Self-tuning controller for farm tractor guidance

Kwang-Mo Noh
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd
Part of the Agriculture Commons, and the Bioresource and Agricultural Engineering Commons

Recommended Citation
https://lib.dr.iastate.edu/rtd/9874

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Self-tuning controller for farm tractor guidance

Noh, Kwang-Mo, Ph.D.

Iowa State University, 1990
Self-tuning controller for farm tractor guidance

by

Kwang-Mo Noh

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Department: Agricultural Engineering
Major: Agricultural Engineering
(Agricultural Power and Machinery)

Approved:
Signature was redacted for privacy.

In Charge of Major Work
Signature was redacted for privacy.

By the Major Department
Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1990
# TABLE OF CONTENTS

**GENERAL INTRODUCTION** ........................................... 1  
  Dissertation Format ........................................... 4  
  Objectives .................................................. 5

**PART I. STEERING CONTROLLERS FOR A FARM TRACTOR** .................. 7  
**INTRODUCTION** .................................................. 8  
**POSITION SENSING SYSTEM** ...................................... 10  
  Mechanical Systems ............................................ 10  
  Leader-Cable Systems ........................................... 11  
  Ultrasonic Systems ............................................. 13  
  Photoelectric Systems ......................................... 15  
  Navigational Systems .......................................... 16  
**STEERING CONTROLLER DESIGN** .................................... 22  
  Control Variables of a Guidance System ....................... 22  
  PID Controller ............................................... 26  
  Optimal Controller ............................................ 27  
  Adaptive Controller .......................................... 29
LIST OF TABLES

Table 1.1: Characteristics of position sensing systems .......... 21
Table 2.1: Description of the modeled tractor ............... 68
Table 3.1: Maximum, mean, standard deviation, and RMS values for absolute position errors of the center of tractor front wheel at speed 18 km/h .................. 145
Table 3.2: Estimated parameters, position errors, and steering angles at steady-state circular path with a radius of 36 m when one-step-ahead prediction model with unknown future paths was used ............... 152
Table 3.3: The closed-loop characteristics of the self-tuning steering controller at steady-state circular path with a radius of 36 m ... 152
Table 3.4: Characteristics of the gain $K_0$, the initial value $\hat{b}_1$, and the position error at test speeds ............... 155
LIST OF FIGURES

Figure 1.1: Basic configuration of an adaptive system ..................... 31
Figure 1.2: Adaptive preview control model of driver-car system. $y, \dot{y}, \ddot{y}$ are position, velocity, and acceleration of the vehicle, $\beta$ is slip angle, $\gamma$ is yaw rate, $\delta$ is steering angle, and $R$ is turning radius 32
Figure 2.1: A single-chain rigid body ........................................ 49
Figure 2.2: Multi-chain rigid bodies ......................................... 49
Figure 2.3: Revolute joint .................................................. 54
Figure 2.4: Translational joint .............................................. 54
Figure 2.5: Spherical joint .................................................. 55
Figure 2.6: Kinematic chain with a closed-loop subsystem .............. 55
Figure 2.7: Schematic diagram of a two-wheel drive tractor .......... 63
Figure 2.8: Graphical representation of Figure 2.7 ....................... 63
Figure 2.9: Steering angle input of the tractor ......................... 66
Figure 2.10: The lateral position of the tractor ......................... 66
Figure 2.11: The lateral acceleration of the tractor .................... 67
Figure 2.12: The vertical acceleration of the tractor .................. 67
Figure A.1: Revolute-translational joint ................................. 76
Figure A.2: Universal joint ............................................ 76
Figure 3.1: Linearized tractor model at constant speed. $\alpha$, $\beta$, $\psi$, and $\delta$ are tire-slip, side-slip, yaw, and steering angles, $l$ is distance between tractor c.g. and wheel center, $S$ and $U$ are lateral and longitudinal forces, and $v$ is tractor velocity.

Figure 3.2: Block diagram of self-tuning tractor steering controller. $e$ is disturbances, $y$ and $y_r$ are actual and reference paths, $y_e$ and $y_m$ are path error and amplified error signal.

Figure 3.3: A composite desired path to test self-tuning steering control algorithm.

Figure 3.4: Position error of the center of front wheel for three estimation methods.

Figure 3.5: Parameter estimates of the model predicting path error without one-step-ahead desired path.

Figure 3.6: Parameter estimates of the model predicting path error with one-step-ahead desired path.

Figure 3.7: Variable forgetting factor in response to path changes in the prediction model with one-step-ahead desired path.

Figure 3.8: Trace of covariance matrix $D$ in response to path changes in the prediction model with one-step-ahead desired path.

Figure 3.9: Steering angle command from the controller that predicts path errors without one-step-ahead desired path.

Figure 3.10: Steering angle command from the controller that predicts path error with one-step-ahead desired path.
Figure 3.11: Position error of the center of front wheel in the model having normal random disturbance with zero mean and variance 0.02

Figure 3.12: Steering angle command from the controller of the system subject to normal random disturbance

Figure 3.13: Block diagram for the closed-loop system of self-tuning steering controller at steady state. $y$ and $y_r$ are actual and reference paths, $y_e$ is path error, and $\delta$ is steering angle
GENERAL INTRODUCTION

Recent developments in the electronics industry and current trends in agricultural equipment indicate that automatic guidance of agricultural machine systems is feasible and should prove a desirable option on tomorrow's field machines. Automatic guidance systems should not only greatly reduce fatigue but also result in more efficient field operation of the machine. The operator can concentrate on maintaining peak implement performance rather than on guiding the machine. Applications include machine guidance through hazardous areas, material distribution along preset routes, and most other operations associated with agricultural production (Smith and Schafer, 1981).

Ideally automatic guidance systems should be reliable, flexible, versatile, easy to maintain, simple to operate, and reasonably priced. A reliable system must have sufficient safety devices to assure the prevention of damage and injury resulting from malfunctions. Capability is needed for quick conversion from automatic control to manual operation and vice versa. Sufficient versatility would permit the use of various implements which require the tractor to be navigated through different field patterns.

The purpose of automatic guidance is to cause a vehicle to follow a desired path, and the vehicle must do so when various disturbances, such as implement forces and/or surface irregularities, are present. A closed-loop system for such control must
possess three essential components: 1) a reference system to produce measurable quantities which can be sensed by a vehicle so that its present and future state can be accurately determined, 2) auxiliary sensors mounted on the vehicle which measure the signals necessary for determining the vehicle's state, and 3) a steering control system which operates on the sensed signals so as to maintain the vehicle in a desired state.

First, selection of the reference system depends upon the type of sensor for detecting vehicle position. According to the position of the reference system, it is generally divided into global and local position sensing systems. Local position sensing systems use mechanical, leader cable, ultrasonic, and photo-electric sensors. Gerrish and Surbrook (1984) researched vision sensors for tractor guidance systems. Simple mechanical sensors have been commercially used for tractor or implement guidance systems (Vogt, 1990; Johnson, 1988 and 1989). Some of these systems have been developed for ridge farming. Although these systems are simple to operate and reasonably priced, they do not always sense ridge or crop row position correctly and often lack sufficient reliability and versatility.

To make guidance systems completely reliable and versatile, the guidance function should not be limited to particular field operations or row pattern, and the guidance system should steer the tractor accurately to allow for repeating the established path with an allowable tracking error. Such a system requires a global position sensing system independent both of traditional machine operation and of field-installed guidance directrix such as furrows, standing crops, or buried wires (Smith and Schafer, 1981). The global position sensing system must determine precise position and orientation of the tractor. This capability is essential for achieving
desired results. This system also can help visualizing soil and crop variation with aid of photo-electric sensors.

Recently the global position sensing system using navigational technology has been researched and applied to control a vehicle in field conditions (Shmulevich et al., 1987; Choi et al., 1989). Because the navigational system uses non-contact scanning methods having high speed response and long measurement range, it can be a reliable reference system for controlling vehicle motion. However, navigational systems usually do not have tolerable control errors under agricultural field conditions.

Second, auxiliary sensors are necessary for determining or predicting the vehicle motion. In general, the parameters include position, velocity, yaw and side-slip. Because some parameters are dependent and environments are dynamic with respect to sensors, there is no efficient and economical method to obtain these quantities. Therefore, existing guidance systems usually involve the acquisition and use of only limited information.

Finally, the steering control algorithm plays a critical role in the automatic guidance system. To design a stable steering controller, theoretical and experimental research (Ellis, 1969; Cormier and Fenton, 1980; Rehkugler, 1982; Furukawa and Nakaya, 1985; Ge, 1987) has been done to determine design parameters affecting the steering control system. Considering current sensor technology and the presence of unexpected disturbances, the steering control algorithm should be robust enough to reduce and compensate for measurement errors.
Dissertation Format

This dissertation consists of three parts. Each part was written as a separate paper. The candidate conducted the research, and authored the papers under the supervision of his major professor, Dr. Donald C. Erbach, and the assistance of Drs. Richard J. Smith and Stephen J. Marley. Part I (Steering Controllers for a Farm Tractor) was presented as ASAE paper number MC90-103 in March at 1990 Mid-Central Conference of the American Society of Agricultural Engineers in St. Joseph, Missouri. Part II (Semi-Recursive Formulation of Mechanical Systems) is being submitted for publication in the Transactions of the American Society of Agricultural Engineers. Part III (Self-Tuning Steering Controller Design for Farm Tractor Guidance) will be submitted for publication in the Transactions of the American Society of Agricultural Engineers.

Part I reports on the review of automatic steering controllers used in both industrial and agricultural vehicles. Available position-sensing techniques are described in terms of the type of positioning systems, and steering control algorithms are presented including design parameters affecting steering guidance systems.

Part II reports on the development of analytical techniques for semi-recursive formulation, which is used to develop a tractor dynamic simulator and to verify the adaptive steering control algorithm developed in Part III. Relative coordinate kinematics are developed by using a variational vector approach, and typical joints are formulated to systematically assemble equations of motion. Cut-constraint Jacobians are developed for solving closed-loop mechanisms, and equations of motion are assembled in the differential-algebraic matrix form. A numerical example of a tractor mechanism illustrates how to formulate equations of motion and verifies the
algorithm performance by simulating a lane change maneuver.

Part III reports on the development of self-tuning steering controller for a tractor steering guidance system. Two degrees-of-freedom model of a farm tractor is developed in discrete-time space, and a recursive least-squares method is modified to estimate the parameters of the 2-DOF tractor model. A variable forgetting factor is implemented to cope with time-varying nonlinear systems, and its algorithm is developed to make the system stable under normal operation or sudden changes of the system by external disturbances. A self-tuning regulator is modified to minimize the variations in tractor position and yaw angle with respect to the desired ones. The algorithm is analyzed and verified by the tractor dynamic simulator.

Objectives

The goal of this study is to develop an adaptive steering control algorithm for controlling tractor path within ± 5 cm of the desired path.

The major objectives are:

1. To review position sensing systems which have been used for navigational vehicle guidance.

2. To investigate steering controllers for both industrial and agricultural vehicle guidance and to study feasibility on a navigational tractor guidance system in an agricultural environment.

3. To develop a semi-recursive dynamic algorithm, based on the variational vector approach, that uses relative generalized coordinates in Cartesian space.
4. To evaluate the dynamic algorithm developed by modeling an agricultural tractor.

5. To determine suitable model structure and recursive parameter estimation algorithm for a tractor guidance system.

6. To develop a self-tuning regulator which minimizes the variations in tractor position and yaw angle with respect to desired ones.

7. To analyze and verify the self-tuning steering control algorithm developed by using the tractor dynamic simulator.
PART I.

STEERING CONTROLLERS FOR A FARM TRACTOR
INTRODUCTION

During the past 30 years there has been increased concern about soil compaction. Loss of crop yields, run-off of water, erosion of soil, and excessive costs of subsoiling or deep tillage are primary concerns. Studies (Gaultney et al., 1982; Erbach et al., 1988) in the Midwest have shown that soil compaction with heavy loads can have a detrimental effect on crop yields.

Optimum soil conditions for tractive efficiency and for growing plants are entirely different. A cropping system called "controlled traffic" was proposed which permanently separated the traffic lanes and the cropping area. Controlled traffic along permanent traffic lanes lowers the need for primary tillage and improves yields. Cooper et al. (1969) reported progress in the development of a traffic control system, and their results showed that cotton yields were increased 15 to 37 percent.

However, the traffic paths must be controlled year after year in exactly the same location, and the path width should be minimized to reduce soil unsuitable for crop growth. An automatic guidance system that can guide a tractor and implement system with respect to a global reference frame, could make it possible to keep the same traffic paths every year.

An automatic guidance system satisfying this criterion should consist of two parts; the navigational system and the steering control algorithm. The navigational
system can use radio signals, lasers, and optoelectronics to determine the position of moving vehicles. Although they have been successfully used in manufacturing environments, much research and development are needed to develop or adapt position sensing systems for the agricultural environment.

Steering control algorithms in a navigational vehicle guidance system are necessary to minimize position errors. Proportional-integral-derivative (PID) control is the most used controller type in industry. However, its use is so diversified that the control engineer must tune the PID values according to specific needs (Kaya and Scheib, 1988). Studies have guided industry by providing quantitative data for tuning PID controllers for the given process, operational conditions, and performance criteria. Unfortunately, operational conditions in agricultural environments include variable soil conditions, inconsistent field geometries, and different types of tractors and implements. Therefore, an improved control method, adaptable to various operational conditions, is needed for accurate steering guidance.

The objectives of this part are: 1) to review position sensing systems which have been used for vehicle guidance, and 2) to investigate steering controllers used for both industrial and agricultural vehicle guidance, and 3) to evaluate feasibility of navigational tractor guidance systems in agricultural environments.
POSITION SENSING SYSTEM

The major part of an automatic guidance system is the sensor to detect the position and orientation of the vehicle to be controlled. This position can be absolute with respect to fixed references or relative with respect to objects sensed. The sensor must be capable of accurately sensing necessary information and operating under a harsh environment. Position sensing systems used in vehicle guidance can be classified into mechanical, leader-cable, ultrasonic, photoelectric, and navigational systems, according to characteristics of the position sensor.

Mechanical Systems

The first attempts at mechanical sensing systems were to develop contact-type crop sensors. Richey (1959) and Liljedahl and Strait (1962) designed a system with mechanical feelers mounted on the tractor to detect a crop row. The feelers actuated microswitches that controlled hydraulic valves or relays of an electric motor which adjusted the angle of tractor steering wheels.

A hydrostatic self-propelled vehicle with an automatic steering system was developed by Parish and Goering (1971). The crop sensor consisted of two microswitches operated by the upward pressure of the hay. There was an allowable dead band between switches. If the edge of the standing crop lay outside the dead band, the signal
generated by switches was sent to the control box for steering corrections.

Suggs et al. (1972) installed steering equipment on a three-wheel high clearance tractor on which a mechanical tobacco harvester was mounted. The row was sensed by a contact arm which operated a pair of microswitches mounted so that one was closed when the machine was too close to the row and the other was closed when the machine was too far from the row. The system was capable of controlling machine position within ± 0.05 m under normal conditions.

A steering control system using open-center hydraulics was introduced by Pool et al. (1984). The system deactivated the hydraulic steering system of the tractor and allowed the direction control of the tractor to be managed by a set of sensing disks, which followed the furrow. A minimum furrow depth of 51 mm was required to retain the sensing disks in the furrow.

Mechanical sensing systems, including furrow and crop-edge followers, are simple and low-cost. However, two problems have prevented adoption of this mechanism: 1) they protrude from the vehicle and are thus nuisance at the ends of a field, and 2) errors cause a great deal of steering activity that often results in magnified errors on subsequent passes (Gerrish et al., 1986). Because they require contact with the crop edge, they are prone to wear. This type of guidance system can only be used for field operations where some physical characteristic of the path can be sensed (Kirk and Krause, 1976; Smith et al., 1985).

**Leader-Cable Systems**

Leader-cables have been used for tractor and automated transit vehicle control. A leader-cable is a wire laid on or under the surface of the ground, and energized with
an audio-frequency current, which is detected by a set of search coils mounted on the vehicle (Gilmour, 1960). Two techniques for sensing magnetic field and processing the resulting signals were developed: a) field amplitude sensing using simple two-sensor arrays and b) field phase sensing requiring a more complex multiple sensor array. An amplitude sensing technique had a good performance on nonreinforced roadways or fields; however, the presence of a nearby conducting sheet distorted the sensed signal severely. Consequently, the phase sensing technique was developed (Olson, 1977).

Brooke (1968) developed a sensing head system which could control the movement of a vehicle along predetermined paths by the magnetic field distribution around a guide wire. The head consisted of three ferrite-cored coils with vertical axes. A 1/6 scale model of the sensing head was made and tested in the laboratory. No stability problems were met and it was possible to detect movements of less than 0.13 cm, which corresponded to 0.64 cm on full scale.

Rushing (1971) developed a guidance system that used a series of buried wires excited by a low power, low frequency electric generator. The field was excited at a frequency 2.8 KHz by 100 W audio amplifier. The steering sensor consisted of two identical coils made by winding a number of turns of wire on ferrite rods. Steering accuracy and repeatability of ± 2.5 cm were observed at tractor speeds up to 9.7 km/h.

The tractor guidance system developed by Schafer and Young (1979) had three pairs of ferrite-core, resonant-circuit antennas mounted near the right front wheel of the tractor. These antennas sensed the location of a buried wire that was excited by a low-current (100 mA), low-frequency (2.5 KHz) signal. Two antenna pairs lay in a horizontal plane and were similar to the ones described by Rushing (1971). One
of these pairs was essentially insensitive to orientation with respect to the controller. The other horizontal pair had the greatest sensitivity to angular orientation with respect to the wire. The third antenna pair lay in a vertical plane and had the greatest sensitivity to lateral position with respect to the wire. The lateral, angular, and reference antenna signals were fed to digital circuits for tractor steering. With no implement load on the tractor and for straight line operation at speeds to 10 km/h, the front wheels of the tractor deviated less than ± 50 mm from the buried wire.

Leader-cable systems have advantages. The control signals are definite and are not likely to be distorted significantly by the environment. The equipment is simple, and can be easily maintained. However, a disadvantage is that the system does not allow easy changing of field row patterns as agricultural production systems are changed (Gilmour, 1960; Smith et al., 1985). Research was undertaken to determine which factors were most dominate in the economics of leader-cable automatic guidance and to study the economic feasibility of such systems. Goering et al. (1972) concluded that the most dominant factor was the percentage yield increase attributable to controlled traffic. Other important factors were the life of leader-cables and control packages, and the hourly cost of labor.

Ultrasonic Systems

Ultrasonics is widely used for remote actuators and for object detectors. Newer systems with refined transducer and circuit designs can also determine the distance and velocity of objects. As a result, ultrasonics has become a reasonable alternative to more costly and complex optoelectronic and radio systems (Gross, 1978).

Julian (1971) adopted the ultrasonic transducer to detect the plowed furrow
wall. Essentially the system directed short pulses of 150 Khz acoustic energy at the ground. The time lapse between transmitting and receiving the reflected signal indicated the presence of high or low ground. By repeating the process at a number of points perpendicular to the direction of motion, a ground discontinuity can be located. Tennes and Murphy (1984) also used ultrasonic devices for position sensing. The sonar measuring system was temperature compensated to be reliable from —5 to 55°C. However, both tests could not be undertaken in an agricultural field due to erroneous measurements.

McMahon et al. (1983) used an array of five ultrasonic units to measure the distance from each ultrasonic unit to trunks of apple trees as the harvester drove over each tree. The five sonar units were positioned on the left side of the harvester in an array which was parallel to the harvester’s centerline. To simulate a row of trees, metal stands were used for the performance tests. Test results showed that the guidance system was effective at keeping each tree stand within the harvester’s allowable zone.

A planter guidance system using ultrasonic sensors was developed and tested under laboratory conditions (Patterson et al., 1985). Two ultrasonic units mounted at the end of the planter frame sensed the position of the mock planter with respect to an existing directrix. To test the system a straight 38 m two by four directrix was laid out on a wooden floor. Although the planter could be guided through the test course without losing the directrix, planter travel oscillated about the directrix. They concluded that the reason of the oscillating path was the inability of the ultrasonic guidance system to reference the directrix ahead of the planter.

Unlike electromagnetic radiation, ultrasonic beams are affected by several prop-
erties of air and soil. The biggest effect is produced by water vapor, which absorbs ultrasonic energy and decreases range. Furthermore, the velocity of ultrasonic wave propagation is highly sensitive to temperature (Gross, 1978). Some research has shown that inadequate reflection of ultrasound from soil made ultrasonic systems unusable in agricultural conditions (Kirk and Krause, 1976; Harries and Ambler, 1981).

Photoelectric Systems

Developments in photoelectric technology made it possible to use infra-red emitting diodes or image sensors. MacHardy (1967) initially planned to use two infra-red detectors for locating the tractor exhaust pipe by scanning, and to determine the position of tractor by triangulation. Kirk and Krause (1976) used an infra-red sensing system to control the steering of a self-propelled swather. They used a commercial infra-red proximity sensor that had a dust and moisture proof sensing head. The unit operated by reflecting pulsed infra-red light off of the crop edge. Intensity of the reflected light indicated the distance of the crop edge from the sensing head. This sensor could detect a crop edge up to 0.35 m away, and the system could follow the crop edge at speeds up to 12 km/h with an accuracy of 0.1 m.

Harries and Ambler (1981) used a range meter to plow automatically. The range meter measured phase difference between a projected modulated light signal and the reflection received from a reflecting post. The projected light was produced by an infra-red diode energized with a 5 MHz square wave. The returned optical energy was converted by a silicon detector to an electrical signal whose phase was compared with that of a reference signal derived from the projector.
In agricultural field operations, another popular photoelectric-sensing technique is to use image processing of the crop row. The advantage of image processing using computer-vision is 1) a look-ahead capability which may enable efficient open-loop steering corrections while avoiding the over-steering which troubles "near-sighted" or tactile systems, 2) no out-board rigging, and 3) a potential for adaptation to a number of field crops and operations (Gerrish et al., 1986).

Reid et al. (1985) investigated image sensing for determining guidance information from row crop images. Field images of cotton were recorded using a solid state camera and video cassette recorder. An 850 nm filter with a 100 nm bandwidth was used to optically preprocess the field data. The camera was mounted on the frame of a tractor in the plane passing through the outside of the left wheels of the tractor. A set of camera height angles were selected for data recording so that the camera field-of-view only recorded intensities from the plant canopy and soil background. To reduce image processing time, which is the main parameter affecting the success of any real time application, a subsampling procedure was tested to compare the effectiveness of systematic random sampling with that of total image processing for estimating classification threshold and distribution parameters. They stated that subsampling was effective for estimating the distributional properties of an image.

Navigational Systems

Several field navigational systems have been used to precisely indicate a machine's position in a field at any time. Navigational guidance techniques have high speed response and a long measurement range. They generally use spatial coordinates obtained by continuous measurement with non-contact scanning methods, that
use gyroscopes, radio signals, radar, lasers, and satellites.

Gilmour (1960) designed a navigational tractor guidance system with a dead-reckoning control scheme. The system performed all operations by reference to internal standards of heading and distance. The measurement of direction was performed by a magnetic compass, and the distance was measured by a trailing track. The test failed because the transmitting compass was insufficiently damped and the repeaters tended to oscillate excessively. A similar system for automobiles used a mileage sensor counting tire revolutions and a helium gas-rate gyro detecting vehicle direction (Tamagi et al., 1983).

An inertial system using gyroscopes has several limitations. This system can not eliminate errors such as those introduced by round-off in integration. Error can only be eliminated by resetting the gyroscope at a reference point (Gordon and Holmes, 1988). Another problem encountered using gyros as direction sensors is that they tend to drift from the reference direction. Therefore, gyros which are practically drift-free should be used (Grovum and Zoerb, 1970).

Choi et al. (1988) developed an automatic guidance system, based on field mapping, that consisted of one or two spatial position-sensing systems to locate the center of the front of tractor and the center of a 3-point hitch mounted implement. The position-sensing system called AGNAV\(^1\) consisted of a computer/transmitter-receiver module and a pair of repeater units. This module generated and transmitted VHF radio signals (154.565–154.605 MHz) to the repeaters where the signals were delayed and returned to the module. They concluded that more accurate measurement of

\(^1\)AGNAV units are manufactured by D & N Micro Products, Inc. Trade and company names used here are solely for providing specific information. Their mention does not imply recommendation or endorsement over others not mentioned.
tractor position and yaw angle, and faster error processing were required to guide the tractor with acceptable precision for field operations.

Microwave systems have demonstrated workable accuracy and permit several vehicles to operate from one transmitter. As with all electromagnetic distance measuring techniques, two transmitters are required and must be spaced a known distance apart. To maintain accuracy, the instruments require crystal ovens which must be allowed to warm up before use and require frequent recalibration (Gordon and Holmes, 1988). Heil et al. (1986) developed a microwave positioning system for an agricultural machine and reported good results for distances up to 578 m.

The drawback of existing systems has resulted in limited adoption and prompted the experimentation with laser based systems. Laser technology dominates the grade control and elevation measurement market. In the existing configuration, a beam of laser light is projected horizontally outward from a rotating mirror on a laser transmitter. Useful operating range exceeds 300 m from the laser transmitter to the working vehicles. Nominally, laser transmitter mirrors rotate at 5 or 10 revolutions per second.

Mizrach et al. (1987) chose a laser transmitter-receiver method to evaluate a suitable guidance means for the MJMT (Multi-Jointed Mobile Truss) system. A single rotating laser beam transmitter was mounted on a mast in a fixed position in the field, and four receivers (infra-red photo detectors) were mounted on a rigid rectangular frame attached to the main power unit of the MJMT. The angular velocity of the transmitter was between 0.1655 (1.58) and 0.1968 rad/s (1.88 rev/min). The test results showed that performance was limited due to the improper rate of transmitting signal by uneven laser head rotation. The sampling resolution could not be
better than 4 msec because of 250 Hz laser pulse rate. They concluded that a higher rate laser was needed to improve resolution and obtain better precision, and a laser transmitter should have a precise constant angular velocity.

To achieve good accuracy using existing laser control technology, Gordon and Holmes (1988) developed a prototype system for determining the position of a vehicle as it moved about a field. The objective of obtaining accuracy of ± 0.6 m was not fully satisfied; however, up to the range of 305 m, the accuracy was within ± 1.2 m. Angular accuracy of ± 0.19° degrees was achieved using a constant adjustment. Operation in daylight conditions was possible only by shielding of the photocells from direct sunlight. Large photocells produced the best signal. Considering the weak signal occurring at 244 and 305 m during the tests, a small amount of light blockage might prevent measurements. Repeatability of the data was not as good as had been hoped. At 305 m, the measurements varied over 6 m. Similar accuracy in automobile laser positioning system was obtained on a 40×100 m test track by Sakai (1978).

Positioning systems using satellites are becoming more accessible as satellite coverage becomes more complete. Ideally the systems will determine the position of the machine relative to a fixed station in or near the field. The precise position can then be used by operators or guidance systems to control machine direction and speed. Although many problems including cost and accuracy still exist in using this system, it seems that satellite positioning system would be the best alternative for automatic guidance systems in the future.

Larsen et al. (1988) suggested a system for satellite field navigation. Two satellite receivers will be required; one will be located at a permanent known position
relative to the field and the other mounted on the moving equipment. Currently available receivers are instruments used by surveyors for geodetic surveys and will need slight modifications for real time use on mobile equipment. The satellite signal has a high frequency and the receivers may be modified to obtain a more rapid location determination frequency than every two seconds. A radio transmitter will be used to transmit satellite data from the stationary satellite receiver to the mobile unit. The radio receiver on the mobile unit will collect these data for use on the mobile unit.

The characteristics of position sensing systems are summarized in Table 1.1.
Table 1.1: Characteristics of position sensing system

<table>
<thead>
<tr>
<th>System</th>
<th>Sensor</th>
<th>Response</th>
<th>Cost</th>
<th>Accuracy</th>
<th>Versatility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanical</td>
<td>Feeler</td>
<td>Fast</td>
<td>Low</td>
<td>Average</td>
<td>Poor</td>
</tr>
<tr>
<td>Leader-cable</td>
<td>Energized wire</td>
<td>Fast</td>
<td>Average</td>
<td>Excellent</td>
<td>Poor</td>
</tr>
<tr>
<td>Ultrasonic</td>
<td>Sonic transducer</td>
<td>Fast</td>
<td>Low</td>
<td>Average</td>
<td>Poor</td>
</tr>
<tr>
<td>Photoelectric</td>
<td>Infra-red</td>
<td>Average</td>
<td>High</td>
<td>Good</td>
<td>Average</td>
</tr>
<tr>
<td></td>
<td>Image sensor</td>
<td>Slow</td>
<td>High</td>
<td>Good</td>
<td>Average</td>
</tr>
<tr>
<td>Navigational</td>
<td>Gyroscope</td>
<td>Fast</td>
<td>Average</td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td></td>
<td>Radio signal</td>
<td>Average</td>
<td>High</td>
<td>Average</td>
<td>Excellent</td>
</tr>
<tr>
<td></td>
<td>Radar</td>
<td>Fast</td>
<td>High</td>
<td>Average</td>
<td>Average</td>
</tr>
<tr>
<td></td>
<td>Laser</td>
<td>Fast</td>
<td>High</td>
<td>Good</td>
<td>Average</td>
</tr>
<tr>
<td></td>
<td>Satellite</td>
<td>Slow</td>
<td>High</td>
<td>Poor</td>
<td>Excellent</td>
</tr>
</tbody>
</table>
Controller design effort is limited by resources that can be spent on the design. In many cases, it is not economically feasible to spend much effort on design of controllers with fixed parameters that do not require adjustments. For such applications it is common to use a standard, general-purpose regulators (controllers) with adjustable parameters (Åström and Wittenmark, 1984).

The possibilities for designing flexible, general purpose regulators have increased with computer control. When a regulator is implemented on a computer, it is possible to provide the system with computer-aided tools, which simplify design and tuning. In the design phase, a choice of control variables and measurements generally comes first. Different controllers are then introduced until a closed-loop system, with the desired properties, is obtained. The regulators used to build up the system are standard types that use feedback, feedforward, prediction, estimation, optimization, and adaption.

Control Variables of a Guidance System

Motion of a vehicle as a rigid body has six degrees of freedom. Three degrees of freedom are translational motions in vertical, lateral, and longitudinal directions, and the others are rotational degrees of freedom consisting of yaw, roll, and pitch.
All six motions are controlled by external forces acting on the tire contact surfaces, which consist of vertical forces supporting the vehicle weight, longitudinal forces of tractive or braking effort, and lateral cornering forces.

A number of approaches have been made toward deriving the differential equations which describe the lateral motion of an individual vehicle (Fenton et al., 1976; Shladover et al., 1978). These linearized models range from the complete set in three degrees of freedom to simple two-degree-of-freedom “bicycle” dynamics in which roll motion is neglected. The steering controller for the automobile guidance system is generally based on this linearized model. However, most steering controllers for the tractor have been developed using geometric and kinematic relationships of the tractor because dynamic effects can be neglected due to the slow operating speed of the tractor.

Much research has been done to determine the parameters affecting automatic guidance system stability. Grovum and Zoerb (1970) tested the marker-follower system as a fully automatic, preset guidance system. A computer simulation verified that a continuous measurement of the tractor displacement and transverse velocity with respect to a preset guide line contains sufficient information for automatic tractor guidance. Computer results indicated that the guidance system was a second-order type of control system for low forward speeds. At low speed the guidance system was very stable and displayed good following characteristics. Optimum performance required that displacement and velocity gains be set at specific values. He concluded that the following parameters affected the stability of the guidance system and must be included in any analytical stability study—tractor forward speed, displacement sensor location, servomechanism response characteristics, displacement sensor char-
acteristic, displacement gain, and velocity gain.

Shukla et al. (1970) investigated the relationship between tractor parameters and automatic steering accuracy. A kinematic mathematical model was developed to relate tractor parameters, speed, and position. Because the model was strictly kinematic, the vehicle speed and type of drive had negligible effect. Choi et al. (1988) also developed the steering control algorithm to analyze the parameters affecting the tractor guidance system. The proper front wheel angle was computed based upon the lateral position error and the curvature of desired path. The algorithm was evaluated by use of computer simulation at different steering rates, tractor forward speeds, and error sampling rates. The results showed that the tractor speed, in the range of 3–12 km/h, had only a small effect on guidance errors, and steering turning rate had negligible effect on error. When the tractor operated at high speed, errors were significant at low sampling rates.

Young et al. (1983) used two algorithms for steering control. One was the NSD (Not Speed Dependent) algorithm. The time interval for steering update was held constant. The other was the SD (Speed Dependent) algorithm. The time interval was adjusted so that the distance traveled by the tractor during this time interval remained constant. The major reason for investigating a speed dependent algorithm was to try to minimize the effect of tractor speed on the steering response for all paths, but it was unsuccessful. The tractor speed was the major factor affecting the steering accuracy for both algorithms, which was consistent with the findings of Tennes and Murphy (1984).

Control algorithms were developed for guiding tractor-implement combinations such that the implement followed a desired path (Smith et al., 1985). The algorithms,
which were based on "constant-turn" geometric relationship, were evaluated through the use of computer simulation. Simulation results indicated that absolute position error generally decreased as the steering-gain factor increased, and guidance stability was highly dependent on the magnitude of the steering gain factor. Maximum absolute implement errors increased as the steering-gain factor was reduced and decreased as the distance traveled per update interval was reduced. Position error magnitude tended to increase as the number of links in the basic mechanism increased.

Julian (1971) researched the feasibility of applying the linearized dynamic model to an agricultural tractor. If this theory is applied to a conventional rear wheel tractor operating at speeds up to 16 km/h, then a step change in steered wheel angle, for the vehicle considered in 2 degrees of freedom (yaw and side slip), results in a critically damped response, the time constant being of the order 0.125 sec. Tire turning force vs slip angle relationships used were obtained from an empirical formula. Calculated data, which assumed an asphalt surface, corresponded well with experimental values, for similar size tires operating on soil. Therefore, neglecting implement loading effects, and provided the soil is reasonably firm, there should be little error in assuming vehicle response on soil to be similar to that on asphalt.

Classical mechanics was applied to study the lateral rigid-body motions produced by steering control of an automobile (Segel, 1956). The automobile was modeled as a linear dynamic system. The desired mathematical model was obtained on equating the inertia reactions (side forces, yawing moment, and rolling moment) to their respective external force and moment summations. There are many advantages to be gained by doing so. In particular, the application of many experimental techniques developed for studying dynamic systems requires the existence of linearity. This
assumption is adequate for lateral motions of a reasonable magnitude. However, Takasaki and Fenton (1977) reported that the linearized model appeared inadequate because it was only valid for small-signal conditions at some prescribed operating point. For large-signal situations, considerable deviation from a steady-state condition were present.

**PID Controller**

The feedback loops used in controllers include simple PID regulators and their cascade combinations. Most steering controllers for automatic guidance system use a fixed gain with single or double closed-loops from the measurable state variables, e.g., lateral position error and yaw rate.

The simplest controllers are of the on-off type. This controller controls the steering wheel with a fixed turning rate and without use of compensators. Horio (1984) reported that success of guidance systems using on-off controllers depends on determination of optimum steering gain. Parish and Goering (1970, 1971) have developed an on-off steering controller for a hydrostatic swather, using a mechanical contact-type crop sensor. A kinematic mathematical model was developed and computer simulation was used to predict the performance of the automatic guidance system. This system was capable of following the crop edge at a speed of 5.0 km/h in the fully automatic mode, with a rms error of 0.5 m.

Smith et al. (1987) used instrumented scale models to verify the adequacy of guidance algorithms (Smith et al., 1985) for controlling tractor implements along predefined straight-line and sinusoidal paths. The position error at or ahead of the front axle of the machine was used to maintain guidance stability. The performance
of the guidance algorithms were influenced by the distance traveled between steering angle computations, and the magnitude of the steering angle gain factor.

Fenton et al. (1976) developed single-loop and multiple-loop controllers using a linearized model and tested them under full-scale conditions wherein a wire-reference configuration was employed. The basic control variables were automobile position and velocity, yaw angle and rate. The fixed-gain and cascade compensator was used in controller tests. The test vehicle was automatically steered on both straight and curving roads at speeds up to 35.8 m/s. The maximum tracking error was 6.35 cm both when a sidewind was present and when the vehicle entered a curving section of roadway.

The PID control technique satisfied a required set of performance specifications. These requirements may be met by a number of different designs, combining cascade and feedback combinations. However, these designs do not simultaneously meet a defined optimal performance criterion such as minimum variance.

**Optimal Controller**

By the nature of the optimal control technique a desired performance criterion is selected by optimizing the performance index or the Riccati equation and a unique design is obtained (D'Azoo and Houpis, 1981). The performance is then said to be optimal in terms of the defined performance criterion. Sometimes this technique would result in a design which may not be practically useful because it often requires feedback of state variables which can not be efficiently obtained.

Shladover et al. (1978) used optimal control techniques to synthesize controllers which minimized a performance index, $J$, consisting of mean square lateral accelera-
tion and tracking error at a given point on the vehicle located \( l' \) forward of the center of mass:

\[
J = (\ddot{y} + l' \ddot{\psi})^2 + \rho^2(y + l' \psi - y_0)^2
\]

where \( y \) is lateral position of vehicle from inertial reference, \( y_0 \) is lateral position of guideway reference from inertial reference, \( \ddot{y} \) is rms lateral acceleration, \( \psi \) is vehicle yaw angle, and \( \ddot{\psi} \) is vehicle yaw acceleration. The quantity \( \rho \) weighs tracking error relative to acceleration in the performance index. The Wiener filter optimization technique was used to determine an optimum steering controller with respect to \( J \), which generated a steering angle in response to the measured lateral error between a point on the vehicle and the guideway reference.

Often observer theory is included in the lateral controller when inaccessible states result from the use of state feedback control. Fenton and Selim (1988) employed an optimal control approach to design a velocity-adaptive, full-state feedback configuration wherein a reduced-order observer is employed. He minimized the feedback control law suggested by Bonderson (1974)

\[
J = \frac{1}{2} \int_0^\infty \left\{ \rho_1 y^2 + \rho_2 \dot{y}^2 + \rho_3 \ddot{y}^2 + (\rho_4 \psi^2 + \rho_5 \dot{\psi}^2 + \rho_6 \ddot{\psi}^2) \left[ \frac{a + b}{2} \right]^2 + \rho_7 \delta_{W}^2 + \rho_8 u^2 \right\} dt
\]

where the \( \rho_i \) (\( i = 1, \ldots, 8 \)) are weighting factors, on the state variables and various corresponding derivatives, \( a \) and \( b \) are longitudinal distances from the mass center to the front and rear axles, \( y, \dot{y}, \) and \( \ddot{y} \) are lateral position, velocity, and acceleration of the mass center from guidelane center, \( \psi, \dot{\psi}, \) and \( \ddot{\psi} \) are yaw angle, velocity, and acceleration relative to guidelane center, \( \delta_W \) is steering angle, and \( u \) is the voltage
of an electrohydraulic unit corresponding to the steering angle. The state variables were $y, \dot{y}, \Psi, \dot{\Psi}$, and $\delta \Psi$.

The Riccati equation resulting from the 5th-order dynamic model and Eq. (1) were solved to obtain feedback gain vector $\mathbf{K}$ of the control law

$$U = -\mathbf{KX}$$

(2)

where $\mathbf{X}$ is a state vector. Since the state variables except output variable $y$ are not available, a reduced fourth order observer was constructed to estimate all states. Excellent lateral control ($|y| < 0.024$ m in curve tracking at 30 m/s) and a good insensitivity to disturbance forces were obtained.

Adaptive Controller

Applying PID and optimal control techniques to a steering control problem needs a substantial effort. It is necessary to carry out the steps of modeling, identification, control design, and sensitivity analysis. Sometimes these steps should be repeated until satisfactory results are obtained (Åström and Wittenmark, 1984). Although these control techniques are quite suitable for designing automobile guidance systems, they may not be suitable when designing a steering controller for tractor guidance, which is likely to have large parameter variations, due to the unpredictable disturbances resulting from heavy implements, variable soil conditions, and an inaccurate positioning system. Therefore, it is desirable to provide the controller with algorithms for parameter estimation and control design.

A control method having such capability is adaptive control, which can tune itself and can handle systems with large parameter variations. This technique is relatively
new, but the growing number of papers and doctoral dissertations in this area have indicated that this type of system has a wide range of application. The adaptive controller is more complex than constant-gain controller, but it can be conveniently implemented by using a microprocessor. By introducing this technique into a vehicle guidance system, it is possible to compensate automatically for the change in vehicle dynamics and to keep the dynamics always around the optimum design point (Iguchi, 1986).

The specific definition of adaptive systems by Landau (1974) is:

An adaptive system measures a certain index of performance (IP) using the inputs, the states and the outputs of the adjustable system. From the comparison of the measured index of performance (IP) values and a set of given ones, the adaptation mechanism modifies the parameters of the adjustable system or generate an auxiliary input in order to maintain the index of performance (IP) values close to the set of given ones.

This definition is illustrated in Fig. 1.1.

According to the methods of parameter estimation in response to changes in process and disturbance dynamics, the adaptive controller can be classified as self-tuning regulator (STR), model-reference adaptive system (MRAS), gain-scheduling system, or dual controller.

The purpose of self-tuning regulators is to control systems with unknown but constant parameters. The regulators can also be applied to systems with varying parameters. The control algorithm is obtained by introducing a recursive parameter estimator. The controller's parameters are then adjusted continuously until the tracking or model error is nullified. There are many possible self-tuning regulators
Figure 1.1: Basic configuration of an adaptive system
Figure 1.2: Adaptive preview control model of driver-car system. $y, \dot{y}, \ddot{y}$ are position, velocity, and acceleration of the vehicle, $\beta$ is slip angle, $\gamma$ is yaw rate, $\delta$ is steering angle, and $R$ is turning radius.
depending on the system to be controlled and the design and parameter estimation
techniques (Åström and Wittenmark, 1984).

The model-reference method was originally developed by Whitaker et al. (1958). The
specifications are given in terms of the reference model which tells how the
system output ideally responds to the command signal. The controller’s parameters
are updated with a parameter adaptive algorithm. The main difficulty is how to
determine the adaptive algorithm so that a stable system is obtained. In the original
MRAS, the MIT rule was used for parameter adjustment mechanism:

\[
\frac{d\theta}{dt} = -\alpha e^T \nabla_{\theta} e
\]

where \(e\) is a model error vector and \(\theta\) is an adjustable parameter vector. The num-
ber \(\alpha\) is a parameter that determines the adaptation rate. However, there exists
another parameter estimation method which can make the system stable (Ljung and
Söderström, 1984).

There are major differences between the STR and MRAS control schemes. For
example, for the former parameter convergence is not necessary for closed-loop sta-
bility, but it is essential for the latter. In other words, the equations describing the
closed-loop behavior of MRAS can be written in a compact form and regarded as
a set of nonlinear time-varying differential equations. The stability analysis can be
done by using any of the standard nonlinear system analysis techniques such as Lyap-
unov functions and functional analysis. Meanwhile, the self-tuning model can not
be analyzed in a similar manner because the error minimized is expressed in terms
of system model estimates (Canudas de wit, 1988).

Application of the adaptive control technique to vehicle guidance is still rare,
but much research will be devoted to this area as highway automation is developed.
Nagai and Mitschke (1985) and Nagai (1987) developed an adaptive preview control model of driver-car system. In this adaptive control model, shown in Fig. 1.2, the driver steering gain was not constant, but changeable depending on road situations or disturbing forces. Lateral dynamics of driver-car systems in critical road situations, i.e., when a car runs from a dry to a wet road surface on a curve, was studied by use of a driving simulator. A new mathematical model of human driver was developed by use of adaptive control theory, to explain the adaptive behavior. The experimental test results showed that course tracking performance on a wet surface was maintained by the steering gain adaptation, but the stability was not sufficiently improved by the adaptation. The influence of the driver steering gain was greater on the course tracking performance than on other variables.
SUMMARY AND CONCLUSIONS

Position sensing systems used in vehicle guidance were reviewed and classified into mechanical, leader-cable, ultrasonic, photoelectric, and navigational systems, according to types and characteristics of the positioning sensor. The navigational system, which can keep the same traffic paths every year, is the most efficient system for a tractor steering controller, but much research and development are needed to adapt the system so it will have positioning errors small enough for tractor-implement guidance in the agricultural environment.

Control variables affecting the guidance system were investigated through the review of automobile and tractor controllers. The most commonly used parameters were lateral position, velocity, and acceleration error, and yaw angle, velocity, and acceleration with respect to the guide path. Although they are all implemented in the controller design of high speed automobiles, acceleration information may not be necessary to design guidance systems for tractors with relatively low speed.

Finally, available steering controllers were investigated to study their feasibility for tractor guidance system. Most controllers for automobiles have been devised based upon fixed-gain PID or optimal control techniques because the major disturbance force results from the wind. However, the tractor disturbance forces may have extremely large variations due to heavy implements and variable soil conditions.
Therefore, a gain-adaptation technique using adaptive control is desirable for designing tractor steering controllers because it has the potential of self-tuning and can handle systems with large parameter variations.
REFERENCES


Parish, R. L. and C. E. Goering. 1971. Simulation of an automatic steering system

Patterson, R. J., B. W. Fehr and L. P. Sheets. 1985. Electronic guidance system for a planter. ASAE Paper No. 85-1587. ASAE, St. Joseph, MI.


Suggs, C. W., B. K. Huang, and P. E. Davis. 1972. Automatic steering of field machine. ASAE Paper No. 72-122. ASAE, St. Joseph, MI.


PART II.

SEMI-RECURSIVE FORMULATION OF MECHANICAL SYSTEMS
INTRODUCTION

General-purpose programs have become popular for applications in machine dynamics, robotics, and spacecraft dynamics. In agricultural engineering, Kim and Rehkugler (1987) showed that general-purpose programs are useful for research on the design and stability of a tractor. Researchers (McConville and Angell, 1984; Antoun et al., 1986; Song et al., 1985) have used commercially available general-purpose dynamic programs (e.g., ADAMS, DADS, HVOSM, and MCADA) to study tractor dynamics.

Two criteria should be met by general-purpose dynamics programs: automatic computer-code generation and computational efficiency. The most difficult part of developing general-purpose dynamic programs is determining how to assemble the equations of motion efficiently. Often 80 percent of the execution time is spent on computation of acceleration routines. Most tractor dynamic models (Davis and Rehkugler, 1974; Larson et al., 1976; Feng and Rehkugler, 1986) have focused on specific motions by using Lagrangian or Newtonian approaches and do not satisfy the criteria for a general-purpose program. Although the Lagrangian approach is suitable for study of a specific motion, it requires formulation of kinetic energy and calculation of its derivative with respect to the generalized coordinates and velocities. Therefore, the procedure is messy, and its computational efficiency is generally poor.
compared with that of the Newtonian approach, which uses both Cartesian and relative generalized coordinates.

A variational vector approach (Haug and McCullough, 1986) was used to exploit the linear structure of both vector and differential calculus. Because this approach uses Cartesian coordinates, the system equations of motion obtained are easily implemented into computer codes. The computational efficiency is reduced, however, when a maximal set of coordinates and associated kinematic-constraint equations are introduced.

The recursive formulation is not a new concept, and it has been extensively used in the open-loop manipulator of robots. Bae (1986) used the full-recursive formulation to handle closed-loop mechanical systems and reduced the system equations of motion by eliminating the relative generalized coordinates. Therefore, the system equations of motion could be assembled in a compact form because they did not include relative coordinates. The procedure to eliminate relative coordinates is not computationally efficient, however, because it requires extra, complex computations.

The objectives of this part are: 1) to develop a semi-recursive dynamic algorithm, based on the variational vector approach, that uses relative generalized coordinates in Cartesian space, and 2) to evaluate the algorithm by modeling an agricultural tractor.
RELATIVE COORDINATE KINEMATICS

A Pair of Bodies

A pair of coupled bodies treated here has general relative motion, as shown in Fig. 2.1. Consider that the relative motion of body $j$ is constrained by a joint with respect to body $i$. Body $i$ is located and oriented in space by the position vector $r_i$ from the global-reference frame $O$ to the body-fixed frame $O_i'$ and by a set of generalized coordinates defining the orientation of this frame relative to the global frame. Joint-coordinate frames are defined on each of the bodies at joint-definition points $O_{ij}'$ and $O_{ji}'$. The $x_{ij}' - y_{ij}' - z_{ij}'$ and $x_{ji}' - y_{ji}' - z_{ji}'$ frames are fixed in bodies $i$ and $j$, respectively.

Euler angles or Euler parameters are used to represent relative generalized coordinates of joints. But, Euler angles can not be determined uniquely for some configurations, for example, a spherical joint. To avoid this difficulty, four relative Euler parameters (Nikravesh and Chung, 1982)

$$
P_{ij} = \begin{bmatrix} e_{ij0}, e_{ij1}, e_{ij2}, e_{ij3} \end{bmatrix}^T = \begin{bmatrix} e_{ij0}, e_{ij}^T \end{bmatrix}^T
$$

are defined with satisfying a normalization constraint

$$
e_{ij0}^2 + e_{ij1}^2 + e_{ij2}^2 + e_{ij3}^2 - 1 = 0
$$
In this paper the boldface denotes a vector quantity.

A vector \( v'_i \) in the frame \( O'_i \) is transformed to the global frame by

\[
v_i = A_i v'_i
\]  

(3)

where orthogonal transformation matrix \( A_i \) depends on the orientation generalized coordinates of body \( i \). The transformation of a vector \( v''_i \) from \( O''_{ij} \) frame to \( O'_i \) frame is

\[
v'_i = A_{ij}' v''_i
\]

(4)

where \( A_{ij}' \) is a constant orthogonal transformation matrix from the \( O''_{ij} \) to the \( O'_i \) frame. Similarly, a vector \( v''_j \) in the \( O''_{ij} \) frame is transformed to the \( O'_j \) frame by

\[
v'_j = A_{ij}' v''_j
\]

(5)

Finally, a vector \( v''_j \) in the \( O''_{ij} \) frame is transformed to the \( O''_{ij} \) frame by

\[
v''_i = A''_{ij} v''_j
\]

(6)

where the orthogonal transformation matrix \( A''_{ij} \) depends on the relative coordinates of joints.

The centroid of body \( j \) in Fig. 1 is located by the vector

\[
r_j = r_i + s_{ij} + d_{ij}(A_i, q_{ij}) - s_{ji} = r_i + s_{ij} + A_i d_{ij}'(q_{ij}) - s_{ji}
\]

(7)

where \( d_{ij} \) is a vector from the origin of the \( O''_{ij} \) frame to the origin of \( O''_{ji} \) frame and a function of the orientation matrix \( A_i \) of body \( i \) and \( q_{ij} \), the relative generalized coordinate between bodies \( i \) and \( j \), and \( s_{ij} \) and \( s_{ji} \) are body-fixed vectors in the global-reference frame. The vector \( s_{ij} \) can be written, using Eq. (3), as

\[
s_{ij} = A_i s_{ij}'
\]
By using Eqs. (3), (4), (5), and (6), the vector $s'_{ji}$ in the $O'_j$ frame can be sequentially transformed to the global frame through body $i$. Then a global vector $s_{ji}$ can be written as

$$s_{ji} = A_j s'_{ji} = A_i A'_{ij} A''_{ij} A'_{ji} T'_{ji}s'_{ji}$$

Therefore, the orientation matrix for the centroidal frame to the global frame for body $j$ is represented by

$$A_j = A_i A'_{ij} A''_{ij} A'_{ji} T'_{ji}$$

The angular velocity of body $j$ can be obtained as

$$\omega_j = \omega_i + \omega_{ij} = \omega_i + \sum_{m=1}^{k} u_{ijm} \dot{q}_{ijm} \equiv \omega_i + U_{ij} \dot{q}_{ij}$$

where $\omega_{ij}$ is relative angular velocity between bodies $i$ and $j$, $u_{ij}$ is a unit vector defining a joint axis transformed from the $O''_{ij}$ frame to the global reference frame, and $k$ is the number of generalized coordinates used to define a joint.

The time derivative of Eq. (7) is

$$\dot{r}_j = \dot{r}_i + \tilde{\omega}_i (s'_{ij} + d_{ij}) + \frac{\partial d_{ij}}{\partial q_{ij}} \dot{q}_{ij} - \tilde{\omega}_j s'_{ji}$$

Here $\tilde{\omega}$ denotes a skew symmetric matrix defined as

$$\tilde{\omega} = \begin{bmatrix}
0 & -\omega_z & \omega_y \\
\omega_z & 0 & -\omega_x \\
-\omega_y & \omega_x & 0
\end{bmatrix}$$

Noting the relationship $\tilde{\omega} s = -\dot{s} \omega$, substitution of Eq. (8) into Eq. (9) gives

$$\dot{r}_j = \dot{r}_i + \tilde{\omega}_i (r_j - r_i) + (\frac{\partial d_{ij}}{\partial q_{ij}} + \tilde{s}_{ji} U_{ij}) \dot{q}_{ij}$$
Combining Eqs. (8) and (10) can yield a compact form

\[ \ddot{\mathbf{Z}}_j = \mathbf{B}_{ij} \dot{\mathbf{Z}}_i + \mathbf{C}_{ij} \dot{\mathbf{q}}_{ij} \]

(11)

where

\[ \dot{\mathbf{Z}}_i = \begin{bmatrix} \dot{\mathbf{r}}_i^T, \omega_i^T \end{bmatrix}^T \]

\[ \mathbf{B}_{ij} = \begin{bmatrix} I & \dot{\mathbf{r}}_i - \dot{\mathbf{r}}_j \\ 0 & I \end{bmatrix} \equiv \begin{bmatrix} I & \ddot{\mathbf{g}}_{ij} \\ 0 & I \end{bmatrix} \]

\[ \mathbf{C}_{ij} = \begin{bmatrix} \frac{\partial d_{ij}}{\partial q_{ij}} + \ddot{q}_j U_{ij} \\ U_{ij} \end{bmatrix} \]

(12)

The translational and angular acceleration can be obtained, by taking a derivative of Eq. (11), as

\[ \ddot{\mathbf{Z}}_j = \mathbf{B}_{ij} \dot{\mathbf{Z}}_i + \mathbf{C}_{ij} \ddot{\mathbf{q}}_{ij} + \mathbf{D}_{ij} \]

(13)

where \( D_{ij} = \dot{\mathbf{B}}_{ij} \dot{\mathbf{Z}}_i + \dot{\mathbf{C}}_{ij} \dot{\mathbf{q}}_{ij} \).

Virtual displacement and rotation can be represented by replacing time derivatives with variational operators in Eq. (11) as

\[ \delta \mathbf{Z}_j = \mathbf{B}_{ij} \delta \mathbf{Z}_i + \mathbf{C}_{ij} \delta \mathbf{q}_{ij} \]

(14)

where \( \delta \mathbf{Z} = [\delta \mathbf{r}^T, \delta \omega^T]^T \). By Euler's theorem (Goldstein, 1980) virtual rotation of bodies can be represented as a vector \( \delta \pi \) about which the rotation occurs and whose magnitude is the angle of rotation.

**Multi-Bodies Linked with Joints**

When several bodies are connected by joints, velocity and acceleration of an arbitrary body may be represented with respect to the particular body of interest. It
Figure 2.1: A single-chain rigid body

Figure 2.2: Multi-chain rigid bodies
can be a junction body connected with more than one body or a base body having no preceding body. For example, the chassis of a vehicle can be considered as both.

Suppose that bodies of a kinematic chain shown in Fig. 2.2 are linked by joints. Using Eq. (11), the velocity of body $j$ with respect to body $i$ may be expressed as

$$\dot{Z}_j = B_{ij} \dot{Z}_i + \sum_{i}^{j-1} B_{i+1,j} C_{i,i+1,i,i+1} = B_{ij} \dot{Z}_i + H_i^j q_i^j$$

where

$$H_i^j = \begin{cases} [B_{i+1,j} C_{i,i+1}, B_{i+2,j} C_{i+1,i+2}, \ldots, B_{j,j} C_{j-1,j}] & i \neq j \\ 0 & i = j \end{cases}$$

$$q_i^j = \begin{cases} [q_{i+1,i+1}^T, q_{i+2,i+2}^T, \ldots, q_{j-2,j-1}^T, q_{j-1,j}^T] & i \neq j \\ 0 & i = j \end{cases}$$

Similarly, the virtual displacement and rotation of body $j$ is

$$\delta Z_j = B_{ij} \delta Z_i + H_i^j \delta q_i^j \quad (15)$$

The acceleration of body $j$ with respect to body $i$ can be written, by using Eq. (13), as

$$\ddot{Z}_j = B_{ij} \ddot{Z}_i + H_i^j \ddot{q}_i^j + K_{ij} \quad (16)$$

where

$$K_{ij} = \begin{cases} \sum_{i}^{j-1} B_{i+1,j} D_{i,i+1} & i \neq j \\ 0 & i = j \end{cases}$$

Equations (15) and (16) can be used to reduce and assemble the equations of motion rapidly with respect to the base body and generalized relative coordinates.
JOINT FORMULATIONS

Several joints can be formulated for the relative coordinate kinematics developed in the previous chapter. A vector $d_{ij}$ and matrices $U_{ij}$, $C_{ij}$, and $D_{ij}$ in Eqs. (8), (11), and (13) are formulated for three typical joints—translational, rotational, and spherical joints. Formulations for revolute-translational and universal joints are listed in Appendix A.

Revolute Joint

A typical revolute joint is shown in Fig. 2.3. The relative coordinate between bodies $i$ and $j$ is $q_{ij}$, and $u_{ij}$ is a unit vector defining a joint axis transformed from the $O''_{ij}$ frame to the global-reference frame. Since $z''$ axes and origins of $O''_{ij}$ and $O''_{ji}$ are coincident, the revolute joint has one relative degree of freedom, and $d_{ij}$ is a null vector. Then using Eqs. (3) and (4) gives

$$U_{ij} = u_{ij} = A_{ij} A_{ij}' u_{ij}''$$

where $u_{ij}'' = [0,0,1]^T$. The transformation matrix $A_{ij}''$ from $O''_{ji}$ to $O''_{ij}$ frame is

$$A_{ij}'' = \begin{bmatrix} \cos q_{ij} & -\sin q_{ij} & 0 \\ \sin q_{ij} & \cos q_{ij} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (17)$$
Therefore, the matrices $C_{ij}$ and $D_{ij}$ can be written as

\[
C_{ij} = \begin{bmatrix}
\hat{s}_{ji}u_{ij} \\
u_{ij}
\end{bmatrix}
\]
\[
D_{ij} = \begin{bmatrix}
\hat{g}_{ij}\omega_i + (\hat{s}_{ji} + \hat{s}_{ji}\hat{u}_{ij})\hat{q}_{ij} \\
\hat{u}_{ij}\hat{q}_{ij}
\end{bmatrix}
\]  \hspace{1cm} (18)

**Translational Joint**

In the translational joint shown in Fig. 2.4, similarly, the distance vector $d_{ij}$ can be expressed as

\[
d_{ij} = q_{ij}u_{ij}
\]

where $q_{ij}$ is a relative translational generalized coordinate, and $u_{ij}$ is a unit vector along the translational axis. Because there is no relative rotational motion, $U_{ij}$ is a null vector, and $A_{ij}^{(1)}$ is an identity matrix. Therefore, the matrices $C_{ij}$ and $D_{ij}$ are

\[
C_{ij} = \begin{bmatrix}
u_{ij} \\
0
\end{bmatrix}
\]
\[
D_{ij} = \begin{bmatrix}
\hat{g}_{ij}\omega_i + \hat{u}_{ij}\hat{q}_{ij} \\
0
\end{bmatrix}
\]

**Spherical Joint**

In the instance of the spherical joint, shown in Fig. 2.5, Euler parameters defined in Eqs. (1) and (2) are used as the relative generalized coordinates. Transformation matrix $A_{ij}^{(2)}$ can be represented by using Euler parameter identity (Nikravesh and
Chung, 1982) as

$$A''_{ij} = E_{ij} G'^{T}_{ij}$$

where $E_{ij} = \begin{bmatrix} -e_{ij} & \bar{e}_{ij} + e_{ij0}I \end{bmatrix}^{T}$ and $G_{ij} = \begin{bmatrix} -e_{ij} & -\bar{e}_{ij} + e_{ij0}I \end{bmatrix}^{T}$. Because of the coincident origin of two joint frames, $d_{ij}$ is a null vector. The relative angular velocity may be written as

$$\omega_{ij} = A_{ij} \omega''_{ij}$$

where $\omega''_{ij}$ is relative angular velocity in $O''_{ij}$ frame. This relative angular velocity is substituted for the relative generalized velocity $\bar{q}_{ij}$ of Eq. (11) to avoid normalization constraint in velocity and acceleration analysis. Therefore, $U_{ij}$ can be obtained as

$$U_{ij} = A_{ij} A_{ij}^T = u_{ij}$$

and vector $U_{ij}$ represents the global triad transformed from $O''_{ij}$ frame. Then the matrix $C_{ij}$ has the same form of Eq. (18). Substituting the $\bar{q}_{ij}$ of Eq. (14) into $\omega''_{ij}$ yields the acceleration of body $j$ as

$$\ddot{Z}_{j} = B_{ij}\ddot{Z}_{i} + C_{ij}\dot{\omega}_{ij}'' + D_{ij}$$

where $D_{ij} = \begin{bmatrix} \ddot{g}_{ij}\omega_{ij} + \ddot{s}_{ji}(u_{ij} + \dot{u}_{ij})\omega_{ij}'' \\ \dot{u}_{ij}\omega_{ij}'' \\ \ddot{u}_{ij}\omega_{ij}'' \end{bmatrix}$.
Figure 2.3: Revolute joint

Figure 2.4: Translational joint
Figure 2.5: Spherical joint

Figure 2.6: Kinematic chain with a closed-loop subsystem
SEMI-RECURSIVE EQUATIONS OF MOTION

Cut-Constraint Jacobians

Joint relative coordinates used to describe the motion of a closed-loop system are not generally independent, but are related by cut-constraint equations (Wittenburg, 1977). In other words, concepts of cut-constraint equations and Lagrange multipliers may be introduced to solve the closed-loop system. The Jacobian matrix of cut-constraints is generated in relative coordinate space and used for position, velocity, and acceleration analysis.

The most commonly used equation to describe the kinematic constraints between bodies $i$ and $j$ is of the form

$$\Phi_{ji} = \Phi(r_j, A_j, r_i, A_i) = 0$$  \hspace{1cm} (19)

These constraints can be a holonomic, scleronomic, or bilateral constraint equations. The constraint equations are formulated by geometric compatibility conditions, such as orthogonality or parallelism of pairs of vectors. Differentiating Eq. (19) with respect to time yields the constraint velocity equation

$$\dot{\Phi}_{ji} = \Phi_{Z_j} \ddot{r}_j + \Phi_{Z_i} \ddot{r}_i = 0$$  \hspace{1cm} (20)
The variational form of Eq. (20) can be expressed as

$$\delta \Phi_{ji} = \Phi_{Zj} \delta Z_j + \Phi_{Zi} \delta Z_i = 0$$  \hspace{1cm} (21)

where $\Phi_{Zj}$ and $\Phi_{Zi}$ are constraint Jacobians. For example, the cut-constraint equation for the spherical joint between bodies $m$ and $n$ is

$$\Phi_{mn}^s = r_m + s_{mn} - r_n - s_{nm} = 0$$  \hspace{1cm} (22)

The variation of Eq. (22) can be written as

$$\delta \Phi_{mn}^s = [I, -s_{mn}] \delta Z_m + [-I, s_{nm}] \delta Z_n \equiv \Phi_{zm} \delta Z_m + \Phi_{zn} \delta Z_n = 0$$  \hspace{1cm} (23)

The basic Jacobians for other types of joints or for other constraints are listed in Haug (1989).

Equation (21) can be reduced recursively by using Eq. (15), which provides both computational efficiency and automatic Jacobian generation by using relative coordinates. As shown in Fig. 2.6, consider that the spherical joint between bodies $m$ and $n$ is cut. The bodies $m$ and $n$ become tree-end bodies connected to the junction body $i$ through separate kinematic chains. Then, Eq. (23) can be reduced by using Eqs. (15) and (22), and the reduced form is

$$\delta \Phi_{mn}^s = \Phi_{zm} \dot{H}_i^m \ddot{q}_i^m + \Phi_{zn} \dot{H}_i^n \ddot{q}_i^n = 0$$

where the first term represents recursive reduction with respect to the relative coordinates from body $m$ to body $i$, and the second term from body $n$ to body $i$.

Differentiating Eq. (20) with respect to time yields the constraint acceleration equation

$$\Phi_{zm} \dot{H}_i^m \ddot{q}_i^m + \Phi_{zn} \dot{H}_i^n \ddot{q}_i^n = \gamma$$
Therefore, the cut-constraint acceleration equation is only the function of relative accelerations.

**System Equations of Motion**

Mechanical systems can be divided into two basic categories: open-loop and closed-loop mechanisms. Because the open-loop system can be treated as a special case of the closed-loop system, the closed-loop system is used to derive semi-recursive system equations of motion. Furthermore, because the semi-recursive algorithm does not eliminate the relative generalized coordinates, the system equations of motion for the base body are not independent and produce a larger matrix because they are functions of both base-body accelerations and relative generalized coordinates. Nonetheless, their assembly can be performed easily because elimination of relative coordinates is not necessary, and the Cartesian acceleration of each body can be quickly calculated by using the relative generalized accelerations obtained from the system equations of motion.

The variational equation of motion for a rigid body \( i \) (Wittenburg, 1977) is

\[
\delta Z_i^T (M_i \ddot{Z}_i - Q_i) = 0
\]

where mass matrix \( M_i \) and generalized applied forces \( Q_i \) are

\[
M_i = \begin{bmatrix}
m_i I & 0 \\
0 & J_i
\end{bmatrix}
\]
\[ Q_i = \begin{bmatrix} F_i \\ N_i - \omega_i J_i \omega_i \end{bmatrix} \]

\( J_i \) is the 3x3 global-inertia matrix transformed from the centroidal frame of the rigid body. For a closed-loop subsystem, shown in Fig. 2.6, the variational equation of motion can be written as

\[
\sum_{j=i}^{n} \delta Z_i^T (M_j \ddot{Z}_j - Q_j) + (\delta Z_m^T \Phi Z_m^T + \delta Z_n^T \Phi Z_n^T) \lambda = 0
\]

(24)

where \( \delta Z \)'s are kinematically consistent with Eq. (21). It is assumed that the arbitrary cut-joint constraint exists between bodies \( m \) and \( n \), and other constraints may act on the junction body \( i \). Farkas Lemma (Haug, 1989) guarantees the existence of the Lagrange multiplier vector \( \lambda \) associated with the cut-joint constraint.

Substituting Eqs. (15) and (16) into Eq. (24) gives the reduced variational equations of motion transformed to the junction body \( i \) as

\[
\delta Z_i^T (M_z \ddot{Z}_i + M_q^T \ddot{q} + \Phi M_z^T \lambda - \text{RHS}_z) + \\
\delta q^T (M_c \ddot{Z}_i + M_q \ddot{q} + \Phi M_q^T \lambda - \text{RHS}_q) = 0
\]

(25)

where

\[
M_z = \sum_{j=i}^{n} B_{ij}^T M_j B_{ij}
\]

(26)

\[
M_c^T = \sum_{j=i}^{n} B_{ij}^T M_j H_i^j
\]

(27)

\[
M_q = \sum_{j=i}^{n} H_i^j T M_j H_i^j
\]

(28)

\[
\Phi M_z = \Phi z_m B_{im} + \Phi z_n B_{in} = 0
\]

(29)

\[
\Phi M_q = [\Phi z_m H_i^m, \Phi z_n H_i^n]
\]

(30)
The $M_{ij}^T$ consists of $6 \times k$ matrix, and $k$ indicates the number of generalized relative coordinates used in defining joints between bodies. The matrix $H_i^T M_j H_i^T$ is a submatrix of $k \times k$ matrix $M_q$. The submatrix $H_i^T M_j H_i^T$ and each column of the matrix $B_{ij}^T M_j H_i^T$ should be properly located in the position corresponding to the associated relative coordinate.

While this algorithm directly reduces accelerations of any body in sub-chains with respect to the junction or base bodies, the full-recursive method can not utilize this direct reduction because each relative joint acceleration between any two bodies should be eliminated in the backward reduction. Otherwise, the next reduction can not be performed because the joint relative accelerations are the function of accelerations of inboard body and Lagrange multipliers from the cut-joint constraint of a tree-end body. Moreover, elimination of Lagrange multipliers is not efficient if the system has coupled loops, i.e., two closed-loop chains have more than one common body. In this case, simultaneous linear equations should be solved to eliminate Lagrange multipliers.

If the junction body $i$ is a base body without any constraint, then $\delta z_i$ and $\delta q$ are arbitrary and Eq. (25) becomes

$$M_z \ddot{Z}_i + M_{ij}^T \ddot{q} - \text{RHS}_z = 0$$  \hspace{1cm} (33)

$$M_e \ddot{Z}_i + M_q \ddot{q} + \Phi M_q^T \lambda - \text{RHS}_q = 0$$  \hspace{1cm} (34)

Because Eqs. (33) and (34) have fewer equations than unknowns, cut-joint constraint
acceleration equations must be introduced to complete the reduced equations of motion. Differentiating Eq. (19) twice with respect to time and substituting Eq. (16) yield the cut-constraint acceleration equation

$$\Phi M_q \ddot{q} = \gamma$$  \hspace{1cm} (35)

where $$\gamma = -(\Phi z_m \dot{z}_m + \Phi z_n \dot{z}_n + \Phi z_m K_i m + \Phi z_n K_{in})$$.

The complete set of reduced system equations of motion can be obtained by combining Eqs. (33), (34), and (35):

$$
\begin{bmatrix}
M_z & M_e & 0 \\
M_e & M_q & \Phi M_q^T \\
0 & \Phi M_q & 0
\end{bmatrix}
\begin{bmatrix}
\ddot{z}_i \\
\ddot{q} \\
\lambda
\end{bmatrix}
= 
\begin{bmatrix}
\text{RHS}_z \\
\text{RHS}_q \\
\gamma
\end{bmatrix}
$$  \hspace{1cm} (36)

This system is called a set of differential-algebraic equations. Note that the coefficient matrix is symmetric. If the motion of base body is constrained by external constraints, the constraint Jacobian with respect to the base body must be considered with the constraint acceleration equation for the base body. Compared with the full-recursive formulation, this method yields a bigger system matrix, but it is not necessary to solve the full set of equations. Because Eq. (36) can be divided into two independent equations, the elimination of relative accelerations and Lagrange multipliers is not required. Noting that the constraint equation is only function of the relative acceleration vector, $$\ddot{q}$$ can be directly obtained from the constraint equation. Substitution of them into the equations of motion produces a reduced equations of motion with respect to base body accelerations and Lagrange multipliers. Then, base body accelerations and Lagrange multipliers can be obtained in sequential order.
Numerical Example

A two-wheel-drive John Deere 4430 tractor, shown in Fig. 2.7, was modeled by using the presented approach. The modeled tractor consisted of four bodies; a chassis (body 1), a steering arm (body 2), a right wheel spindle (body 3), and a left wheel spindle (body 4). Revolute joints are connected between the chassis and each body, and two distance constraints are imposed between the steering arm and wheel spindles. A kinematic configuration is graphically shown in Fig. 2.8. $D_{ij}$, $l_{ij}$, and $R_{ij}$ in the figure represent the distance constraint, the distance vector, and the revolute joint between bodies $i$ and $j$, respectively. The system has two closed-loop subsystems and ten relative generalized coordinates, seven for the chassis and three for revolute joints. The chassis is defined as the base body, hence three chains emanate from the chassis by cutting two distance constraints. Because they have a relative driver, two distance constraints, and an Euler parameter normalization constraint for the chassis, the total degrees of freedom are six. The four tires were modeled as an internal force element (i.e., a spring and damper system). Table 2.1 shows description of the modeled tractor. The mass of tractor chassis was obtained from the Nebraska Tractor Test Report (NTTR), and the locations of joints and wheel centers were measured. Other data were estimated because the dynamic behavior of the tractor is not a major interest in this paper.

Equations (26)-(32) and (35) for the system can be derived as

$$M_{z} = \begin{bmatrix} \sum_{i=2}^{4} m_{i} I_{i} + \sum_{i=2}^{4} m_{i} \ddot{g}_{1i} \\ - \sum_{i=2}^{4} m_{i} \dddot{g}_{1i} - \sum_{i=2}^{4} (J_{i} - m_{i} \dddot{g}_{1i} \dddot{g}_{1i}) \end{bmatrix}$$

(37)
Figure 2.7: Schematic diagram of a two-wheel drive tractor

Figure 2.8: Graphical representation of Figure 2.7

\[ q = [q_1, q_2, ..., q_6, q_7] \]
\[
M_c^T = \begin{bmatrix}
-m_3 \ddot{s}_{13} u_{13} & -m_4 \ddot{s}_{14} u_{14} \\
(m_3 \dddot{s}_{13} + J_3) u_{13} & (m_4 \dddot{s}_{14} + J_4) u_{14}
\end{bmatrix}
\] (38)

\[
M_q = \begin{bmatrix}
u_{13}^T (J_3 - m_3 \dddot{s}_{13} \dddot{s}_{13}) u_{13} & 0 \\
0 & u_{14}^T (J_4 - m_4 \dddot{s}_{14} \dddot{s}_{14}) u_{14}
\end{bmatrix}
\] (39)

\[
\Phi M_z = 0
\] (40)

\[
\Phi M_q = \begin{bmatrix}
1_{23}^T (\ddot{s}_{23} - \dddot{s}_{13}) u_{13} & 0 \\
0 & 1_{24}^T (\ddot{s}_{24} - \dddot{s}_{14}) u_{14}
\end{bmatrix}
\] (41)

\[
RHS_z = \sum_{i=1}^{4} Q_i - \sum_{i=2}^{4} \left[ m_i D_{1i1} \right]
\] (42)

\[
RHS_q = \begin{bmatrix}
u_{13}^T [\dddot{s}_{31} (m_3 D_{131} - Q_{31}) + Q_{32} - J_3 D_{132}] \\
u_{14}^T [\dddot{s}_{41} (m_4 D_{141} - Q_{41}) + Q_{42} - J_4 D_{142}]
\end{bmatrix}
\] (43)

\[
\gamma = -\sum_{i=2}^{4} (\Phi z_i D_{1i} + \Phi z_i \dot{z}_i)
\] (44)

where \( \dot{Z}_i = [r_i^T, \omega_i^T]^T \), \( q = [q_{13}, q_{14}]^T \), and \( D_{1i} = [D_{1i1}^T, D_{1i2}^T]^T \). The generalized tire forces calculated were added in the matrix \( Q \). Here a linear relationship between normal force and tire slip angle was used for computing lateral force, but longitudinal force was assumed to be zero, as though no slip occurred.

To validate the semi-recursive algorithm and compare it with the full-recursive algorithm, a typical lane change maneuver was performed on both models. The numerical integration method used here was the Adams-Bashford third-order formula, and the time-step size was 0.002 seconds. Tractor velocity was 5 m/sec on a serial bump whose height and width were 0.05 m and 0.4 m, respectively. It was assumed that this uni-directional sinusoidal bump was applied to all four tires simultaneously at each time-step. The bump was made 5 m ahead of the initial starting point. After
3 seconds passed, the steering arm angle changed continuously during 6 seconds, shown in Fig. 2.9. Simulation procedure is illustrated in Appendix B, and subroutine codes written in C programming language are listed in Appendix C. The outputs obtained from both models were the same, and the lateral position and acceleration, and vertical acceleration of the tractor with respect to the center of mass were plotted in Figs. 2.10 to 2.12. Because the sampling interval was 0.05 sec, the aliasing effect occurred (Figs. 2.11 and 2.12). Therefore, the original frequency 12.5 Hz was lowered to 8 Hz. For one time-step simulation, the execution time was 48 msec with the semi-recursive model, whereas it was 92 msec with the full-recursive model, when a 16-Mz 80386 microprocessor with a math coprocessor was used.
Figure 2.9: Steering angle input of the tractor

Figure 2.10: The lateral position of the tractor
Figure 2.11: The lateral acceleration of the tractor

Figure 2.12: The vertical acceleration of the tractor
Table 2.1: Description of the modeled tractor

<table>
<thead>
<tr>
<th>Tractor mass, kg</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$</td>
<td>5079.0</td>
</tr>
<tr>
<td>$m_2 = m_3 = m_4$</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mass moments of inertia, kg·m</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J'_1$</td>
<td>2800.0</td>
<td>900.0</td>
<td>3500.0</td>
</tr>
<tr>
<td>$J'_2 = J'_3 = J'_4$</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Body-fixed position vector, cm</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s'_{12}$</td>
<td>0.0</td>
<td>179.4</td>
<td>-23.6</td>
</tr>
<tr>
<td>$s'_{21}$</td>
<td>0.0</td>
<td>8.8</td>
<td>0.0</td>
</tr>
<tr>
<td>$s'_{31}$</td>
<td>65.6</td>
<td>184.4</td>
<td>-23.6</td>
</tr>
<tr>
<td>$s'_{41}$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$s'_{14}$</td>
<td>-65.6</td>
<td>184.4</td>
<td>-23.6</td>
</tr>
<tr>
<td>$s'_{24}$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$s'_{32}$</td>
<td>4.0</td>
<td>-8.8</td>
<td>0.0</td>
</tr>
<tr>
<td>$s'_{42}$</td>
<td>0.0</td>
<td>-20.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Joint definition unit vector</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u''_{12}$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$u''_{13}$</td>
<td>-0.1736</td>
<td>0.0</td>
<td>0.9848</td>
</tr>
<tr>
<td>$u''_{14}$</td>
<td>0.1736</td>
<td>0.0</td>
<td>0.9848</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Orientation matrix</th>
<th>$A'<em>{12} = A'</em>{13} = A'<em>{14} = A'</em>{12}$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$A''_{13}$</td>
<td>0.9848</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>0.1736</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$A''_{14}$</th>
<th>$A''_{13}$</th>
</tr>
</thead>
</table>
Table 2.1 (Continued)

| Body-fixed tire-center position vector | 11.3 | 0.0 | -45.9 |
| Front right tire-center from body 3   | -11.3| 0.0 | -45.9 |
| Front left tire-center from body 4    | 85.1 | -84.4| -13.9 |
| Rear right tire-center from body 1    | -85.1| -84.4| -13.9 |
| Rear left tire-center from body 1     | 96.5 | 40.6|
| Cornering stiffness, N/rad            | $6.48 \times 10^4$ |
| Front tire spring coeff., N/m         | $2.5 \times 10^5$ |
| Rear tire spring coeff., N/m          | $2.5 \times 10^5$ |
| Front tire damping coeff., N·s/m      | 5000.0 |
| Rear tire damping coeff., N·s/m       | 1800.0 |
SUMMARY AND CONCLUSIONS

A semi-recursive dynamic algorithm was developed, based on the variational vector approach, that uses relative generalized coordinates in Cartesian space. This algorithm is not only suitable for general-purpose tractor simulation programs, which require automatic computer-code generation and better computational efficiency, but also is applicable for both open-loop and closed-loop mechanical systems.

To evaluate the algorithm, a two-wheel-drive John Deere 4430 tractor was modeled, and the system equations of motion was presented. A lane-change maneuver with semi-recursive and full-recursive models was performed on sinusoidal bump terrain. The result showed that about execution time was reduced about 48 percent by using the semi-recursive dynamic model.

Finally, including the effects of tire forces and of power train would make the tractor simulator more versatile. Because major external forces are applied through tires, and the ground has a high non-linearity, tire forces have substantial influence on tractor behavior. Although techniques of the tire modeling have been extensively developed, there is still a need to define tire-soil interactions.
REFERENCES


Bae, D. S. 1986. A recursive formulation for constrained mechanical system dynamics. Ph.D. Thesis, the University of Iowa, Iowa City, IA.


McConville, J. B. and J. C. Angell. 1984. The dynamic simulation of a moving
vehicle subject to transient steering inputs using the ADAMS computer program. ASME Paper No. 84-DET-2. ASME, New York, NY.


APPENDIX A. OTHER JOINT FORMULATIONS

Revolute-Translational Joint

A revolute-translational joint can be considered as a combination of revolute and translational joints. The two generalized coordinates are defined as

\[ \mathbf{q}_{ij} = [q_{ij1}, q_{ij2}]^T \]

where \( q_{ij1} \) and \( q_{ij2} \) are relative translational and rotational generalized coordinates, respectively. As shown in Fig. A.1, \( \mathbf{u}_{ij1} \) is a unit vector along the translational axis, and \( \mathbf{u}_{ij2} \) is a unit vector defining a joint axis. Vectors \( \mathbf{u}_{ij1} \) and \( \mathbf{u}_{ij2} \) are the same as for a translational and a revolute joints, respectively. The transformation matrix \( A''_{ij} \) has the same form of eq. (17), with the relative coordinates \( q_{ij2} \). Then matrix \( U_{ij} \) is

\[ U_{ij} = [0, u_{ij2}] = [0, A; A''_{ij}: u''_{ij2}] \]  

where \( u''_{ij2} = [0, 0, 1]^T \). The partial derivative of vector \( \mathbf{d}_{ij} \) with respect to \( \mathbf{q}_{ij} \) is obtained as

\[ \frac{\partial \mathbf{d}_{ij}}{\partial \mathbf{q}_{ij}} = \begin{bmatrix} \frac{\partial \mathbf{d}_{ij}}{\partial q_{ij1}} & \frac{\partial \mathbf{d}_{ij}}{\partial q_{ij2}} \end{bmatrix} = [u_{ij1}, 0] \]
Substituting eqs. (A.1) and (A.2) into eq. (12) gives

\[ C_{ij} = \begin{bmatrix} u_{ij1} & \tilde{s}_{ji} u_{ij2} \\ 0 & u_{ij2} \end{bmatrix} \]

and \( D_{ij} \) becomes

\[ D_{ij} = \begin{bmatrix} \tilde{s}_{ij} \omega_i + \dot{\tilde{s}}_{ij} \ddot{q}_{ij1} + (\tilde{s}_{ji} u_{ij2} + \tilde{s}_{ji} \dot{u}_{ij2}) \ddot{q}_{ij2} \\ \dot{u}_{ij2} \ddot{q}_{ij2} \end{bmatrix} \]

**Universal Joint**

A universal joint shown in Fig. A.2 has two rotational generalized coordinates

\[ q_{ij} = [q_{ij1}, q_{ij2}]^T \]

The vector \( d_{ij} \) is null because the origins of two joint reference frames are coincident. The \( x_{ij}'' - y_{ij}'' - z_{ij}'' \) frame can be obtained by two sequential rotational transformation—a rotation \( q_{ij2} \) about the \( y_{ij}'' \) axis after a rotation \( q_{ij1} \) about the \( z_{ij}'' \) axis. Therefore, the transformation matrix \( A_{ij}'' \) is

\[
A_{ij}'' = \begin{bmatrix}
\cos q_{ij1} & -\sin q_{ij1} & 0 \\
\sin q_{ij1} & \cos q_{ij1} & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos q_{ij2} & 0 & -\sin q_{ij2} \\
0 & 1 & 0 \\
\sin q_{ij2} & 0 & \cos q_{ij2}
\end{bmatrix}
\]

\[ \equiv A_{ij1}'' A_{ij2}'' \]

The rotation axis matrix \( U_{ij} \) can be written as

\[ U_{ij} = [u_{ij1}, u_{ij2}] = A_i A_{ij}^T [u_{ij1}, A_{ij1}'' u_{ij2}] \]
where $u_{ij1}' = [0, 0, 1]^T$ and $u_{ij2}'' = [0, 1, 0]^T$. Then $C_{ij}$ and $D_{ij}$ are

$$C_{ij} = \begin{bmatrix}
\bar{s}_{ji}u_{ij1} & \bar{s}_{ji}u_{ij2} \\
\bar{u}_{ij1} & \bar{u}_{ij2}
\end{bmatrix}$$

$$D_{ij} = \begin{bmatrix}
\bar{g}_{ij}\omega_i + (\bar{s}_{ji}u_{ij1} + \bar{s}_{ij}u_{ij2})\dot{q}_{ij1} + (\bar{s}_{ji}u_{ij2} + \bar{s}_{ij}u_{ij1})\dot{q}_{ij2} \\
\dot{u}_{ij1}\dot{q}_{ij1} + \dot{u}_{ij2}\dot{q}_{ij2}
\end{bmatrix}$$
Figure A.1: Revolute-translational joint

Figure A.2: Universal joint
APPENDIX B. SIMULATION PROCEDURE

The name shown in parentheses indicates the source code programmed by C language.

1. Define initial positions \( q_0 \) and velocities \( \dot{q}_0 \) that satisfy kinematic constraints (INPUT.DAT).

2. Compute the steering rack position, velocity, and acceleration depending upon the maneuver type (DRIVER).

3. Calculate orientations of all bodies, global components of the locally defined joint vectors (ORIENTATION).

4. Compute Cartesian positions in forward path sequence and check whether constraint equations are violated beyond user-defined error tolerance. If violations occur, do step 5. Otherwise, go to step 6 (POSITION).

5. Calculate the Jacobians \( \Phi_q \) (JACOBIAN) and solve the equation

\[
\Phi_u \Delta u = -\Phi
\]

by using Newton-Raphson iteration method (POSITION). \( u \) is dependent relative coordinates. Update dependent relative coordinates and go to step 3.
6. If Newton-Raphson iteration was performed, update dependent relative velocities by using the constraint velocity equation

\[ \Phi_u \dot{u} = -\Phi_v \dot{v} - \Phi_t \]

where \( v \) is independent relative coordinates (VELOCITY).

7. Recover Cartesian translational and rotational velocities in forward path sequence (RECOVER VELOCITY).

8. Calculate generalized forces acting on bodies and tires and global inertia matrices acting on bodies (FORCE, TIRE FORCE, and GEN. RHS).

9. Reduce the variational equations in backward path sequence to assemble the base-body system equations of motion (ASSEMBLE).

10. Use Gaussian method to solve the system equations of motion for base-body accelerations and Lagrange multipliers (GAUSS).

11. Recover Cartesian translational and rotational accelerations in forward path sequence (RECOVER ACCEL).

12. Integrate relative velocities and accelerations by using Adams-Bashford third-order formula to obtain relative positions and velocities (INTEGRATION). Go to step 2.
APPENDIX C. SIMULATION PROGRAM CODES

/* ******************************************************/
/* Procedure MAIN*/
/* ******************************************************/
#include <stdio.h>
#include <globals.h>

main(int argc, char *argv [])
{
    int i;
    if ((In = fopen(argv[1], "r")) == NULL) {
        printf("Can't open file %s
", argv[1]);
        abort();
    }
    if ((Path_dat = fopen(argv[2], "r")) == NULL) {
        printf("Can't open file %s
", argv[2]);
        abort();
    }
    if ((Parm = fopen(argv[3], "r")) == NULL) {
        printf("Can't open file %s
", argv[3]);
        abort();
    }
    if ((P = fopen(argv[4], "r")) == NULL) {
        printf("Can't open file %s
", argv[4]);
        abort();
    }
    if ((Out = fopen(argv[5], "w")) == NULL) {
        printf("Can't open file %s
", argv[5]);
        abort();
    }
    if ((Out1 = fopen(argv[6], "w")) == NULL) {
        printf("Can't open file %s
", argv[6]);
        abort();
    }
/*
 * Close all open files
 */

close();

driver();

test, max, step, fresh

read gulp, fresh

/*
 * open all fresh
 */

for (int i = 0; i < 17; i++)
{
    (open = open)[]

    if (open = open)[]
    {
        abort()
        print("Can't open file\n")
    
    if (open = open)[]
    {
        abort()
        print("Can't open file\n")
    
    if (open = open)[]
    {
        abort()
        print("Can't open file\n")
    
}
double stime, amp, wid;

  k = 0;
  t = tstart;
  stime = t + hmax;
  while (t <= tend) {
    /* Calculate steering wheel input. */
    if (t>=3.0 && t<=9.0) {
      amp = 50.0 * 0.001429;
      wid = 1.047197551*(t-3);
      q[0] = amp * sin(wid);
      amp = amp * 1.047197551;
      qd[0] = amp * cos(wid);
      amp = -amp * 1.047197551;
      qdd[0] = amp * sin(wid);
    } else {
      q[0] = 0.0;
      qd[0] = 0.0;
      qdd[0] = 0.0;
    }
    /* Normalize Euler parameters and recover orientations and
       Cartesian coordinates. */
    pos_orient();
    /* Position analysis routine. */
    position();
    /* Velocity analysis routine. */
    velocity();
    /* Recover Cartesian velocities. */
    rec_vel();
    /* Acceleration analysis routine. */
    /* Initialize system matrix. */
    for (i=0; i<=7; i++)
      for (j=0; j<=8; j++)
        asmb[i][j] = 0.0;
    /* Compute global inertia, gravitational and coliolis forces. */
    preacc();
    /* Compute pitch, roll, and yaw angles of the chassis. */
    motion();
    /* Compute tire forces. */
    tire();
/* Compute right-hand side forces due to constraint equations. */
genrhs();
/* Assemble system matrix. */
assemble();
/* Solve the system equation by Gaussian reduction. */
gauss();
/* Recover translational and angular accelerations from the chassis acceleration. */
recover();
/* Output the information. */
output(&stime);
/* Integration Routine. */
/* Update yp variables. */
for (i=0; i<=5; i++)
    yp[i] = qd[i];
/* Compute p(dot) from relative angular velocities. */
/* Update accelerations. */
for (i=0; i<=2; i++)
    yp[i+10] = qdd[i];
    yp[i+13] = cdd[0][i];
    yp[i+16] = omd[0][i];
/* Update y arrays. */
for (i=0; i<=8; i++)
    y[i] = q[i];
    y[i+10] = qd[i];
y[9] = q[9];
/* Integrate yp arrays */
integ();
/* Transfer y to q & qd. */
for (i=1; i<=8; i++)
    q[i] = y[i];
    qd[i] = y[i+10];
q[9] = y[9];
```c
k = k + 1;
}
}

/*******************************************************************************/
Procedure ORIENTATION
/*******************************************************************************/
#include <stdio.h>
#include <math.h>

void pos_orient()
{
int i, j, k;
extern double q[10], s[10][3], u[3][3], cqr[4][3], a[4][3][3],
sp[10][3], up[3][3], cp[2][3][3];
double c[3][3][3], sq_rt;
/* Normalize Euler parameters. */
    q[6] = q[6]/sq_rt;
    q[7] = q[7]/sq_rt;
    q[8] = q[8]/sq_rt;
    q[9] = q[9]/sq_rt;
/* Compute orientation matrix from Euler parameters. */
    egt(ôq[6], a[0]);
/* Compute global vectors. */
    mat3331(a[0], sp[0], s[0]);
    mat3331(a[0], sp[2], s[2]);
    mat3331(a[0], sp[4], s[4]);
    mat3331(a[0], up[0], u[0]);
    mat3331(a[0], up[1], u[1]);
    mat3331(a[0], up[2], u[2]);
/* Compute Aij transformation matrix */
    z_rotate(&q[0], c[0]);
    z_rotate(&q[1], c[1]);
    z_rotate(&q[2], c[2]);
/* Compute global orientation matrices. */
    matmul(a[0], c[0], a[1]);
    orient3(a[0], cp[0], c[1], a[2]);
    orient3(a[0], cp[1], c[2], a[3]);
/* Compute global s vectors. */
```
mat3331(a[1], sp[1], s[1]);
mat3331(a[1], sp[6], s[6]);
mat3331(a[1], sp[8], s[8]);
mat3331(a[2], sp[3], s[3]);
mat3331(a[2], sp[7], s[7]);
mat3331(a[3], sp[5], s[5]);
mat3331(a[3], sp[9], s[9]);

/* Compute global position vectors. */
for (i=0; i<=2; i++) {
    cqr[0][i] = q[3+i];
    cqr[1][i] = cqr[0][i] + s[0][i] - s[1][i];
    cqr[2][i] = cqr[0][i] + s[2][i] - s[3][i];
    cqr[3][i] = cqr[0][i] + s[4][i] - s[5][i];
}

/***************************************************************************/
Procedure POSITION
/***************************************************************************/
#include <stdio.h>
#include <math.h>

void positionO
{
int i, k;
extern double q[10], esprn, t;
double aj[2], fn[2];
void pos_orient();

/* Maximum number of iteration = 25 */
for (i=1; i<=25; i++) {
    k = 0;
    /* Calculate Jacobian and Constraint violatiion. */
    jacob(aj, fn, 1);
    /* Solve correction terms. */
    fn[0] = fn[0]/aj[0];
    fn[1] = fn[1]/aj[1];
    /* Update generalized coordinates. */
    q[1] = q[1] - fn[0];
if (fabs(fn[0]) >= espnr || fabs(fn[1]) >= espnr) {
    k += 1;
}

/* Transform joint coordinates into Cartesian coordinates
and find orientations of bodies from joint coordinates. */
pos_orient();
if (k == 0)
    return;
printf("Fail to assemble the position!\n");
}

Procedure VELOCITY

#include <stdio.h>
#include <math.h>

void velocity()
{
    extern double qd[9];
double aj[2], fn[2];
    /* Compute Jacobian and Phi(dot t). */
jacob(aj, fn, 2);
    /* Update qd[1] and qd[2]. */
    qd[1] = fn[0] / aj[0];
}

Procedure JACOBIAN

#include <stdio.h>
#include <math.h>

jacob(aj, fn, n)
double aj[2], fn[2];
int n;
{
    extern double cqr[4][3], s[10][3], u[3][3], qd[9], phr3[3], phr4[3];
procedure RECOVER VELOCITY

*******************************************************/
{
{
{

'}(f) 0 + [f] 000 b = [f] 0
'}(f) 0 + [f] 00 b = [f] 0

'}(f) 0 + [f] 00 b = [f] 0

{;

}

}

/* Compute tmp */
{

}

/* Compute constraint Jacobians. */
{

}


```c
#include <math.h>

void rec_vel()
{
extern double cqr[3][3], cqd[3][3], s[10][3], u[3][3],
        oma[3][3], q[10], qd[9], t;
int i;
/* Compute translational and angular velocities of bodies. */
for (i=0; i<=2; i++) {
    cqd[i][i] = qd[i+3];
    oma[i][i] = qd[i+6];
    oma[1][i] = oma[0][i] + qd[0] * u[0][i];
    oma[2][i] = oma[0][i] + qd[1] * u[1][i];
    oma[3][i] = oma[0][i] + qd[2] * u[2][i];
}
atmbt(oma[0], s[0], oma[1], s[1], cqd[1]);
atmbt(oma[0], s[2], oma[2], s[3], cqd[2]);
atmbt(oma[0], s[4], oma[3], s[5], cqd[3]);
for (i=0; i<=2; i++) {
    cqd[1][i] += cqd[0][i];
    cqd[2][i] += cqd[0][i];
    cqd[3][i] += cqd[0][i];
}
}

/********************************************************************************
Procedure FORCE
********************************************************************************/
#include <stdio.h>
#include <math.h>

void preacc()
{
int j, nbody;
double gravity = -9.806659;
extern double a[3][3][3], cqr[3][3], qd[9], qdd[9], s[10][3],
        u[3][3], oma[3][3], qd[3][3], frc[3][6], g[3][3],
        h[6][3], inertia[3][3][3], mass[4], cqd[3][3];
double temp[3], temp1[3], vel;
/* Compute global inertia vectors. */
```
for (nbody=0; nbody<=3; nbody++) {
    jg[nbody] [0] [0] =
        inertia[nbody] [0] [0] * a[nbody] [0] [0] * a[nbody] [0] [0]
        + inertia[nbody] [1] [1] * a[nbody] [0] [1] * a[nbody] [0] [1]
        + inertia[nbody] [2] [2] * a[nbody] [0] [2] * a[nbody] [0] [2];
    jg[nbody] [1] [1] =
        inertia[nbody] [0] [0] * a[nbody] [1] [0] * a[nbody] [1] [0]
        + inertia[nbody] [1] [1] * a[nbody] [1] [1] * a[nbody] [1] [1]
        + inertia[nbody] [2] [2] * a[nbody] [1] [2] * a[nbody] [1] [2];
    jg[nbody] [2] [2] =
        inertia[nbody] [0] [0] * a[nbody] [2] [0] * a[nbody] [2] [0]
    jg[nbody] [0] [1] = jg[nbody] [1] [0] =
        inertia[nbody] [0] [0] * a[nbody] [1] [0] * a[nbody] [1] [0]
        + inertia[nbody] [1] [1] * a[nbody] [1] [1] * a[nbody] [1] [1]
        + inertia[nbody] [2] [2] * a[nbody] [1] [2] * a[nbody] [1] [2];
    jg[nbody] [0] [2] = jg[nbody] [2] [0] =
        inertia[nbody] [0] [0] * a[nbody] [2] [0] * a[nbody] [2] [0]
    jg[nbody] [1] [2] = jg[nbody] [2] [1] =
        inertia[nbody] [0] [0] * a[nbody] [2] [0] * a[nbody] [2] [0]
}

/* Compute velocity coupling terms. */
atilb(oma[0], u[0], temp);
for (j=0; j<=2; j++)
    h[3][j] = qdd[0] * u[0][j] + temp[j]*qd[0];
atilb(s[0], oma[0], s[0], oma[1], s[0], oma[1], temp);
atilb(s[1], h[3], h[0]);

/* Compute g vectors together. */
for (j=0; j<=2; j++) {
    h[0][j] += temp[j];
    g[0][j] = cqr[1][j] - cqr[0][j];
    g[1][j] = cqr[2][j] - cqr[0][j];
    g[2][j] = cqr[3][j] - cqr[0][j];
}
couple_revolt2(oma[0],oma[2],s[2],s[3],u[1],&qd[1],h[1],h[4]);
couple_revolt2(oma[0],oma[3],s[4],s[5],u[2],&qd[2],h[2],h[5]);
/* Compute gravitational and coliolis forces. */
for (nbody=0; nbody<=3; nbody++) {
    mat3331(jg[nbody], oma[nbody], temp);
    atilb(oma[nbody], temp, temp);
    frc[nbody][0] = 0.0;
    frc[nbody][1] = 0.0;
    frc[nbody][2] = mass[nbody] * gravity;
    frc[nbody][3] = -temp[0];
    frc[nbody][4] = -temp[1];
    frc[nbody][5] = -temp[2];
}
/* Force compensation to keep constant 5 m/sec velocity. */
vel = a[0][0][1]*cqd[0][0]+a[0][1][1]*cqd[0][1]+
     a[0][2][1]*cqd[0][2];
temp[0] = temp[2] = 0.0;
vel = 200000* (5 - vel);
mat3331(a[0],temp1,temp);
frc[0][0] +=temp[0];
frc[0][1] +=temp[1];
frc[0][2] +=temp[2];

**************************************************************************
Procedure MOTION
**************************************************************************
#include <stdio.h>
#include <math.h>
motion()
{
extern double cqr[4][3], cdd[4][3], q[10], qd[9], a[4][3][3],
    pitch, yaw, roll, yaw_vel, laccel, ssangle;
double sinth, costh, x1, x2;
/* Compute pitch, roll, and yaw angles. */
sinth = a[0][2][1];
if (fabs(sinth) > 1.0)
    sinth = 0.99999 * sign(1.0, &sinth);
costh = sqrt(1.0 - sinth*sinth);
pitch = atan(sinth/costh);
\( x_1 = -a[0][0][1] / \cos \theta; \)
\( x_2 = a[0][1][1] / \cos \theta; \)
\( \text{yaw} = \text{atan2}(x_1, x_2); \)
\( x_1 = -a[0][2][0] / \cos \theta; \)
\( x_2 = a[0][2][2] / \cos \theta; \)
\( \text{roll} = \text{atan2}(x_1, x_2); \)
\( \text{yaw.vel} = a[0][0][2]*\text{qd}[6]+a[0][1][2]*\text{qd}[7]+a[0][2][2]*\text{qd}[8]; \)
\( x_1 = a[0][0][0]*\text{qd}[3]+a[0][1][0]*\text{qd}[4]+a[0][2][0]*\text{qd}[5]; \)
\( x_2 = a[0][0][1]*\text{qd}[3]+a[0][1][1]*\text{qd}[4]+a[0][2][1]*\text{qd}[5]; \)
\( \text{ssangle} = \text{atan2}(-x_1, x_2); \)
\( \text{laccel} = a[0][0][0]*\text{cdd}[0][0] + a[0][1][0]*\text{cdd}[0][1] + \)
\( a[0][2][0]*\text{cdd}[0][2]; \)

```
/**
 Procedure GEN_RHS
 */
#include <stdio.h>
#include <math.h>

void genrhs()
{
 int i;
 extern double cqr[4][3], cqd[4][3], s[10][3], oma[4][3], phr2[2][3],
   phr3[3], phr4[3], phr5[3],php2[2][3], php3[3], php4[3], qddd[2];
 double z23[3], z32[3], z24[3], z42[3], ldl[3], ld2[3];
 /* Compute right-hand side force terms. */
 /* Compute distant constraint equations: phr3=ll[3], phr4=12[3] */
 /* 1(trans)*s^ */
 atilb(phr3, s[6], z23);
 atilb(phr3, s[7], z32);
 atilb(phr4, s[6], z24);
 atilb(phr4, s[7], z42);
 /* Generate constraint Jacobians. */
 for (i=0; i<=2; i++) {
   phr2[0][i] = -phr3[i];
   phr2[1][i] = -phr4[i];
   phr2[0][i] = z23[i];
   phr2[1][i] = z24[i];
   phr3[i] = -z32[i];
   }```

```c
#include <math.h>
#include <stdio.h>

 /**************************************************************************
  *  Procedure TIME FORCE
  **************************************************************/


} (+1) \( t = \alpha \)


Compute right-hand side terms. */


16
are attached. */
for (j=0; j<=3; j++) {
    switch(j) {
    case 0: k=2; rad=f_rad; sp=sp_f; damp=damp_f; break;
    case 1: k=3; rad=f_rad; sp=sp_f; damp=damp_f; break;
    case 2: k=0; rad=r_rad; sp=sp_r; damp=damp_r; break;
    case 3: k=0; rad=r_rad; sp=sp_r; damp=damp_r; break;
}

    /* Compute global coordinates from body centers to wheel center. */
    sw[0] = a[k][0][0]*spw[j][0] + a[k][0][1]*spw[j][1] +
        a[k][0][2]*spw[j][2];
    sw[1] = a[k][1][0]*spw[j][0] + a[k][1][1]*spw[j][1] +
        a[k][1][2]*spw[j][2];
    sw[2] = a[k][2][0]*spw[j][0] + a[k][2][1]*spw[j][1] +
        a[k][2][2]*spw[j][2];

    /* Global wheel center position. */
    cqrw[0] = cqr[k][0] + sw[0];
    cqrw[1] = cqr[k][1] + sw[1];

    /* Bump generation. */
    cqrwy = a[k][0][0]*cqrw[0]+a[k][1][1]*cqrw[1]+a[k][2][2]*cqrw[2];
    if (cqrw[1] >= 10.0) {
        bump = 31.41592654 * cqrwy / 4.0 - 157.0796327;
        bump = 0.05 * fabs(sin(bump));
        cqrw[2] += bump;
    }

    /* Global wheel center velocity. */
    cqdw[0] = cqdw[k][0] + sw[2]*oma[k][1] - sw[1]*oma[k][2];
    cqdw[1] = cqdw[k][1] - sw[2]*oma[k][0] + sw[0]*oma[k][2];
    cqdw[2] = cqdw[k][2] + sw[1]*oma[k][0] - sw[0]*oma[k][1];

    /* Global steering angle. */
    x1 = a[0][0][0]*a[k][0][0] + a[0][1][0]*a[k][1][0] +
         a[0][2][0]*a[k][2][0];
    x2 = a[0][0][1]*a[k][0][0] + a[0][1][1]*a[k][1][0] +
         a[0][2][1]*a[k][2][0];
    s_angle[j] = atan(x2 / x1);

    /* Tire deflection. */
    defl = -cqrw[2] + rad * a[0][2][2];
    if ( defl <= 0.0 ) {
        defl = 0.0;
printf("Tire[%d] is off ground!!\n", j);
return;
}

/* Tire Normal force. */
nforce = sp * defl - damp * cqdw[2];
if (nforce <= 0.0 ) {
  nforce = 0.0;
  printf("Tire normal force is zero!!\n");
  return;
}

/* Wheel center velocity in wheel coordinates. */
cdel = cos(s_angle[j]);
sdel = sin(s_angle[j]);
cqdww[0] = (a[0][0][0]*cdel + a[0][0][1]*sdel)*cqdw[0] + (a[0][1][0]*cdel + a[0][1][1]*sdel)*cqdw[1] + (a[0][2][0]*cdel + a[0][2][1]*sdel)*cqdw[2];
cqdww[1] = (-a[0][0][0]*sdel + a[0][0][1]*cdel)*cqdw[0] + (-a[0][1][0]*sdel + a[0][1][1]*cdel)*cqdw[1] + (-a[0][2][0]*sdel + a[0][2][1]*cdel)*cqdw[2];
cqdww[2] = a[0][0][2]*cqdw[0] + a[0][1][2]*cqdw[1] + a[0][2][2]*cqdw[2];

/* No longitudinal slip, no force. */
flong = 0.0;

/* Lateral slip and force (linear relationship). */
if (fabs(cqdww[1]) <= 0.0001) {
  if (fabs(cqdww[0]) <= 0.0001 )
    strslp = 0.0;
  else
    strslp = -cqdw[0];
} else
  strslp = -atan2(cqdww[0], cqdw[1]) - yaw - s_angle[j];
fmax = mulat * nforce;
thn = 2.5 * fmax / calp;
arg = fabs(strslp);
sig = sign(1.0, &strslp);
if (arg > thn)
  flat = fmax * sig;
else
  flat = strslp * calp;
if (fabs(flat) > fmax)
flat = fabs(fmax) * sign(1.0, &flat);

/* Force computation. */
force[0] = a[0][0][0] * (cdel*flat - sdel*flong) + a[0][0][1] * (sdel*flat + cdel*flong) + a[0][0][2] * nforce;
force[1] = a[0][1][0] * (cdel*flat - sdel*flong) + a[0][1][1] * (sdel*flat + cdel*flong) + a[0][1][2] * nforce;
force[2] = a[0][2][0] * (cdel*flat - sdel*flong) + a[0][2][1] * (sdel*flat + cdel*flong) + a[0][2][2] * nforce;

/* Compute torque. */
r = rad - defl;
x1 = r * flong;
x2 = -r * flat;
force[3] = a[0][0][0] * (cdel*x1 - sdel*x2) + a[0][0][1] * (sdel*x1 + cdel*x2) - sw[2]*force[1] + sw[1]*force[2];
force[4] = a[0][1][0] * (cdel*x1 - sdel*x2) + a[0][1][1] * (sdel*x1 + cdel*x2) - sw[0]*force[2] + sw[2]*force[0];
force[5] = a[0][2][0] * (cdel*x1 - sdel*x2) + a[0][2][1] * (sdel*x1 + cdel*x2) - sw[1]*force[0] + sw[0]*force[1];
for (i=0; i<6; i++)
  frc[k][i] +=force[i];
}

Procedure ASSEMBLE
******************************************************************************/
#include <stdio.h>
#include <math.h>

void assemble()
{
  int i, j;
  extern double s[10][3], u[3][3], g[3][3], h[6][3], jg[4][3][3],
        frc[4][6], qdd[9], phr2[2][3], php2[2][3], phr3[3],
95

php3[3], phr4[3], php4[3], asmb[8][9], mass[4],
qddd[2];
double z23[3][3], z32[3][3], z24[3][3], temp1[2][3];
/* Solve generalized acceleration qdd first. */
atil(s[3], z23);
atil(s[5], z32);
mat1333(phr3, z23, z24[0]);
mat1333(phr4, z32, z24[1]);
for (i=0; i<=2; i++) {
    z24[0][i] = z24[0][i] + php3[i];
    z24[1][i] = z24[1][i] + php4[i];
    asmb[6][6] += z24[0][i]*u[1][i];
    asmb[7][7] += z24[1][i]*u[2][i];
}
qddd[0] +=
    -phr2[0][0]*h[0][0]-phr2[0][1]*h[0][1]-phr2[0][2]*h[0][2]-
    php2[0][0]*h[3][0]-php2[0][1]*h[3][1]-php2[0][2]*h[3][2]-
    phr3[0]*h[1][0]-phr3[0]*h[1][1]-phr3[0]*h[1][2]-
    php3[0]*h[4][0]-php3[0]*h[4][1]-php3[0]*h[4][2];
qddd[1] +=
    -phr2[1][0]*h[0][0]-phr2[1][1]*h[0][1]-phr2[1][2]*h[0][2]-
    php2[1][0]*h[3][0]-php2[1][1]*h[3][1]-php2[1][2]*h[3][2]-
    phr4[0]*h[2][0]-phr4[0]*h[2][1]-phr4[0]*h[2][2]-
    php4[0]*h[5][0]-php4[0]*h[5][1]-php4[0]*h[5][2];
qdd[1] = qddd[0] / asmb[6][6];
/* Write mass matrix. */
asmb[0][0] = asmb[1][1] = asmb[2][2] =
asmb[0][4] = asmb[4][0] =
    mass[1]*g[0][2] + mass[2]*g[1][2] + mass[3]*g[2][2];
asmb[1][3] = asmb[3][1] = -asmb[0][4];
asmb[2][3] = asmb[3][2] =
    mass[1]*g[0][1] + mass[2]*g[1][1] + mass[3]*g[2][1];
asmb[0][5] = asmb[5][0] = -asmb[2][3];
asmb[1][5] = asmb[5][1] =
    mass[1]*g[0][0] + mass[2]*g[1][0] + mass[3]*g[2][0];
asmb[2][4] = asmb[4][2] = -asmb[1][5];
asmb[3][3] = jg[0][0][0]+jg[1][0][0]+jg[2][0][0]+jg[3][0][0] +
    mass[1] * (g[0][1]*g[0][1] + g[0][2]*g[0][2]);
mass[2] * (g[1][1]*g[1][1] + g[1][2]*g[1][2]) +
mass[3] * (g[2][1]*g[2][1] + g[2][2]*g[2][2]);
asmb[4][4] = jg[0][1]*g[1][1] + jg[1][1][1] + jg[2][1][1] + jg[3][1][1] +
mass[1] * (g[0][0]*g[0][0] + g[0][2]*g[0][2]) +
mass[2] * (g[1][0]*g[1][0] + g[1][2]*g[1][2]) +
mass[3] * (g[2][0]*g[2][0] + g[2][2]*g[2][2]);
mass[1] * (g[0][0]*g[0][0] + g[0][1]*g[0][1]) +
mass[2] * (g[1][0]*g[1][0] + g[1][1]*g[1][1]) +
mass[3] * (g[2][0]*g[2][0] + g[2][1]*g[2][1]);
asmb[3][4]=asmb[4][3]=
jg[0][0][1] + jg[1][0][1] + jg[2][0][1] +
jg[3][0][1] - mass[1]*g[0][0]*g[0][0] -
mass[2]*g[0][0]*g[0][0] - mass[3]*g[0][0]*g[0][0];
asmb[3][5]=asmb[5][3]=
jg[0][0][2] + jg[1][0][2] + jg[2][0][2] +
jg[3][0][2] - mass[1]*g[0][1]*g[0][1] -
mass[2]*g[0][1]*g[0][1] - mass[3]*g[0][1]*g[0][1];
asmb[4][5]=asmb[5][4]=
jg[0][1][2] + jg[1][1][2] + jg[2][1][2] +
jg[3][1][2] - mass[1]*g[0][1]*g[0][1] -
mass[2]*g[0][1]*g[0][1] - mass[3]*g[0][1]*g[0][1];
asmb[3][8]= -mass[1]*h[0][1] - mass[2]*h[1][1] - mass[3]*h[2][1] +
frc[0][1] + frc[1][1] + frc[2][1] + frc[3][1];
asmb[i+3][8]=
-mass[1]*z23[0][1] - mass[2]*z23[1][1] - mass[3]*z23[2][1] -
z32[0][1] - z32[1][1] - z32[2][1] +
z24[0][1] + z24[1][1] + z24[2][1] +
frc[0][1+3] + frc[1][1+3] + frc[2][1+3] + frc[3][1+3];

atilb(s[3], u[1], z23[0]);
atilb(s[5], u[2], z23[1]);
atbt(g[1], s[3], z32);
atbt(g[2], s[5], z24);
for (i=0; i<=2; i++)
    for (j=0; j<=2; j++) {
        z32[i][j] = mass[2]*z32[i][j] + jg[2][i][j];
        z24[i][j] = mass[3]*z24[i][j] + jg[3][i][j];
    }
mat3331(z32, u[1], templ[0]);
mat3331(z24, u[2], tempi[1]);
for (i=0; i<=2; i++) {
    asmb[6][i] = mass[2]*z23[0][i];
    asmb[7][i] = mass[3]*z23[1][i];
    asmb[6][i+3] = temp1[0][i];
    asmb[7][i+3] = temp1[1][i];
    asmb[i][8] += -asmb[6][i]*qdd[1] - asmb[7][i]*qdd[2];
    asmb[i+3][8] += -temp1[0][i]*qdd[1] - temp1[1][i]*qdd[2];
}
/* Equations from two generalized coordinates */
revolt(&mass[2], jg[2], frc[2], s[3], u[1], h[1], h[4], 6);
revolt(&mass[3], jg[3], frc[3], s[5], u[2], h[2], h[5], 7);

revolt(mass, jg, f, s, u, hi, h2, flag)
double *mass, jg[3][3], f[6], s[3], u[3], hi[3], h2[3];
int flag;
{
    int i, j, l;
    extern double asmb[8][9];
double uj[3], us[3], temp[3], tempi[3][3], temp2[3];

    atbt(s, s, temp1);
    for (i=0; i<=2; i++)
        for (j=0; j<=2; j++)
            temp1[i][j] = jg[i][j] - (*mass)*temp1[i][j];
    mat3331(temp1, u, temp2);
    mat3331(jg, h2, us);
    for (i=0; i<=2; i++)
        uj[i] = *mass*h1[i] - f[i];
atilb(s, uj, temp);
    for (i=0; i<=2; i++)
        temp[i] = temp[i] - us[i] + f[i+3];
asmb[flag][8] = u[0]*(temp[0]-temp2[0])+u[1]*(temp[1]-temp2[1])+u[2]*(temp[2] - temp2[2]);
}

/******************************************************************************************
 Procedure GAUSS
******************************************************************************************/
#include <stdio.h>
#include <math.h>
void gauss()
{
    int i, j, k;
    extern double asmb[8][9];
    double mult;
    for (k=0; k<=6; k++)
        for (i=k+1;i<=7; i++) {
            mult = asmb[i][k] / asmb[k][k];
            asmb[i][8] = asmb[i][8] - mult * asmb[k][8];
            for (j=k+1; j<=7; j++)
                asmb[i][j] = asmb[i][j] - mult * asmb[k][j];
        }
    /* Back substitution. */
    for (i=7; i>=0; i--)
        for (j=i+1; j<=7; j++)
            asmb[i][8] = asmb[i][8] - asmb[i][j] * asmb[j][8];
    asmb[i][8] = asmb[i][8] / asmb[i][i];
}

/******************************************************************************************
 Procedure RECOVER_ACCEL
******************************************************************************************/
void recover()
{
    int i;
    extern double cdd[4][3], s[10][3], u[3][3], qdd[9], omd[4][3], g[3][3], h[6][3], asmb[8][9];
    double z23[3], z32[3], z24[3], z42[3];
    /* Update base body equation and recover joint accelerations. */
/* base body */
for (i=0; i<=2; i++) {
    cdd[0][i] = asmb[i][8];
    omd[0][i] = asmb[i+3][8];
}
/* body[0] */
    atilb(g[0], omd[0], z24);
for (i=0; i<=2; i++) {
    cdd[1][i] = cdd[0][i] - z24[i] + h[0][i];
    omd[1][i] = omd[0][i] + h[3][i];
}
atilb(g[1], omd[0], z23);
atilb(s[3], u[1], z24);
atilb(g[2], omd[0], z32);
atilb(s[5], u[2], z42);
for (i=0; i<=2; i++) {
    cdd[2][i] = cdd[0][i] - z23[i] + qdd[1]*z24[i] + h[1][i];
    omd[2][i] = omd[0][i] + qdd[1]*u[1][i] + h[4][i];
    cdd[3][i] = cdd[0][i] - z32[i] + qdd[2]*z42[i] + h[2][i];
    omd[3][i] = omd[0][i] + qdd[2]*u[2][i] + h[5][i];
}

/***********************************************************/
 Procedure INTEGRATION
/***********************************************************/
#include <stdio.h>

void integ()
{
    int i;
    extern double y[19], yp[19], t, hstep;
    static double w[2][19], wl[2][19];
    static int count = 1;
    if (count == 1) {
        for (i=0; i<=18; i++) {
            y[i] = y[i] + hstep*yp[i] / 4.0;
            w[1][i] = yp[i];
            wl[1][i] = yp[i];
        }
t = t + hstep / 4.0;
++count;
return;
}
if (count == 2) {
    for (i=0; i<=18; i++) {
        y[i] = y[i] + hstep * (3.0*yp[i] - w0l[i]) / 8.0;
        w0l[i] = yp[i];
    }
    t = t + hstep / 4.0;
    ++count;
    return;
}
if (count == 3) {
    for (i=0; i<=18; i++) {
        y[i] = y[i] + hstep * (23.0*yp[i] - 16.0*w0l[i] 
            + 5.0*w1[i]) / 48.0;
        w1[i] = w0l[i];
        w0l[i] = yp[i];
    }
    t = t + hstep / 4.0;
    ++count;
    return;
}
if (count == 4) {
    for (i=0; i<=18; i++) {
        y[i] = y[i] + hstep * (23.0*yp[i] - 16.0*w0l[i] 
            + 5.0*w1[i]) / 24.0;
        w0l[i] = yp[i];
    }
    t = t + hstep / 4.0;
    ++count;
    return;
}
if (count == 5) {
    for (i=0; i<=18; i++) {
        y[i] = y[i] + hstep * (23.0*yp[i] - 16.0*w0l[i] 
            + 5.0*w1[i]) / 24.0;
        w0l[i] = yp[i];
    }
}
double a[3][3], b[3][3] = a; /* (a) */

#include <math.h>

#include <stdio.h>

PROCEDURE MATH_LIB

*******************************************************************/

{ /*...*/

    int count;
    int step = 2 + 1;
    
    f[t][0]A = [f][0][0]A
    f[t][0]A = [f][0][1]A
    
    0.0 / (f[t][0]W + [f][0]W + 16.0 - [f][0]W) * step + [f][0]A = [f][0]A
    } (++) for: (0 = i) (0 = i) (0 = i) (0 = i) (0 = i) (0 = i) (0 = i)
    
    return
    count++
    step = 2 + 1;
    
    f[t][0]A = [f][0][0]A
    f[t][0]A = [f][0][1]A
    
    0.0 / (f[t][0]W + [f][0]W + 16.0 - [f][0]W) * step + [f][0]A = [f][0]A
    } (++) for: (0 = i) (0 = i) (0 = i) (0 = i) (0 = i) (0 = i) (0 = i)
    
    return
    count++
    step = 2 + 1;
    
    f[t][0]A = [f][0][0]A
    f[t][0]A = [f][0][1]A

101
b[0][0] = b[1][1] = b[2][2] = 0.0;
b[0][1] = -a[2]; b[0][2] = a[1];
b[1][0] = a[2]; b[1][2] = -a[0];
b[2][0] = -a[1]; b[2][1] = a[0];
}
/* c = tilde(a) * b */
atilb(a, b, c)
double a[], b[], c[];
{
    c[0] = -b[1]*a[2] + a[1]*b[2];
    c[1] = b[0]*a[2] - a[0]*b[2];
    c[2] = -b[0]*a[1] + a[0]*b[1];
}
/* c = tilde(a) * tran(b) */
atilb32(a, b, c)
double a[3], b[2][3], c[3][2];
{
    c[0][0] = a[1]*b[0][2] - a[2]*b[0][1];
    c[1][0] = a[2]*b[0][0] - a[0]*b[0][2];
    c[2][0] = a[0]*b[0][1] - a[1]*b[0][0];
    c[0][1] = a[1]*b[1][2] - a[2]*b[1][1];
    c[1][1] = a[2]*b[1][0] - a[0]*b[1][2];
    c[2][1] = a[0]*b[1][1] - a[1]*b[1][0];
}
/* g = tilde(a) * tilde(b) * c - tilde(d) * tilde(e) * f */
atimtb(a, b, c, d, e, f, g)
double a[3], b[3], c[3], d[3], e[3], f[3], g[3];
{
    g[0] = a[1]*(b[0]*c[1]-c[0]*b[1])-a[2]*(b[2]*c[0]-b[0]*c[2])
        -d[1]*(e[0]*f[1]-e[1]*f[0])+d[2]*(e[2]*f[0]-e[0]*f[2]);
    g[1] = a[2]*(b[1]*c[2]-c[1]*b[2])-a[0]*(b[0]*c[1]-b[1]*c[0])
        -d[2]*(e[1]*f[2]-e[2]*f[1])+d[0]*(e[0]*f[2]-e[2]*f[0]);
    g[2] = a[0]*(b[2]*c[0]-c[2]*b[0])-a[1]*(b[1]*c[2]-b[2]*c[1])
        -d[0]*(e[2]*f[0]-e[0]*f[2])+d[1]*(e[1]*f[2]-e[2]*f[1]);
}
/* e = tilde(a) * b - tilde(c) * d */
atmb(a, b, c, d, e)
double a[3], b[3], c[3], d[3], e[3];
{

} /* Orientation by Euler parameters. */

gte(p, a)
double p[4], a[3][3];
{
   a[0][0] = 2.0*(p[0]*p[0] + p[1]*p[1] - 0.5);
   a[0][1] = 2.0*(p[1]*p[2] - p[0]*p[3]);
   a[0][2] = 2.0*(p[0]*p[2] + p[1]*p[3]);
   a[1][0] = 2.0*(p[1]*p[2] + p[0]*p[3]);
   a[1][1] = 2.0*(p[0]*p[0] + p[2]*p[2] - 0.5);
   a[2][0] = 2.0*(p[1]*p[3] - p[0]*p[2]);
   a[2][2] = 2.0*(p[0]*p[0] + p[3]*p[3] - 0.5);
}

mat333(a, b, c)
double a[], b[3][3], c[];
{
   c[0] = a[0][0]*b[0][0] + a[1][0]*b[1][0] + a[2][0]*b[2][0];
   c[1] = a[0][1]*b[0][1] + a[1][1]*b[1][1] + a[2][1]*b[2][1];
   c[2] = a[0][2]*b[0][2] + a[1][2]*b[1][2] + a[2][2]*b[2][2];
}

/* d = a[1][3] * b[3][3] * c[3][1] */
mat331(a, b, c, d)
double a[3], b[3][3], c[3], *d;
{
   *d = a[0] * (b[0][0]*c[0] + b[0][1]*c[1] + b[0][2]*c[2]) +
      a[1] * (b[1][0]*c[0] + b[1][1]*c[1] + b[1][2]*c[2]) +
}

mat3331(a, s, as)
double a[3][3], s[3], as[3];
{
   as[0] = a[0][0]*s[0] + a[0][1]*s[1] + a[0][2]*s[2];
   as[1] = a[1][0]*s[0] + a[1][1]*s[1] + a[1][2]*s[2];
   as[2] = a[2][0]*s[0] + a[2][1]*s[1] + a[2][2]*s[2];
}
/* c[3][3] = a[3][1] * b[1][3] */
mat3113(a, b, c)
double a[3], b[3], c[3][3];
{
    c[0][0] = a[0]*b[0];
    c[1][0] = a[1]*b[0];
    c[2][0] = a[2]*b[0];
    c[0][1] = a[0]*b[1];
    c[1][1] = a[1]*b[1];
    c[2][1] = a[2]*b[1];
    c[0][2] = a[0]*b[2];
    c[1][2] = a[1]*b[2];
    c[2][2] = a[2]*b[2];
}

mat333(a, b, c, d)
double a[3][3], b[3][3], c[3][3], d[3][3];
{
    double t[3][3];
    t[0][0] = a[0][0]*b[0][0] + a[0][1]*b[1][0] + a[0][2]*b[2][0];
    t[0][1] = a[0][0]*b[0][1] + a[0][1]*b[1][1] + a[0][2]*b[2][1];
    t[0][2] = a[0][0]*b[0][2] + a[0][1]*b[1][2] + a[0][2]*b[2][2];
    t[1][0] = a[1][0]*b[0][0] + a[1][1]*b[1][0] + a[1][2]*b[2][0];
    t[1][1] = a[1][0]*b[0][1] + a[1][1]*b[1][1] + a[1][2]*b[2][1];
    t[1][2] = a[1][0]*b[0][2] + a[1][1]*b[1][2] + a[1][2]*b[2][2];
    t[2][0] = a[2][0]*b[0][0] + a[2][1]*b[1][0] + a[2][2]*b[2][0];
    t[2][1] = a[2][0]*b[0][1] + a[2][1]*b[1][1] + a[2][2]*b[2][1];
    t[2][2] = a[2][0]*b[0][2] + a[2][1]*b[1][2] + a[2][2]*b[2][2];
    d[0][0] = t[0][0]*c[0][0] + t[0][1]*c[1][0] + t[0][2]*c[2][0];
    d[0][1] = t[0][0]*c[0][1] + t[0][1]*c[1][1] + t[0][2]*c[2][1];
    d[0][2] = t[0][0]*c[0][2] + t[0][1]*c[1][2] + t[0][2]*c[2][2];
    d[1][0] = t[1][0]*c[0][0] + t[1][1]*c[1][0] + t[1][2]*c[2][0];
    d[1][1] = t[1][0]*c[0][1] + t[1][1]*c[1][1] + t[1][2]*c[2][1];
    d[1][2] = t[1][0]*c[0][2] + t[1][1]*c[1][2] + t[1][2]*c[2][2];
    d[2][0] = t[2][0]*c[0][0] + t[2][1]*c[1][0] + t[2][2]*c[2][0];
    d[2][1] = t[2][0]*c[0][1] + t[2][1]*c[1][1] + t[2][2]*c[2][1];
    d[2][2] = t[2][0]*c[0][2] + t[2][1]*c[1][2] + t[2][2]*c[2][2];
}

/* Compute orientation of body. */


```c
orient3(a, b, c, d)
double a[3][3], b[3][3], c[3][3], d[3][3];
{

double t[3][3];

t[0][0] = a[0][0]*b[0][0] + a[0][1]*b[1][0] + a[0][2]*b[2][0];
t[0][1] = a[0][0]*b[0][1] + a[0][1]*b[1][1] + a[0][2]*b[2][1];
t[1][0] = a[1][0]*b[0][0] + a[1][1]*b[1][0] + a[1][2]*b[2][0];
t[1][1] = a[1][0]*b[0][1] + a[1][1]*b[1][1] + a[1][2]*b[2][1];
t[2][0] = a[2][0]*b[0][0] + a[2][1]*b[1][0] + a[2][2]*b[2][0];
t[2][1] = a[2][0]*b[0][1] + a[2][1]*b[1][1] + a[2][2]*b[2][1];
t[0][2] = t[1][0]*t[2][1] - t[2][0]*t[1][1];
t[1][2] = t[2][0]*t[0][1] - t[0][0]*t[2][1];
t[2][2] = t[0][0]*t[1][1] - t[1][0]*t[0][1];
d[0][0] = t[0][0]*c[0][0] + t[0][1]*c[1][0] + t[0][2]*c[2][0];
d[0][1] = t[0][0]*c[0][1] + t[0][1]*c[1][1] + t[0][2]*c[2][1];
d[1][0] = t[1][0]*c[0][0] + t[1][1]*c[1][0] + t[1][2]*c[2][0];
d[1][1] = t[1][0]*c[0][1] + t[1][1]*c[1][1] + t[1][2]*c[2][1];
d[2][0] = t[2][0]*c[0][0] + t[2][1]*c[1][0] + t[2][2]*c[2][0];
d[2][1] = t[2][0]*c[0][1] + t[2][1]*c[1][1] + t[2][2]*c[2][1];
d[0][2] = d[1][0]*d[2][1] - d[2][0]*d[1][1];
d[1][2] = d[2][0]*d[0][1] - d[0][0]*d[2][1];
d[2][2] = d[0][0]*d[1][1] - d[1][0]*d[0][1];
}

/*
  | cos(theta) -sin(theta) 0 |
  a[3][3] = | sin(theta)  cos(theta) 0 |
  | 0       0     1     |
*/
z_rotate(theta, a)
double *theta, a[3][3];
{

int i, j;
double co, si;

co = cos(*theta);
si = sin(*theta);

a[0][0] = a[1][1] = co;
a[0][2] = a[1][2] = a[2][0] = a[2][1] = 0.0;
a[0][1] = -si; a[1][0] = si; a[2][2] = 1.0;
}

matmul(a, b, c)
```
double a[3][3], b[3][3], c[3][3];
{
    c[0][0] = a[0][0]*b[0][0]+a[0][1]*b[1][0]+a[0][2]*b[2][0];
    c[0][1] = a[0][0]*b[0][1]+a[0][1]*b[1][1]+a[0][2]*b[2][1];
    c[0][2] = a[0][0]*b[0][2]+a[0][1]*b[1][2]+a[0][2]*b[2][2];
    c[1][0] = a[1][0]*b[0][0]+a[1][1]*b[1][0]+a[1][2]*b[2][0];
    c[1][1] = a[1][0]*b[0][1]+a[1][1]*b[1][1]+a[1][2]*b[2][1];
    c[1][2] = a[1][0]*b[0][2]+a[1][1]*b[1][2]+a[1][2]*b[2][2];
    c[2][0] = a[2][0]*b[0][0]+a[2][1]*b[1][0]+a[2][2]*b[2][0];
    c[2][1] = a[2][0]*b[0][1]+a[2][1]*b[1][1]+a[2][2]*b[2][1];
    c[2][2] = a[2][0]*b[0][2]+a[2][1]*b[1][2]+a[2][2]*b[2][2];
}

/* Velocity coupling term by revolute joint. */
couple_revolt2(omai, omaj, si, sj, u, qd, hi, h2)
    double omai[3], omaj[3], si[3], sj[3], u[3], *qd, hi[3], h2[3];
{
    int i;
    double temp[3];
    for (i=0; i<=2; i++)
        temp[i] = *qd * u[i];
    atilb(omai, temp, h2);
    atimtb(omai, omaj, si, omaj, omaj, sj, temp);
    atilb(sj, h2, hi);
    for (i=0; i<=2; i++)
        hi[i] += temp[i];
}

/* sign(a, b) = sign(b) * abs(a) */
sign(a, b)
    double a, *b;
{
    double t;
    t = fabs(a);
    if (*b >= 0.0 )
        return(t);
    else
        return(-t);
}

/* c[3][3] = tran(a[1][3]) * tran(b[3][1]) */
atbt(a,b,c)
    double a[3], b[3], c[3][3];
\{ 
  c[0][0] = -a[1]*b[1] - a[2]*b[2]; 
  c[1][1] = -a[0]*b[0] - a[2]*b[2]; 
  c[2][2] = -a[1]*b[1] - a[0]*b[0]; 
  c[0][1] = a[1] * b[0]; c[0][2] = a[2]*b[0]; 
  c[1][0] = a[0] * b[1]; c[1][2] = a[2]*b[1]; 
  c[2][0] = a[0] * b[2]; c[2][1] = a[1]*b[2]; 
\}

/*********************************************************
Common GLOBAL.H
*******************************************************************/
double
/* Time control variables. */
t, tstart, tend, hstep, espnr, hmax,
/* Position, translational and angular velocity and acceleration. */
cqr[4][3], cqd[4][3], cdd[4][3], g[3][3], oma[4][3], omd[4][3],
/* Relative generalized position, velocity, and acceleration. */
q[10], qd[9], qdd[9],
/* Integration variables. */
y[19], yp[19],
/* Body fixed position and joint vectors, orientation vectors. */
s[10][3], u[3][3], a[4][3][3],
s[10][3] = {{ 0.0, 1.7940, -0.2355 },
{ 0.0, 0.0878, 0.0 },
{ 0.6595, 1.844, -0.2355 },
{ 0.0, 0.0, 0.0 },
{-0.6595, 1.844, -0.2355 },
{ 0.0, 0.0, 0.0 },
{ 0.04, -0.0878, 0.0 },
{ 0.0, -0.2, 0.0 },
{-0.04, -0.0878, 0.0 },
{ 0.0, -0.2, 0.0 }},
up[3][3] = {{ 0.0, 0.0, 1.0 },
{-0.1736, 0.0, 0.9848 },
{ 0.1736, 0.0, 0.9848 }},
cp[2][3][3] = {{ 0.9848, 0.0, -0.1736,
0.0, 1.0, 0.0,
0.1736, 0.0, 0.9848 },
{ 0.9848, 0.0, 0.1736},
Inertial matrix, mass, force vectors. */

\[
\begin{bmatrix}
0.0, & 1.0, & 0.0, \\
0.0, & 0.0, & 0.0
\end{bmatrix}
\]

Jacobian matrix. */

System mass matrix. */

Tire characteristics. */

Variables for chassis motion. */

pitch, yaw, roll, yaw_vel, laccel, ssangle, s_angle[4];

Initial Condition INPUT.DAT

0.0001 0.0 12.0 0.001 0.05
0.0000 0.000 0.000 0.0000 0.0000 1.037546 1.0000
0.00 0.0 0.0
0.0000 0.0000 0.0000 0.0000 5.0 0.0000 0.0000 0.0 0.0
PART III.

SELF-TUNING STEERING CONTROLLER DESIGN FOR FARM TRACTOR GUIDANCE
Adaptive control has been studied for design of high-performance control systems since the early 1950s. However, its practical application was not very successful until the early 1970s. With the development of low cost, reliable digital computers adaptive control algorithms have received much attention and have worked well for many applications. The advantage of an adaptive regulator is the capability to change its behavior in response to changes in the dynamics of a system and to outside disturbances. Because ordinary feedback was originally introduced for the same purpose, the difference between the two control methods was not clear, but there is a general consensus that a constant-gain feedback is not an adaptive control (Isermann, 1982; Åström, 1983).

There are different approaches and several techniques for designing adaptive controllers. They vary from simple gain scheduling, where the regulator is adjusted from a gain table which describes the system static characteristics, to more complex control laws which are time-varying and non-linear functions of the system states. Two widely recognized adaptive control families among them are a self-tuning regulator (STR) and a model-reference adaptive system (MRAS).

Ideally, a tractor guidance system should work over a wide range of operating conditions because it is likely to have parameter variations, due to the unpredictable
disturbances resulting from heavy implements, variable soil conditions, and inaccurate positioning systems. Ordinary constant-gain, linear feedback can work well in one operating condition, but difficulties can be encountered when operating conditions change. A more sophisticated adaptive regulator which works well over a wide range of operating conditions is therefore needed to control a tractor over a predetermined path.

Selection of the proper control method for a particular system depends upon the control objective, the model structure, and the input and output. In MRAS, the parameters are adjusted so that errors between the system and model outputs are minimized. If the model structure is well-known and its parameters are precise, MRAS has better performance than the STR because noise effects are considered in the parameter adaptive mechanism. When the parameters are unknown and the states are inaccessible, an adaptive regulator with an adjustable model should be used. Selection of a STR would be more desirable for tractor guidance because it is very flexible with respect to the design methods, easy to understand, and to implement with microprocessors.

A study of system structure should be performed when applying two control schemes to the tractor guidance system. The dynamics of rubber-tired vehicles have been studied for at least two decades, ranging from the complex eleven degrees-of-freedom (DOF) model (McHenry, 1968) to a simple 2-DOF model used in control studies (Pasternack, 1973). Where lateral accelerations are less than 0.3g, a 3-DOF model is quite accurate for automotive vehicles, which often have significant roll characteristics (Shladover et al., 1978). The choice between 3-DOF and 2-DOF models depends upon the importance of roll motion in tractor steering characteristics and ve-
Vehicle's handling. The lateral acceleration of the tractor is small under normal working environments. Therefore, the simple 2-DOF model is adequate for tractor guidance systems because roll steer can be neglected.

In the linear 2-DOF tractor model, the two degrees of freedom are the yaw and lateral (or side-slip) motions of the tractor. The state variables of the control system can be these parameters and their derivatives with respect to time. In an actual system, it is often unrealistic to assume that all states of the system can be measured. This is due to the disturbance and measurement noise or to practical measurement difficulties. The unmeasurable states can be estimated by using an observer, which can reconstruct the states from the measured outputs by a dynamical system. The tractor guidance controller may also need a prediction of some states to control a tractor following a predetermined path.

These problems often can be solved by introducing prediction and filtering theory into the guidance system. The method best suited for a particular system depends on the nature of disturbances and measurement noises, but it is natural to use stochastic or random concepts to describe them. If their models are known, optimal observers and predictors like the Kalman filter can be designed. This method can cover a wide range of disturbances, and can yield good control for filtering and prediction problems.

The objectives of this research are: 1) to determine a suitable model structure by using a 2-DOF linear dynamic model of a tractor in discrete-time space, 2) to estimate model parameters satisfying design criterion, by using a recursive least-squares parameter estimation algorithm with system input and output, 3) to design an algorithm for setting variable forgetting factors which can cope with time-varying
nonlinear systems, 4) to design a self-tuning regulator which minimizes the deviations of actual from desired tractor position and yaw angle, and 5) to analyze and verify the self-tuning adaptive controller by using a tractor dynamic simulator.
SELECTION OF MODEL STRUCTURE

The selection of the proper model structure is the necessary first step in self-tuning regulator design. The selection will greatly influence the characteristics of the system identification problem, such as the computational effort, the order of the system, and the possibility of finding unique solutions. In automatic control and signal processing, the model of a dynamic system is a mathematical description of the relationship between inputs and outputs of the system, which can be obtained from physical laws. The advantage of model-building from physics is that it gives insight about the system, and about parameters and variables which have physical interpretations (Åström and Eykhoff, 1971).

The dynamic model of rubber-tire vehicles ranges from the several DOF model to simple three and two DOF models often used in control studies. The choice between 2-DOF and 3-DOF models are vehicle-specific, depending on the importance of rolling motion. Because the speed of agricultural tractors is slow and their rolling motions are not significant under normal operation, the simpler 2-DOF model is adequate for use in the initial design of steering systems.
Continuous-Time Model

A linear 2-DOF model, shown in Fig. 3.1, is used to determine the model type for steering guidance system of a farm tractor. The two degrees of freedom are the yaw \( \psi \) and either lateral motion \( y \) or sideslip \( \beta \) of the tractor. The assumptions used here are as follows:

1. All angles are small.
2. Forward velocity is constant.
3. Dynamic changes in the wheel loads are neglected.
4. Longitudinal tire forces are zero.
5. Lateral tire forces are proportional to the tire slip angle \( \alpha \).
6. Front wheels are steered, and the steering is parallel.
7. Aerodynamic effects are neglected.
8. Counterclockwise rotation is positive.

The lateral force on tires can be written as

\[
S_f = -K_f \alpha_f \\
S_r = -K_r \alpha_r
\]  \hspace{1cm} (1)

where \( S \) is the lateral force (N) of the tire, \( K \) is the tire cornering stiffness (N/rad), \( \alpha \) is the tire slip angle (rad), and subscripts \( f \) and \( r \) represent the front and rear tires,
Figure 3.1: Linearized tractor model at constant speed. $\alpha$, $\beta$, $\psi$, and $\delta$ are tire-slip, side-slip, yaw, and steering angles, $l$ is distance between tractor c.g. and wheel center, $S$ and $U$ are lateral and longitudinal forces, and $v$ is tractor velocity.
respectively. The slip-angles of the tire are
\[
\alpha_f = \beta + \frac{f \psi}{v} - \delta \\
\alpha_r = \beta - \frac{r \psi}{v}
\]
where \(\beta\) is side-slip angle of the center of mass (rad), \(\psi\) is yaw velocity of the tractor (rad/sec), \(v\) is a tractor forward velocity (m/sec), \(\delta\) is steering angle (rad), and \(l_f\) and \(l_r\) are distances from the center of mass to front and rear axles (m), respectively. Lateral velocity, \(\dot{x}\), of the center of mass can be approximated as
\[
\dot{x} \approx -(\psi + \beta)v
\]
Newton’s law for the lateral forces and Euler equation for the yaw moment yields the equations of motion
\[
M \ddot{x} = -f_f - f_r \\
I_z \ddot{\psi} = l_f f_f - l_r f_r
\]
where \(M\) is tractor mass (kg), \(I_z\) is the yaw moment of inertia (kg-m^2), and \(\ddot{\psi}\) is tractor yaw acceleration (rad/sec^2). Substituting Eqs. (1), (2) and the time derivative of Eq. (3) into Eq. (4) gives the continuous-time state-space form
\[
\dot{x} = Ax + B\delta
\]
where \(x = [\beta, \psi]^T\) is the state vector and
\[
A = \begin{bmatrix} K_f + K_r & M v^2 + f_f l_f - f_r l_r \\
-\frac{f_f l_f - f_r l_r}{I_z} & \frac{M v^2}{I_z} + \frac{f_f^2 l_f^2 + f_r^2 l_r^2}{I_z v} \end{bmatrix} \\
B = \begin{bmatrix} K_f^T \\
\frac{M v}{I_z} \end{bmatrix}
\]
The model has one input and two outputs and is of second order.

Discrete-Time Model

A common situation in computer control is that the D-A converter is constructed so that it holds the analog signal constant until a new conversion is commanded. Therefore, it is natural to choose the sampling instants, $t_k$, as the times when the control changes. The control signal is represented by the sampled signal $\{u(t_k) : k = \ldots, -1, 0, 1, \ldots\}$.

A input-output model has a very convenient structure because it leads to characterization of the input-output behavior based upon the measurements. Assuming that the input and the output are sampled at the same times, the control signal is constant between the sampling constants, and the sampling is periodic with period $h$, the model of Eq. (5) is simplified to the time-invariant zero-order-hold sampled system (Åström and Wittenmark, 1984)

$$x(kh + h) = \Phi^T x(kh) + \Gamma^T \delta(kh)$$
$$y(kh) = Cx(kh)$$

where

$$\Phi_{11}^T = \frac{(A_{22} - p_1)e^{p_1h} - (A_{22} - p_2)e^{p_2h}}{p_2 - p_1}$$
$$\Phi_{12}^T = \frac{-A_{12}(e^{p_1h} - e^{p_2h})}{p_2 - p_1}$$
$$\Phi_{21}^T = \frac{-A_{21}(e^{p_1h} - e^{p_2h})}{p_2 - p_1}$$
$$\Phi_{22}^T = \frac{(A_{11} - p_1)e^{p_1h} - (A_{11} - p_2)e^{p_2h}}{p_2 - p_1}$$
Here $p_1$ and $p_2$ are the eigenvalues of the matrix $\Phi$ and the poles of the system

$$p_{1,2} = \frac{1}{2} \left( (A_{11} + A_{22}) \pm \sqrt{(A_{11} + A_{22})^2 - 4(A_{11}A_{22} - A_{12}A_{21})} \right)^{0.5}$$

For convenience of notation, omitting the sampling interval $h$ from Eq. (6) and rewriting yields

$$x(k+1) = \Phi^t x(k) + \Gamma^t \delta(k)$$

$$y(k) = C x(k) \quad (7)$$

The outputs of the discrete-time system (7) are side-slip angle $\beta$, with respect to tractor-fixed coordinates, and yaw rate $\dot{\psi}$, with respect to the inertial reference frame. To get the outputs of lateral position and yaw angle, the lateral velocity $\dot{x}'$ can be approximated by the kinematic relationship

$$\dot{x}' \approx \beta v \quad (8)$$

Substituting Eq. (8) into Eq. (7) gives

$$x(k+1) = \Phi x(k) + \Gamma \delta(k)$$

$$y(k) = C x(k) \quad (9)$$
where

\[
\begin{align*}
x(k) &= [x'(k), \psi(k)] \\
\Phi &= \begin{bmatrix}
\Phi_{11}^T & \Phi_{12}^T \\
\Phi_{21} & \Phi_{22}
\end{bmatrix} \\
\Gamma &= \begin{bmatrix}
\Gamma_1^T \\
\Gamma_2^T
\end{bmatrix}
\end{align*}
\]

The difference equation for lateral position and yaw angle can be approximated by using Tustin's approximation method (Åström and Wittenmark, 1984)

\[
\frac{dx(t)}{dt} \approx \frac{2}{h} \left( 1 - q^{-1} \right) \frac{1}{1 + q^{-1}}
\]  

which corresponds to the trapezoidal rule for numerical integration. The backward-shift operator is denoted by \(q^{-1}\), and has the property

\[
q^{-1}g(k) = g(k - 1)
\]

Because Tustin's approximation has the advantage that the left-s-plane is transformed into the unit circle in z-domain, a stable discrete-time system can be obtained. Applying Eq. (10) into Eq. (9) yields

\[
\begin{align*}
x(k + 1) &= \Phi x(k) + \frac{h(1 + q^{-1})}{2(1 - q^{-1})} \Gamma \delta(k) \\
y(k) &= C x(k)
\end{align*}
\]

where \(x(k) = [x'(k), \psi(k)]^T\).

To obtain the input-output relationship, the state vector must be eliminated.
Using Eqs. (11) and (12) gives the pulse transfer function

\[ H(q) = \frac{Y(k)}{\delta(k)} = \frac{q^{-1} \left[ b_{11} + b_{12}q^{-1} + b_{13}q^{-2} \right]}{1 + a_1q^{-1} + a_2q^{-2} + a_3q^{-3}} = \frac{q^{-1} B(q^{-1})}{A(q^{-1})} \]  

(13)

where

\[
\begin{align*}
    b_{11} & = \frac{\Gamma_1 h}{2} \\
    b_{12} & = \beta_{11} \left( 1 - \Phi_{22} + \frac{\Phi_{12}\Gamma_2}{\Gamma_1} \right) \\
    b_{13} & = \beta_{11} \left( \frac{\Phi_{12}\Gamma_2}{\Gamma_1} - \Phi_{22} \right) \\
    b_{21} & = \frac{\Gamma_2 h}{2} \\
    b_{22} & = \beta_{21} \left( 1 - \Phi_{11} + \frac{\Phi_{21}\Gamma_2}{\Gamma_1} \right) \\
    b_{23} & = \beta_{21} \left( \frac{\Phi_{21}\Gamma_2}{\Gamma_1} - \Phi_{11} \right) \\
    a_1 & = -\left( 1 + \Phi_{11} + \Phi_{22} \right) \\
    a_2 & = \left( \Phi_{11} + \Phi_{22} - \Phi_{12}\Phi_{21} \right) \\
    a_3 & = \Phi_{12}\Phi_{21}
\end{align*}
\]

Therefore, the model (13) gives input-output relationship in polynomial form

\[ A(q^{-1})y(k) = B(q^{-1})\delta(k - 1) \]  

(14)

where \( A \) is a diagonal matrix whose elements are

\[ a_{11} = a_{22} = 1 + a_1q^{-1} + a_2q^{-2} + a_3q^{-3} \]
Model Modification for Tractor Guidance

The discrete-time tractor model (14) has a two-dimensional form of one input and two outputs. This is not an appropriate form for a self-tuning steering controller for a tractor. Although self-tuning control algorithms with a multi-dimensional model have been studied and developed intensively, it is still hard to implement for practical application because theoretical research has not reached its final form. Furthermore, the algorithms are so complicated and time-consuming that they may not be useful in practical point of view. Therefore, reducing them to simpler forms would be desirable if it does not sacrifice dynamic characteristics and stability of closed-loop system.

Rewriting the model (14) explicitly yields

\[ y(k) = -a_1 y(k - 1) - a_2 y(k - 2) - a_3 y(k - 3) - b_1 \delta(k - 1) - b_2 \delta(k - 2) - b_3 \delta(k - 3) \]  

(15)

where \( y(i) = [x'(i), \psi(i)]^T \) and \( b_i = [b_{1i}, b_{2i}]^T \). Multiplying \( b_i^T \) in both sides of (15) yields a scalar input-output model

\[ y(k) = -a_1 y(k - 1) - a_2 y(k - 2) - a_3 y(k - 3) + b_1 \delta(k - 1) + b_2 \delta(k - 2) + b_3 \delta(k - 3) \]  

(16)

where \( y(i) = b_1^T y(i) \) and \( b_i = b_1^T b_i \). If the vector \( b_1 \) is known, \( y(i) \) is a scalar variable which is a linear function of \( y(i) \). With the same sign convention for the lateral position \( (x') \) and the yaw angle \( (\psi) \), the single-input single-output model (16) can be substituted for the single-input two-output model (15).

Equation (16) has the form of a servo model, i.e., the task is to make output variables \( x' \) and \( \psi \) responsive to changes in input signal \( u \). In design of steering
controllers for tractor guidance, the steering angle $\delta$ is the input for the tractor, but the control variable for the entire control system. For a steering controller using leader-cable system, the control objective is to nullify the position and yaw errors which are directly determined by the sensing system. If a global positioning system is used to guide the tractor, however, the model can be described as a servo problem because the input of the control system is the desired tractor path and the output is the actual path. Considering that the output variable is the error between desired and actual paths, it can also be regarded as a regulation problem.

Substituting reference variables $\delta_r$ and $y_r$ into the model (16) yields

$$y_r(k) = -a_1y_r(k-1) - a_2y_r(k-2) - a_3y_r(k-3) + b_1\delta_r(k-1) + b_2\delta_r(k-2) + b_3\delta_r(k-3)$$

where $y_r(i) = b_1^Ty_r(i)$. Subtracting Eq. (16) from Eq. (17) makes regulator problem

$$y_e(k) = -a_1y_e(k-1) - a_2y_e(k-2) - a_3y_e(k-3) + b_1\delta_e(k-1) + b_2\delta_e(k-2) + b_3\delta_e(k-3)$$

where $y_e(i) = y_r(i) - y(i)$ and $\delta_e(i) = \delta_r(i) - \delta(i)$.

In practical aspects, the steering angle error $\delta_e(i)$ is the difference between $\delta(i)$ and $\delta(i-1)$ because the desired steering angle $\delta_r(i)$ is a control variable. In addition, the regulator model (18) may have some problems on the parameter estimation routine. A fundamental result of system identification theory shows that the input signal to the system must be “persistently exciting” or “sufficiently rich” (Åström and Bohlin, 1966). In the adaptive systems whose input signal is generated by feedback, there is no guarantee that the system will be properly excited. If the input signal
is weak, the estimated parameters will be poor. The input signal $\delta_e$ of model (18) may not be rich under normal operation because the steering angle in radian is used. Therefore, if the steering angle error of model (18) is replaced by the steering angle, the final form is a mixed servo and regulator model

$$y_e(k) = -a_1y_e(k - 1) - a_2y_e(k - 2) - a_3y_e(k - 3)$$

$$+ b_1 \delta(k - 1) + b_2 \delta(k - 2) + b_3 \delta(k - 3)$$

(19)

where $y_e = b_1^T [x_r - x, \psi - \psi]^T = b_1^T [\Delta y, \Delta \psi]^T$.

The sign for the lateral position error $\Delta y$ was considered positive if the center of front wheel was located on the left side of the desired path. The clockwise rotation of the yaw angle error $\Delta \psi$ with respect to the desired path was positive. With $b_1^T = [1, 1]$, the output error is the yaw angle error when the lateral position error is equal to zero. Therefore, the steering signal is proportional to the magnitude of yaw angle errors. When the signs of both position and yaw errors are the same, the tractor tends to move away from the desired path. The output error, the sum of position and yaw errors, will generate a big steering command to correct the deviation of the tractor from the desired path. When signs are opposite, the tractor is approaching the desired path. In this situation, the steering command is small because the output error $y_e$ is small. The criterion of this method is when both signs are opposite and their magnitudes are the same. In this case, the output error is equal to zero. Because this criterion depends upon the tractor speed, it can be adjusted by modifying the gain vector $b_1$ to yield the best tracking.
RECURSIVE PARAMETER ESTIMATION

For the identification of parametric system models the use of parameter estimation methods is straightforward. Much research has been done to study system parameter estimation techniques (Åström and Eykhoff, 1971; Isermann, 1982; Åström et al., 1977). Many textbooks also discuss them from theoretical and practical points of view (Isermann, 1980; Ljung, 1987; Ljung and Söderström, 1984; Mendel, 1973).

In on-line systems, the model is needed to support decisions during the operation. Then the model should be inferred and updated at the same time as the new data become available. Recursive parameter estimation methods are best suited for these operations in real time because the observations are generally obtained in sequential order, and the results obtained for \( k \) observations are used to get the estimates for the \( k+1 \) observation. Therefore, these methods have been applied to design adaptive controllers for time-variant systems.

There are several methods for recursive parameter estimation in open-loop systems; recursive least-squares (RLS), recursive extended least squares (RELS), recursive instrumental variables (RIV), and recursive maximum likelihood (RML). RLS algorithm is the most widely used method because it is robust and easily implemented.
Recursive Least-Squares Method

It is assumed that the system to be controlled can be described stochastically by

\[ A(q^{-1})y(t) = B(q^{-1})u(t - d) + C(q^{-1})e(t) \]  \hspace{1cm} (20)

where

\[ A(q^{-1}) = 1 + \alpha_1 q^{-1} + \cdots + \alpha_p q^{-p} \]
\[ B(q^{-1}) = \beta_1 + \beta_2 q^{-1} + \cdots + \beta_r q^{-r+1} \]
\[ C(q^{-1}) = 1 + \gamma_1 q^{-1} + \cdots + \gamma_p q^{-p} \]

are polynomials in the backward shift operator \( q^{-1} \). \( u \) and \( y \) are scalar input and output signals, \( e(t) \) is a disturbance which is a sequence of independent random variables, and \( d \) is a discrete time delay.

Assuming that no disturbance exists, the prediction model for the least-squares method can be written

\[ \hat{y}(t) = -A(q^{-1})y(t - 1) + B(q^{-1})u(t - d) \]  \hspace{1cm} (21)

where \( \hat{y} \) is the predicted output, and

\[ A(q^{-1}) = \alpha_1 + \alpha_2 q^{-1} + \cdots + \alpha_p q^{-p+1} \]
\[ B(q^{-1}) = \beta_1 + \beta_2 q^{-1} + \cdots + \beta_r q^{-r+1} \]

are also polynomials in backward shift operator \( q^{-1} \). For convenience, let's define

\[
\phi(t) = [-y(t - 1), \ldots, -y(t - p), u(t - d), \ldots, u(t - d - r + 1)]^T \\
\theta = [\alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_r]^T
\]
Then, Eq. (21) can be written

\[ \hat{y}(t) = \phi^T(t) \theta \]

According to Gauss the principle of least squares is that the unknown parameters of a model should be chosen in such a way that the sum of the squares of the difference between the actually observed and computed values multiplied by numbers that measure the degree of precision is a minimum (Åström and Wittenmark, 1984). Therefore, the least squares method is based on the minimization of the functional

\[ J(\theta) = \sum_{i=1}^{k} \lambda^{k-i}\|y(i) - \phi^T(i)\theta\|^2 \]  

(22)

where \( \lambda \) is a weight coefficient called forgetting factor. If the parameters are time-varying, a forgetting factor of less than one lets the model track variations by eliminating the influence of old data exponentially with a factor of \( \lambda \). Minimizing the functional (22) with respect to the parameter vector \( \theta \) gives the estimate

\[ \hat{\theta}(k) = \left[ \sum_{i=1}^{k} \lambda^{k-i} \phi(i) \phi^T(i) \right]^{-1} \sum_{i=1}^{k} \lambda^{k-i} \phi(i)y(i) \]  

(23)

provided the inverse exists. If the inverse matrix is nonsingular, then the functional \( J \) has a minimum and the minimum is unique.

Noting that Eq. (23) is not written recursively, define

\[
R(k) = \sum_{i=1}^{k} \lambda^{k-i} \phi(i) \phi^T(i) \\
P(k) = R^{-1}(k) \\
L(k) = \lambda R^{-1}(k) \phi(k) = \lambda P(k) \phi(k) 
\]

(24)

Then the matrix \( P(k) \) can be updated instead of inverting the matrix \( R(k) \). With the aid of matrix inversion lemma (Ljung, 1987), Eqs. (23) and (24) can be modified
to yield the RLS algorithm (Ljung and Söderström, 1984)

\[
\hat{\theta}(k) = \hat{\theta}(k-1) + L(k) \left( y(k) - \phi^T(k) \hat{\theta}(k-1) \right)
\]  
\[
L(k) = \frac{P(k-1)\phi(k)}{1/\lambda + \phi^T(k)P(k-1)\phi(k)}
\]  
\[
P(k) = \left( I - L(k)\phi^T(k) \right) P(k-1)
\]  

A matrix inversion is also necessary to compute Eq. (26), but the denominator of \( L(k) \) has the same dimension as the number of measurements. Hence, it is a scalar for a single-output system.

The RLS method is a good and simple identification method. Its main disadvantage is that it can give biased estimates unless the true system can be described by Eq. (20) with \( C(q^{-1}) = 1 \) (Åström, 1983; Kurz et al., 1980). As the parameter converges with recursive least-squares estimation, however, Fortescue et al. (1981) reported that better control can be achieved and the optimal control law can even result in the case of convergence to biased estimates of the system parameters. To avoid the bias problem, an extended least-squares model suggested by Panuska (1969) can be used. The RELS method is exactly the same as RLS method, except that the estimated prediction errors are included in the parameter vector \( \theta \).

**Variable Forgetting Factors**

The original self-tuning regulator was designed to operate on a system with constant but unknown parameters (Åström and Wittenmark, 1973). In such a time-invariant system, a forgetting factor of unity is used so that the past information is not discounted. However, most systems exhibit time-varying and nonlinear dynamics which violates the assumption of a linear, time-invariant system. To track time-
varying parameters, the measurements should be weighted properly. Generally, the
forgetting factor is chosen between zero and unity.

A smaller value of \( \lambda \) gives a faster response for the system change but a large
steady-state variance. A higher value near unity results in slow changes of param­
eters due to the slow forgetting of old information. Although a system is stable
under normal operation, the matrix \( P \) will increase exponentially with a fixed \( \lambda \).
The very large matrix \( P \) may lead to a system which is extremely sensitive to dis­
turbances and susceptible to numerical and computational difficulties (Åström and
Wittenmark, 1980). This burst can happen when the parameters are estimated with
constant forgetting factor in an open-loop unstable system. Therefore, the parameter
estimation routine requires an algorithm to control the forgetting factor depending
on system conditions.

To set the forgetting factor properly, the prime interest is the minimal squared
residual error of the functional (22)

\[
E(k) = \inf_{\hat{\theta}} \sum_{i=1}^{k} \lambda^{k-i} \| y(i) - \phi^T(i) \hat{\theta} \|^2 = \sum_{i=1}^{k} \lambda^{k-i} \| y(i) - \phi^T(i) \hat{\theta}(k) \|^2
\]

where \( \hat{\theta} \) is the minimum solution. Albert and Sittler (1966) proved that \( E(k) \)
related to \( E(k-1) \) by a recursion of the form

\[
E(k) = \lambda E(k-1) + \frac{\| y(k) - \phi^T(k) \hat{\theta}(k-1) \|^2}{1 + \phi^T(k)P(k-1)\phi(k)}
\]  \hfill (28)

A strategy for choosing a forgetting factor was proposed by Fortescue et al.
(1981), keeping residual errors such that

\[
E(k) = E(k-1) = \cdots = E_0
\]
Then, the amount of forgetting at each step will correspond to the amount of new information in the latest measurement, ensuring that the estimation is always based on the same amount of information. Equation (28) can be rewritten, in terms of $\lambda$, as

$$\lambda(k) = 1 - \frac{1}{N(k)}$$  \hspace{1cm} (29)

where

$$N(k) = \frac{(1 + \phi^T(k)P(k-1)\phi(k)) E_0}{\epsilon(k)^2}$$  \hspace{1cm} (30)

A memory time constant $N$ indicates that the information dies away with a time constant of $N$ sample intervals (Clarke and Gawthrop, 1975).

If the matrix $P$ goes to infinity or zero, the system becomes unstable. To avoid problems related to the matrix $P$, an ad hoc method which constrains the trace of $P$ to a certain boundary are considered here. Equation (27) are modified as

$$P(k) = \left( I - L(k)\phi^T(k) \right) P(k-1)$$  \hspace{1cm} (31)

If \( \frac{\text{trace} \ P(k)}{\lambda(k)} \leq C_0 \)

$$P(k) = \frac{P(k)}{\lambda(k)}$$

where $C_0$ is a constant value which can be determined by experiment or simulation. If the system changes suddenly, the memory time constant $N$ of Eq. (30) decreases by the increased prediction error $\epsilon$. This decreases forgetting factor $\lambda$ of Eq. (29). The matrix $P$ decreases rapidly, indicating that the current and the previous measurements have little correlation. If the change of system continues, it tends to be a singular matrix which results in complete instability. To prevent the system from
being unstable, matrix $\mathbf{P}$ is increased by dividing it by the decreased forgetting factor $\lambda$, which increases matrix $\mathbf{L}$. This increased gain matrix leads to rapid adaptation of the system. During normal steady-state operation, $\lambda$ becomes near one. Thus, $N$ increases and the matrix $\mathbf{P}$ remains constant.

**Practical Estimation Technique**

The purpose of self-tuning controller is to keep systems unattended over long periods of time without intervention of human operator. This puts an emphasis on the robustness of the estimation algorithm. There are some difficulties with the use of recursive least-squares method as formulated in Eqs. (25), (26), and (31).

Equation (31) for updating $\mathbf{P}$ is numerically ill-conditioned because $\mathbf{P}$ may become negative definite. The result is instability in the parameter estimates which will not be removed until a sufficiently large control signal is sent out to make $\mathbf{P}$ positive definite again. This control signal will be enough to affect the system output seriously and result in poor closed-loop performance.

The solutions to this numerical problem depend on the factorization of $\mathbf{P}$ into an upper triangular matrix and its transpose, which is possible due to the symmetric and positive-definite matrix $\mathbf{P}$. This method called square root factorization was proposed by Potter, who recognized that numerical instability in Kalman-filter algorithm is often accompanied by a computed covariance matrix $\mathbf{P}$ that loses its positiveness (Battin, 1964). The algorithm requires $(4m^2+5m)/2$ multiplications plus $m$ square root calculations, where $m$ is the number of estimated parameter (Clarke, 1980).

To eliminate the square root calculation, $\mathbf{UD}$ factorization method was developed by Bierman (1977). The matrix $\mathbf{P}$ is factorized as $\mathbf{UDU}^T$ where $\mathbf{U}$ is an
upper triangular matrix with unities on the diagonal and D is a diagonal matrix corresponding to the variance of the individual parameter estimates. This algorithm requires only \((3m^2+3m)/2\) multiplications. In addition, the diagonal matrix D can be used for diagnostic purpose without extra computations that would be required in the square root method. Therefore, the UD factorization method is preferable when using recursive parameter estimation.

In the tracking problem, a large output error can occur from the poorly estimated parameters when the input signal is not rich. To get a better tracking satisfying the design criterion, the proportional gain may be considered to amplify the controller output \(\delta(k)\). Because this violates the idea of minimum variance control, however, the controller becomes unstable. To eliminate this problem, a proportional gain can be inserted in the parameter estimation routine to amplify the output error of the system. This can reduce tracking errors by increasing the adaptation rate of the system.
SELF-TUNING STEERING CONTROLLER

Some important theoretical problems related to self-tuning control strategy have been solved during the last years. There has been considerable progress in stability and convergence proofs for simple self-tuning control algorithms (Egardt, 1980; Goodwin et al., 1980; Morse, 1980). Nevertheless, there are no results available for more realistic cases, and active research is being done in this area.

It is important to consider both theoretical and practical aspects when implementing self-tuning control algorithms. The theory deals with ideal situations where all conditions hold under the assumptions made. In practical applications, however, the assumptions made for development of theory are frequently violated.

Controller Design Criterion

Various forms of self-tuning control schemes have been designed. These variations usually lie in the choice of recursive parameter estimator, but the various estimation methods have the same structure stemming from the recursive least-squares method. The situation for designing the controller algorithm is different. Depending upon the choice of design criterion, the controllers have different performance and characteristics. The frequently used criterion are pole-placement method (Zarrop and Fischer, 1985; Åström and Wittenmark, 1980), linear quadratic Gaussian (LQG) con-
trol (Clarke and Gawthrop, 1975), and minimum-variance (MV) control (Åström et al., 1977; Åström and Wittenmark, 1973).

The design principle of the pole-placement method is to assign the closed-loop poles and zeros of the system to desired positions and to set parameters of the controller by solving the linear polynomial equation resulting from pole-zero assignments. Since this method is not based on the optimization technique of the cost function related to system inputs or outputs, it does not yield an asymptotically optimal system. Therefore, the regulation performance is much worse than the other control schemes. But, it has the advantage that the system's closed-loop response is specified by the user, when the desired pulse-transfer function of the system is known, and is not sensitive to the large control signal (Wellstead et al., 1979).

For the system model (20), the linear quadratic Gaussian control is to minimize the cost function (Davis and Vinter, 1985)

\[
J(u) = \lim_{N \to \infty} E \left\{ \frac{1}{N} \sum_{k=1}^{N} \left( y^2(k) + \lambda u^2(k) \right) \right\}
\]  

(32)

where \( E \) represents the expectation or the mean value function of a random variable and \( \lambda \) is a forgetting factor. With \( 0 < \lambda < 1 \) the functional is quadratic when the model (20) is represented in state-space form. The minimization of the functional (32) requires spectral factorization or steady-state solution of Riccati equations. The result is a controller whose parameters are functions of the parameters of the model (20). The LQG controller has several good properties. It is applicable to multi-variable and time-variant systems. Moreover, it always gives a stable closed-loop system if the quadratic function is symmetric and positive definite, and if the system is reachable, i.e., if it is possible to find a control sequence such that an
arbitrary state can be reached from any initial state in finite time (Åström and Wittenmark, 1984). However, there are several practical problems: it is hard to compute because of complexity, and to determine the weightings in the cost function.

When there is no weighting in the functional (32), i.e., \( \lambda = 0 \), the cost functional only includes the variance of the output. The controller obtained by minimizing it is called minimum-variance controller. Therefore, the MV controller can be considered as a special case of LQG controllers. The most important property of the MV control is that it can yield an excellent self-tuning controller because the control sequences are made to keep the output as small as possible. This strategy can be applied to the input-output model with unknown parameters and easily extended to mixed servo and regulator problems by small modifications. It has also disadvantage that a large control signal, which is unrealistic in practical cases, may be generated if the output is suddenly changed to large extent. However, due to the good self-tuning property under normal operation and design flexibilities, the MV control technique has a broad application, and has been widely used when implementing a self-tuning regulator.

**Minimum Variance Controller**

It is assumed that the system to be regulated is a sampled linear stochastic and has single-input and single-output. Then, the input-output model (20) can be rewritten by the difference equation

\[
A(q^{-1})y(k) = B(q^{-1})u(k - d) + C(q^{-1})e(k)
\]  

(33)

where polynomials \( A(q^{-1}) \), \( B(q^{-1}) \), and \( C(q^{-1}) \) are defined by

\[
A(q^{-1}) = 1 + a_1q^{-1} + \cdots + a_nq^{-n}
\]
\[ B(q^{-1}) = b_0 + b_1 q^{-1} + \cdots + b_m q^{-m} \quad (b_0 \neq 0) \]
\[ C(q^{-1}) = 1 + c_1 q^{-1} + \cdots + c_n q^{-n} \]

and \( e(k) \) is a sequence of independent normal random variables with zero mean and \( \sigma^2 \) variance. The control criterion is to determine the signal \( u(k) \) such that the variance of the output is as small as possible. The following assumptions are made about the system:

1. The discrete time delay \( d \) is known.
2. The upper bound for \( n \) and \( m \) is known.
3. \( B(q^{-1}) \) and \( C(q^{-1}) \) have all zeros inside the unit circle.
4. The value of \( b_0 \) is known.

The minimum variance controller can be derived by using the polynomial identity (Peterka, 1972)
\[ C(q^{-1}) = A(q^{-1})F(q^{-1}) + q^{-d}G(q^{-1}) \]  
(34)

where \( F \) and \( G \) are polynomials of degrees \( d - 1 \) and \( n - 1 \) defined by
\[ F(q^{-1}) = 1 + f_1 q^{-1} + \cdots + f_{d-1} q^{1-d} \]
\[ G(q^{-1}) = g_0 + g_1 q^{-1} + \cdots + g_{n-1} q^{1-n} \]

Some manipulation of Eq. (33) with Eq. (34) gives the minimum variance controller (Åström, 1970)
\[ u(k) = -\frac{G(q^{-1})}{B(q^{-1})F(q^{-1})} y(k) \]  
(35)
An advantage of the minimum-variance controller is that it is extremely easy to compute on-line, because the recursion (35) expresses the current control signal \( u(k) \) as a linear combination of a finite number of previous inputs and outputs. For minimum-phase systems (the polynomial \( B \) has stable zeros inside the unit circle), the MV controller gives the signal \( u(k) \) which is asymptotically stable. In case of non-minimum phase systems (the polynomial \( B \) has unstable zeros outside the unit circle), the controller (35) can not be used because the control signal may diverge. However, if the suboptimal strategy of factorizing \( B \) into stable and unstable zeros is used, the unstable zeros outside the unit circle can be moved to the origin so that the closed-loop system is stable (Åström and Wittenmark, 1984).

For a single-input single-output, deterministic, discrete time, time-invariant system, the Eq. (33) is simplified as

\[
A(q^{-1})y(k) = B(q^{-1})u(k - d) \tag{36}
\]

By repeated substitution Eq. (36) can be written as

\[
y(k + d) = \alpha(q^{-1})y(k) + \beta(q^{-1})u(k) \tag{37}
\]

where

\[
\alpha(q^{-1}) = \alpha_0 + \alpha_1 q^{-1} + \cdots + \alpha_{n-1} q^{-n+1}
\]
\[
\beta(q^{-1}) = \beta_0 + \beta_1 q^{-1} + \cdots + \beta_{m+d-1} q^{-m-d+1} \quad \beta_0 \neq 0
\]

The advantage of the assumption \( \beta_0 \neq 0 \) is to avoid division by zero in the calculation of the input \( u(k) \) and numerical instability by keeping the zeros of \( \beta(q^{-1}) \) inside the unit circle. For estimation purpose, Eq. (37) is rewritten as

\[
y(k + d) = \phi^T(k)\theta + \beta_0 u(k)
\]
where $\phi(k)$ and $\theta$ are defined by

$$
\phi(k) = [y(k), \ldots, y(k-n+1), u(k-1), \ldots, u(k-m-d+1)]^T
$$

$$
\theta = [\alpha_0, \ldots, \alpha_{n-1}, \beta_1, \ldots, \beta_{m+d-1}]^T
$$

If the estimated $\hat{\beta}_0$ is known by experiment or simulation, then the minimum variance controller $u(k)$ is simply

$$
u(k) = \frac{-\phi^T(k)\hat{\theta}}{\hat{\beta}_0}
$$

Goodwin et al. (1980) proved that $y(k)$ and $u(k)$ of the model (37) are bounded and

$$
\lim_{k \to \infty} [y(k) - y_r(k)] = 0
$$

if the above assumptions hold and if $0 < \beta_0/\hat{\beta}_0 < 2$. The condition $0 < \beta_0/\hat{\beta}_0 < 2$ can always be satisfied if the sign $\beta_0$ and an upper bound for the magnitude of $\beta_0$ are known.

In summary, the block diagram of tractor self-tuning steering controller is shown in Fig. 3.2. Assuming that at time $k$ the measurements $y_e(k), y_e(k-1), \cdots$ and all previous control signals $\delta(k-1), \delta(k-2), \cdots$ are known, two estimation models with no-step-ahead and one-step-ahead path predictions are implemented to determine the control signal $\delta(k)$. The no-step-ahead model estimates the parameters by using the measurements up to current time, while the one-step-ahead model utilizes an additional one-step-ahead predicted path error for parameter estimation. Two kinds of one-step-ahead prediction can be considered here: one-step-ahead prediction of path error with and without one-step-ahead desired path data. With one-step-ahead tractor position and yaw angle predicted from the current position, the former calculates the path error with respect to the desired one-step-ahead path, and the latter
predicts it with the path data up to current time. These prediction methods can be useful for the tractor guidance system that uses a global positioning system.

With the model (19) and initial data \( \hat{b}_1, E_0, K_0, C_0, P_0 \) (or \( D_0 \)), \( \hat{\theta}(0) \), these methods are described as follows:

1. Compute the prediction error \( e(k) \).

\[
e(k) = y_m(k) - \phi^T(k - 1)\hat{\theta}(k - 1) - \hat{b}_1 \delta(k - 1)
\]

where \( y_m(k) = K_0 y_e(k) \) and

\[
\phi(k - 1) = [-y_e(k - 1), -y_e(k - 2), -y_e(k - 3), \delta(k - 2), \delta(k - 3)]
\]

\[
\hat{\theta}(k - 1) = [\hat{a}_1(k - 1), \hat{a}_2(k - 1), \hat{a}_3(k - 1), \hat{b}_2(k - 1), \hat{b}_3(k - 1)]
\]

In step-ahead prediction, the prediction error is

\[
\hat{e}(k + 1) = \hat{y}_m(k + 1) - \phi^T(k)\hat{\theta}(k - 1)
\]

where \( \hat{y}_m(k + 1) = K_0 \hat{y}_e(k + 1) \).

2. Compute the gain matrix \( L(k) \). This step can be implemented by using UD factorization method.

\[
L(k) = \frac{P(k - 1)\phi(k - 1)}{1/\lambda(k - 1) + \phi^T(k - 1)P(k - 1)\phi(k - 1)}
\]

In step-ahead prediction, \( \phi(k - 1) \) is substituted to \( \phi(k) \).

3. Update the parameter vector \( \hat{\theta} \). \( \hat{e}(k + 1) \) is used in step-ahead prediction.

\[
\hat{\theta}(k) = \hat{\theta}(k - 1) + L(k)\hat{e}(k)
\]
4. Compute the memory time constant $N(k)$. $\phi(k)$ and $\hat{\phi}(k+1)$ replace $\phi(k-1)$ and $\hat{\phi}(k)$ in step-ahead prediction.

$$N(k) = \frac{(1 + \phi^T(k-1)P(k-1)\phi(k-1))E_0}{\epsilon(k)^2}$$

5. Set the forgetting factor $\lambda(k)$.

$$\lambda(k) = 1 - \frac{1}{N(k)}$$

6. Update the covariance matrix $P$.

$$P(k) = \left( I - L(k)\phi^T(k-1) \right) P(k-1)$$

If $\frac{\text{trace}\ P(k)}{\lambda(k)} \leq C_0$

$$P(k) = \frac{P(k)}{\lambda(k)}$$

If UD factorization method is used, the diagonal matrix $D(k)$ obtained from step 2 are updated here.

If $\frac{\text{trace}\ D(k)}{\lambda(k)} \leq C_0$

$$D(k) = \frac{D(k)}{\lambda(k)}$$

7. Determine the control signal $\delta(k)$ based on the estimated parameters.

$$\delta(k) = -\frac{\phi^T(k)\hat{\theta}(k)}{\hat{b}_1}$$
Figure 3.2: Block diagram of self-tuning tractor steering controller. $e$ is disturbances, $y$ and $y_r$ are actual and reference paths, $y_e$ and $y_m$ are path error and amplified error signal.
COMPUTER SIMULATION

Startup

The self-tuning steering controller was evaluated with the tractor dynamic simulator developed in Part II. The controller program listed in Appendix was used with source codes of Appendix C of Part II.

As shown in Fig. 3.3, a composite test path including lane change and sinusoidal maneuvers was generated by feeding user-supplied steering inputs to the tractor dynamic simulator. The tractor speed was constant at maximum design speed 18 km/h (5 m/sec) during 70 seconds, and the steering input used was

$$\delta(t) = \begin{cases} 
0 & 0 \leq t \leq 2, \quad 10 \leq t \leq 12, \\
0.0715 \sin \left( \frac{\pi}{4} (t - 2) \right) & 16 \leq t \leq 18, \quad 66 \leq t \leq 70 \\
0.1 \sin \left( \frac{\pi}{2} (t - 12) \right) & 2 \leq t \leq 10 \\
0.0857 \sin \left( \frac{\pi}{3} (t - 18) \right) & 12 \leq t \leq 16 \\
\end{cases} \quad (38)$$

During some periods of the time $18 \leq t \leq 66$, the sign of steering inputs was reversed to make alternating sinusoidal paths. The UD factorization was used for parameter estimation. Three deterministic models with no-step-ahead and one-step-ahead predictions were tested with initial data $\hat{b}_1 = 2, \ E_0 = 0.001, \ K_0 = 6, \ C_0 = 100,$
\[ D_0 = 50I, \text{ and } \dot{\theta}(0) = 0. \] Because pre-simulation showed that the execution time of the controller program was about 4 msec when 16-MHz 80386 microprocessor was used, a sampling interval was set to 0.1 second. A steering command was implemented by the exponential function

\[ \delta(t) = \delta(t - 1) + \Delta\delta(k) \left( 1.0 - e^{-50h} \right) \] (39)

where \( \Delta\delta(k) = \delta(k) - \delta(k - 1) \) is the steering command to tractor and \( h \) is a time increment used in tractor dynamic simulator.

As expected, the position error decreased as path errors were predicted more accurately. Figure 3.4 indicates that the smallest position error is obtained by predicting path deviation with respect to known one-step-ahead desired path. For poor initial parameter values, Figs. 3.5 and 3.6 show how model parameters adapt to tractor movements. In the model predicting path error without one-step-ahead desired path, the parameters already started converging after two lane change maneuvers because these maneuvers provided enough information for path curvatures and position errors. If the radius of curvature becomes shorter than that of the given test path, parameters adapt to new environments and path errors increase. When one-step-ahead desired path is known, however, the path error can be predicted more accurately than when it is computed based upon the known paths up to the current time. Therefore, the actual path error is reduced because this predicted error directly forces parameters to adapt in response to environment changes. This situation illustrates that the prediction of path errors depends on the use of future desired paths, not error prediction scheme of the original self-tuner which uses the paths and measurements up to the current time. The fluctuation of parameters shown in Fig. 3.6 results from this fact.
Other supports for this fact can be found through Figs. 3.7 and 3.8. Sudden drops of forgetting factor and decreases of covariance matrix $D$ indicate that prediction model tends to be very sensitive as the radius of curvature suddenly becomes shorter. Dividing $D$ by forgetting factor $\lambda$ increases matrix $D$, which increases gain vector $L$ to adapt parameters rapidly. Steering angle command set by the controller has the same tendency. When the path error is predicted with path data up to the current time, the steering command shown in Fig. 3.9 is a smooth and continuous curve. However, Fig. 3.10 shows that the steering command signal is fluctuated when the path error is predicted based upon one-step-ahead desired path. This is also due to the uncertainty of one-step-ahead prediction of path errors.

A stochastic model was implemented to see effects of measurement errors or noises. A sequence of normal pseudo-random numbers with zero mean and variance 0.02 was generated, which corresponded to $\pm 5$ cm position error. At every sampling instant two random numbers were added to actual $x$ and $y$ positions of tractor. The path error was computed from the one-step-ahead desired path and the tractor position predicted with respect to current disturbed position. The initial data were $\hat{b}_1 = 2$, $E_0 = 0.001$, $K_0 = 4$, $C_0 = 100$, and

$$\hat{\theta}(0) = \begin{bmatrix} -0.904, 0.261, 0.570, -0.830, 0.524 \end{bmatrix}^T$$

$$D_d = [0.005, 2.683, 2.207, 260.910, 1281.418]$$

where $D_d$ is diagonals of matrix $D$. To allow more time delays for steering command, a time constant of the driver model (39) was set to 0.05 instead of 0.02. Figures 3.11 and 3.12 show the position error and the steering command for the stochastic model. Maximum absolute position error was 9.23 cm, but the percentage of position errors bigger than 5 cm was relatively small 6.01%. Because of random deviations of
Table 3.1: Maximum, mean, standard deviation, and RMS values for absolute position errors of the center of tractor front wheel at speed 18 km/h

<table>
<thead>
<tr>
<th>Model</th>
<th>Maximum cm</th>
<th>Mean cm</th>
<th>STD cm</th>
<th>RMS cm</th>
<th>5 cm error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-step-ahead</td>
<td>6.10</td>
<td>2.80</td>
<td>1.82</td>
<td>3.33</td>
<td>8.16</td>
</tr>
<tr>
<td>Step-ahead $(y_T(k))^c$</td>
<td>3.39</td>
<td>1.49</td>
<td>0.94</td>
<td>1.75</td>
<td>0.00</td>
</tr>
<tr>
<td>Step-ahead $(y_T(k+1))^d$</td>
<td>1.46</td>
<td>0.26</td>
<td>0.21</td>
<td>0.34</td>
<td>0.00</td>
</tr>
<tr>
<td>Stochastic</td>
<td>9.23</td>
<td>2.20</td>
<td>1.59</td>
<td>2.72</td>
<td>6.01</td>
</tr>
</tbody>
</table>

\(^a\)Root Mean Square error = $\sqrt{\frac{\sum y^2}{N}}$ where $N$ is the number of data points.

\(^b\)Indicates the percentage of absolute position errors bigger than 5 cm.

\(^c\)Path error prediction by using paths up to current time.

\(^d\)Path error prediction by using one-step-ahead desired path.

tractor position from the desired path, the steering command had more fluctuation compared with that obtained from the deterministic model. However, it appears that implementation of this steering command into the actuating signal does not have major difficulties because the signal tends to be attenuated more or less in practical worlds.

For the deterministic and stochastic prediction models used, Table 3.1 summarizes maximum, mean, RMS, and 5 cm error values, and standard deviation for absolute position errors of the center of tractor front wheel at tractor speed 18 km/h. It shows that one-step-ahead deterministic prediction models can control the tractor within 5 cm position error.

The proportional gain $K_0$ for path error was directly related to reduction of the path error. For both deterministic and stochastic models at the speed 18 km/h, the maximum $K_0 = 8$ could be used to reduce path errors without instability of the controller. In stochastic case, however, the lower gain around 4 gave the stable steer-
Figure 3.3: A composite desired path to test self-tuning steering control algorithm

Figure 3.4: Position error of the center of front wheel for three estimation methods
Figure 3.5: Parameter estimates of the model predicting path error without one-step-ahead desired path

Figure 3.6: Parameter estimates of the model predicting path error with one-step-ahead desired path
Figure 3.7: Variable forgetting factor in response to path changes in the prediction model with one-step-ahead desired path

Figure 3.8: Trace of covariance matrix $D$ in response to path changes in the prediction model with one-step-ahead desired path
Figure 3.9: Steering angle command from the controller that predicts path errors without one-step-ahead desired path.

Figure 3.10: Steering angle command from the controller that predicts path error with one-step-ahead desired path.
Figure 3.11: Position error of the center of front wheel in the model having normal random disturbance with zero mean and variance 0.02

Figure 3.12: Steering angle command from the controller of the system subject to normal random disturbance
ing command and reduced the position error because high gains amplified random measurement errors or noises. The value $E_0$ affected the magnitude of forgetting factor, but raised no remarkable problems when it was selected so that the forgetting factor could vary between one and zero. The $C_0$ value, minimum trace of matrix $D$, was not critical, but too high values bigger than $10^6$ or low values near zero should be avoided because they could make the system unstable.

**Effects of Tractor Velocity**

A variable speed test was designed to study the characteristics of estimated parameters and closed-loop system. The test speeds were 18.0, 14.4, 10.8, 7.2, and 3.6 km/h, and a circular path with a radius of 36 m was applied to the one-step-ahead prediction model with unknown future paths. During all tests, the sampling time was fixed at 0.1 second, and the driver model (39) was used. The initial data used were $\hat{b}_1 = 2, E_0 = 0.001, K_0 = 6, C_0 = 100, D_0 = 50I$, and $\dot{\theta}(0) = 0$.

At maximum design speed 18 km/h, model parameters were investigated at every sampled constant until they converged. If the parameters have six significant digits with respect to the previous ones, they were assumed to be stationary. Then, the tractor speed was set to the next test speed with the same initial data. The same procedure was used for all test speeds.

Table 3.2 shows estimated parameters, lateral position errors, and steering angles at steady-state condition when the prediction model with unknown future desired paths was used. These parameters are coefficients of open-loop pulse-transfer function of tractor model at each test speed. However, the system property is determined by the location of the roots of the characteristic equation of the closed-loop sys-
Table 3.2: Estimated parameters, position errors, and steering angles at steady-state circular path with a radius of 36 m when one-step-ahead prediction model with unknown future paths was used

<table>
<thead>
<tr>
<th>Speed km/h</th>
<th>( \hat{a}_1 )</th>
<th>( \hat{a}_2 )</th>
<th>( \hat{a}_3 )</th>
<th>( \hat{b}_2 )</th>
<th>( \hat{b}_3 )</th>
<th>( \Delta x_{ss}^a )</th>
<th>( \delta_{ss} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.0</td>
<td>-1.225</td>
<td>1.212</td>
<td>-0.483</td>
<td>0.897</td>
<td>-0.122</td>
<td>1.019</td>
<td>0.044</td>
</tr>
<tr>
<td>14.4</td>
<td>-1.157</td>
<td>1.286</td>
<td>-0.521</td>
<td>1.209</td>
<td>-0.083</td>
<td>-1.005</td>
<td>0.058</td>
</tr>
<tr>
<td>10.8</td>
<td>-1.092</td>
<td>1.229</td>
<td>-0.447</td>
<td>1.347</td>
<td>0.000</td>
<td>-2.581</td>
<td>0.068</td>
</tr>
<tr>
<td>7.2</td>
<td>-0.968</td>
<td>0.958</td>
<td>-0.326</td>
<td>1.461</td>
<td>-0.148</td>
<td>-3.708</td>
<td>0.076</td>
</tr>
<tr>
<td>3.6</td>
<td>-0.944</td>
<td>0.664</td>
<td>-0.044</td>
<td>1.162</td>
<td>0.187</td>
<td>-4.385</td>
<td>0.080</td>
</tr>
</tbody>
</table>

\( ^a \)Steady-state lateral position error.

Table 3.3: The closed-loop characteristics of the self-tuning steering controller at steady-state circular path with a radius of 36 m

<table>
<thead>
<tr>
<th>Speed km/h</th>
<th>Gain  ( K )</th>
<th>Poles</th>
<th>Zeros</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.0</td>
<td>-4.350</td>
<td>0.258, -0.303±j0.390</td>
<td>0.109, -0.558</td>
</tr>
<tr>
<td>14.4</td>
<td>5.762</td>
<td>-0.867, 0.109±j0.190</td>
<td>0.062, -0.667</td>
</tr>
<tr>
<td>10.8</td>
<td>2.652</td>
<td>-1.005, 0.133±j0.231</td>
<td>0.000, -0.674</td>
</tr>
<tr>
<td>7.2</td>
<td>2.049</td>
<td>-1.066, 0.145±j0.198</td>
<td>0.090, -0.821</td>
</tr>
<tr>
<td>3.6</td>
<td>1.836</td>
<td>-0.765, 0.053±j0.096</td>
<td>-0.291±j0.095</td>
</tr>
</tbody>
</table>

In self-tuning controller, it is very difficult to analyze the characteristics of the closed-loop system because of the complexity of the controller and time-variant parameters. Assuming that the self-tuner can be represented by a single proportional gain, the closed-loop system can be constructed and analyzed at steady state. As shown in Fig. 3.13, the estimated parameters were used in tractor model, and the proportional gain  \( K \) was computed from the steady-state lateral position error and steering command of Table 3.2.

Table 3.3 summarizes the proportional gain  \( K \), the closed-loop poles and zeros for test tractor speeds. The estimated parameter  \( \hat{b}_1 = 2 \) gave stable closed-loop zeros.
Figure 3.13: Block diagram for the closed-loop system of self-tuning steering controller at steady state. $y$ and $y_r$ are actual and reference paths, $y_e$ is path error, and $\delta$ is steering angle.
at all test speeds. As tractor speed decreased, the proportional gain tended to drop, which resulted in the increase of lateral position error. When the tractor speeds were 10.8 and 7.2 km/h, one closed-loop pole at each speed was slightly located outside stable boundary. Despite of unstable closed-loop poles, stable steering commands were obtained because the performance of minimum variance controller is affected by the location of zeros. At the speed 3.6 km/h, the pole was shifted to the right, within stable region. If the gain $K$ is increased or if the steering signal is persistently exciting, those unstable poles are moved to the right. The system performance can be also improved by changing the sampling time or the initial value $\hat{b}_1$.

**Effects of Sampling Time**

The sampling time is an important factor which affects the overall closed-loop characteristics of the system. Three sampling intervals, 0.1, 0.2, and 0.3 seconds, were chosen to study the effects of sampling interval on the proportional gain $K_0$, the initial value $\hat{b}_1$, and the position error of the system.

The steering input of Eq. (38) was used between 0 and 20 seconds, and the initial data were $E_0 = 0.001$, $C_0 = 100$, $D_0 = 100001$, and $\hat{\theta}(0) = 0$. At each test speed, proportional gain $K_0$ and initial value $\hat{b}_1$ were adjusted so that they could yield the minimum lateral position error and the smooth steering command.

Table 3.4 shows the characteristics of the gain $K_0$, the initial value $\hat{b}_1$, and the lateral position error at various test speeds. In general, the longer the sampling interval is, the bigger the position error. For sampling intervals 0.1 and 0.2 seconds, the controller could control the tractor path at all test speeds within ± 5 cm of the desired path. At the speeds below 10.8 km/h, position errors were within ± 5 cm
Table 3.4: Characteristics of the gain $K_0$, the initial value $\hat{b}_1$, and the position error at test speeds

<table>
<thead>
<tr>
<th>Speed km/h</th>
<th>Sampling time sec.</th>
<th>Gain $K_0$</th>
<th>$\hat{b}_1$</th>
<th>Max. cm</th>
<th>Mean cm</th>
<th>Std. cm</th>
<th>RMS cm</th>
<th>5 cm error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.0</td>
<td>0.1</td>
<td>6</td>
<td>2</td>
<td>1.46</td>
<td>0.26</td>
<td>0.21</td>
<td>0.34</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>5</td>
<td>5</td>
<td>3.59</td>
<td>1.28</td>
<td>1.09</td>
<td>1.68</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>1</td>
<td>2</td>
<td>8.41</td>
<td>3.15</td>
<td>2.73</td>
<td>4.16</td>
<td>33.33</td>
</tr>
<tr>
<td>14.4</td>
<td>0.1</td>
<td>6</td>
<td>2</td>
<td>1.31</td>
<td>0.27</td>
<td>0.26</td>
<td>0.38</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>5</td>
<td>5</td>
<td>4.70</td>
<td>1.59</td>
<td>1.45</td>
<td>2.15</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>3</td>
<td>5</td>
<td>7.71</td>
<td>2.82</td>
<td>2.54</td>
<td>3.79</td>
<td>27.71</td>
</tr>
<tr>
<td>10.8</td>
<td>0.1</td>
<td>6</td>
<td>2</td>
<td>1.37</td>
<td>0.32</td>
<td>0.31</td>
<td>0.45</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>7</td>
<td>5</td>
<td>2.85</td>
<td>0.93</td>
<td>0.89</td>
<td>1.29</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>5</td>
<td>5</td>
<td>4.64</td>
<td>1.55</td>
<td>1.41</td>
<td>2.10</td>
<td>–</td>
</tr>
<tr>
<td>7.2</td>
<td>0.1</td>
<td>14</td>
<td>2</td>
<td>1.42</td>
<td>0.40</td>
<td>0.36</td>
<td>0.51</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>13</td>
<td>5</td>
<td>1.27</td>
<td>0.36</td>
<td>0.33</td>
<td>0.48</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>7</td>
<td>5</td>
<td>3.06</td>
<td>0.99</td>
<td>0.97</td>
<td>1.39</td>
<td>–</td>
</tr>
<tr>
<td>3.6</td>
<td>0.1</td>
<td>14</td>
<td>2</td>
<td>1.27</td>
<td>0.31</td>
<td>0.30</td>
<td>0.43</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>17</td>
<td>5</td>
<td>1.27</td>
<td>0.32</td>
<td>0.28</td>
<td>0.43</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>12</td>
<td>5</td>
<td>1.57</td>
<td>0.51</td>
<td>0.36</td>
<td>0.63</td>
<td>–</td>
</tr>
</tbody>
</table>

$\hat{b}_1 = b_1^T b_1$. $b_1 = [1,1]^T$ or $[2,1]^T$. 
for all sampling intervals. As sampling interval increased at each test speed, the gain $K_0$ decreased and the initial guess $\hat{b}_1$ increased. For all sampling intervals, the gain $K_0$ was increased as the test speed was reduced. These explain that the gains for both position and yaw angle errors should be increased with the decrease of the test speed.

In a given test speed, however, Table 3.4 shows different trends, depending upon the choice of the initial guessed parameter $\hat{b}_1$ and the sampling interval. When the sampling interval was 0.1 second, the guess $\hat{b}_1 = 2$ gave the minimum position error, but the best value was $\hat{b}_1 = 5$ at 0.2 and 0.3 second sampling intervals except the speed 18 km/h. This indicates that the position error should have a bigger weighting than the yaw angle error if the sampling interval is increased over 0.2 second. Moreover, it supports that the characteristics of the controller change between the sampling interval 0.1 and 0.2 seconds.

Referenced on maximum gains at the sampling interval 0.2 second, gain drops at the sampling interval 0.3 second are related to the magnitude of errors. Because the increase of path errors resulting from the increase of the sampling interval provided rich input signals enough to estimate the model parameters, the magnitude of gain $K_0$ was reduced. If a higher gain is used, position errors tend to decrease, but the steering command from the controller starts fluctuating. At the speed 18 km/h and sampling interval 0.3 second, the selection of the initial guess $\hat{b}_1 = 5$ produced larger position errors than when $\hat{b}_1 = 2$. This indicates that amplification is not necessary at all because the high speed and long sampling interval results in large output errors.
SUMMARY AND CONCLUSIONS

A self-tuning steering controller was designed to control the tractor along the desired path. A 2-DOF linear dynamic model of a tractor, which had one-input (steering angle) and two outputs (lateral position and yaw angle), was developed in both continuous- and discrete-time spaces. This model was modified to single-input single-output system for recursive parameter estimation. The single output consisted of a linear combination of the lateral position and the yaw angle errors.

A recursive least-squares method was used to estimate parameters of the system model developed. To handle non-linear time-varying systems, a variable forgetting factor was implemented into the estimation scheme. The algorithm to set the variable forgetting factor was modified to avoid the instability of the system. A minimum variance controller was designed to minimize the deviations of actual from desired tractor position and yaw angle.

To analyze and evaluate the self-tuning steering controller, the tractor dynamic simulator developed in Part II was used. Two estimation models with no-step-ahead and one-step-ahead prediction errors were implemented to test performance of the controller. The no-step-ahead model estimated the parameters by using the measurements up to the current time. Two kinds of one-step-ahead prediction were considered: one-step-ahead prediction of path error with and without one-step-ahead
desired path, with respect to the one-step-ahead position and yaw angle predicted from actual path data up to the current time. To see the effects of measurement errors on parameter estimation and controller performance, a sequence of normal pseudo-random numbers with zero mean and variance 0.02 was generated and added to the current position data.

A composite test path including lane change and sinusoidal maneuvers was generated from the dynamic simulator at tractor speed 18 km/h. All estimation models were started up for 70 seconds by using this path. The speed and the sampling interval used in start-up were 18 km/h and 0.1 second. To study the steady-state closed-loop characteristics of the controller, a circular path with a radius of 36 m was used. In this study, test speeds were 3.6, 7.2, 10.8, 14.4, and 18.0 km/h, and the sampling interval was fixed at 0.1 second. To evaluate overall performance of the self-tuning controller, the composite path was used for initial 20 seconds. For five test speeds and three sampling intervals, 0.1, 0.2, and 0.3 seconds, the proportional gain $K_0$ and the initial value of the parameter $\hat{b}_1$ were adjusted to obtain the minimum position error. The values which yielded the minimum position error but unrealistic fluctuating signals were excluded from this evaluation.

On the basis of the results from computer simulation, the following conclusions and observations were made:

1. The execution time of the controller program was about 5 msec when 16-Mz 80386 microprocessor was used. Hence, a fast sampling rate can be achieved in practical applications.

2. The self-tuning controller can be used to guide a tractor with any types of positioning system, if they can measure the position or the position error with
respect to the desired path.

3. The initial value of residual error $E_0$ affected the magnitude of a forgetting factor. When it was selected so that the forgetting factor could vary between one and zero, the differences in position error were negligible.

4. The initial value $C_0$ for minimum trace of matrix $D$ was not critical in the performance of the controller, but too high values bigger than $10^6$ or low values near zero should not be used because they could make the system unstable.

5. In the existence of measurement errors up to $\pm$ 5 cm, the maximum position error at speed 18 km/h and sampling interval 0.1 second was 9.23 cm in the prediction model with known one-step-ahead desired path, but RMS error was 2.72 cm. If the path is moderately smooth, the controller can control the tractor within $\pm$ 5 cm position error.

6. Some poles of the closed-loop system under a steady-state circular path were located outside stable boundary, but it did not affect the performance of the controller because the minimum variance control technique was only affected by the location of zeros.

7. When sampling intervals were 0.1 and 0.2 seconds, the controller could control the tractor path at all test speeds within $\pm$ 5 cm of the desired path. At speeds below 10.8 km/h, the position error was less than $\pm$ 5 cm at all sampling intervals.

8. In a given speed, the maximum gain was obtained when the sampling interval was 0.2 second.
9. As the tractor speed decreased, the proportional gain $K_0$ was increased to compensate for the small prediction error resulting from the decreased speed.

The position error should have a bigger weighting than the yaw angle error if the sampling time is bigger than 0.1 second.

To control the tractor path within ± 5 cm of the desired path, an accurate position-sensing is the most important factor. Furthermore, if the navigational system is used, at least 0.3 second of the sampling interval is required to control the tractor within the desired path error. The measurement error of positioning systems should also be less than 5 cm.
REFERENCES


183:61–81.


APPENDIX SELF-TUNING CONTROLLER PROGRAM

/*****************************************************************************/
Procedure SELF_TUNER
******************************************************************************/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define n_parm 5
#define n_U n_parm*(n_parm-1)/2
#define sum_trace 100
#define min_ramda 0.2
#define E_0 0.001
#define out_gain 7.0
#define prediction_gain 7.0
/* Flags for various tests. 1=true, 0=false */
#define estimation 1
#define prediction 1
#define step_ahead_path 0
#define step_ahead_nopath
#define disturbance 0
#define velocity_effect 0
/* Estimation */
/* Prediction */
/* Step-ahead desired path */
/* No step-ahead desired path */
/* Disturbance generator */
/* Circular path */

void steering(double *delta)
{
extern double t, hmax, yaw, cqr[4][3], cqd[4][3], oma[4][3],
q[10], s[10][3];
float normal_random(float *variance);
int i, j, k=-1, k_f=-1, k_u=-1, tcount;
static double data[n_parm], parm[n_parm], U[n_U], D[n_parm],
U_p[n_U], D_p[n_parm], out_error, prev_steer,
ramda=0.995, cur_yaw, future_yaw, prev_path[2],
cur_path[2], future_path[2], pos[2], temp[2];
double dis_vec[2][2], outl[2], dummy, L[n_parm], f_j, v_j,
alpha_j, p_j, error, pos_error, ajlast, trace_D=0.0,
dPd[n_parm], vel[2], predicted_path[2], prediction_error,
temp_yaw;
float variance=0.02;
static int count=0, flag=1;

/* Get initial parameters and D matrix */
if (!count) {
    for (i=0;i<n_parm;i++) {
        dataCi] = 0.0;
        fscanf(Farm, "%.If", &parm[i]);
        fscanf(CP, "%.If", &D[i]);
        D_p[i] = D[i];
    }
    if (!velocity_effect)
        fscanf(Path_dat, "%.If %lf %lf %lf %lf\n",
            &dummy,&cur_path[0],&cur_path[1],&cur_yaw);
    prev_path[0]=prev_path[1]=0.0;
    out.error = 0.0;
    count++;
    if (!step_ahead_path) goto jump.once;
}
/* compute current path and yaw errors. */
if (flag && !(velocity_effect))
    if (!step.ahead.path)
        fscanf(Path_dat, "%.If %lf %lf %lf %lf\n",
            &dummy,&cur_path[0],&cur_path[1],&cur_yaw);
    else
        fscanf(Path_dat, "%.If %lf %lf %lf %lf\n",
            &dummy,&future_path[0],&future_path[1],&future_yaw);
jump.once:
/* Compute current position and yaw errors */
temp_yaw = yaw;
for (i=0;i<2;i++)
    pos[i] = cqr[0][i]+s[0][i];
if (!velocity_effect) {
    for (i=0;i<2;i++) /* Desired path vector a /
        dis_vec[0][i]=cur_path[i]-prev_path[i];
/* Pseudo random number generator - measurement noise */
for (i=0; i<2; i++) { /* Actual path vector b */
    if (disturbance)
        pos[i] += normal_random(&variance);
    dis_vec[1][i]=pos[i]-prev_path[i];
}

/* Error = b*\sin(\theta) = \text{vector}(b) \times \text{vector}(a) / a */
outl[0] = sqrt(dis_vec[0][0]*dis_vec[0][0]+dis_vec[0][1]*dis_vec[0][1]);
dummy = outl[0] * sqrt(dis_vec[1][0]*dis_vec[1][0]+dis_vec[1][1]*dis_vec[1][1]);
dummy = (dis_vec[0][0]*dis_vec[1][0]+dis_vec[0][1]*dis_vec[1][1])/dummy;
if (acos(dummy) >= 1.570796)
    flag = 0;
else
    flag = 1;
pos_error=outl[0]=(-dis_vec[0][0]*dis_vec[1][1]+dis_vec[0][1]*dis_vec[1][0])/outl[0];

/* Error calculation for the circular path */
if (velocity_effect) {
    dis_vec[0][0] = pos[0] + 36.0;
    dis_vec[0][1] = pos[1];
pos_error = outl[0] = sqrt(dis_vec[0][0]*dis_vec[0][0]+dis_vec[0][1]*dis_vec[0][1]) - 36.0;
    cur_yaw = atan2(dis_vec[0][1], dis_vec[0][0]);
cur_path[0] = 36.0 * (cos(cur_yaw) - 1.0);
cur_path[1] = 36.0 * sin(cur_yaw);
if (cur_yaw<0.0 || (cur_yaw>0 && yaw<0))
    cur_yaw += 2.0*3.141592654;
if (yaw<0.0 || (cur_yaw<0.0 && yaw>0.0))
    temp_yaw += 2.0*3.141592654;
}
out1[1]=cur_yaw-temp_yaw;
out_error = out_gain * (outl[0] + outl[1]);
/* Predict step-ahead position and yaw errors */
/* First compute velocity components of the center front wheel */
if (prediction) {
    vel[0] = cqd[0][0] - oma[0][2]*s[0][1] + oma[0][1]*s[0][2];
vel[1] = cqd[0][1] + oma[0][2]*s[0][0] - oma[0][0]*s[0][2];
predicted_path[0] = pos[0] + vel[0]*hmax;
if (step_ahead_path)
  for (i=0; i<2; i++) {
    dis_vec[0][i] = future_path[i] - cur_path[i];
    dis_vec[1][i] = predicted_path[i] - cur_path[i];
  }
else
  for (i=0; i<2; i++) {
    dis_vec[0][i] = cur_path[i] - prev_path[i];
    dis_vec[1][i] = predicted_path[i] - prev_path[i];
  }
out1[0] = sqrt(dis_vec[0][0]*dis_vec[0][0] +
               dis_vec[0][1]*dis_vec[0][1]);
out1[0] = (-dis_vec[0][0]*dis_vec[1][1] +
          dis_vec[0][1]*dis_vec[1][0])/out1[0];
if (step_ahead_path)
  out1[1] = future_yaw - temp_yaw;
else
  out1[1] = cur_yaw - temp_yaw;
prediction_error = prediction_gain*(out1[0] + out1[1]);
}
if (!estimation) goto no_estimate;
/* Parameter estimation routine with UD filter. */
/* Measurement vector update */
data[2] = data[1];
data[1] = data[0];
data[0] = -out.error;
data[4] = data[3];
if (!prediction) {
  error = out.error;
data[3] = prev.steer;
  prev.steer = q[0];
} else {
  error = prediction_error;
data[3] = q[0];
}
/* Prediction error */
for (i=0; i<n_parm; i++)
error = error - parm[i]*data[i];
if (!prediction)
    error = error - 2.0 * q[0];
/* First parameters of L and D. */
f_j = data[0];
v_j = D[0] * f_j;
L[0] = v_j;
alpha_j = 1.0 + v_j * f_j;
D[0] = D[0] / alpha_j;
/* From second to the last of L, U, D. */
for (j=1; j<n_parm; j++) {
    f_j = dataCj] ;
    for (i=0; i<j; i++) {
        ++k_f;
        f_j += data[i] * U[k_f];
    }
    v_j = f_j * D[j];
    L[j] = v_j;
    ajlast = alpha_j;
    alpha_j = ajlast + v_j * f_j;
    D[j] = D[j] * ajlast / (alpha_j * ramda);
    p_j = -f_j / ajlast;
    for (i=0; i<j; i++) {
        ++k_u;
        dummy = U[k_u] + L[i] * p_j;
        L[i] += U[k_u] * v_j;
        U[k_u] = dummy;
    }
}
/* Update parameters. */
for (i=0; i<n_parm; i++)
    parm[i] += error * L[i] / alpha_j;
/* Variable forgetting factor. */
for (i=0; i<n_parm; i++)
    dPd[i] = 0.0;
for (i=0; i<n_parm; i++) {
    dPd[i] += data[i];
    for (j=i+1; j<n_parm; j++)
        dPd[i] += data[i] * U[++k];
} /* transpose (data) * U */
ramda = 1.0;
for (i=0; i<n_parm; i++)
    ramda += D[i] * D[i] * D[i];
ramda = 1.0 - error * error / (ramda * E_0);
if (ramda < 0.2) ramda = min_ramda;

/* Prevent blow-up of P matrix. */
for (i=0; i<n_parm; i++)
    trace_D += D[i];
if ( (trace_D/ramda) <= sum_trace )
    for (i=0; i<n_parm; i++)
        D[i] = D[i]/ramda;
else {
    for (i=0; i<n_parm; i++)
        D[i] = D_p[i];
    for (i=0; i<n_U; i++)
        U[i] = U_p[i];
}

/* Save previous U and D values. */
for (i=0; i<n_parm; i++)
    D_p[i] = D[i];
for (i=0; i<n_U; i++)
    U_p[i] = U[i];

no_estimate:
if (flag)
    for (i=0; i<2; i++) {
        prev_path[i] = cur_path[i];
        if (step_ahead_path)
            cur_path[i] = future_path[i];
    }

/* Set the steering angle. */
*delta = 0.0;
for (i=0; i<n_parm; i++)
    *delta += -data[i]*parm[i];
*delta = -(*delta)/2.0;

/*
fprintf(Out1, "%5.3f %10.6f %10.6f %10.6f %10.6f %d\n",
    t, ramda, trace_D, pos_error, q[0], flag);
fprintf(Out2, "%5.3f %10.6f %10.6f %10.6f %10.6f %10.6f\n",
    t, ramda, trace_D, pos_error, q[0], flag);
*/
t, parm[0], parm[1], parm[2], parm[3], parm[4]);
fprintf(Out3, "%5.3f %10.6f %10.6f %10.6f %10.6f %10.6f\n",
    t, D[0], D[1], D[2], D[3], D[4]);
 */
}

/* Normal deviates by using Box-Muller method */
float normal_random(float *variance)
{
    static int iset = 0;
    static float gset;
    float fac, r, v1, v2;
    float uniform_random();

    if (iset==0) {
        do {
            v1 = 2.0*uniform_random()-1.0;
            v2 = 2.0*uniform_random()-1.0;
            r = v1*v1+v2*v2;
        } while (r >= 1.0);
        fac = sqrt(-2.0*log(r)/r);
        gset = v1*fac;
        iset = 1;
        return (*variance)*v2*fac;
    } else {
        iset = 0;
        return (*variance)*gset;
    }
}

/* Uniform random number generator based on the algorithm
    of Bays and Durhams */
float uniform_random()
{
    static float y, maxran, v[98];
    float dum;
    static int iff = 0, seed = 1;
    int j;
    unsigned i,k;
if (iff==0) {
    iff = 1;
    i = 2;
    do {
        k = i;
        i <<= 1;
    } while (i);
    maxran = k;
    seed = seed*12347;
    srand(seed);
    for (j=1; j<=97; j++) dum = rand();
    for (j=1; j<=97; j++) v[j] = rand();
    y = rand();
}
    j = 1+97.0*y/maxran;
    y = v[j];
    v[j] = rand();
    return y/maxran;
}
GENERAL SUMMARY

Automatic guidance of farm tractors would improve productivity of many field operations by reducing operator fatigue and increasing machine performance. A specific need is to control tractor position when planting and cultivating row crops, especially those grown with conservation tillage systems. Effective guidance requires control of position to within ± 5 cm of the desired path.

A global position-sensing system using navigational technology has been researched and applied to control a vehicle in field conditions. Besides guiding a tractor in conservation tillage systems, navigational positioning systems can be used to generate field maps which can help in the application of chemicals and in visualizing variation of soil and crop conditions.

Position-sensing systems used in both industrial and agricultural vehicles were reviewed. Control variables affecting the guidance system were investigated through the review of automobile and tractor controllers, and the control method was discussed to select a proper technique which can be easily implemented to guide a tractor-implement system.

A tractor dynamic simulator was developed by using a semi-recursive formulation which uses the variational vector approach and relative coordinates in Cartesian space. The emphasis was focused on the computational efficiency and automatic
code generation. Typical joints were formulated for automatic assembly of equations of motion, and cut-joint Jacobians were used to handle with a closed-loop mechanism. To verify the algorithms, a numerical example of a tractor system was given.

A self-tuning steering controller, which can be used for all non-contact types of the positioning systems, was designed for tractor guidance systems. A simple two degrees-of-freedom model of a tractor was chosen to develop a prediction model used in recursive least-squares parameter estimation. A variable forgetting factor was implemented, and its algorithm was modified to cope with time-varying nonlinear systems. The self-tuning steering controller based on minimum variance control was designed and verified by using the tractor dynamic simulator. Test paths used were a circular path with a radius of 36 m and a composite path which consisted of two lane-change and continuous sinusoidal maneuvers. The test speeds considered were in the range of 0–18 km/h.

Following conclusions were drawn from this study:

1. Navigational positioning is the most efficient system for a tractor steering controller, but much research and development are needed to reduce a measurement error of the system and to achieve a fast sampling rate.

2. Control parameters affecting tractor guidance system were lateral position and velocity, yaw angle and velocity with respect to the guide path.

3. A tractor dynamic simulator by a semi-recursive algorithm could be efficiently used to develop a self-tuning steering controller.

4. Because the execution time of the controller program was about 5 msec with a 16-Mz 80386 microcomputer, a fast sampling rate can be achieved in practical
5. The self-tuning controller that can be used to guide a tractor with any non-contact types of a positioning system can measure the position or the position error with respect to the desired path.

6. When sampling intervals were 0.1 and 0.2 seconds, the controller could control the tractor path within ± 5 cm of the desired path, up to the tractor speed 18 km/h.
GENERAL REFERENCES


ACKNOWLEDGEMENTS

The author expresses his appreciation to:

The United States Department of Agriculture and the Department of Agricultural Engineering of Iowa State University for providing financial support and facilities for this research.

Dr. E. C. Erbach for serving as major professor and for his guidance and support during the years spent at Iowa State University.

Dr. R. J. Smith for his constant assistance in the area of the controller design and for serving as a graduate committee member.

Drs. S. J. Marley and M. H. Mericle for their interest and suggestion about self-tuning controller for tractor guidance and for serving as graduate committee members.

Dr. M. J. Vanderploeg for his suggestion to improve the conversation skill and for serving as a graduate committee member.

S. Thangavadivelu for his continuous discussion and encouragement as a roommate.

G. Rajaram, K. M. Adam, and other graduate students for their help and friendship.
His parents, Jin-young and Jeong-ok Noh, for their continuous and patient support and encouragement.

His wife, Hee-seon, for her patience, support, and love and for taking care of his son, Dale.