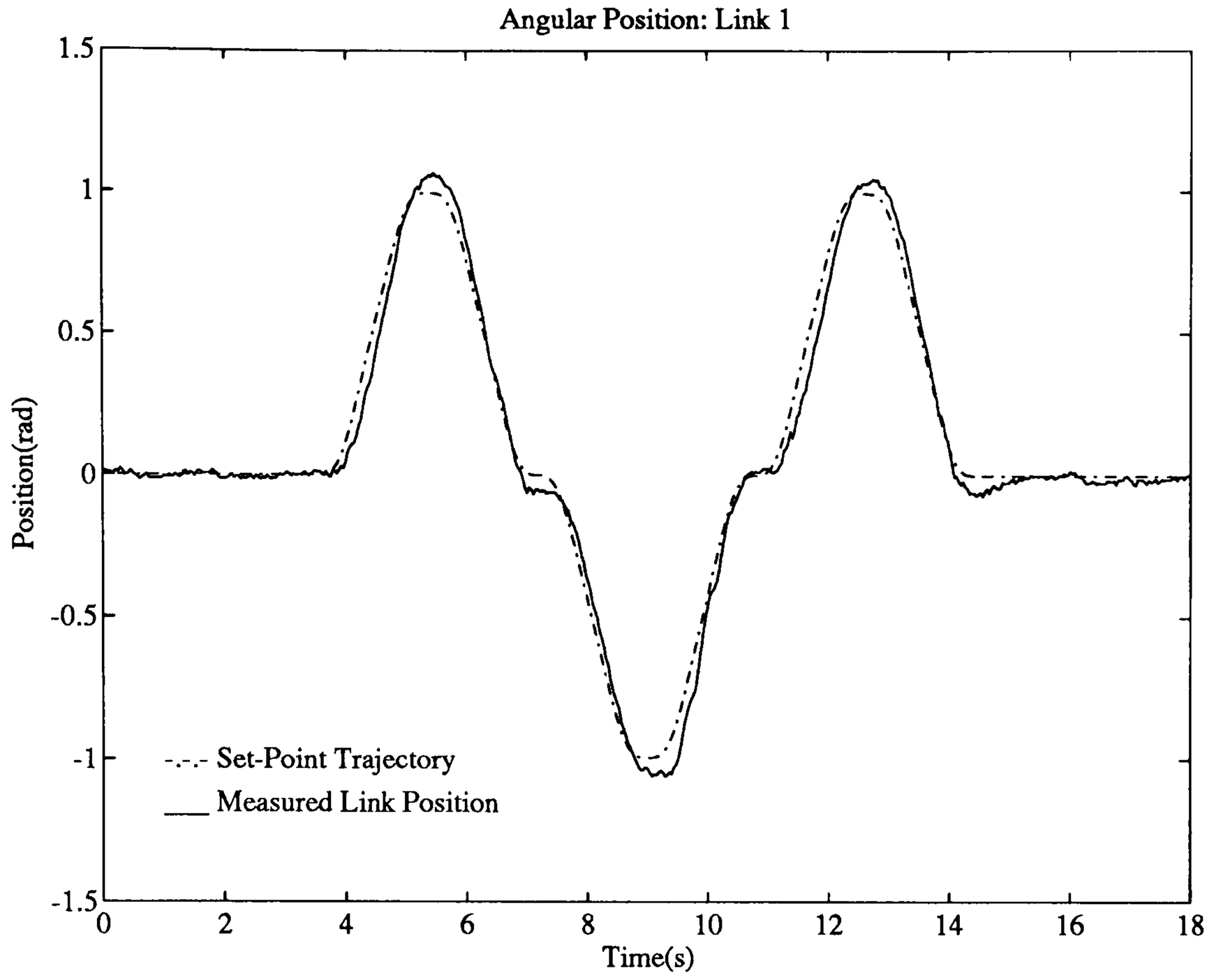


Figure 6.13: Link Positions using Independent Joint Controller with Velocity Feed-Forward.

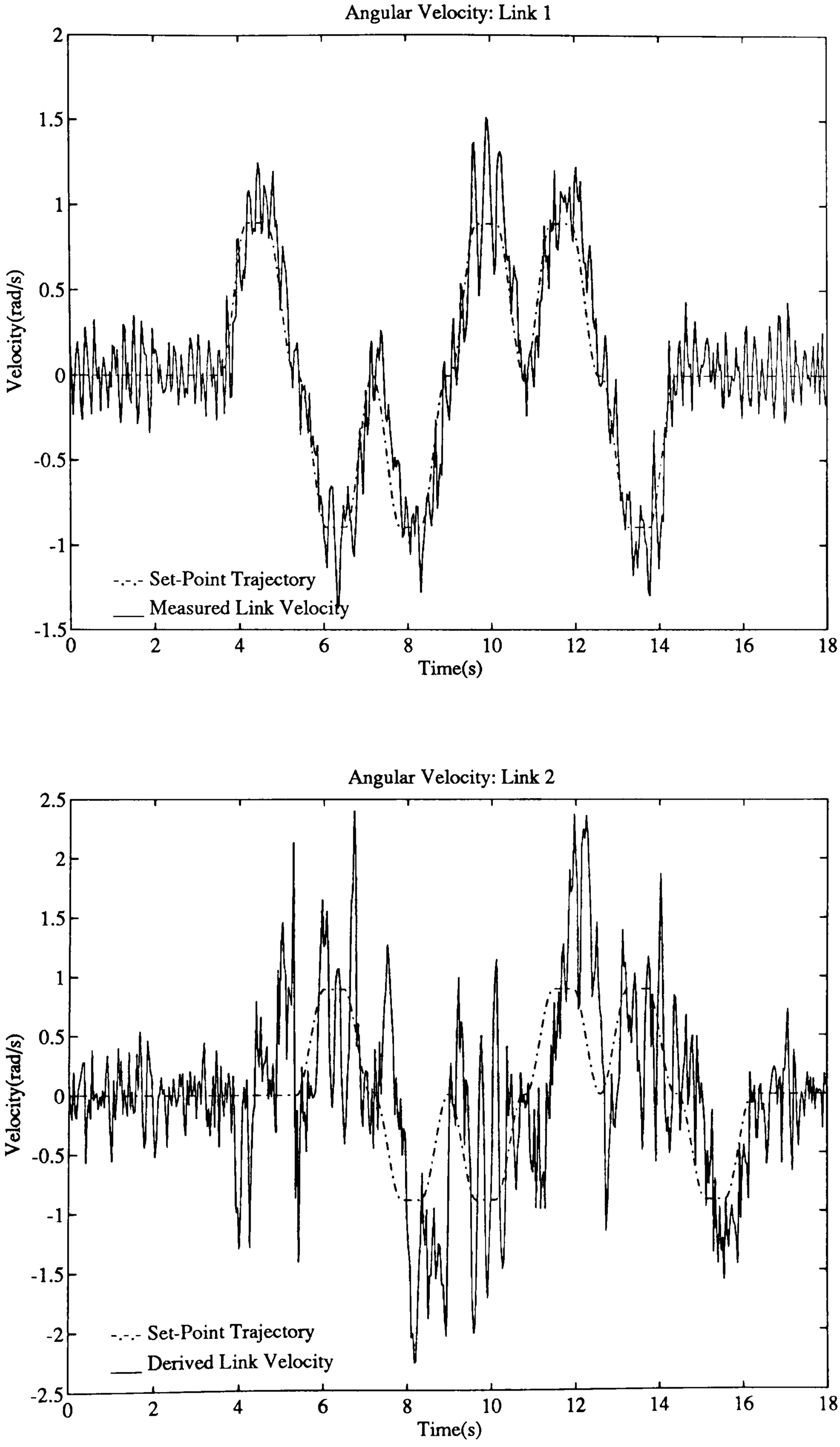


Figure 6.14: Link Velocities using Independent Joint Controller with Velocity Feed-Forward.

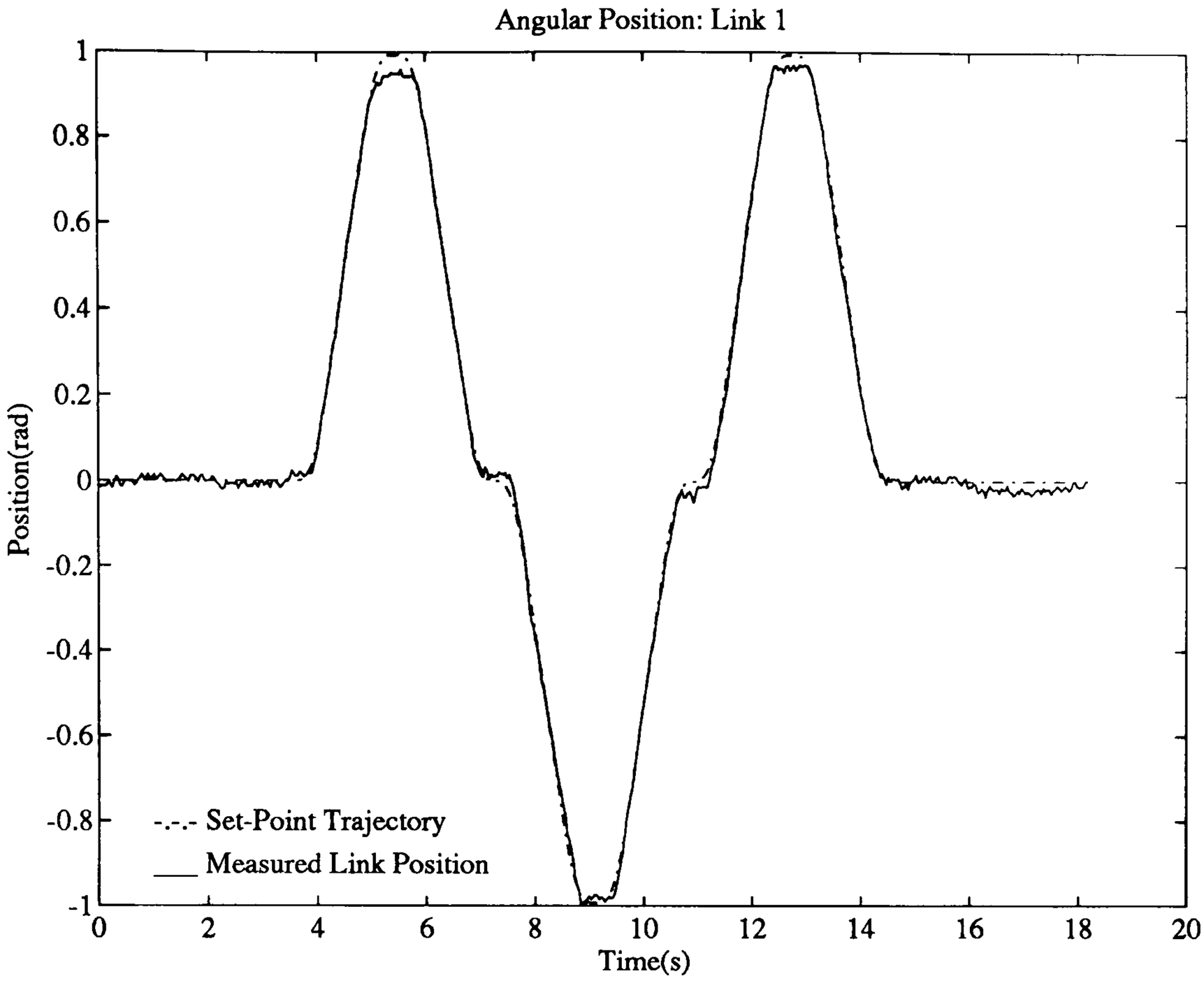


Figure 6.15: Link Positions using Computed Torque Controller.

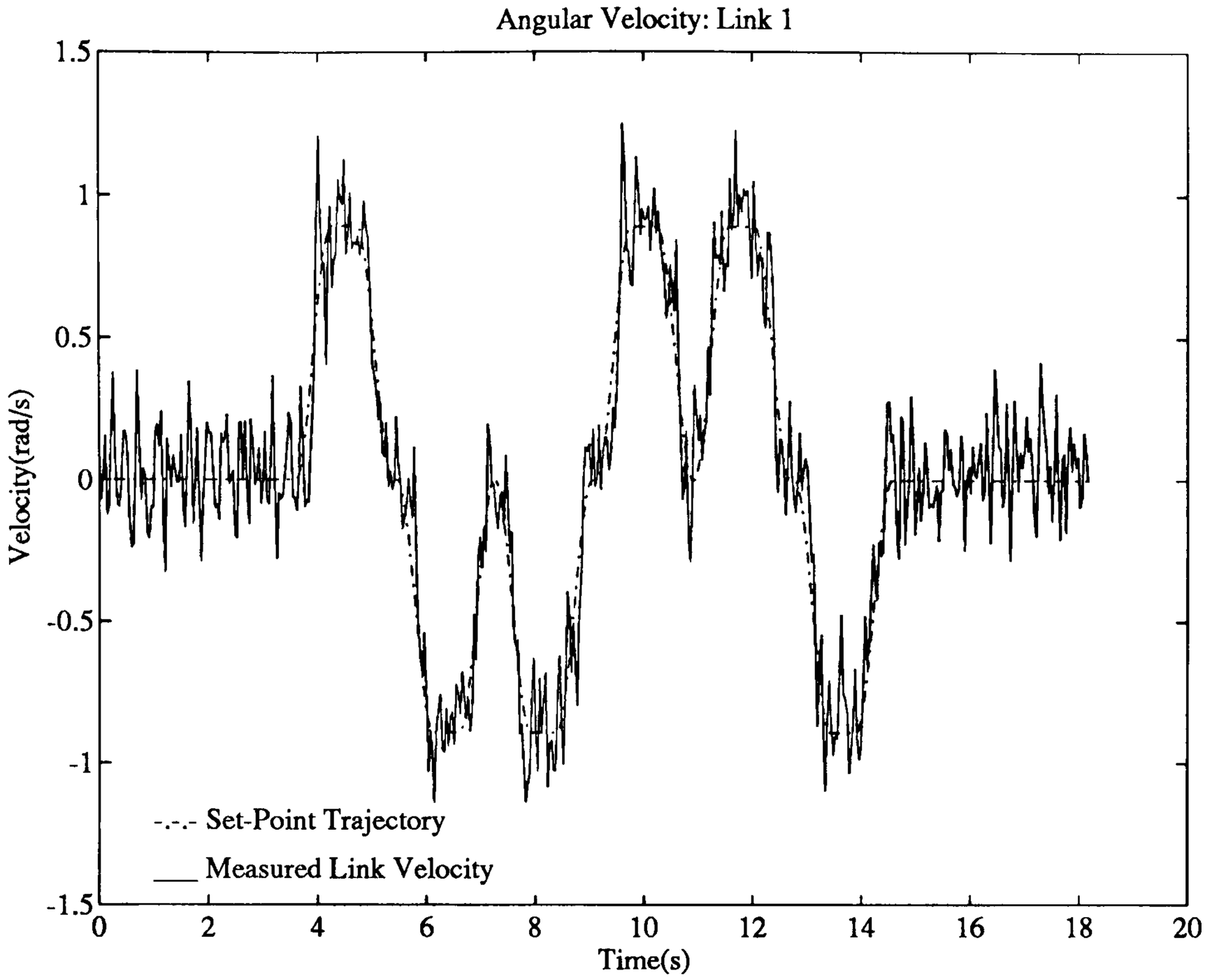


Figure 6.16: Link Velocities using Computed Torque Controller.

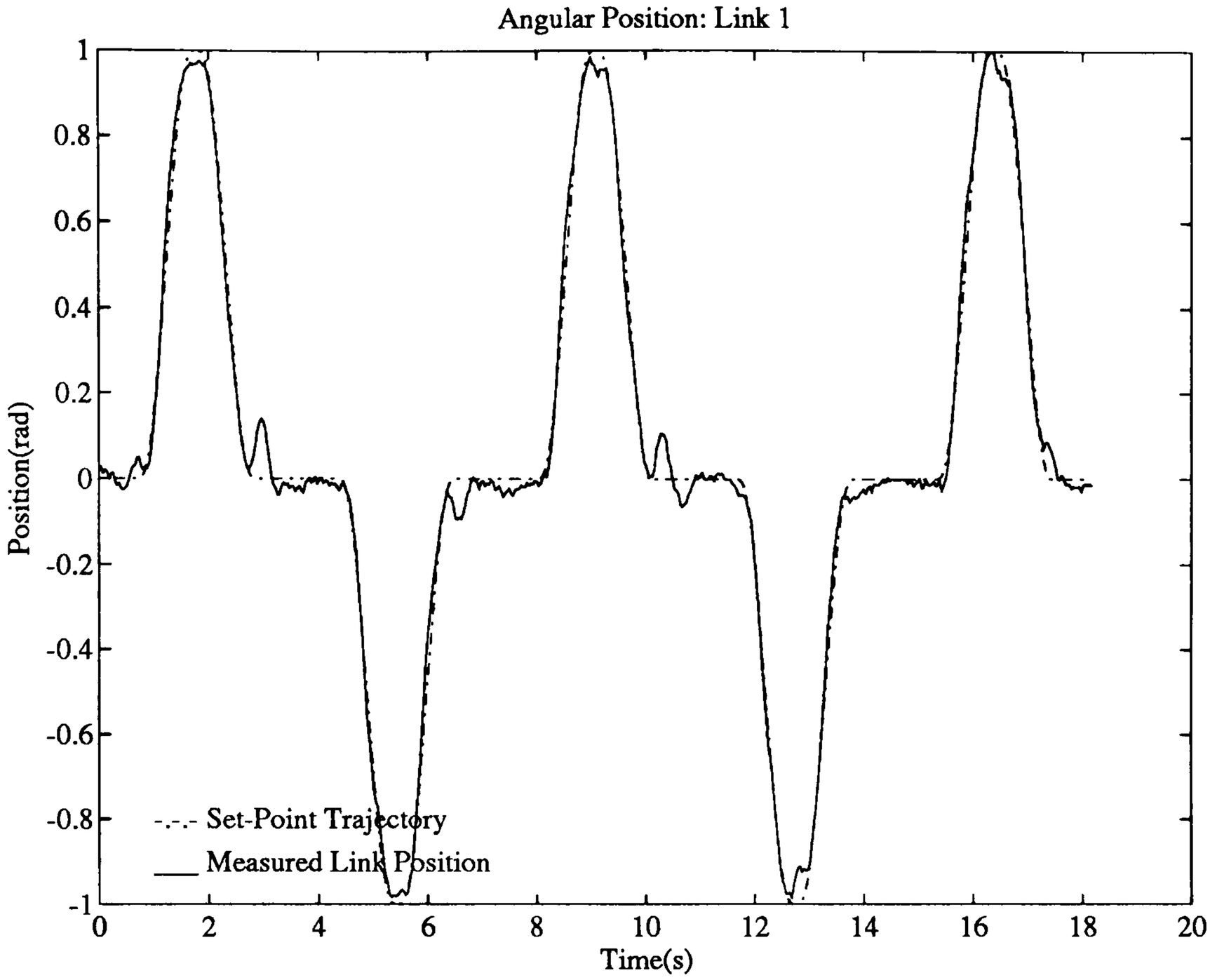


Figure 6.17: Link Positions using Computed Torque Controller to follow a more demanding Set-Point Profile.

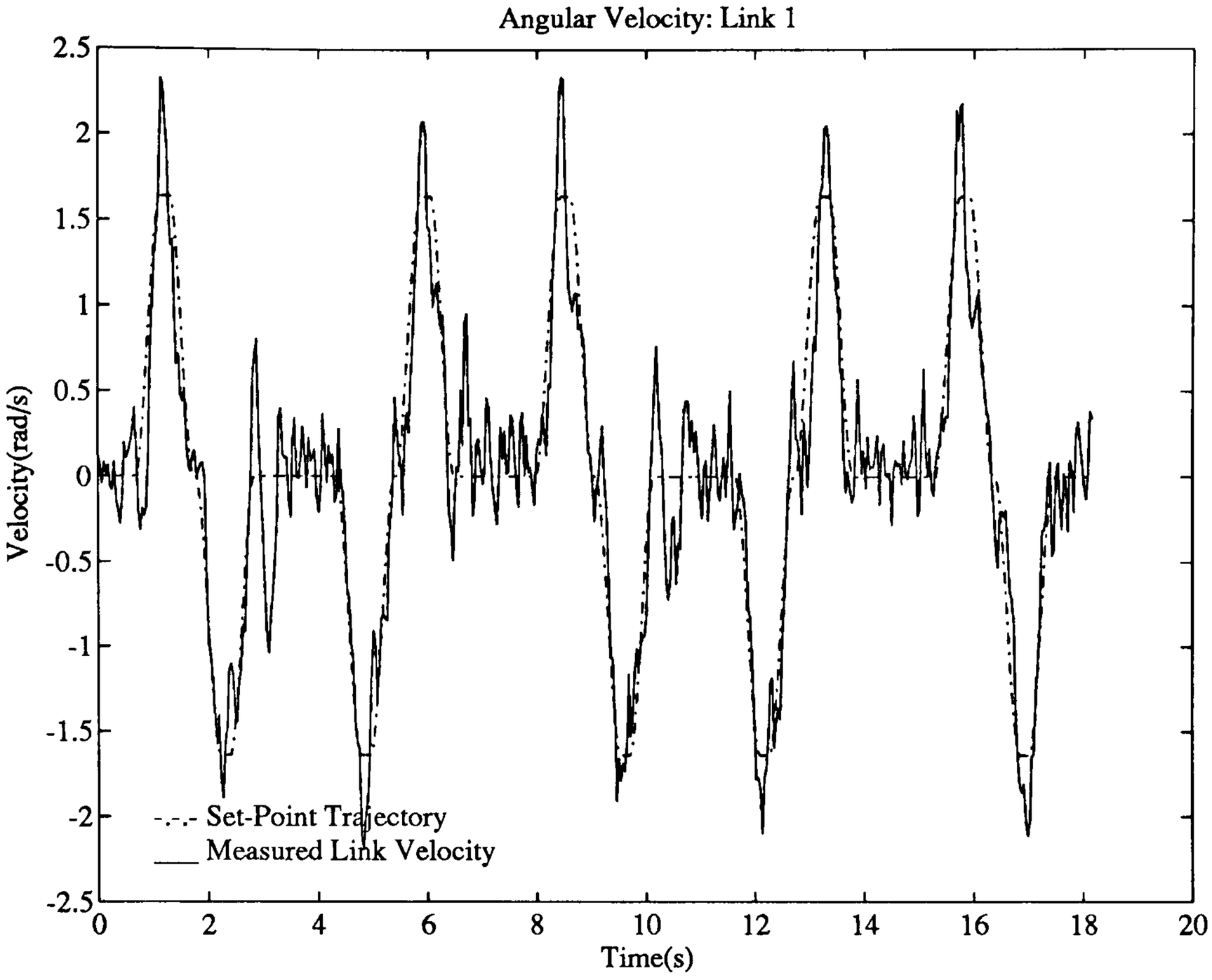


Figure 6.18: Link Velocities using Computed Torque Controller to follow a more demanding Set-Point Profile.

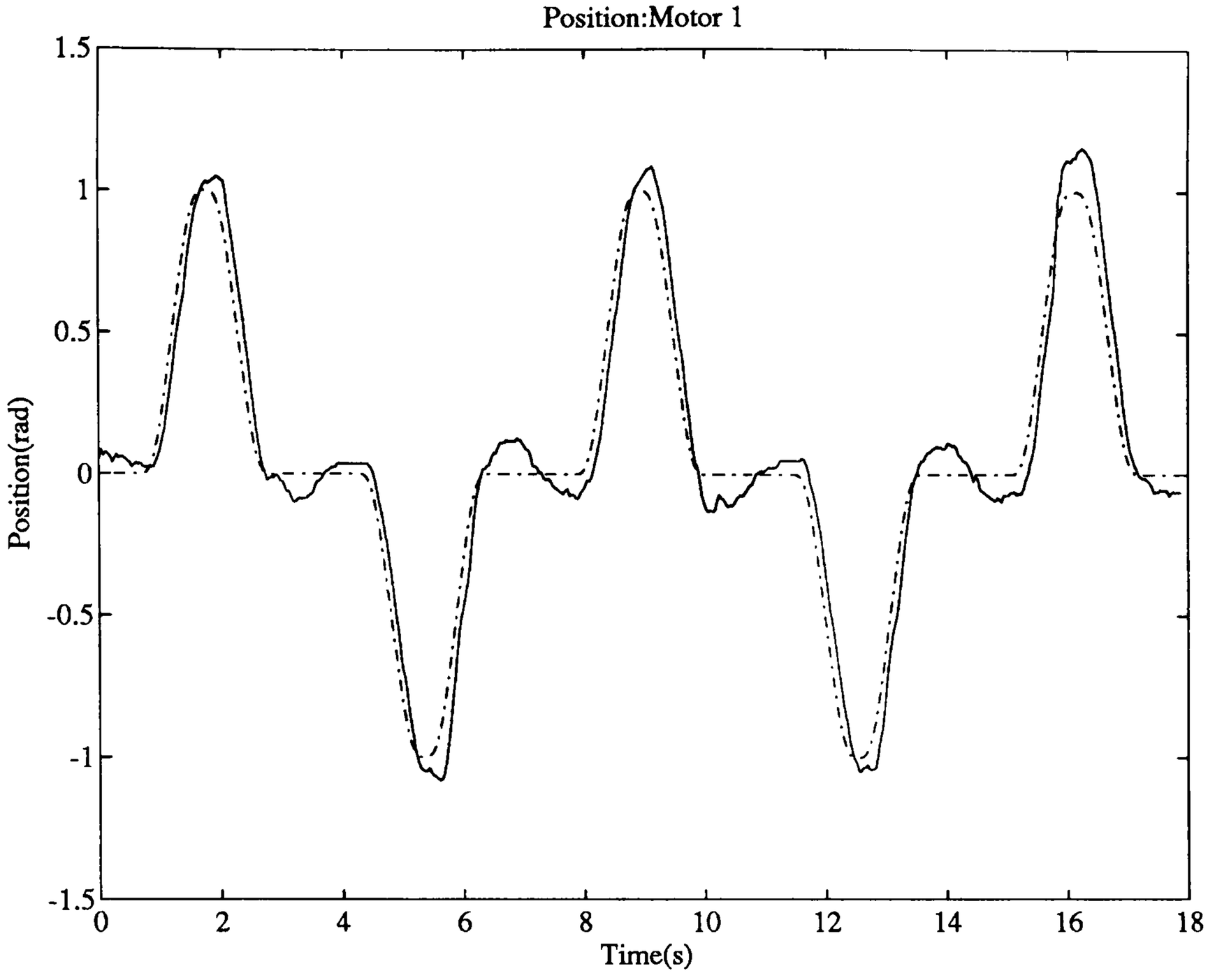


Figure 6.19: Link Positions using Independent Joint Controller with Velocity Feed-Forward to follow a more demanding Set-Point Profile.

6.5 Conclusion.

It has been shown that it is possible to create an inverse model for a robotic manipulator through relatively simple manipulation of the inputs and outputs of its bond graph representation. For the experimental two-link manipulator, this inverse bond graph was used to create the equations required to implement a computed torque controller. On implementation, this controller was seen to improve the control of the manipulator considerably over standard independent joint controllers. Furthermore, use of the computed torque control scheme simplifies the selection of the gains for the feedback controller as the trajectory error equations for each link are decoupled.

It is important to note that whilst this research has been implemented using a simple, rigid, two-link manipulator, the techniques are generic and may be used to create inverse model based controllers for complex six-degree of freedom manipulators once bond graphs for these robots have been created.

Chapter 7

Conclusion.

It has been shown how the creation of sophisticated model-based control schemes for robotic manipulators might be automated starting from a bond graph representation. To demonstrate that these methods work in practice, as well as in simulation, the bond graph derived controllers have been implemented on an experimental, rigid, planar, direct drive two-link manipulator. Whilst the manipulator is capable of exhibiting the highly non-linear nature of the robot dynamics, it is sufficiently simple and accessible for small scale laboratory experimentation.

The modelling of robots using bond graphs offers some useful advantages over algebraic based derivations

- Graphical format.
- Ability to create bond graphs of large, complex systems by connecting the bond graphs of simple sub-systems.
- Graphs are constructed through consideration of the kinematics of the robot.
- Dynamic equations of motion may be derived automatically from the causally complete bond graph.

The main disadvantage of using bond graphs is that, for most people, a new and non-intuitive modelling technique must be learnt.

With a validated bond graph of the experimental two-link manipulator, two methods of improving its control were investigated:

- use of a model-based observer,
- use of a bond graph derived 'inverse system' controller.

7.1 Model Based Observer.

A common problem in the control of robotic manipulators lies in the implementation of link velocity feedback loops to improve the damping of the system. Tachometer measurements of link angular velocity are often contaminated by measurement noise due to discontinuities in the magnetic field of the tachometer stator at low velocities together with ripple torques and other high frequency phenomena [46]. This severely limits the reachable closed-loop bandwidth and constrains the use of high gain controllers. Differentiation of position measurements offers only a partial solution as this suffers from noise amplification.

One solution to this problem is to use a model based observer; a mathematical model of the robot which is run in parallel with the robot and with the same inputs. If constructed correctly, the outputs of the model, including link velocities, will track the robot outputs accurately but will not be contaminated by measurement noise. The observer outputs may then be input to the feedback controller replacing noise contaminated measurements.

It has been demonstrated how a bond graph representation of a full-order non-linear model based observer can be constructed by making only slight modifications to the basic bond graph of the two-link manipulator. Thus observer dynamic equations of motion and software can be created automatically. By constructing a linear feedback loop around the observer, designed with the use of

a linearised model, the states of the observer could be made to track the states of the experimental manipulator accurately thus causing model outputs to track system outputs.

The observed velocities were indeed less contaminated by noise than the measured velocities, allowing the feedback gains to be increased significantly without instability. Consequently, the bandwidth of the controller was increased enabling the manipulator to be run at much higher speeds and with improved trajectory tracking.

7.2 Inverse Model Based Control.

The second method of improving manipulator control is to use an 'inverse system' type controller in which the input torques (or, more accurately, input voltages) required to drive the manipulator to follow a desired trajectory are calculated by the mathematical model of the manipulator.

The basic bond graph model of the manipulator was modified to give motor torques as outputs with the desired link angular velocities as inputs. From this bond graph, the inverse system equations could be derived automatically as the set of output equations of the inverse bond graph.

The inverse model equations were implemented for the experimental manipulator in the form of a 'computed torque' controller. This form of controller is known as a non-linearity cancellation controller as it computes the torques required to counteract the non-linearities of the system such as link interactions, centrifugal and Coriolis forces. The feedback portion of the controller is left to deal only with trajectory errors which are represented by a set of simple linear decoupled equations. Thus the task of controlling a highly non-linear multi-input multi-output coupled system is reduced to that of controlling a set of decoupled linear systems.

On implementation, the computed torque controller was found to considerably improve the control of the manipulator over standard independent joint controllers. Furthermore, as the feedback control deals with linear decoupled subsystems the task of choosing suitable feedback gains was simplified. The effects of one links' trajectory correction torques on all the other links are compensated for automatically by the controller.

7.3 Summary.

This research has shown that it is possible to start with a bond graph representation of a robotic manipulator and automatically create the software required to implement sophisticated model based controllers. This capability would be highly advantageous in an industrial environment as it avoids the need to derive and manipulate large and complex equations of motion.

Whilst the experimental robot used as the test bed for this research was a simple one, the techniques required to derive the model based controller are generic and may therefore be used to derive controllers for multi-link three degree of freedom robots once bond graphs for these robots have been created.

7.4 Future Work.

The use of bond graphs to help in the creation of model based controllers for robotic manipulators is in its very early stages of development. Whilst the initial results are promising, there are two clear avenues for further research:

- development of bond graph derived controllers for three dimensional multi-degree of freedom industrial robots;
- development of bond graph derived controllers for robots with joint and link flexibility.

7.4.1 Multi-Degree of Freedom Industrial Robots.

The development of a bond graph derived controller for robots capable of operating in three dimensions is still in its infancy. In principle, however, it would not be difficult to extend the results of the present research to these robots as the extra complexity of the dynamic equations of motion would be handled automatically by computers.

Implementation of the bond graph derived controllers for three dimensional robots would require far more capable processors than those used for the control of the two-link manipulator as the equations of motion would be far more complex. With the rapid advancement of distributed computing, and in the capability of processors such as transputers, this should not prove to be a great obstacle to development in this area.

Most industrial robots are, however, driven with the use of gearboxes. It is possible to model gearbox driven manipulators, and this has in fact been done for an existing industrial robot, but it is dubious as to whether sophisticated model based controllers, such as computed torque, offer significant control improvements since the presence of gearboxes reduces the ability to control joint torques accurately.

7.4.2 Flexible Robots.

The development of lighter, faster robots and the demands for higher accuracy have brought the need to control the effects of link and joint flexibility in manipulators. Flexibility adds to the number of degrees of freedom of the robot and necessitates the use of additional sensors such as strain gauges and extra position encoders to measure the state of flexure of the system. This adds to the expense of the robot and decreases its reliability.

The modelling of flexible robots using bond graphs could offer solutions to these problems as they may be used to derive model based observers to observe

rather than measure the state of flexure. It may also be possible to derive inverse model type controllers for flexible robots as well.

Appendix A

Motor and Amplifier Specifications.

A.1 Motor Data.

	S19-1B/T	S372-1A/T
Continuous Stall Torque (Nm)	0.67	0.078
Peak Torque (Nm)	3.35	0.39
Max Stall Current (Amps)	2.9	1.66
Max No Load Speed (rpm)	2500	5000
Max Terminal Voltage (V)	60	30
MECHANICAL DATA		
Rotor Moment of Inertia (kg m^2)	$1.34*10^{-4}$	$3.3 * 10^{-6}$
Mechanical Time Constant (ms)	7.4	5.2
Damping Constant (Nm/K. rpm)	$10.6 * 10^{-3}$	$1.4 * 10^{-3}$
Static Friction Torque (Nm)	$5.7 * 10^{-2}$	$1.4 * 10^{-2}$
Motor Weight (Kg)	2.72	0.44

	S19-1B/T	S372-1A/T
ELECTRICAL DATA		
Torque Constant(k_2)(Nm/A)	$2.3 * 10^{-1}$	$4.4 * 10^{-2}$
Voltage Constant(k_3)(V/K. rpm)	24.0	4.95
Armature Resistance @25C(Ω)	3.8	3.5
Electrical Time Constant (ms)	1.2	4.5
Max Pulse Current (A)	14	7.5
Armature Inductance (L_a)(mH)	4.0	1.3
TACHOMETER DATA		
Voltage Constant(V/K. rpm)	13.2	3.0
Armature Resistance (Ω)	110	48
Ripple (%pk-pk) at 500rpm	2.0	5.0
Linearity(%)	0.2	0.2
Temperature Coefficient(%/C)	0.01	0.01

A.2 Amplifier Data.

	EM100B
Ouput Voltage (Vdc)	± 24
Continuous Current (A)	2
Peak Current (A)	10
CONTROL SUPPLIES	
$\pm 12 Vdc$ rating (mA)	30
Stabilisation (V/deg C)	0.006

Appendix B

Observer Equations of Motion.

B.1 State Vector.

$$x = \begin{pmatrix} h_1 \\ h_2 \\ \theta_1 \\ \theta_2 \end{pmatrix} ; \; z = \begin{pmatrix} m_1\dot{x}_1 \\ m_1\dot{y}_1 \\ m_2\dot{x}_2 \\ m_2\dot{y}_2 \\ m_m\dot{x}t_1 \\ m_m\dot{y}t_1 \end{pmatrix} ; \; y = \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \omega_a 1 \\ \omega_a 2 \\ \theta_1 \\ \theta_2 \end{pmatrix} ; \; u = \begin{pmatrix} v_1 \\ v_2 \\ \omega_a 1 \\ \omega_a 2 \\ \theta_1 \\ \theta_2 \end{pmatrix} \tag{B.1}$$

B.2 Differential Algebraic Equations.

$$\dot{x}_1 = \frac{1}{(2l_1^2 l_2^2 m_1 m_2 ra_1 ra_2)} \begin{pmatrix} -(24(((b_1+b_2)ra_2 + k2_2^2)ra_1 + ra_2 k2_1^2)l_2^2 m_2 x_1 - \ldots \\ \ldots 2((ra_2 u_3 - k2_2 u_2)ra_1 + ra_2 k2_1 u_1)l_1^2 l_2^2 m_1 m_2 - \ldots \\ \ldots 24(ra_2 b_2 + k2_2^2)l_1^2 m_1 ra_1 x_2 + \ldots \\ \ldots (\dot{z}_1 + 2\dot{z}_3 + 2\dot{z}_5)sin(x_3)l_1^3 l_2^2 m_1 m_2 ra_1 ra_2 + \ldots \\ \ldots (\dot{z}_2 + 2\dot{z}_4 + 2\dot{z}_6)cos(x_3)l_1^3 l_2^2 m_1 m_2 ra_1 ra_2) \end{pmatrix} \tag{B.2}$$

$$\dot{x}_2 = \frac{1}{(2l_1^2 l_2^2 m_1 m_2 r a_2)} \begin{pmatrix} -(24(l_1^2 m_1 x_2 - l_2^2 m_2 x_1)(r a_2 b_2 + k2_2^2) - \dots \\ \dots 2(r a_2 u_4 + k2_2 u_2) l_1^2 l_2^2 m_1 m_2 + \dots \\ \dots sin(x_3 + x_4) l_1^2 l_2^3 m_1 m_2 r a_2 \dot{z}_3 + \dots \\ \dots cos(x_3 + x_4) l_1^2 l_2^3 m_1 m_2 r a_2 \dot{z}_4) \end{pmatrix} \quad \text{(B.3)}$$

$$\dot{x}_3 = \frac{(l_1^2 m_1 u_5 + 12 x_1)}{(l_1^2 m_1)} \quad \text{(B.4)}$$

$$\dot{x}_4 = \frac{((l_2^2 m_2 u_6 + 12 x_2) l_1^2 m_1 - 12 l_2^2 m_2 x_1)}{(l_1^2 l_2^2 m_1 m_2)} \quad \text{(B.5)}$$

$$z_1 = \frac{(6 sin(x_3) x_1)}{l_1} \quad \text{(B.6)}$$

$$z_2 = \frac{(6 cos(x_3) x_1)}{l_1} \quad \text{(B.7)}$$

$$z_3 = \frac{(6(sin(x_3 + x_4) l_1 m_1 x_2 + 2 sin(x_3) l_2 m_2 x_1))}{(l_1 l_2 m_1)} \quad \text{(B.8)}$$

$$z_4 = \frac{(6(cos(x_3 + x_4) l_1 m_1 x_2 + 2 cos(x_3) l_2 m_2 x_1))}{(l_1 l_2 m_1)} \quad \text{(B.9)}$$

$$z_5 = \frac{(12 sin(x_3) m_m x_1)}{(l_1 m_1)} \quad \text{(B.10)}$$

$$z_6 = \frac{(12 cos(x_3) m_m x_1)}{(l_1 m_1)} \quad \text{(B.11)}$$

$$y_1 = \frac{(12 x_1)}{(l_1^2 m_1)} \quad \text{(B.12)}$$

$$y_2 = \frac{(12(l_1^2 m_1 x_2 - l_2^2 m_2 x_1))}{(l_1^2 l_2^2 m_1 m_2)} \quad \text{(B.13)}$$

$$y_3 = \frac{(12 x_1)}{(l_1^2 m_1)} \quad \text{(B.14)}$$

$$y_4 = \frac{(12 x_2)}{(l_2^2 m_2)} \quad \text{(B.15)}$$

$$y_5 = x_3 \quad \text{(B.16)}$$

$$y_6 = x_4 \quad \text{(B.17)}$$

B.3 Constrained State Equations.

$$\dot{\chi}_1 = \frac{1}{(l_2^2 m_2 r a_1 r a_2 l_1^2 m_1)} \begin{pmatrix} -(12(((b_1+b_2)ra_2 + k2_2^2)ra_1 + ra_2 k2_1^2)l_2^2 m_2 x_1 - \dots \\ \dots((ra_2 u_3 - k2_2 u_2)ra_1 + ra_2 k2_1 u_1)l_2^2 m_2 l_1^2 m_1 - \\ \dots 12(ra_2 b_2 + k2_2^2)ra_1 l_1^2 m_1 x_2) \end{pmatrix} \quad \text{(B.18)}$$

$$\dot{\chi}_2 = \frac{(12(l_2^2 m_2 x_1 - l_1^2 m_1 x_2)(ra_2 b_2 + k2_2^2) + (ra_2 u_4 + k2_2 u_2)l_2^2 m_2 l_1^2 m_1)}{(l_2^2 m_2 r a_2 l_1^2 m_1)} \quad \text{(B.19)}$$

$$\dot{\chi}_3 = \frac{(l_1^2 m_1 u_5 + 12x_1)}{(l_1^2 m_1)} \quad \text{(B.20)}$$

$$\dot{\chi}_4 = \frac{((l_1^2 m_1 u_6 - 12x_1)l_2^2 m_2 + 12 l_1^2 m_1 x_2)}{(l_2^2 m_2 l_1^2 m_1)} \quad \text{(B.21)}$$

$$y_1 = \frac{(12x_1)}{(l_1^2 m_1)} \quad \text{(B.22)}$$

$$y_2 = \frac{(-12(l_2^2 m_2 x_1 - l_1^2 m_1 x_2))}{(l_2^2 m_2 l_1^2 m_1)} \quad \text{(B.23)}$$

$$y_3 = \frac{(12x_1)}{(l_1^2 m_1)} \quad \text{(B.24)}$$

$$y_4 = \frac{(12x_2)}{(l_2^2 m_2)} \quad \text{(B.25)}$$

$$y_5 = x_3 \quad \text{(B.26)}$$

$$y_6 = x_4 \quad \text{(B.27)}$$

$$\begin{aligned} E(1,1) &= \frac{(4(3m_2 + m_1 + 3m_m))}{m_1} \\ E(1,2) &= \frac{(6cos(x_4)l_1)}{l_2} \\ E(1,3) &= \frac{(-6sin(x_4)l_1 x_2)}{l_2} \\ E(1,4) &= \frac{(-6sin(x_4)l_1 x_2)}{l_2} \\ E(2,1) &= \frac{(6cos(x_4)l_2 m_2)}{(l_1 m_1)} \\ E(2,2) &= 4 \\ E(2,3) &= \frac{(6sin(x_4)l_2 m_2 x_1)}{(l_1 m_1)} \end{aligned}$$

$$\begin{aligned} E(3,3) &= 1 \\ E(4,4) &= 1 \end{aligned}$$

(B.28)

Appendix C

Results Obtained with the use of Anti-Aliasing Filters.

C.1 Introduction.

The results presented in chapters 5 and 6 were obtained without the use of an anti-aliasing filter to prevent frequencies above the Nyquist frequency (which is equivalent to half the sampling frequency) being reflected into the frequency range below the Nyquist frequency as aliased signals. To prevent this, an analog ant-aliasing filter may be used to pre-filter the measurements so that no frequencies at or above the Nyquist frequency are present in the sampled signals.

This appendix repeats the same tests as those presented in chapter 5 but this time the measurements of angular positions and velocities have been pre-filtered to prevent aliasing.

C.2 Anti-Aliasing Filters.

The anti aliasing filters used to pre-filter the measurements from the manipulator were analog eight pole elliptic filters with a cut-off frequency set at 12Hz which

was approximately a third of the sample rate of the system.

The pre-filtered signals were

- link angular position measurements and
- link angular velocity measurements.

C.3 Results.

The results obtained are shown in figures C.1 to C.12. They repeat exactly the tests presented in chapter 5 (figures 5.14 to 5.26) apart from the use of the anti-aliasing filters and the use of the tachometer derived second angular velocity measurement (now filtered) as this provided a better conditioned signal than the differentiated second position measurement.

From the graphs, it is immediately apparent that

- the performance of the manipulator when observed angular velocities are used in the feedback controller is considerably better than when measured angular velocities are used, and
- the performance of the manipulator has been degraded by the use of the anti-aliasing filters whether the observer is used or not.

The degradation in performance when the anti-aliasing filters are used manifests itself as an oscillatory response which only marginally affects the manipulator when the observer is used but produces a highly oscillatory response when it is not.

This degradation is caused by the phase delay introduced into the measured signals by the anti-aliasing filters. This phase delay is clearly apparent in figure C.5 where the observed velocities can be seen to lead the measured angular velocities by a considerable margin. As the outputs of the observer need not be filtered.

they do not suffer from the same phase delay and hence the performance of the manipulator when the observer is used is better than when it is not.

The observer will, however, be affected by the phase delayed measurements as they are used in the observer feedback loop. Furthermore, in practice only the observed velocities are used in the feedback controller to minimize the number of parameters passed over ethernet and hence maximise the sample rate. Consequently, the controller uses the phase delayed position signals. The combination of these two effects means that the performance of the manipulator when the observer is used will also be degraded but it can be seen that this degradation is not severe.

C.3.1 High Sample Rate Controller.

To alleviate the problems caused by phase delay in the anti-aliasing filters a test was performed in which the sample rate was increased to 108Hz by not implementing the observer. This allowed the cut-off frequency of the anti-aliasing filters to be increased to 50Hz which reduces the phase delay of the measured signals in the range of frequencies in which the manipulator is capable of operating. The controller gains used in the test are the same as those used in the tests of the preceeding section.

The results of the test are shown in figures C.13 and C.14. It can be seen that whilst the oscillatory response has been reduced (c.f. figure C.4), the performance of the manipulator is still poor in comparison to the performance of the observer enhanced manipulator (c.f. figures C.1 and C.3), especially with respect to the motor actuation voltages. Raising the cut-of frequency of the filters has reduced the phase delay but has allowed more of the high frequency noise on the tachometer signals through into the controller and hence into the control voltages.

C.4 Conclusion.

The results presented in this appendix have shown that the performance of the manipulator both with and without using the observer is poorer when analogue filters are used to prevent aliasing of the sampled measurements of link angular positions and velocities. This suggests that the measured signals are degraded more by the use of the filters than by the presence of aliased signals.

Given the use of anti-aliasing filters, it has been shown that the observer is capable of producing non-phase delayed observed outputs which can be used effectively in the feedback control of the manipulator despite having the phase delayed measurements used in the observer feedback loop. When the same phase delayed measurements are used directly in the feedback controller, the resulting performance of the manipulator is highly oscillatory.

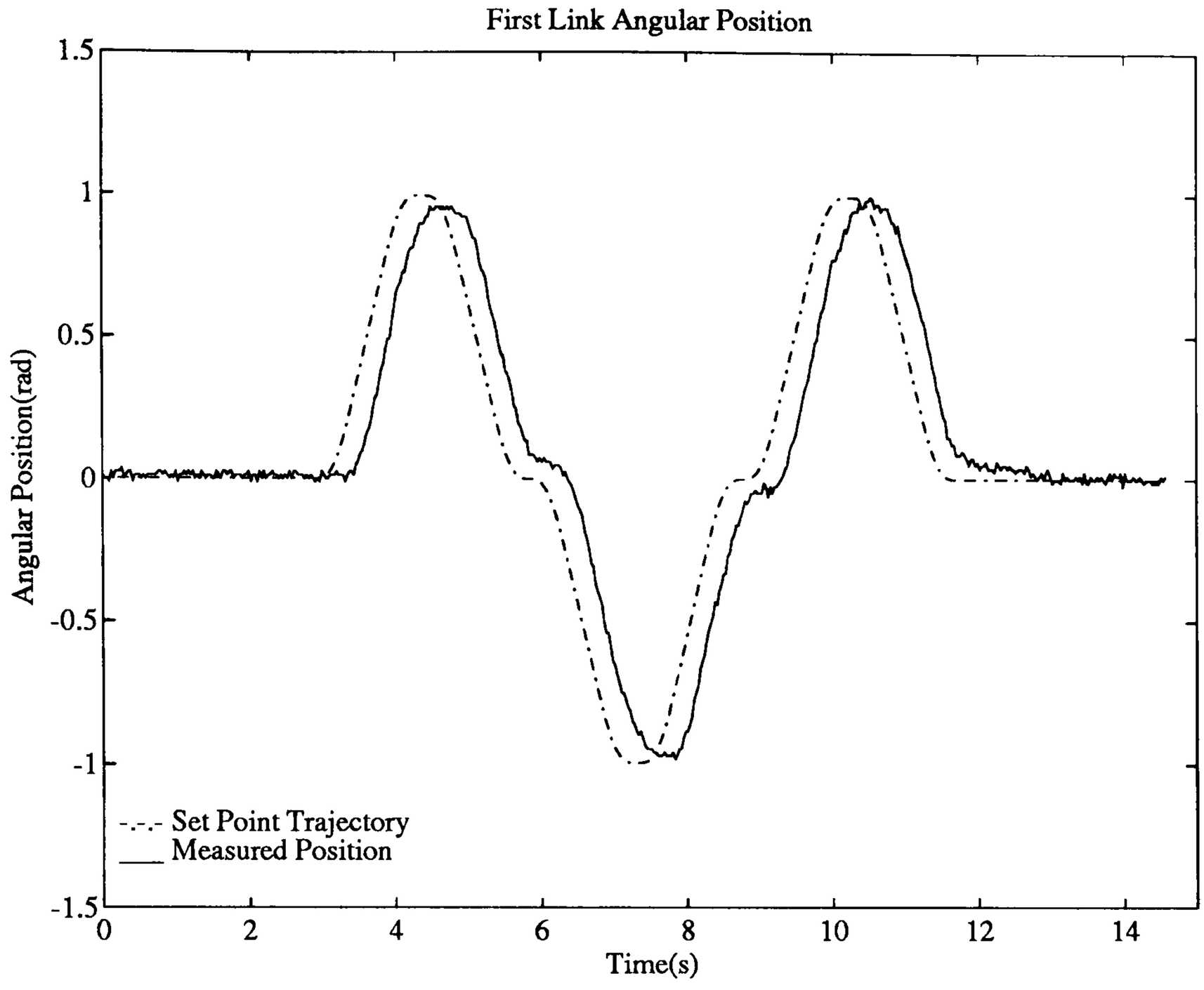


Figure C.1: Test 1af. Link Positions and Set Points using Observed Angular Velocities in the Controller.

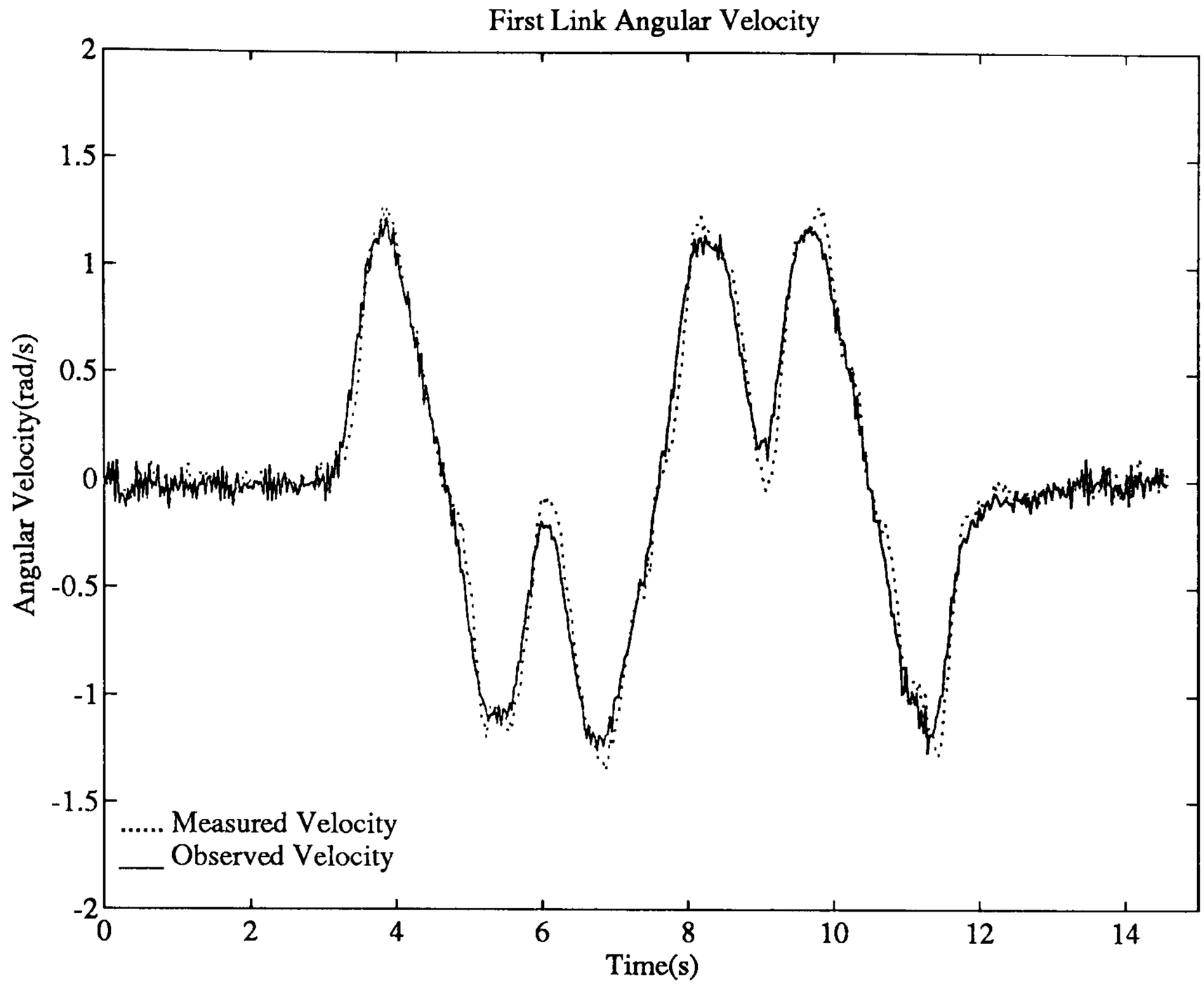
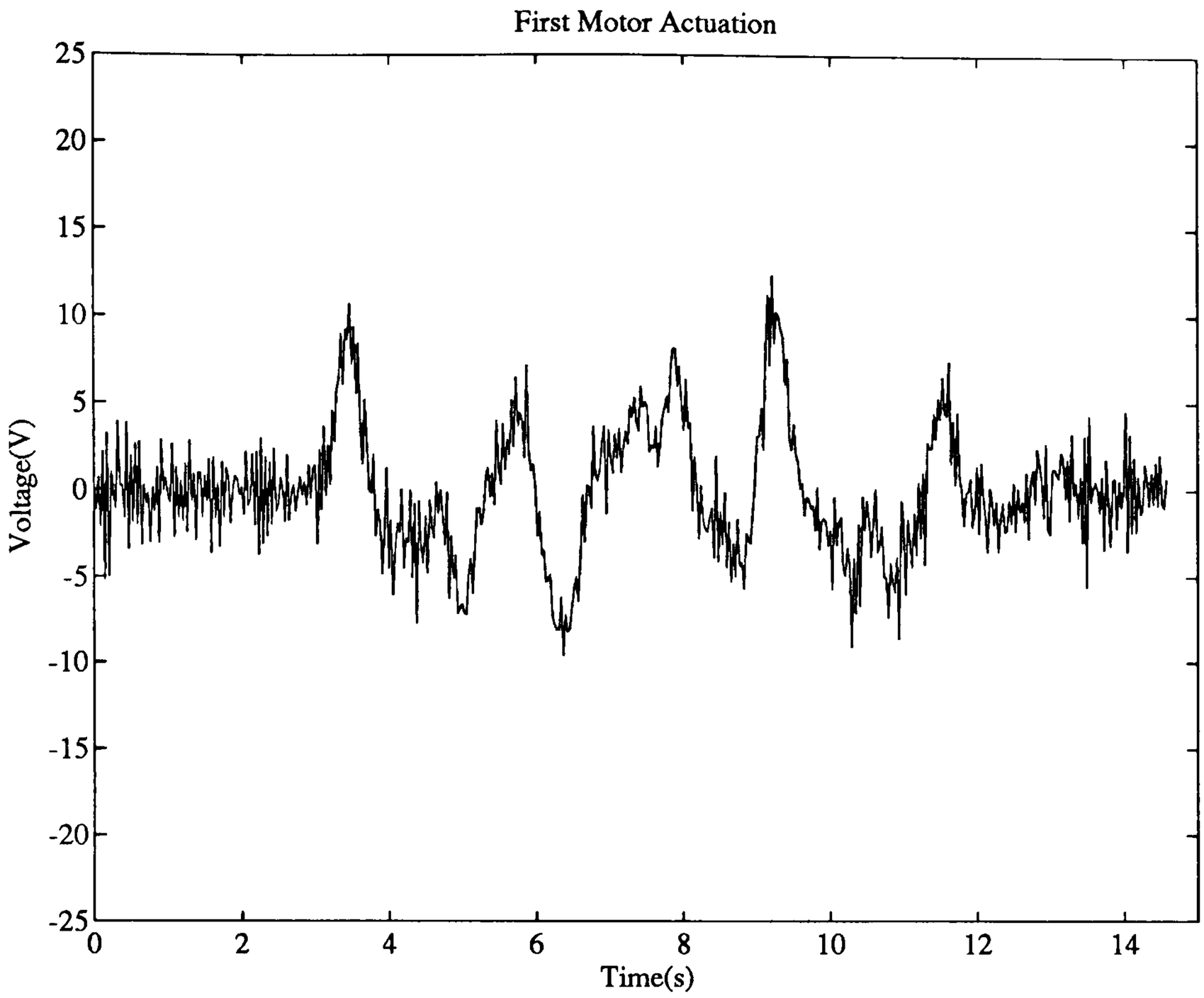


Figure C.2: Test 1af. Link Angular Velocities using Observed Angular Velocities in the Controller.



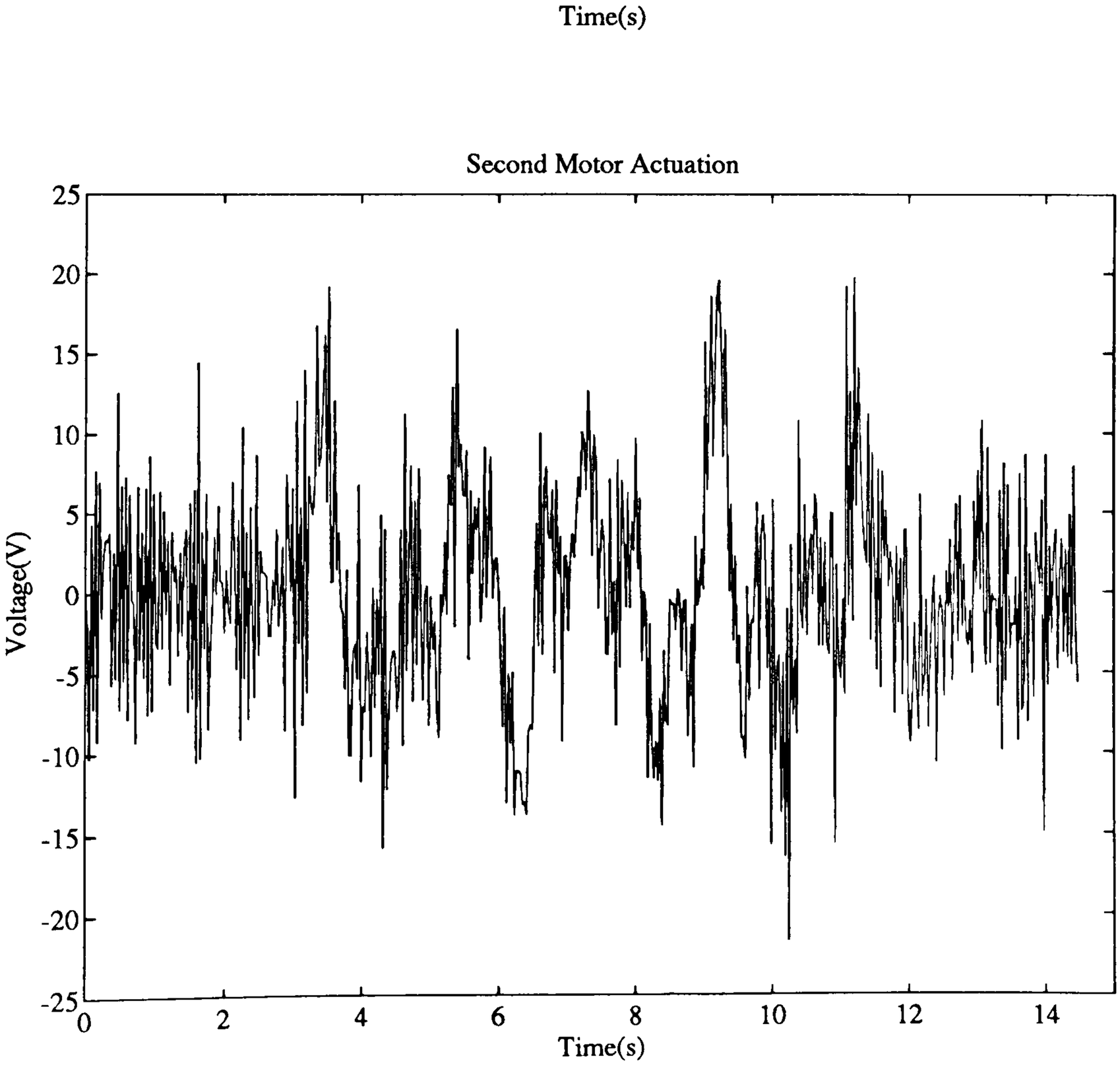


Figure C.3: Test 1af. Motor Actuation Voltages using Observed Angular Velocities in the Controller.

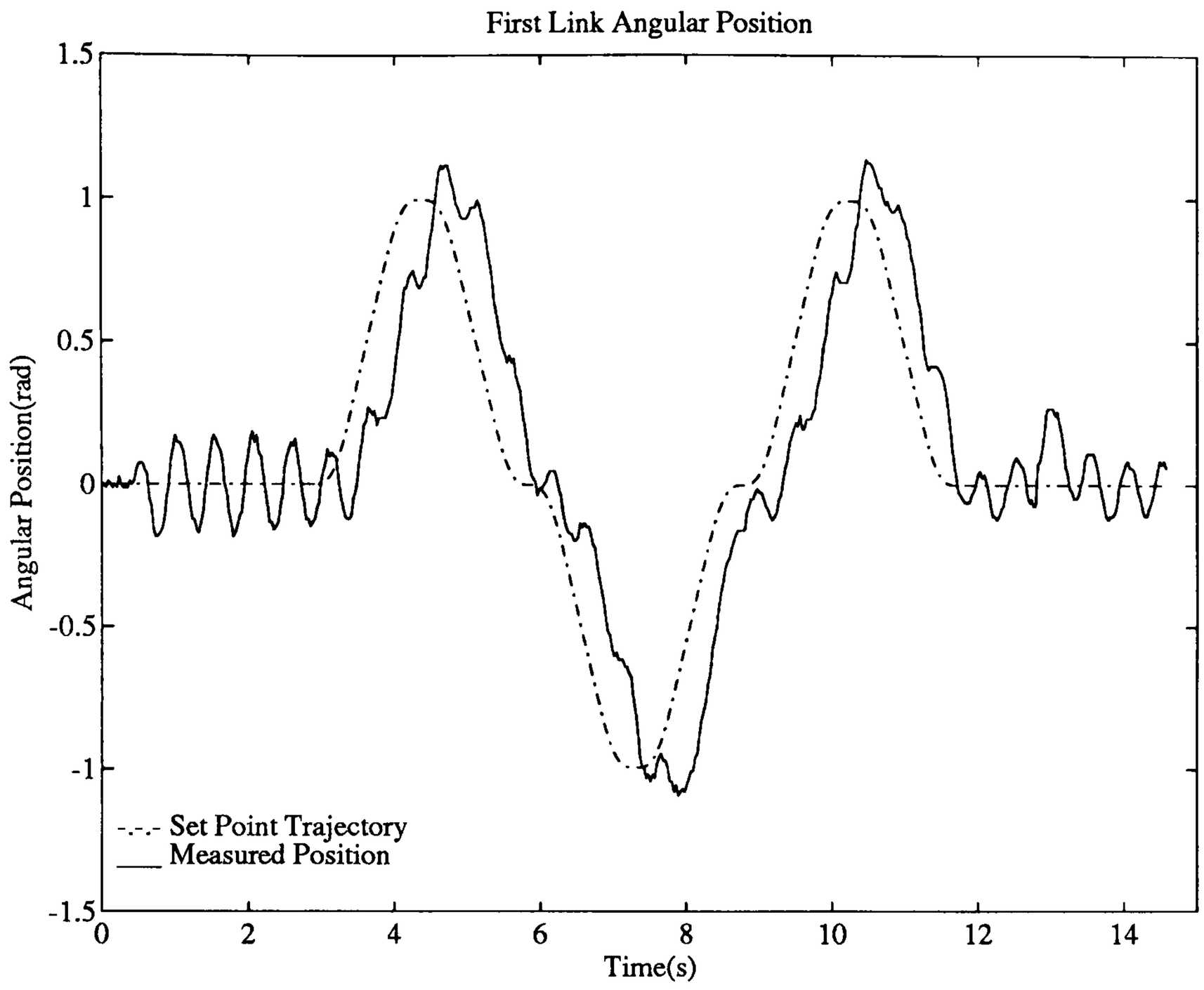
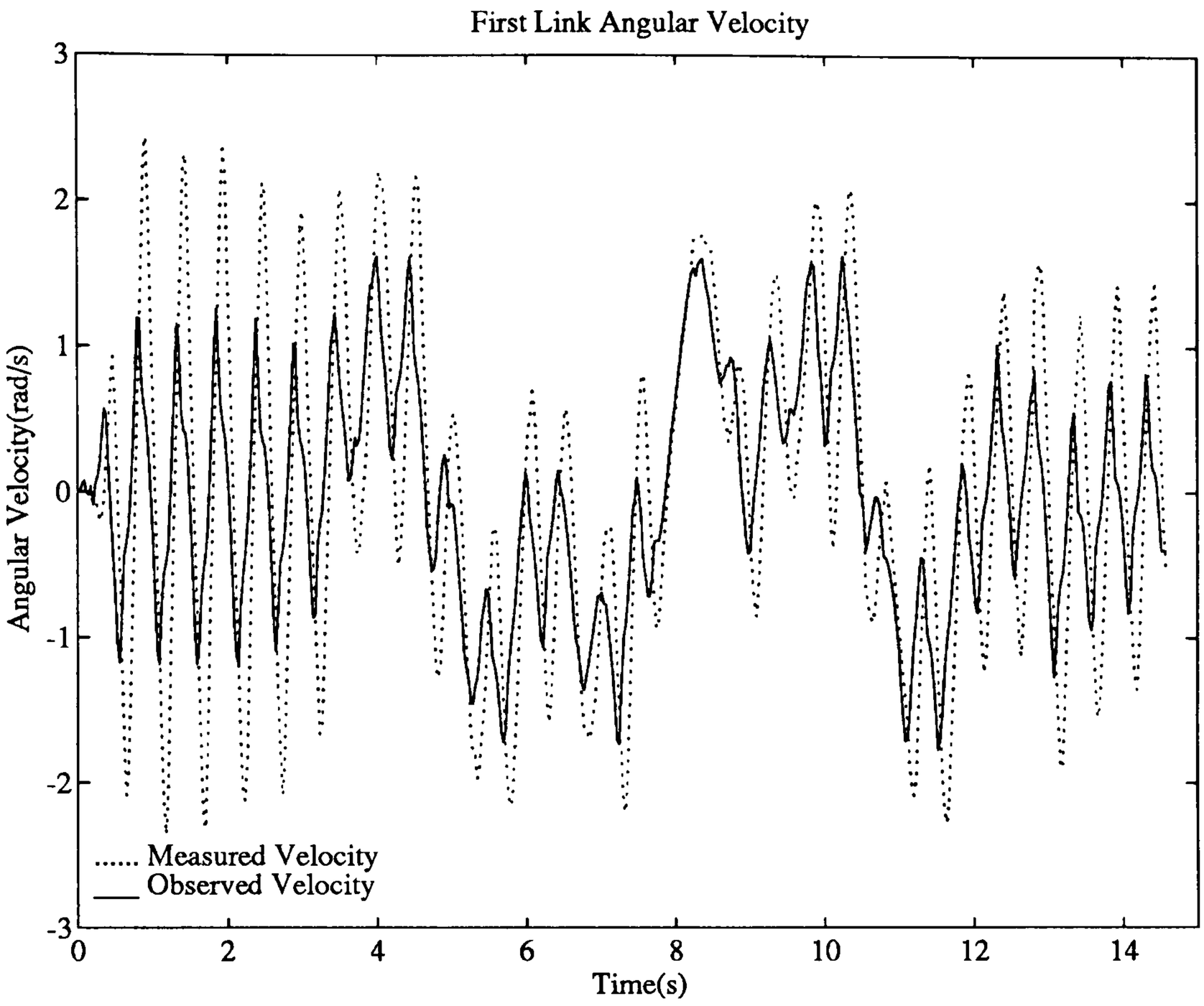


Figure C.4: Test 2af. Link Positions and Set Points using Measured Angular Velocities in the Controller.



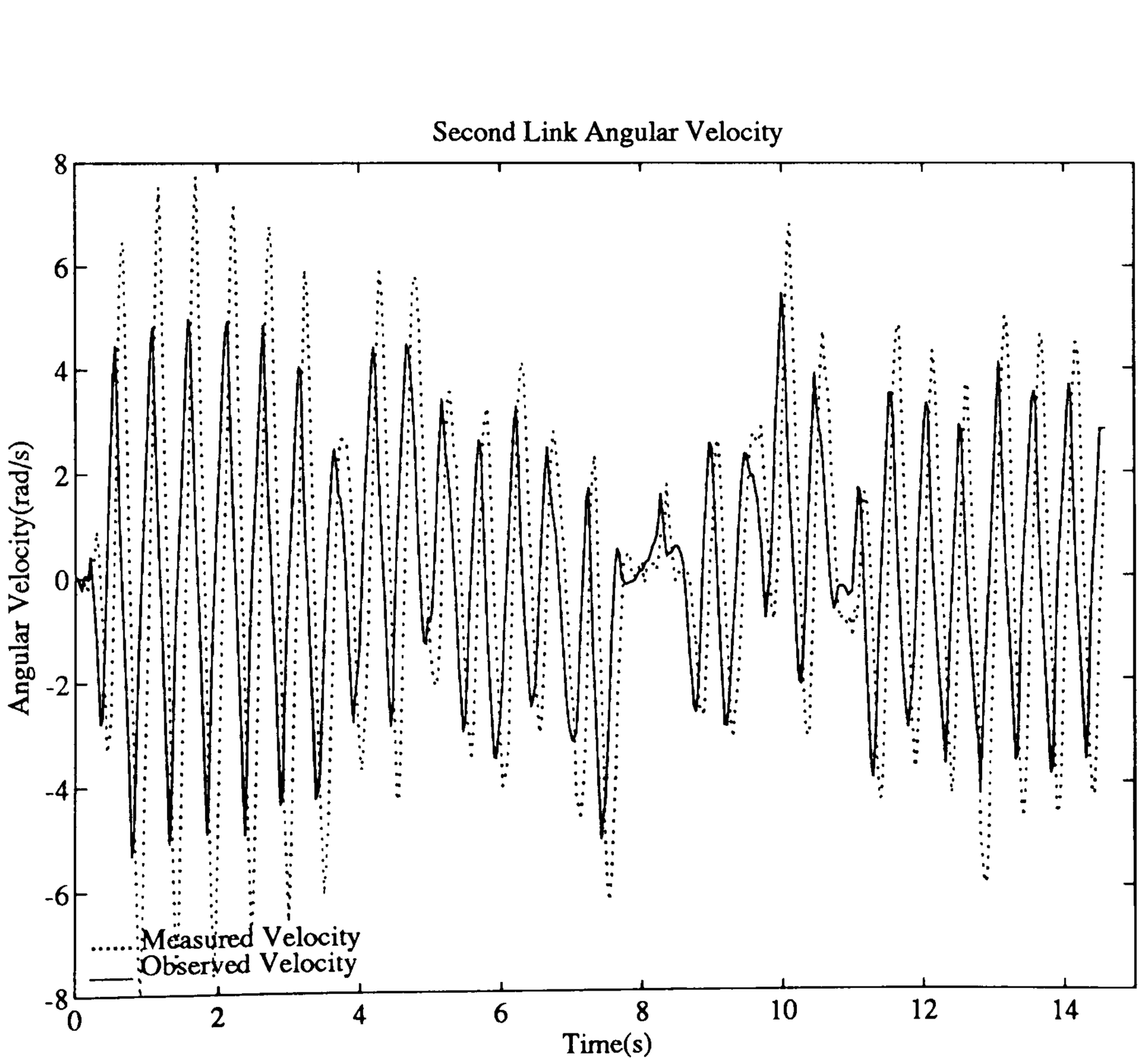
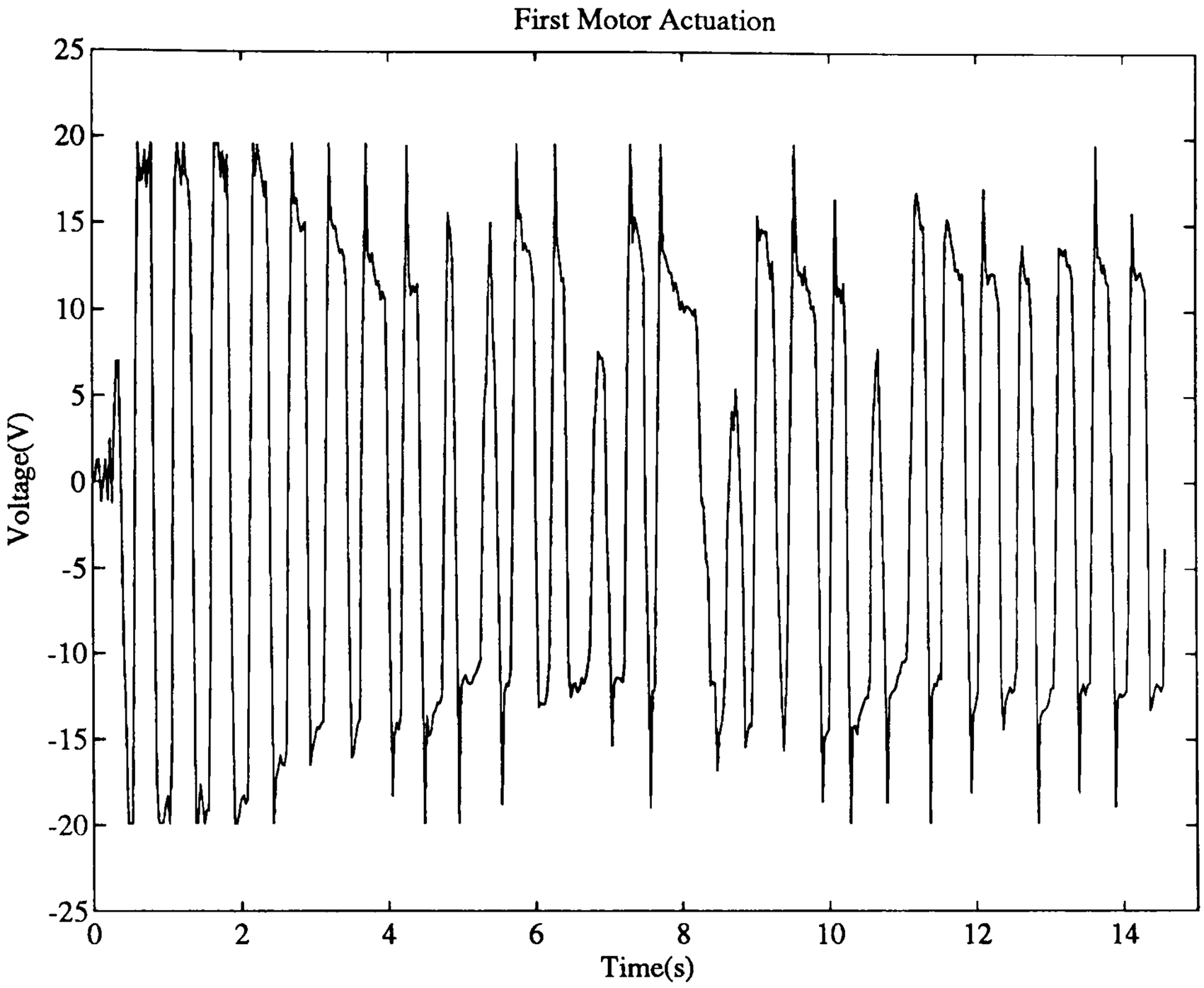


Figure C.5: Test 2af. Link Angular Velocities using Measured Angular Velocities in the Controller.



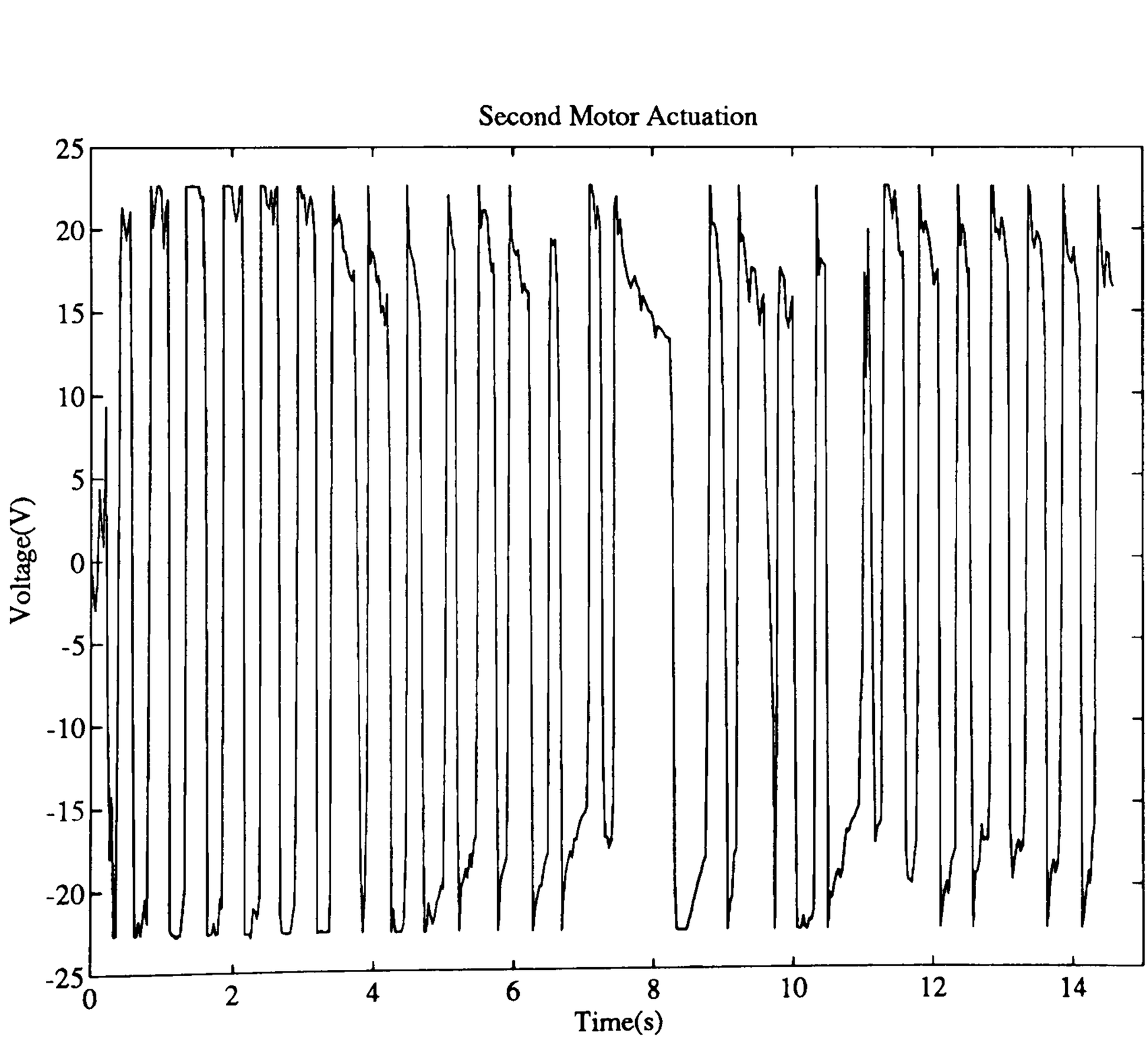


Figure C.6: Test 2af. Motor Actuation Voltages using Measured Angular Velocities in the Controller.

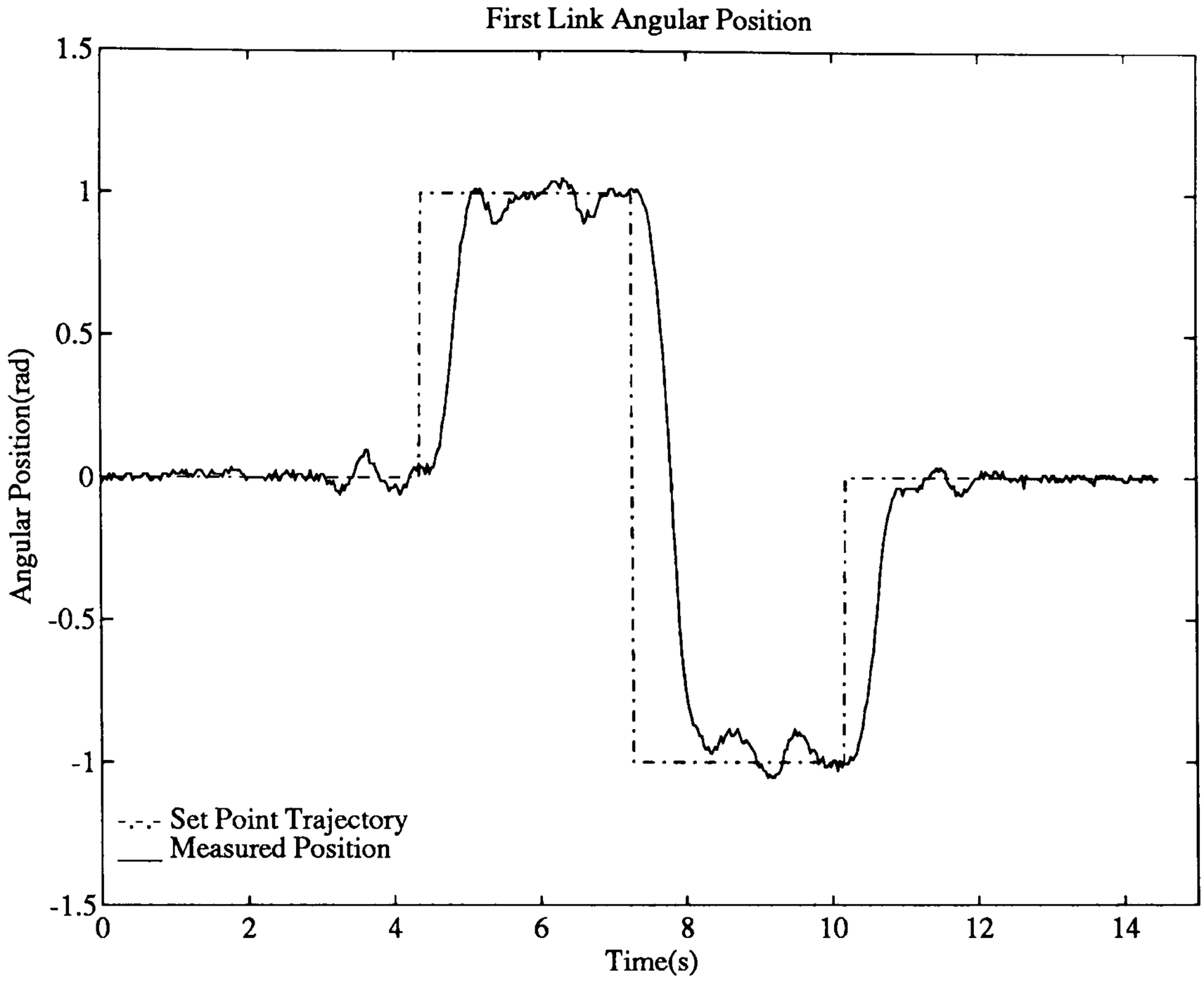


Figure C.7: Test 3af. Link Positions and Set Points using Observed Angular Velocities in the Controller.

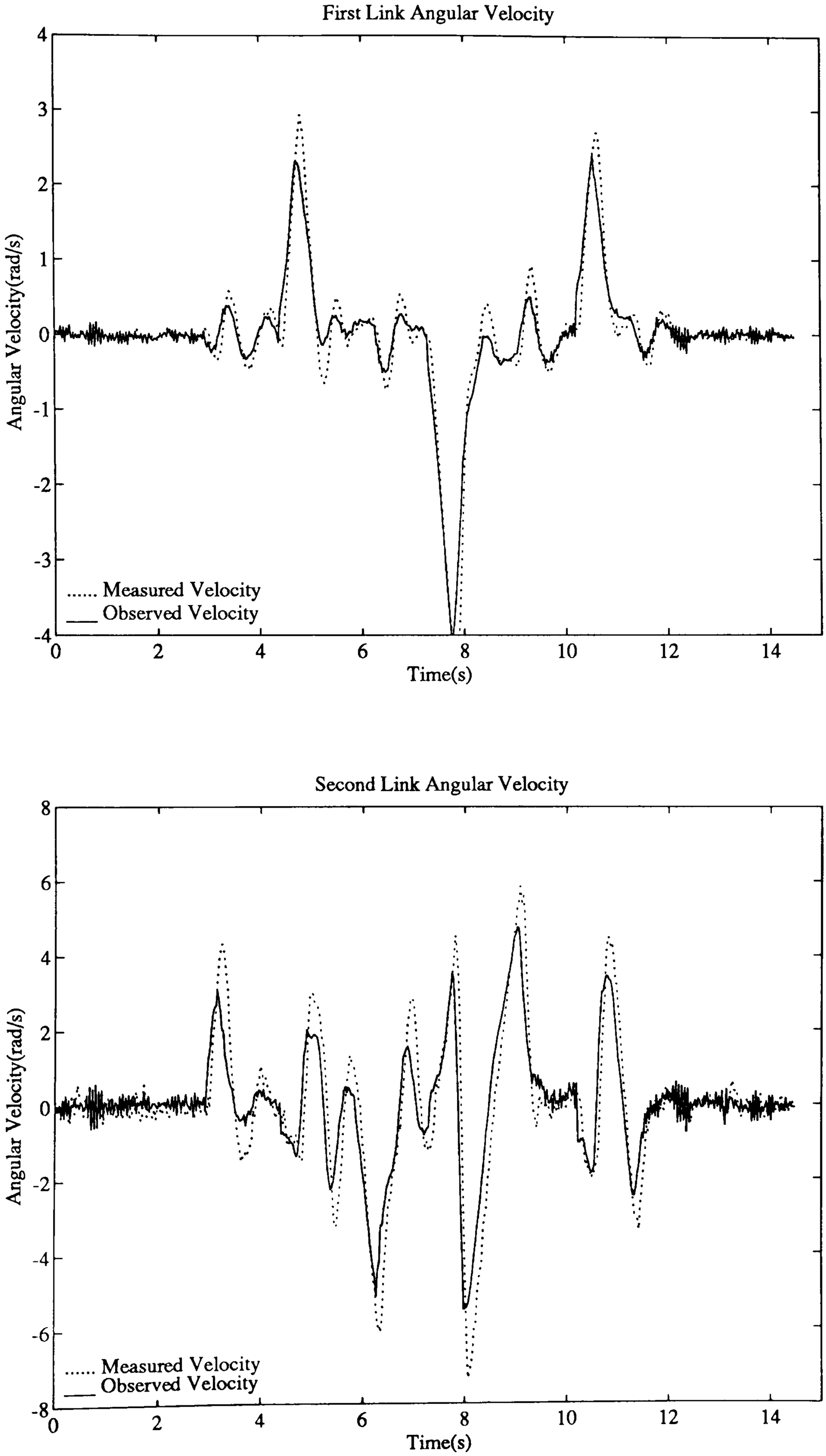
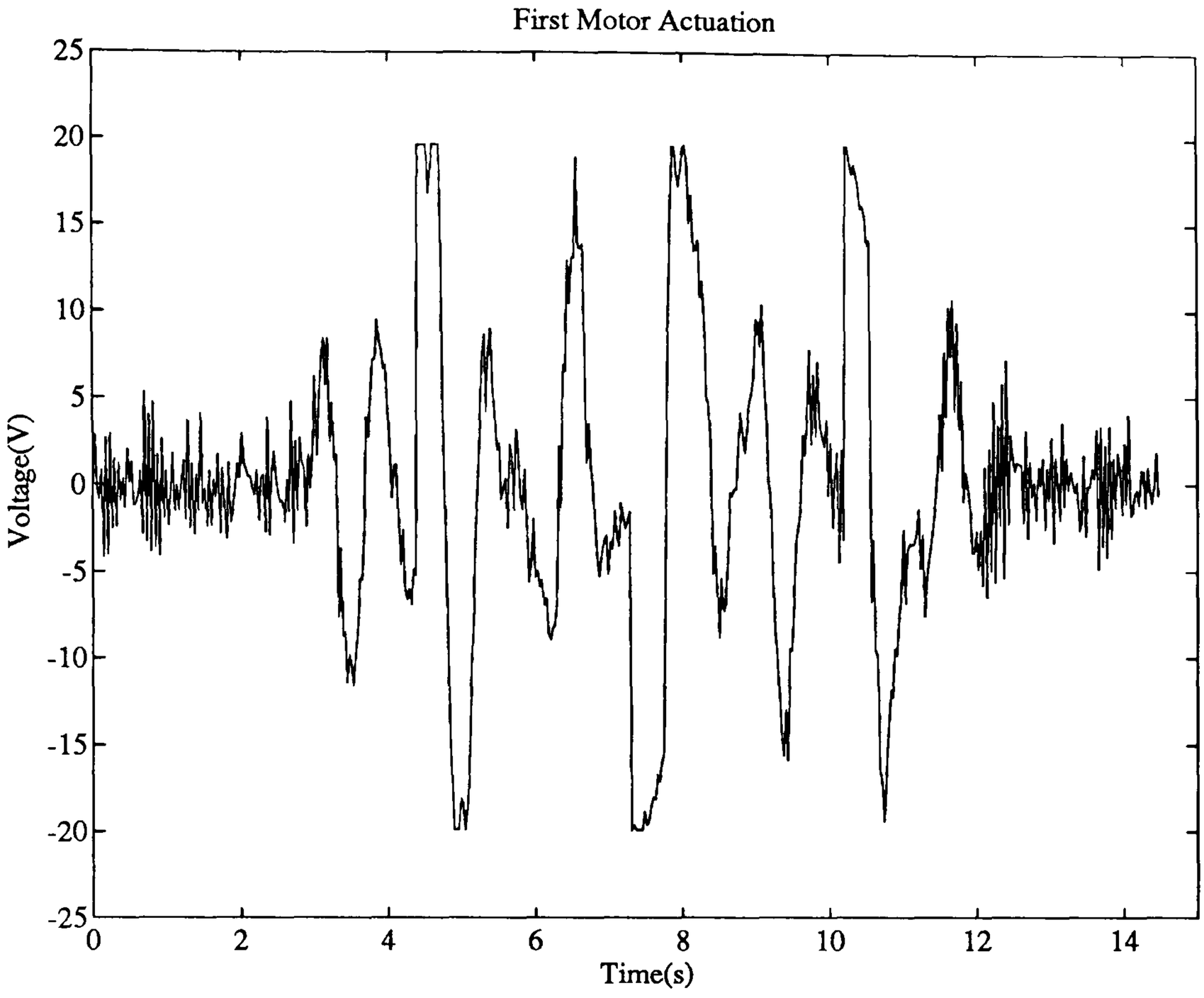


Figure C.8: Test 3af. Link Angular Velocities using Observed Angular Velocities in the Controller.



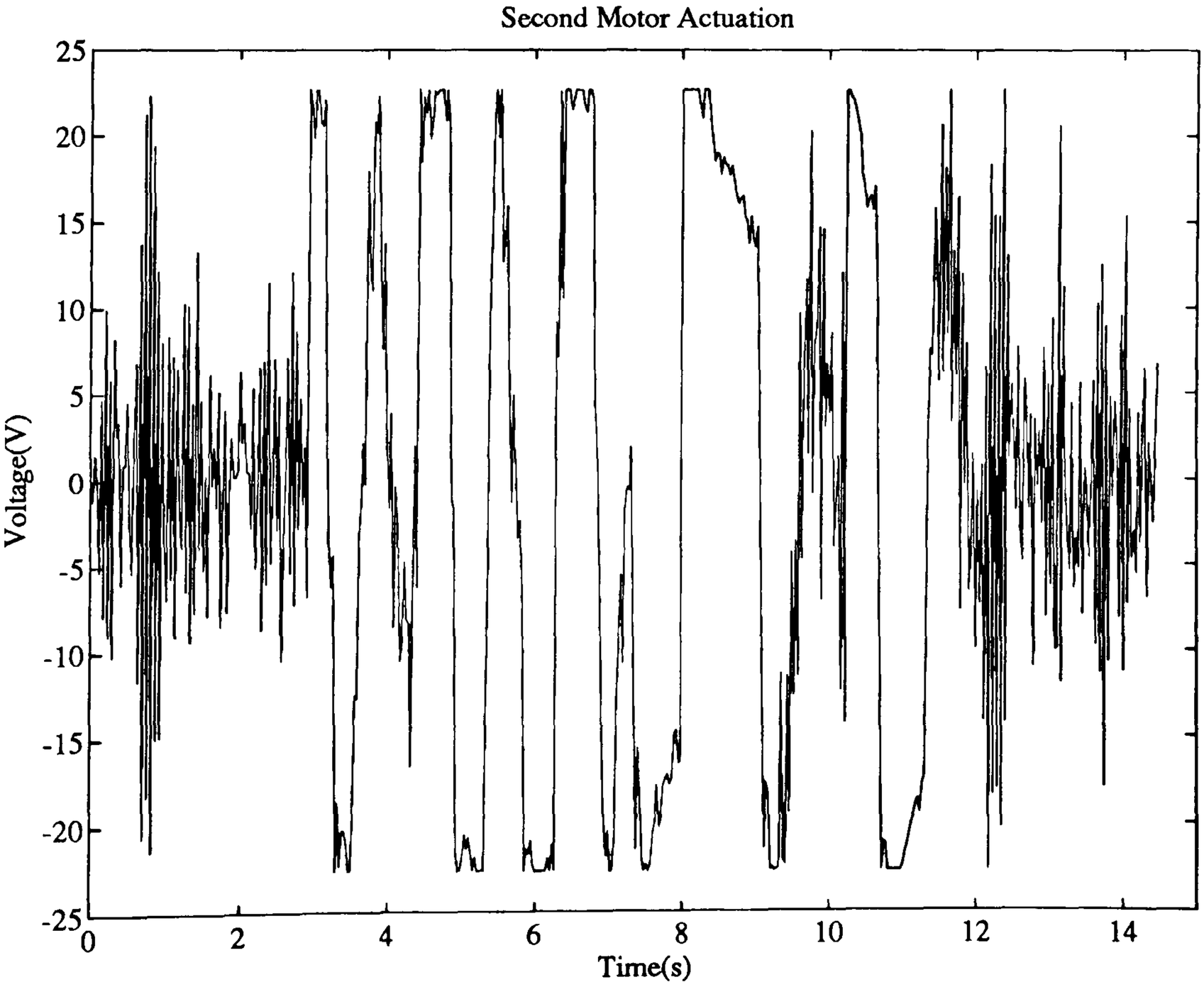


Figure C.9: Test 3af. Motor Actuation Voltages using Observed Angular Velocities in the Controller.

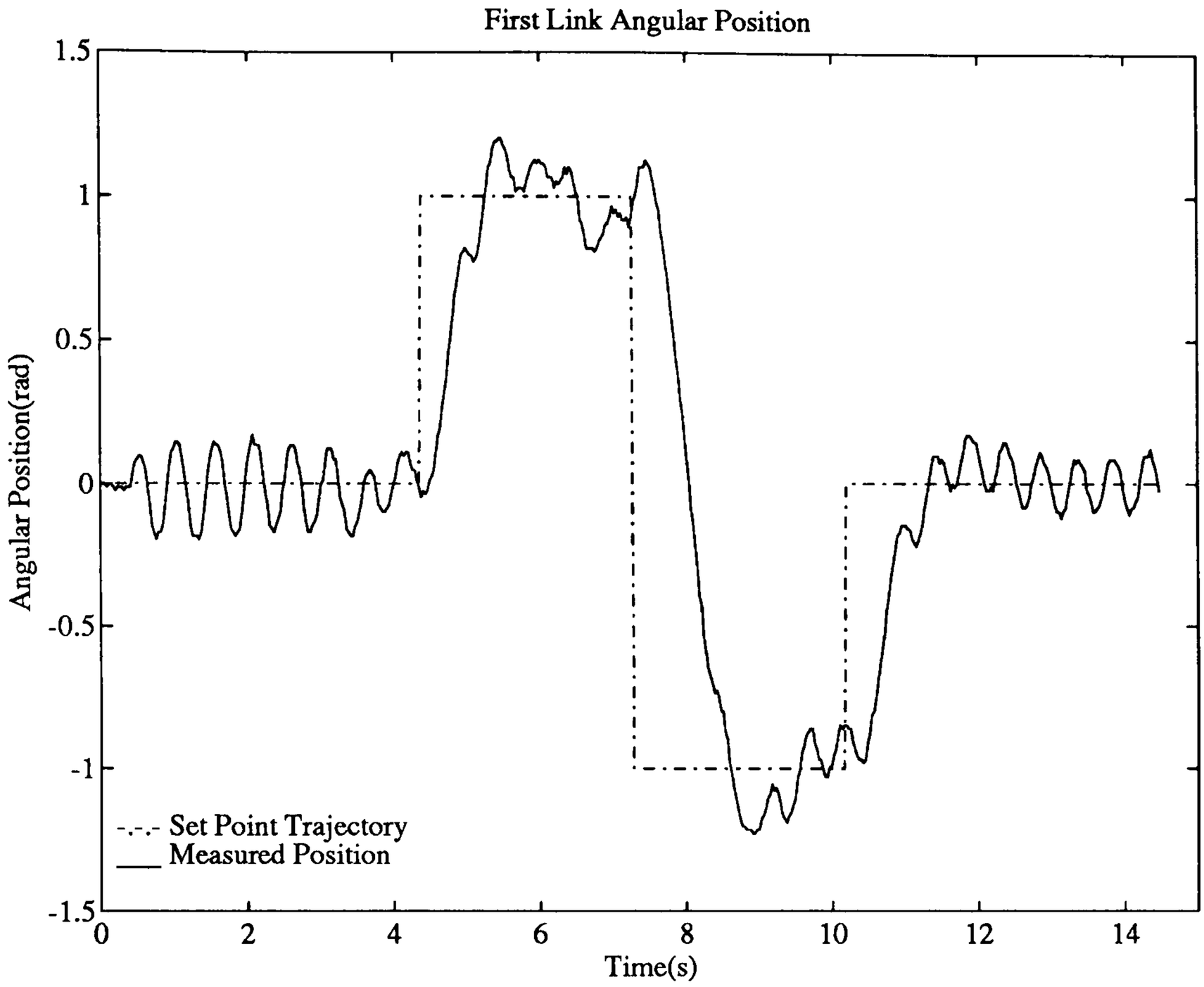


Figure C.10: Test 4af. Link Positions and Set Points using Measured Angular Velocities in the Controller.

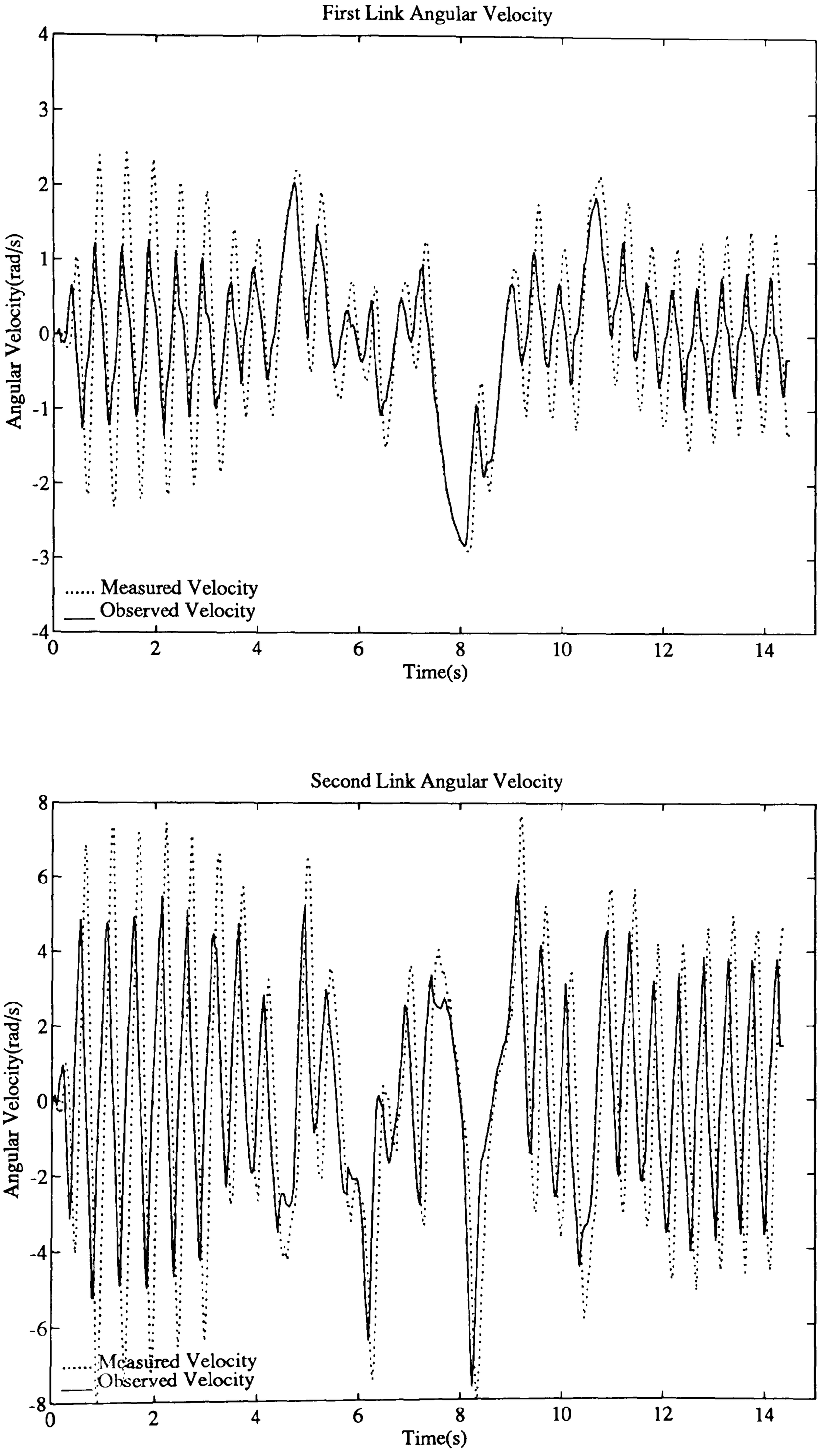
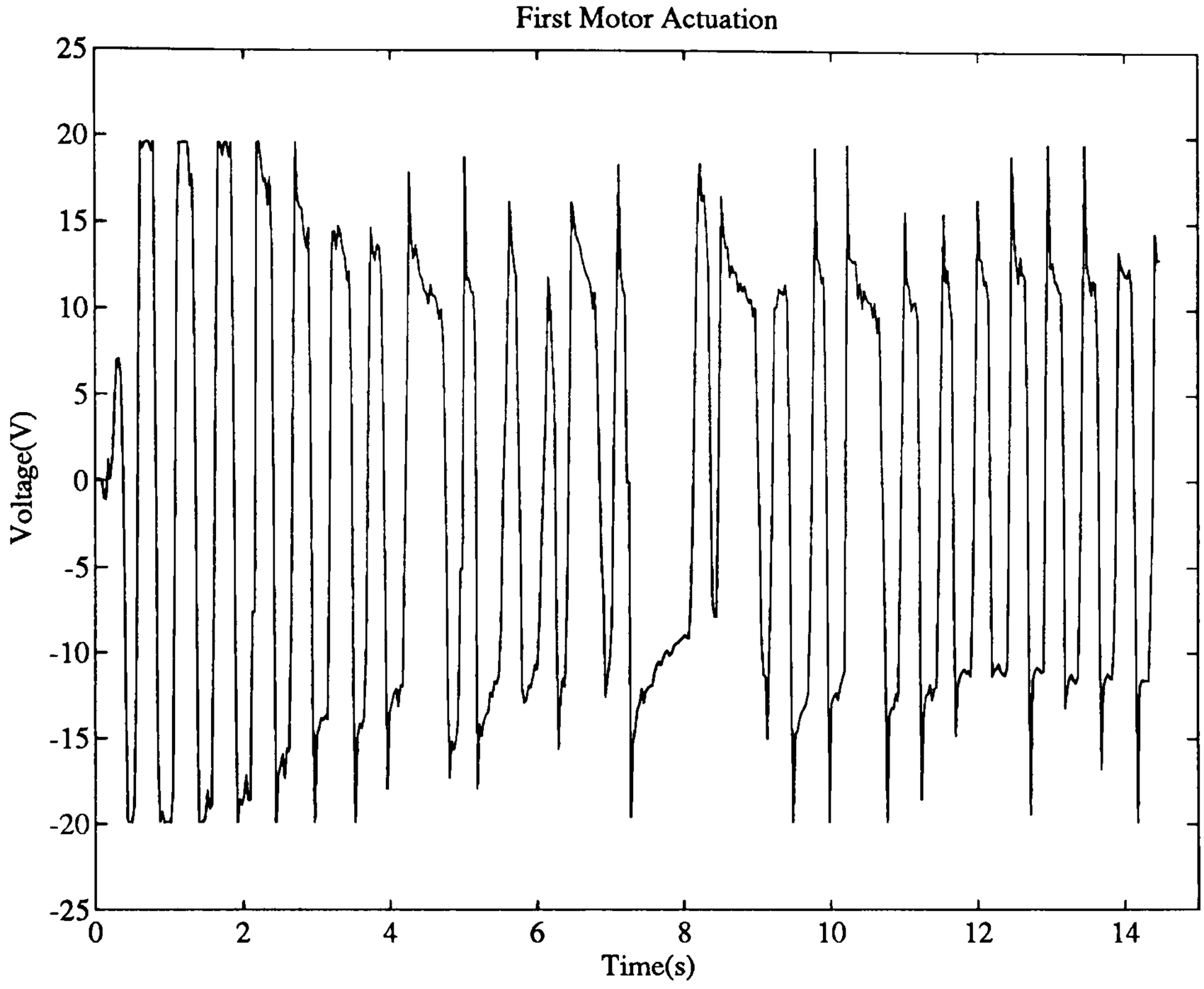


Figure C.11: Test 4af. Link Angular Velocities using Measured Angular Velocities in the Controller.



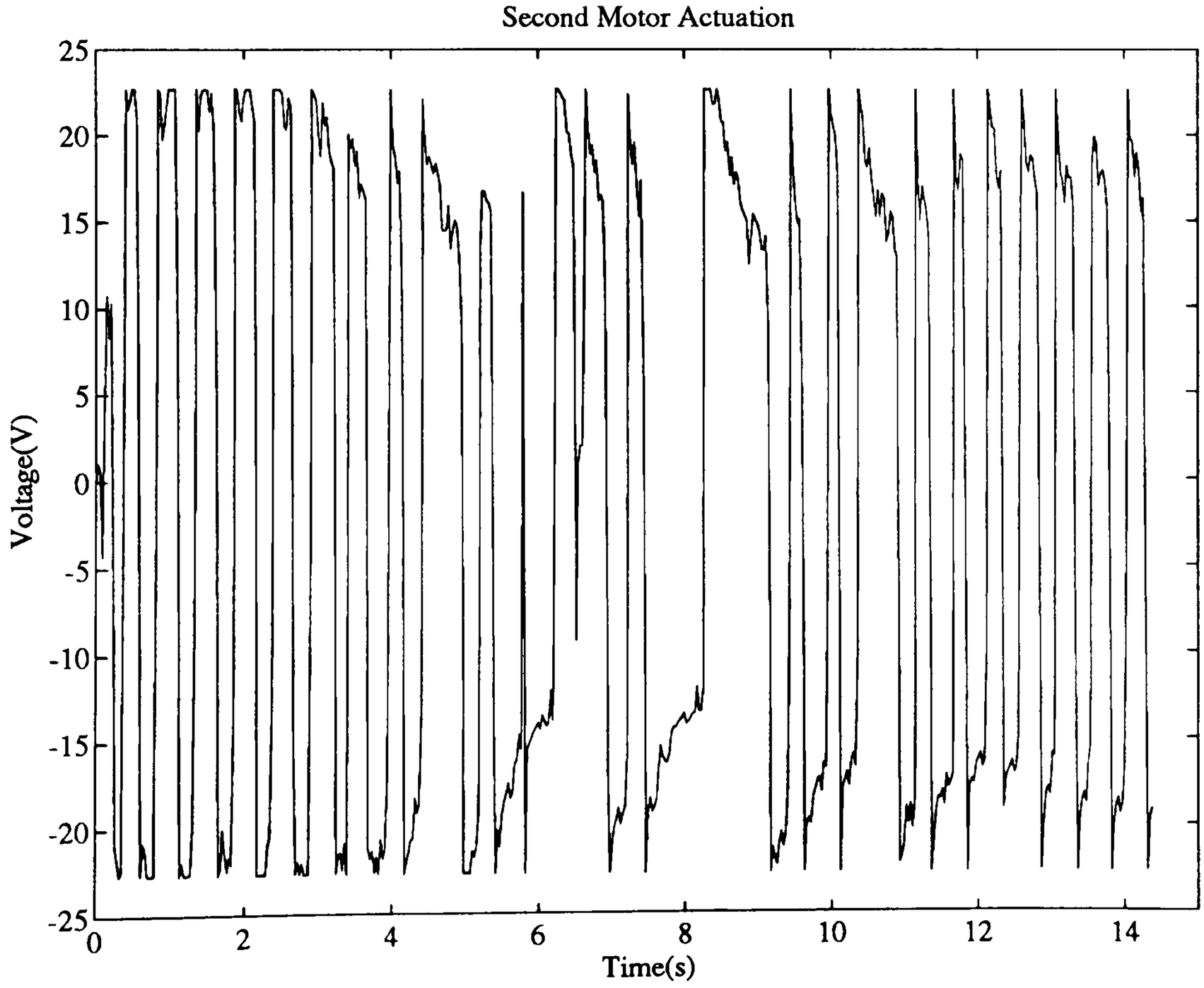


Figure C.12: Test 4af. Motor Actuation Voltages using Measured Angular Velocities in the Controller.

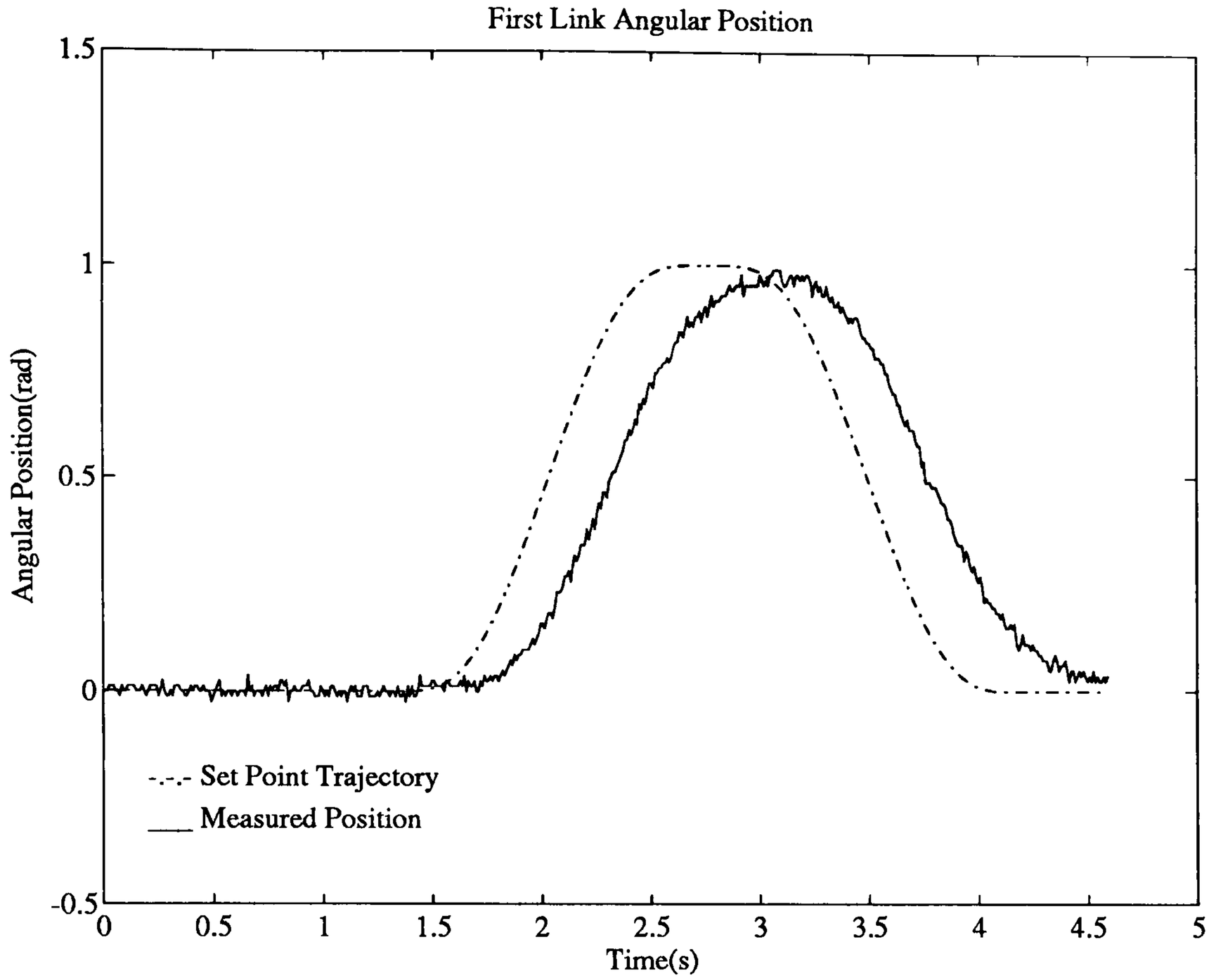
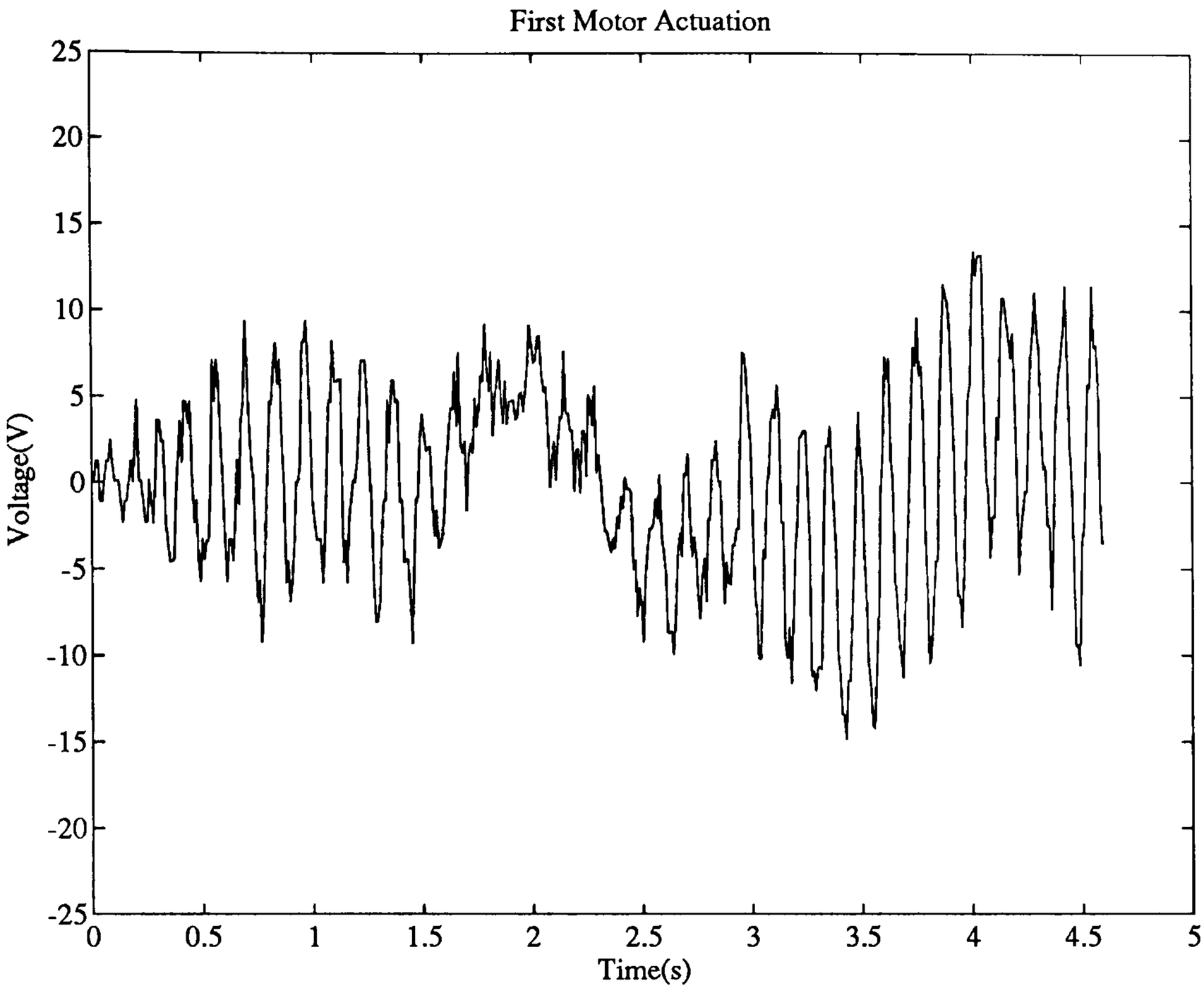


Figure C.13: Test 5af. Link Angular Positions using Measured Angular Velocities in the Controller and a 108Hz Sample Rate.



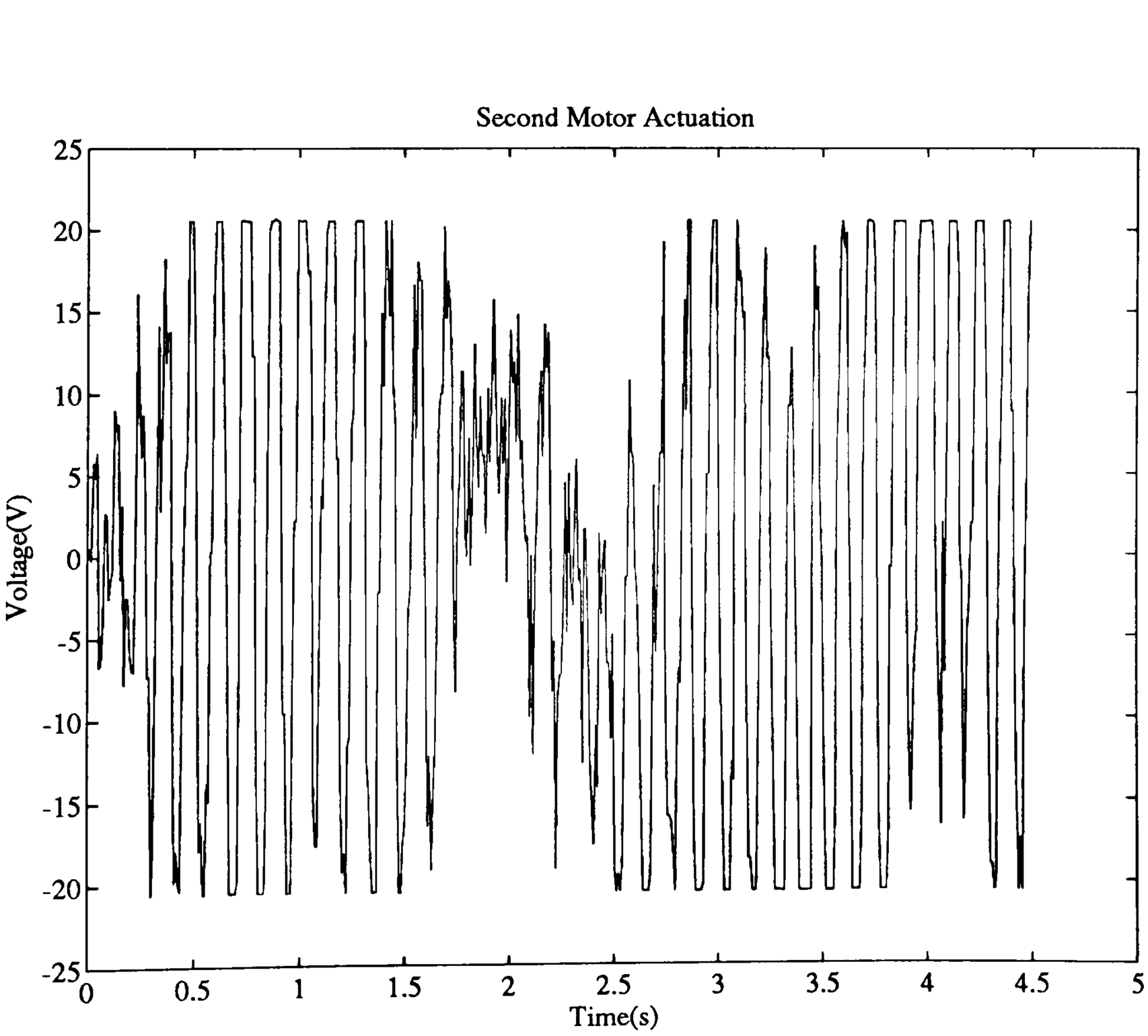


Figure C.14: Test 5af. Motor Actuation Voltages using Measured Angular Velocities in the Controller and a 108Hz Sample Rate.

Appendix D

Observer Software.

D.1 Observer Software Creation Code.

This is the REDUCE code that writes the software for the model based observer.

rcs2con

```
#! /bin/sh

     ##########################################
     ##### Model Transformation Tools #####
     ##########################################

# Bourne shell script: rcs2con
# Reduce constrained-state to conic
# D.W.Roberts
# 20th March 1992

# Output must be filtered by for2mat.

# Remove the old log file
rm -f rcs2con.log

# Use reduce to accomplish the transformation
reduce >rcs2con.log << EOF

%Read the reduce definitions file
in "$1.rde";

%Read the reduce  constrained-state equations   file
```

```
in "$1.rcs_";

ON BigFloat, NumVal;
PRECISION 16; %Compatible with Matlab
OFF Nat;

%ON NERO;          % Suppress zero elements

%Fortran switches - one line expressions
ON fort;
cardno!* := 1;
fortwidth!* := 100;

OFF period;
OFF echo;

%Calculate the inverse of E
MTTiE := MTTE^-1;

OUT "$1.con";
%Headings -  style
write "TASK MODULE $1;";
write "{Robot equations for system $1}";
write "{File $1_scs.m}";
write "{Generated by MTT}";
write " ";
write "USE  SigDefs:RealSignal,QueueLength,OutType,FBtype,StatType;
";
write " ";
write "ENTRYPORT stat1port : StatType;";
write "ENTRYPORT stat2port : StatType;";
write "ENTRYPORT FbackPort : FBtype;";
write "ENTRYPORT startrun  : SignalType;";
write "EXITPORT ObOut : OutType;";
write "EXITPORT Ready : SignalType;";
write "EXITPORT ow1Port : real;";
write "EXITPORT ow2Port : real;";
write " ";
write "CONST";
write "{Model Parameters}";
write " ";
write "VAR";
write " MTTX : PACKED ARRAY [1..",MTTNx,"] OF real;";
write " MTTdX : PACKED ARRAY [1..",MTTNx,"] OF real;";
write " MTTdXe : PACKED ARRAY [1..",MTTNx,"] OF real;";
write " MTTE : PACKED ARRAY [1..",MTTNx,",1..",MTTNx,"] OF real;";
write " MTTiE : PACKED ARRAY [1..",MTTNx,",1..",MTTNx,"] OF real;";
write " Y : OutType;";
```

```
write " ";
IF MTTNu>0 THEN
FOR i := 1:MTTNu-1 DO
BEGIN
write " MTTu",i, ",";
END;
write " MTTu",MTTNu," : real;";
write " ";
write "{_____________________________________________________________}";
write "PROCEDURE CalcInvE;";
write " ";
write "BEGIN";
MTTiE := MttiE;
write "END;";
write " ";
write "{_____________________________________________________________}";
write "PROCEDURE InitVect;";
write "{Initialises the state vector}";
write " ";
write "BEGIN";
FOR i := 1:MTTNx DO
BEGIN
  write " MTTX[", i, "] := 0;";
END;
write "END;";
write " ";
write "{_____________________________________________________________}";
write "PROCEDURE SetInputVector;";
write "{Sets up input vector}";
write " ";
write "BEGIN";
IF MTTNu>0 THEN
FOR i := 1:MTTNu DO
BEGIN
  write " MTTu", i, " := ########";
END;
write "END;";
write "{_____________________________________________________________}";
write "PROCEDURE CalcDerivVector;";
write "{Calculates the derivative of the pseudo state vector}";
write " ";
write "BEGIN";
MTTdxE := MTTdxE;
write "END;";
write " ";
write "{_____________________________________________________________}";
write "PROCEDURE CalcStateVector;";
write "{Calculates the derivative of the state vector}";
```

```
write " ";
write "VAR";
write " row,column : 1..",MTTNx,";";
write " ";
write "BEGIN";
write " FOR row:=1 TO ",MTTNx," DO";
write " MTTdX[row] := 0;";
write " ";
write " FOR row:=1 TO ",MTTNx," DO";
write " BEGIN";
write " FOR column:=1 To ",MTTNx," DO";
write " BEGIN";
write " MTTdX[row]:=MTTdX[row]+(MTTiE[row,column]*MTTdXe[column,row]);";
write " END;";
write " END;";
write "END;";
write "{_____________________________________________________}";
write "PROCEDURE Integrate;";
write "{Integrates the derivative of the state vector}";
write " ";
write "VAR";
write " row : 1..",MTTNx,";";
write " ";
write "BEGIN";
write " FOR row :=1 TO ",MTTNx," DO";
write " MTTX[row] := MTTX[row] + (MTTdX[row]*dt);";
write "END;";
write " ";
write "{_____________________________________________________}";
write "PROCEDURE CalcOutputs;";
write "{Calculates the output vector}";
write " ";
MTTY := MTTY;
write "END;";
write " ";
write "{_____________________________________________________}";
write " ";
write "BEGIN {MainProgram}";
write " ";
write "WHILE True DO BEGIN";
write " InitVect;";
write " ";
write " SEND Signal TO Ready;";
write " RECEIVE Signal FROM startrun;";
write " ";
write " ";
write " LOOP";
write "     SELECT";
```

```
write " RECEIVE stats1 FROM stat1port =>";
write " RECEIVE stats2 FROM stat2port;";
write " RECEIVE fbvals FROM FbackPort;";
write " ";
write " SetInputVector; ";
write " CalcDerivVector;";
write " CalcInvE;";
write " CalcStateVector;";
write " Integrate;";
write " CalcOutputs;";
write " SEND Y TO ObOut;";
write " SEND Y.w1 TO ow1Port;";
write " SEND Y.w2 TO ow2Port;";
write " ";
write " ";
write " OR";
write " TIMEOUT(2000) =>";
write " EXIT;";
write "     END;{select}";
write " ";
write " END;{loop}";
write " ";
write " ";
write "   END;{Infinite While}";
write " ";
write "END.{Program}";


SHUT "$1.con";
```

D.2 Observer Software.

observer.tas

```
TASK MODULE observer;
{Two-Link Manipulator Observer}
{Generated through modification of otwolm_scs.m}

{D.W.Roberts.}
{15th January 1992}

USE  SigDefs:RealSignal,QueueLength,OutType,FBtype,StatType;

USE  debug :dump;
```

```
ENTRYPORT stat1port : StatType;
ENTRYPORT stat2port : StatType;
ENTRYPORT FbackPort : FBtype;
ENTRYPORT startrun  : SignalType;

EXITPORT ObOut : OutType;
EXITPORT Ready : SignalType;
EXITPORT ow1Port : real;
EXITPORT ow2Port : real;

CONST
{Model Parameters}
l_1  = 0.34;
l_2 = 0.355;

m_m = 0.56;

m_1 = 0.360;
m_2  = 0.33;

k2_1 = 0.23;
k2_2 = 0.044;

ra_1 = 3.4;
ra_2 = 5.0;

b_1 = 0.03;
b_2 = 0.01;

noofpts = 500;
dt = 0.0291;


VAR
X : PACKED ARRAY [1..4] OF real;
dXe : PACKED ARRAY [1..4] OF real;

E : PACKED ARRAY [1..4,1..4] OF real;
InvE : PACKED ARRAY [1..4,1..4] OF real;
det : real;
Y : OutType;
u1,u2,u3,u4,u5,u6 : real;

ans1,ans2,ans3,
ans4,ans5,ans6,
ans7,ans8 : real;

stats1,
```

```
stats2 : StatType;

fbvals :FBtype; {Observer feed-back values}


{_____________________________________________________________}
PROCEDURE CalcInvE;

BEGIN
E[1,1] := (12*(m_2+1/3*m_1+m_m))/m_1;
E[1,2] := (6*cos(X[4])*l_1)/l_2;
E[1,3] := -(6*sin(X[4])*X[2]*l_1)/l_2;
E[1,4] := -(6*sin(X[4])*X[2]*l_1)/l_2;
E[2,1] := (6*cos(X[4])*l_2*m_2)/(l_1*m_1);
E[2,2] := 4;
E[2,3] := (6*sin(X[4])*X[1]*l_2*m_2)/(l_1*m_1);
E[3,3] := 1;
E[4,4] := 1;

det := (E[1,1]*E[2,2])-(E[1,2]*E[2,1]);

InvE[1,1] := (1/det)*E[2,2];
InvE[1,2] := (1/det)*(-1)*E[1,2];
InvE[1,3] := (1/det)*((E[1,2]*E[2,3])-(E[1,3]*E[2,2]));
InvE[1,4] := (1/det)*(-1)*(E[1,4]*E[2,2]);
InvE[2,1] := (1/det)*(-1)*E[2,1];
InvE[2,2] := (1/det)*E[1,1];
InvE[2,3] := (1/det)*((E[2,1]*E[1,3])-(E[1,1]*E[2,3]));
InvE[2,4] := (1/det)*(E[1,4]*E[2,1]);
InvE[3,1] := 0;
InvE[3,2] := 0;
InvE[3,3] := (1/det)*((E[1,1]*E[2,2])-(E[1,2]*E[2,1]));
InvE[3,4] := 0;
InvE[4,1] := 0;
InvE[4,2] := 0;
InvE[4,3] := 0;
InvE[4,4] := (1/det)*((E[1,1]*E[2,2])-(E[1,2]*E[2,1]));
END;

{_____________________________________________________________}
PROCEDURE InitVect;
{Initialise the state vector}

BEGIN
X[1] := 0;
X[2] := 0;
X[3] := 0;
X[4] := 0;
```

```
END;
{_____________________________________________________________________}
PROCEDURE SetInputVector;
{Set up the Input vector}

BEGIN
u1 := stats1.v;
u2 := stats2.v;
u3 := fbvals[1];
u4 := fbvals[2];
u5 := fbvals[3];
u6 := fbvals[4];
END;

{_____________________________________________________________________}
PROCEDURE CalcDerivVect;
{Calculates the derivative of the pseudo-state vector}

BEGIN

ans8:=-1/12*u3*l_2*l_2*m_2*ra_1*ra_2*l_1*l_1*m_1;
ans7:=1/12*u2*l_2*l_2*m_2*ra_1*k2_2*l_1*l_1*m_1+ans8;
ans6:=-1/12*u1*l_2*l_2*m_2*ra_2*k2_1*l_1*l_1*m_1+ans7;
ans5:=-X[2]*ra_1*ra_2*b_2*l_1*l_1*m_1-X[2]*ra_1*k2_2*k2_2*l_1*l_1*m_1+ans6;
ans4:=X[1]*l_2*l_2*m_2*ra_1*k2_2*k2_2+X[1]*l_2*l_2*m_2*ra_2*k2_1*k2_1+ans5;
ans3:=X[1]*l_2*l_2*m_2*ra_1*ra_2*b_1+X[1]*l_2*l_2*m_2*ra_1*ra_2*b_2+ans4;
ans2:=12*ans3;
ans1:=ans2/(l_2*l_2*m_2*ra_1*ra_2*l_1*l_1*m_1);
dXe[1]:=-ans1;

ans5:=1/12*u4*l_2*l_2*m_2*ra_2*l_1*l_1*m_1;
ans4:=1/12*u2*l_2*l_2*m_2*k2_2*l_1*l_1*m_1+ans5;
ans3:=-X[2]*ra_2*b_2*l_1*l_1*m_1-X[2]*k2_2*k2_2*l_1*l_1*m_1+ans4;
ans2:=X[1]*l_2*l_2*m_2*ra_2*b_2+X[1]*l_2*l_2*m_2*k2_2*k2_2+ans3;
ans1:=12*ans2;
dXe[2]:=ans1/(l_2*l_2*m_2*ra_2*l_1*l_1*m_1);

dXe[3]:=(12*(X[1]+1/12*u5*l_1*l_1*m_1))/(l_1*l_1*m_1);

ans4:=-1/12*u6*l_2*l_2*m_2*l_1*l_1*m_1;
ans3:=X[1]*l_2*l_2*m_2-X[2]*l_1*l_1*m_1+ans4;
ans2:=12*ans3;
ans1:=ans2/(l_2*l_2*m_2*l_1*l_1*m_1);
dXe[4]:=-ans1;

END;
{_____________________________________________________________________}
PROCEDURE InitX;
```

```
BEGIN
X[1]:=0;
X[2]:=0;
X[3]:=0;
X[4]:=0;
END;
{_____________________________________________________________}
PROCEDURE CalcOutputs;
{Calculates the Output Vector Y}

BEGIN
Y.w1 := (12*X[1])/(l_1*l_1*m_1);

ans2 := 12*(X[1]*l_2*l_2*m_2-X[2]*l_1*l_1*m_1);
ans1 := ans2/(l_2*l_2*m_2*l_1*l_1*m_1);
Y.w2 := -ans1;

Y.th1 := X[3];

Y.th2 := X[4];
END;
{_____________________________________________________________}
PROCEDURE CalcStateVec;
{Calculates the state vector from X=(Inverse of E)*chi}

VAR
n : 1..4;

BEGIN
FOR n:=1 TO 4 DO
BEGIN
ans2 := InvE[n,1]*dXe[1]+InvE[n,2]*dXe[2];
ans1 := ans2+InvE[n,3]*dXe[3]+InvE[n,4]*dXe[4];
X[n] := X[n]+(ans1*dt);
END;
END;

{_____________________________________________________________}

BEGIN {Main Program}

  WHILE true DO BEGIN

InitVect;
InitX;

SEND Signal TO Ready;
```

```
RECEIVE Signal FROM startrun;


LOOP
    SELECT
RECEIVE stats1 FROM stat1port =>
RECEIVE stats2 FROM stat2port;
RECEIVE fbvals FROM FbackPort;

SetInputVector;
CalcDerivVect;
CalcInvE;
CalcStateVec;
CalcOutputs;

SEND Y TO ObOut;
SEND Y.w1 TO ow1Port;
SEND Y.w2 TO ow2Port;


OR
TIMEOUT(2000) =>
EXIT;
    END;{select}

END;{loop}


   END;{Infinite While}

END.{Program}
```

obfb.tas

```
TASK MODULE obfb;
{Feedback Module for the Observer}
{Feedback Gain Matrix T designed by pole-placement}

{D.W.Roberts}
{16th January 1992}

USE  SigDefs:RealSignal,OutType,FBtype,StatType;

ENTRYPORT  SysOutPort : OutType;
ENTRYPORT ObOutPort : OutType;
ENTRYPORT startrun : SignalType;
```

```
EXITPORT FBPort : FBtype;
EXITPORT Ready : SignalType;

CONST
noofpts = 500;

VAR
SysOuts : OutType;
ObOuts : OutType;
D : OutType;
fbvals : FBtype;
T : PACKED ARRAY [1..4,1..3] OF real;

n : 1..4;

{_____________________________________________________________}
PROCEDURE SetT;
{Sets up the Feeed-Back gain matrix T}

BEGIN
T[1,1] := 1.1516;
T[1,2] := 0;
T[1,3] := 1.9888;

T[2,1] := 0.1995;
T[2,2] := 0;
T[2,3] := 1.3778;

T[3,1] := 1;
T[3,2] := 10;
T[3,3] := 0;

T[4,1] := -1;
T[4,2] := 0;
T[4,3] := 19.9667;
END;
{_____________________________________________________________}
PROCEDURE InitObOuts;

BEGIN
ObOuts.w1 := 0;
ObOuts.w2 := 0;
ObOuts.th1 := 0;
ObOuts.th2 := 0;

END;

{_____________________________________________________________}
```

```
BEGIN {Main Program}

SetT;

    WHILE true DO BEGIN

    SEND Signal TO Ready;
    RECEIVE Signal FROM startrun;

InitObOuts;

LOOP
   SELECT
RECEIVE SysOuts FROM SysOutPort =>

{Find differences between system and observer outputs}
D.w1 := SysOuts.w1 - ObOuts.w1;
D.th1 := SysOuts.th1 - ObOuts.th1;
D.th2 := SysOuts.th2 - ObOuts.th2;

{Calculate the feed-back values}
 FOR n := 1 to 4 DO
         fbvals[n] := T[n,1]*D.w1  + T[n,2]*D.th1 + T[n,3]*D.th2;

SEND fbvals TO FBPort;
RECEIVE ObOuts FROM ObOutPort;
    OR
TIMEOUT(2000) =>
EXIT;
   END;{select}
END;{loop}

    END;{Infinite While}

END.{Main Program}
```

io.tas

```
TASK MODULE io;

{Performs input/output handling for the observer group module}
{and logs data}

USE  SigDefs:RealSignal,OutType,StatType;
```

```
USE debug : dump;

ENTRYPORT stat1port : StatType;
ENTRYPORT stat2port : StatType;

ENTRYPORT ObOutPort : OutType;
ENTRYPORT ObReady : SignalType;
ENTRYPORT ObfbReady : SignalType;
ENTRYPORT startrun : SignalType;

EXITPORT SysOutPort : OutType;
EXITPORT Ready : SignalType;

CONST  noofpts = 500;

VAR SysOuts : OutType;
ObOuts : OutType;
ObFile : TEXT;
Output : TEXT;
stats1,
stats2 : StatType;
Counter : integer;
Store : PACKED ARRAY [1..noofpts,1..7] OF real;

{_____________________________________________________________}
PROCEDURE delay;

CONST  dtime = 1000;

VAR n : 1..dtime;

BEGIN
FOR n:=1 TO dtime DO ;

END;
{_____________________________________________________________}
PROCEDURE StoreData;

VAR
n : integer;

BEGIN
ReWrite(ObFile,'observer.dat');

FOR n:=1 TO (Counter-2) DO
BEGIN
Write(ObFile,Store[n,1],' ');
flush(ObFile);
```

```
Write(ObFile,Store[n,2],' ');
flush(ObFile);
Write(ObFile,Store[n,3],' ');
flush(ObFile);
Write(ObFile,Store[n,4],' ');
flush(ObFile);
Write(ObFile,Store[n,5],' ');
flush(ObFile);
Write(ObFile,Store[n,6],' ');
flush(ObFile);
WriteLn(ObFile,Store[n,7]);
flush(ObFile);
END;
close(ObFile);
END;
{_____________________________________________________________}


BEGIN {Main Program}

WHILE true DO BEGIN

RECEIVE Signal FROM ObReady;
RECEIVE Signal FROM ObfbReady;
WriteLn(Output,'Observer ready');
flush(Output);

LOOP
   SEND Signal TO Ready;
SELECT
RECEIVE Signal FROM startrun =>
WriteLn(Output,'observer going');
flush(Output);
EXIT;

OR
TIMEOUT(2000) =>
WriteLn(Output,'observer still ready');
flush(Output);
END;{Select}
END;{loop}

Counter := 0;

LOOP
SELECT
      RECEIVE stats1 FROM stat1port =>
RECEIVE stats2 FROM stat2port;
```

```
SysOuts.w1 := stats1.w;
SysOuts.w2 := stats2.w;
SysOuts.th1 := stats1.th;
SysOuts.th2 := stats2.th;

SEND SysOuts TO SysOutPort;

RECEIVE ObOuts FROM ObOutPort;

Counter := Counter+1;

Store[Counter,1] := time;
Store[Counter,2] := stats1.v;
Store[Counter,3] := stats2.v;
Store[Counter,4] := ObOuts.w1;
Store[Counter,5] := ObOuts.w2;
Store[Counter,6] := ObOuts.th1;
Store[Counter,7] := ObOuts.th2;

OR
TIMEOUT(2000) =>
EXIT;
END;{select}
END;{loop}

StoreData;

    END;{Infinite While}

END.{Main Program}
```

Appendix E

Observer Pole Placement Macros.

E.1 Pole Placement Macro.

getgains.m

```
%Calculates the observer feed-back gain matrix

%Input model parameters
otwolm_mpa

%Generate numerical linearised state space matrices around theta2=0
theta2=0;
otwolm;

a=inv(E)*A;

c=[C(1,:) ; C(3:4,:)];

p=40;

T=place(a',c',[-p -p -p -(p+0.00001)])';

eig(a-T*c)


T=E*T
```

E.2 M.files called by getgains.m

otwolm.m

```
 E=zeros(4,4);;
E(1,1) = (4*(3*m_2 + m_1 + 3*m_m))/m_1;
E(1,2) = (6*cos(theta2)*l_1)/l_2;
E(1,3) = - (6*sin(theta2)*0*l_1)/l_2;
E(1,4) = - (6*sin(theta2)*0*l_1)/l_2;
E(2,1) = (6*cos(theta2)*l_2*m_2)/(l_1*m_1);
E(2,2) = 4;
E(2,3) = (6*sin(theta2)*0*l_2*m_2)/(l_1*m_1);
E(3,3) = 1;
E(4,4) = 1;

A=zeros(4,4);;
A(1,1) = - (12*(ra_1*k2_2^2 + ra_2*k2_1^2))/(ra_1*ra_2*l_1^2*m_1);
A(1,2) = (12*k2_2^2)/(l_2^2*m_2*ra_2);
A(2,1) = (12*k2_2^2)/(ra_2*l_1^2*m_1);
A(2,2) = - (12*k2_2^2)/(l_2^2*m_2*ra_2);
A(3,1) = 12/(l_1^2*m_1);
A(4,1) = - 12/(l_1^2*m_1);
A(4,2) = 12/(l_2^2*m_2);

B=zeros(4,2);;
B(1,1) = k2_1/ra_1;
B(1,2) = - k2_2/ra_2;
B(2,2) = k2_2/ra_2;

C=zeros(4,4);;
C(1,1) = 12/(l_1^2*m_1);
C(2,1) = - 12/(l_1^2*m_1);
C(2,2) = 12/(l_2^2*m_2);
C(3,3) = 1;
C(4,4) = 1;

D=zeros(4,2);
```

otwolmmpa.m

```
l_1  = 0.34;
l_2 = 0.355;

m_m = 0.56;

m_1 = 0.360;
m_2  = 0.33;
```

```
k2_1 = 0.23;
k2_2 = 0.044;

ra_1 = 3.4;
ra_2 = 5;

%b_1 = 0;
%b_2 = 0;

b_1 = 0.03;
b_2 = 0.01;

global l_1 l_2 m_m m_1 m_2 k2_1 k2_2 ra_1 ra_2 b_1 b_2
```

Appendix F

Modified Model Transformation Toolbox Files.

F.1 Modified MTT Files to allow Constrained States to appear in System Output Equations.

rda2rcs

```
#! /bin/sh

     ##########################################
     ##### Model Transformation Tools #####
     ##########################################

# Bourne shell script: rda2rcs
# Differential-algebraic equations to constrained-state equations
# P.J.Gawthrop 14 June 1991, 8 Aug 1991
# Copyright (c) P.J.Gawthrop 1991.

# Remove the old log file
rm -f rda2rcs.log

# Use reduce to accomplish the transformation
reduce >rda2rcs.log << EOF
```

```
OFF Echo;
OFF Nat;
ON NERO;
ON GCD;

in "$1.rde";
in "$1.rpa_";
in "$1.rda_";

% %in "Create F and G matrices from non-states - if such there be
IF MTTNz>0 THEN
BEGIN
% Find MTTF;
matrix MTTF(MTTNx,MTTNz)$
FOR j := 1:MTTNz DO
  BEGIN
  xj := MTTdZ(j,1)$
  FOR i := 1:MTTNx DO
    MTTF(i,j) := df(MTTdX(i,1), xj, 1)$
  END;

% Find MTTG;
matrix MTTG(MTTNz,MTTNx)$
FOR j := 1:MTTNx DO
  BEGIN
  xj := MTTX(j,1)$
  FOR i := 1:MTTNz DO
    MTTG(i,j) := df(MTTZ(i,1), xj, 1)$
  END;


  %% The following gets rid of the dZs; there must be a better way.
  MTTdZ1 := 0;
  MTTdZ2 := 0;
  MTTdZ3 := 0;
  MTTdZ4 := 0;
  MTTdZ5 := 0;
  MTTdZ6 := 0;
  MTTdZ7 := 0;
  MTTdZ8 := 0;
  MTTdZ9 := 0;
  MTTdZ10 := 0;
  MTTdZ11 := 0;
  MTTdZ12 := 0;
  MTTdZ13 := 0;
  MTTdZ14 := 0;
  MTTdZ15 := 0;
  MTTdZ16 := 0;
```

```
  MTTdZ17 := 0;
  MTTdZ18 := 0;
  MTTdZ19 := 0;
END;



%%Create the .rcs file
OUT "$1.rcs";

IF MTTNx>0 THEN
BEGIN
  write "matrix MTTdXE(", MTTNx, ",1)";
END;
MTTdXE := MTTdX;

IF MTTNy>0 THEN
BEGIN
  write "matrix MTTY(", MTTNy, ",1)";
END;
IF MTTNz>=1 THEN  clear MTTdZ1$
IF MTTNz>=2 THEN  clear MTTdZ2$
IF MTTNz>=3 THEN  clear MTTdZ3$
IF MTTNz>=4 THEN  clear MTTdZ4$
IF MTTNz>=5 THEN  clear MTTdZ5$
IF MTTNz>=6 THEN  clear MTTdZ6$
IF MTTNz>=7 THEN  clear MTTdZ7$
IF MTTNz>=8 THEN  clear MTTdZ8$
IF MTTNz>=9 THEN  clear MTTdZ9$
IF MTTNz>=10 THEN  clear MTTdZ10$
IF MTTNz>=11 THEN  clear MTTdZ11$
IF MTTNz>=12 THEN  clear MTTdZ12$
IF MTTNz>=13 THEN  clear MTTdZ13$
IF MTTNz>=14 THEN  clear MTTdZ14$
IF MTTNz>=15 THEN  clear MTTdZ15$
IF MTTNz>=16 THEN  clear MTTdZ16$
IF MTTNz>=17 THEN  clear MTTdZ17$
IF MTTNz>=18 THEN  clear MTTdZ18$
IF MTTNz>=19 THEN  clear MTTdZ19$
MTTY := MTTY;

IF MTTNu>0 THEN
BEGIN
  write "matrix MTTU(", MTTNu, ",1)";
END;
MTTU := MTTU;
```

```
IF MTTNx>0 THEN
BEGIN
  matrix MTTE(MTTNx,MTTNx);
  write "matrix MTTE(", MTTNx, ",", MTTNx, ")";
  IF MTTnZ=0 THEN MTTE := MTTI\$
  IF MTTnZ>0 THEN MTTE := (MTTI - MTTF*MTTG)\$

END;
MTTE := MTTE;

write ";END;";

SHUT "$1.rcs";
quit;

EOF
```

rcs2rcz

```
#! /bin/sh

     ##########################################
     ##### Model Transformation Tools #####
     ##########################################

# Bourne shell script: rcs2rcz
# Differential-algebraic equations to constrained-state equations
# with explicit constrained state derivatives.
# P.J.Gawthrop 14 June 1991, 8 Aug 1991
# Modified from rda2rcs by D.W.Roberts, 10th March 1992
# Copyright (c) P.J.Gawthrop 1991.

# Remove the old log file
rm -f rcs2rcz.log

# Use reduce to accomplish the transformation
reduce >rcs2rcz.log << EOF

OFF Echo;
OFF Nat;
ON NERO;
ON GCD;

in "$1.rde";
in "$1.rpa_";
in "$1.rda_";
in "$1.rcs_";
```

```
IF MTTNz>0 THEN
BEGIN
% Find MTTF (dxdot/dzdot)
matrix MTTF(MTTNx,MTTNz)$
FOR j := 1:MTTNz DO
  BEGIN
  xj := MTTdZ(j,1)$
  FOR i := 1:MTTNx DO
    MTTF(i,j) := df(MTTdX(i,1), xj, 1)$
  END;


%Create dz/dx
% Find MTTG;
matrix MTTG(MTTNz,MTTNx)$
FOR j := 1:MTTNx DO
  BEGIN
  xj := MTTX(j,1)$
  FOR i := 1:MTTNz DO
    MTTG(i,j) := df(MTTZ(i,1), xj, 1)$
  END;

%%Create dz/du
IF MTTNu>0 THEN
BEGIN
matrix MTTH(MTTNz,MTTNu)$
 FOR j := 1:MTTNu DO
BEGIN
uj := MTTU(j,1)$
FOR i := 1:MTTNz DO
MTTH(i,j) := df(MTTZ(i,1),uj,1)$
END;

%%Create a matrix of input derivatives

matrix MTTDU(MTTNU,1)$

IF MTTNU>=1 THEN MTTDU(1,1) := MTTDU1$
IF MTTNU>=2 THEN MTTDU(2,1) := MTTDU2$
IF MTTNU>=3 THEN MTTDU(3,1) := MTTDU3$
IF MTTNU>=4 THEN MTTDU(4,1) := MTTDU4$
IF MTTNU>=5 THEN MTTDU(5,1) := MTTDU5$
IF MTTNU>=6 THEN MTTDU(6,1) := MTTDU6$
IF MTTNU>=7 THEN MTTDU(7,1) := MTTDU7$
IF MTTNU>=8 THEN MTTDU(8,1) := MTTDU8$
IF MTTNU>=9 THEN MTTDU(9,1) := MTTDU9$
```

```
IF MTTNU>=10 THEN MTTDU(10,1) := MTTDU10$
IF MTTNU>=11 THEN MTTDU(11,1) := MTTDU11$
IF MTTNU>=12 THEN MTTDU(12,1) := MTTDU12$
IF MTTNU>=13 THEN MTTDU(13,1) := MTTDU13$
IF MTTNU>=14 THEN MTTDU(14,1) := MTTDU14$
IF MTTNU>=15 THEN MTTDU(15,1) := MTTDU15$

END;

%Must first create dx/dt, state derivatives without zdots
MTTDX := (MTTE^-1)*(MTTdXE+(MTTF*MTTH*MTTDU));



%%Create zdot in terms of states and derivatives of inputs
FOR i := 1:MTTNz DO
BEGIN
MTTDZ(i,1) := FOR j:=1:MTTNx SUM MTTG(i,j)*MTTDX(j,1)$

IF MTTNu>0 THEN
   FOR j:=1:MTTNu DO
MTTDZ(i,1):=MTTDZ(i,1)+(MTTH(i,j)*MTTDU(j,1))$
END;

END;



%%Create the .rcz file
OUT "$1.rcz";

IF MTTNx>0 THEN
BEGIN
  write "matrix MTTdX(", MTTNx, ",1)";
END;
MTTdX := MTTdX;

IF MTTNz>0 THEN
BEGIN
  write "matrix MTTdZ(", MTTNz, ",1)";
END;
IF MTTNz>=1 THEN  MTTdZ1 := MTTDZ(1,1)$
IF MTTNz>=2 THEN  MTTdZ2 := MTTDZ(2,1)$
IF MTTNz>=3 THEN  MTTdZ3 := MTTDZ(3,1)$
IF MTTNz>=4 THEN  MTTdZ4 := MTTDZ(4,1)$
IF MTTNz>=5 THEN  MTTdZ5 := MTTDZ(5,1)$
IF MTTNz>=6 THEN  MTTdZ6 := MTTDZ(6,1)$
IF MTTNz>=7 THEN  MTTdZ7 := MTTDZ(7,1)$
```

```
IF MTTNz>=8 THEN  MTTdZ8 := MTTDZ(8,1)$
IF MTTNz>=9 THEN  MTTdZ9 := MTTDZ(9,1)$
IF MTTNz>=10 THEN  MTTdZ10 := MTTDZ(10,1)$
IF MTTNz>=11 THEN  MTTdZ11 := MTTDZ(11,1)$
IF MTTNz>=12 THEN  MTTdZ12 := MTTDZ(12,1)$
IF MTTNz>=13 THEN  MTTdZ13 := MTTDZ(13,1)$
IF MTTNz>=14 THEN  MTTdZ14 := MTTDZ(14,1)$
IF MTTNz>=15 THEN  MTTdZ15 := MTTDZ(15,1)$
IF MTTNz>=16 THEN  MTTdZ16 := MTTDZ(16,1)$
IF MTTNz>=17 THEN  MTTdZ17 := MTTDZ(17,1)$
IF MTTNz>=18 THEN  MTTdZ18 := MTTDZ(18,1)$
IF MTTNz>=19 THEN  MTTdZ19 := MTTDZ(19,1)$

MTTdZ := MTTdZ;

IF MTTNy>0 THEN
BEGIN
  write "matrix MTTY(", MTTNy, ",1)";
END;

MTTY := MTTY;

IF MTTNu>0 THEN
BEGIN
  write "matrix MTTU(", MTTNu, ",1)";
END;
MTTU := MTTU;


IF MTTNx>0 THEN
BEGIN
  matrix MTTE(MTTNx,MTTNx);
  write "matrix MTTE(", MTTNx, ",", MTTNx, ")";
  END;
MTTE := MTTE;

write ";END;";

SHUT "$1.rcz";
quit;

EOF
```

rcz2scz

```
#! /bin/sh
```

```
#########################################
##### Model Transformation Tools #####
#########################################

# Bourne shell script: rcz2scz
# Reduce constrained-state to simulab  constrained-state equations
# P.J.Gawthrop 14 June 1991
# Copyright (c) P.J.Gawthrop 1991.

# Modified 10th March 1992
# from rcs2scs
# by D.W.Roberts
# to explicitly calculate constrained state derivatives

# Output must be filtered by for2mat.
# Simulab does not handle DAE's - so only ODE bits implemented

# Remove the old log file
rm -f rcz2scz.log

# Use reduce to accomplish the transformation
reduce >rcz2scz.log << EOF

%Read the reduce definitions file
in "$1.rde";

%Read the reduce  constrained-state equations   file
in "$1.rcz_";

ON BigFloat, NumVal;
PRECISION 16; %Compatible with Matlab
OFF Nat;

ON NERO;        % Suppress zero elements

%Fortran switches - one line expressions
ON fort;
cardno!* := 1;
fortwidth!* := 100;

OFF period;
OUT "$1.scz";
%Headings - Simulab style
write "function [sys,x0] = $1_scz(t,x,u,flag);";
write "%Robot equations for system $1";
write "%File $1_scz.m";
write "%Generated by MTT";
```

```
write "if (abs(flag) == 1) | (abs(flag) == 3)";
write "% Set up the State variables";
FOR i := 1:MTTNx DO
BEGIN
  write "MTTx", i, " = x(", i, ");";
END;
write "% Set up the Input variables";
IF MTTNu>0 THEN
FOR i := 1:MTTNu DO
BEGIN
  write "MTTu", i, " = u(", i, ");";
END;
write "end;";


write "if abs(flag) == 1 %State derivative";

write "MTTE = zeros(", MTTNx, ",", MTTNx, ");";
MTTE := MTTE;
MTTdxE := MTTdxE;
write "sys = inv(MTTE)*MTTdxE;";

write "elseif abs(flag) == 3 %Outputs";
MTTdZ := MTTdZ;
MTTy := MTTy;
write "sys = MTTy;";

write "elseif abs(flag) == 0 %Structure";
write " sys = [", MTTNx, ",0,", MTTNy, ",", MTTNu, ",0,0]";
write " x0  = zeros(", MTTNx, ",1)";
write "end;";
SHUT "$1.scz";
```

rcz2tcz

```
#! /bin/sh

     ########################################
     ##### Model Transformation Tools #####
     ########################################

# Bourne shell script: rcz2tcz
# Reduce constrained-state to LaTex constrained-state equations.
# P.J.Gawthrop 10th May 199, 8th August 1991
```

```
# Copyright (c) P.J.Gawthrop, 1991.

# Modified 10th March 1992
# from rcs2tcs
# by D.W.Roberts
# to handle explicit constrained state derivatives

# Remove the old log file
rm -f rcz2tcz.log

# Use reduce to accomplish the transformation
reduce >rcz2tcz.log << EOF

%Read the definitions file
in "$1.rde";

%Read the constrained-state equations file
in "$1.rcz_";

OFF Echo;
OFF Nat;
OFF Exp; ON GCD;
%%%ON BigFloat,numval;

%Change some names - rather yuccy
MTTx1 := MTTx!_1;
MTTx2 := MTTx!_2;
MTTx3 := MTTx!_3;
MTTx4 := MTTx!_4;
MTTx5 := MTTx!_5;
MTTx6 := MTTx!_6;
MTTx7 := MTTx!_7;
MTTx8 := MTTx!_8;
MTTx9 := MTTx!_9;

MTTu1 := MTTu!_1;
MTTu2 := MTTu!_2;
MTTu3 := MTTu!_3;
MTTu4 := MTTu!_4;
MTTu5 := MTTu!_5;
MTTu6 := MTTu!_6;
MTTu7 := MTTu!_7;
MTTu8 := MTTu!_8;
MTTu9 := MTTu!_9;

MTTz1 := MTTz!_1;
MTTz2 := MTTz!_2;
```

```
MTTz3 := MTTz!_3;
MTTz4 := MTTz!_4;
MTTz5 := MTTz!_5;
MTTz6 := MTTz!_6;
MTTz7 := MTTz!_7;
MTTz8 := MTTz!_8;
MTTz9 := MTTz!_9;

MTTdu1 := MTTdu!_1;
MTTdu2 := MTTdu!_2;
MTTdu3 := MTTdu!_3;
MTTdu4 := MTTdu!_4;
MTTdu5 := MTTdu!_5;
MTTdu6 := MTTdu!_6;
MTTdu7 := MTTdu!_7;
MTTdu8 := MTTdu!_8;
MTTdu9 := MTTdu!_9;

MTTy1 := MTTy!_1;
MTTy2 := MTTy!_2;
MTTy3 := MTTy!_3;
MTTy4 := MTTy!_4;
MTTy5 := MTTy!_5;
MTTy6 := MTTy!_6;
MTTy7 := MTTy!_7;
MTTy8 := MTTy!_8;
MTTy9 := MTTy!_9;

OUT "$1.tcz";

%Write out the  constrained-state equations.

write "%File: $1.tcz";

write "%constrained-state equations";
IF MTTNx>0 THEN
FOR Row := 1:MTTNx DO
BEGIN
write"\begin{equation} \label{eq_$1_X_c", Row, "}";
write "\dot MTTX_{", Row, "} = ";
write "{";
write MTTdX(Row,1);
write "}";
write"\end{equation}";
END;
```

```
IF MTTNyz>0 THEN
FOR Row := 1:MTTNyz DO
BEGIN
write"\begin{equation} \label{eq_$1_yz_c", Row, "}";
write "0 = ";
write "{";
write MTTyz(Row,1);
write "}";
write"\end{equation}";
END;

IF MTTNz>0 THEN
FOR Row := 1:MTTNz DO
BEGIN
write"\begin{equation} \label{eq_$1_z_c", Row, "}";
write "MTTdZ_{", Row, "} = ";
write "{";
write MTTdZ(Row,1);
write "}";
write"\end{equation}";
END;

IF MTTNy>0 THEN
FOR Row := 1:MTTNy DO
BEGIN
write"\begin{equation} \label{eq_$1_y_c", Row, "}";
write "MTTy_{", Row, "} = ";
write "{";
write MTTy(Row,1);
write "}";
write"\end{equation}";
END;


write "%  - E matrix";
write "\begin{eqnarray} \label{eq_$1_Ea}";
FOR Row := 1:MTTNx DO
BEGIN
  FOR Col := 1:MTTNx DO IF MTTE(Row,Col) NEQ 0 THEN
  BEGIN
      Write "MTTE(", Row, ",", Col, ") &=& {", MTTE(Row,Col), "}\cr";
  END;
END;
write "\end{eqnarray}";

SHUT "$1.tcz";
quit;
EOF
```

Bibliography

[1] J. J. Craig, *Introduction to Robotics: Mechanics and Control.* Addison-Wesley, 1986.

[2] C. H. An, C. G. Atkeson, and J. M. Hollerbach, *Model-based Control of Robot Manipulators.* The MIT Press, 1988.

[3] R. C. Paul, "Modeling trajectory calculation and servoing of a computer controlled arm," A.I. Memo. 177, Stanford Artificial Intelligence Lab., Stanford Univ., Stanford, CA, 1972.

[4] J. Y. S. Luh, "Conventional controller design for industrial robots -a tutorial," *IEEE Trans, on Systems, Man and Cybernetics*, vol. SMC-13, no. 3, pp. 298–316, 1983.

[5] A. Liegeois, A. Fournier, and M. Aldon, "Model reference control of high velocity industrial robots," in *Joint Automatic Control Conference*, (San Francisco, CA), 1980.

[6] C. H. An, C. G. Atkeson, and J. M. Hollerbach, "Experimental determination of the effect of feedforward control on trajectory tracking errors," in *Proceedings of the 3rd IEEE International Conference on Robotics and Automation, San Francisco,CA*, pp. 55–60, April 1986.

[7] H. Asada, T. Kanade, and I. Takeyama, "Control of a direct-drive arm," *Trans of the ASME Journal of Dynamic Systems, Measurement and Control*, vol. 105, pp. 136–142, 1983.

[8] B. R. Markiewicz, "Analysis of the computed torque drive method and comparison with conventional position servo for a computer controlled manipulator," Tech. Mem. 33-601, Jet Propulsion Lab., Mar 1973.

[9] M. H. Raibert and B. K. P. Horn, "Manipulator control using the configuration space method," *Industrial Robot*, vol. 5, pp. 69–73, 1978.

[10] P. K. Khosla and T. Kanade, "Real-time implementation and evaluation of computed-torque scheme," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 245–253, 1989.

[11] J. J. Uicker, *On the Dynamic Analysis os Spatial Linkages using 4*4 Matrices.* Ph.D. thesis, NorthWestern, 1965.

[12] R. A. Lewis, "Autonomous manipulation on a robot: Summary of manipulator software functions," Tech. Mem. 33-679, Jet Propulsion Lab., Mar 1974.

[13] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "Resolved acceleration control of mechanical manipulators," *IEEE Trans, on Automatic Control*, vol. 25, no. 3, pp. 69–76, 1980.

[14] A. K. Bejczy, "Robot arm dynamics and control," Tech. Mem. 33-669, Jet Propulsion Lab., Feb 1974.

[15] H. M. Paynter, *Analysis and design of engineering systems.* Cambridge, Mass.: MIT Press, 1961.

[16] R. P. Anex, Jr and M. Hubbard, "Modelling and adaptive control of a mechanical manipulator," *journal of Dynamic Systems, Measurement and Control*, vol. 106, Sept. 1984.

[17] P. J. Gawthrop, "Bond graphs: A representation for mechatronic systems," *Mechatronics*, vol. 1, pp. 127–156, April 1991.

[18] P. J. Gawthrop, "Manipulator dynamics: A bond graph approach. part 1: Two dimensional rigid revolute motion," Control Group Research Report R-90/8, Glasgow University, Dept. of Mechanical Engineering, 1990.

[19] P. J. Gawthrop, "Manipulator dynamics: A bond graph approach. part 2: Actuator modelling," Control Group Research Report R-90/9, Glasgow University, Dept. of Mechanical Engineering, 1990.

[20] P. J. Gawthrop, "MTT: Model transformation tools. a bond graph toolbox," Control Group Research Report R-90/13, Glasgow University, Dept. of Mechanical Engineering, 1990.

[21] C. Checkoway and K. Kirk, *SIMULAB Users Guide.* Natick,Massachusetts 01760: The MathWorks,Inc., 1990.

[22] D. C. Karnopp, "Bond graphs in control: Physical state variables and observers," *J. Franklin Institute*, vol. 308, no. 3, pp. 221–234, 1979.

[23] J. M. Hollerbach, "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamic formulation complexity," *IEEE Trans. on Systems, Man And Cybernetics*, vol. 10, 1980.

[24] C. S. G. Lee, B. H. Lee, and R. Nigam, "Development of the generalized d'alembert equations of motion for mechanical manipulators," in *Proceedings of the 22nd Conference on Decision and Control*, December 1983.

[25] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control.* Cambridge, Mass.: The MIT Press series in artificial intelligence, 1981.

[26] C. Canudas de Wit, K. J. Åström, and N. Fixot, "Computed torque control via a non-linear observer," *International Journal of Adaptive Control and Signal Processing*, vol. 4, no. 6, pp. 443–452, 1990.

[27] C. An, C. Atkeson, J. Griffiths, and J. Hollerbach, "Experimental evaluation of feedforward and computed torque control," in *IEEE Conf. Robotics and Automation*, (Raleigh, NC), pp. 165–168, 1987.

[28] P. K. Khosla, *Real Time Control and Identification of a Direct Drive Manipulator.* Ph.D. thesis, Carnegie-Mellon University, 1986.

[29] M. B. Leahy, K. P. Valavanis, and G. N. Saridis, "The effects of dynamic models on robot control," in *Proceedings of the 3rd IEEE International Conference on Robotics and Automation, San Francisco,CA*, pp. 49–54, April 1986.

[30] P. J. Gawthrop, N. A. Marrison, and L. Smith, "MTT: A bond graph toolbox," in *Proceedings of the 5th IFAC/IMACS Symposium on Computer-aided Design of Control Systems:CADCS91, Swansea, Wales*, pp. 274–279, 1991.

[31] P. J. Gawthrop, "Design of mechatronic systems using bond graphs," in *IMechE Conference on Mechatronics, Dundee, Scotland*, 1992.

[32] R. C. Rosenberg and D. C. Karnopp, *Introduction to Physical System Dynamics.* McGraw-Hill, 1983.

[33] P. E. Wellstead, *Introduction to Physical System Modelling.* Academic Press, 1979.

[34] M. J. L. Tiernego and A. M. Bos, "Modelling the kinematics and dynamics of mechanical systems with multibond graphs," *J. Franklin Institute*, vol. 319, no. 1/2, pp. 37–50, 1985.

[35] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man-Mach. Syst.*, vol. 10, pp. 47–53, 1963.

[36] Digital Signal Processing Group, *MPV901 Series Operating Manual.* Burr-Brown Ltd, Livingston, Scotland, 1985.

[37] N. Dulay, J. Magee, and K. Twidle, "Conic under unix user guide - version 3.0," conic programmers manual, Dept. of Computing, Imperial College, Jul 1988.

[38] N. Dulay, J. Kramer, J. Magee, M. Sloman, and K. Twidle, "The conic programming language - version 3.0," conic programmers manual, Dept. of Computing, Imperial College, Jul 1988.

[39] N. Dulay, J. Kramer, J. Magee, M. Sloman, and K. Twidle, "The conic configuration language - version 3.0," conic programmers manual, Dept. of Computing, Imperial College, Jul 1988.

[40] C. Canudas de Wit, *Adaptive control for partially known systems.* Amsterdam: Elsevier, 1988.

[41] X. Li, "Modelling and validation of robot manipulators," M.Sc. Thesis, Glasgow University, Dept. of Mechanical Engineering, 1989.

[42] P. J. Gawthrop, "Automatic tuning of a motion controller," Control Engineering Report 88.5, Glasgow University, Dept. of Mechanical Engineering, 1988.

[43] J. Swevers, "An introduction to system identification and its application in practice. computer controlled motion and robotics," in *Lecture notes of the 1990 Integrated European Course in Mechatronics.*, (Leuven.), 1990.

[44] C. Moler, J. Little, and S. Bangert, *MATLAB user's Guide.* Mathworks Inc., 1992.

[45] C. Canudas de Wit, N. Fixot, and K. J. Åström, "Trajectory tracking in robot manipulators via nonlinear state estimated feedback," *IEEE Transactions on Robot. Automat.*, vol. 8, pp. 138–144, Feb 1992.

[46] C. Canudas de Wit and N. Fixot, "Robot control via robust state estimated feedback," *IEEE Transactions on Automatic Control*, vol. 36, no. 12, pp. 1497–1501, 1991.

[47] C. Canudas de Wit and N. Fixot, "Adaptive control of robot manipulators via velocity estimated feedback," *IEEE Transactions on Automatic Control*, vol. 37, no. 8, pp. 1234–1237, 1992.

[48] S. Salcudean, "A globally convergent angular velocity observer for rigid body motion," *IEEE Transactions on Automatic Control*, vol. 36, pp. 1493–1497, Dec 1991.

[49] S. Nicosia and P. Tomei, "Robot control by using only joint position measurements," *IEEE Transactions on Automatic Control*, vol. 35, no. 9, pp. 1058–1061, 1990.

[50] S. Nicosia, A. Tornambe, and P. Valigi, "Experimental validation of asymptotic observers for robotic manipulators," in *IEEE. Conf. Robot. Automat.*, (Cincinnati,OH.), pp. 1423–1430, May 1990.

[51] C. Canudas de Wit and J. Slotine, "Sliding observers for robot manipulators," in *IFAC Symposium on Non-Linear Control Systems Design*, (Capri,Italy.), pp. 142–147, 1989.

[52] S. Nicosia, P. Tomei, and A. Tornambe, "Dynamic modelling of flexible robot manipulators," in *Proceedings of the 3rd IEEE International Conference on Robotics and Automation, San Francisco,CA*, pp. 365–372, April 1986.

[53] S. Nicosia, P. Tomei, and A. Tornambe, "A non-linear observer for elastic robots," *IEEE Journal on Robotics and Automation*, vol. RA-4, pp. 45–52, 1988.

[54] S. Nicosia and P. Tornambe, "High-gain observers in the state and parameter estimation of robots having elastic joints," *Syst. Control Letters*, vol. 13, pp. 331–337, 1989.

[55] S. Nicosia and P. Tornambe, "A method for the state estimation of elastic joint robots by global position measurements," *Int. Journal of Adaptive Control and Signal Processing*, vol. 4, pp. 475–486, 1990.

[56] P. Tomei, "An observer for flexible joint robots," *IEEE Transactions on Automatic Control*, vol. AC-35, pp. 739–743, 1990.

[57] J. Hung, A. Bortoff, and M. Spong, "Observer-based feedback linearization of flexible joint manipulators," *IEEE Int. Conf. on Robotics and Automation*, pp. 2078–2082, May 1989.

[58] P. J. Gawthrop, "Model-based observer control: an introduction," Control Group Research Report R-91/9, Glasgow University, Dept. of Mechanical Engineering, 1991.

[59] G. F. Franklin, I. Powell, and A. Emami-Naeni, *Feedback Control of Dynamic Systems.* Addison-Wesley, 1986.

[60] J. M. Maciejowski, *Multivariable Feedback Design.* Addison-Wesley, 1989.