A computer model for predicting individual forces in the lower extremity muscles during human movement

by

Michael Stuart Sellberg

A Thesis Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE

Interdepartment Program: Biomedical Engineering
Department: Aerospace Engineering and Engineering Mechanics
Co-majors: Biomedical Engineering
Engineering Mechanics

Iowa State University
Ames, Iowa

1992

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my friend and advisor, Dr. Jeffrey Huston, for his guidance and support throughout my M.S. program. I would also like to thank Professors Patterson, Flatau, and Tant for serving on my committee.

I am grateful to my parents, Bob and Helen, and the rest of my family (Susan, Kari, John, Joshua, and Justin) for their love and their support of my decision to attend graduate school.

Mostly, I would like to thank my girlfriend and best friend, Lorna, for her unselfish love, for her support of my work, and for her incredible patience. And finally, thanks to my dog, Cassidy, for showing me that though life is very complex, happiness is a rather simple thing.

1. INTRODUCTION

The biomechanics of human movement is very important in many fields. Before rehabilitation engineers can design devices to aid a motor function, they must have a keen understanding of the biomechanics related to that function. Physical therapists must know the biomechanics of the knee before prescribing exercises to rebuild strength after an injury. Sports biomechanists apply biomechanical analyses to a variety of athletic events to help train the most talented Olympic athletes.

While much of the biomechanics analyses are performed in terms of joint kinematics and joint torques, to help understand how the neuromuscular system functions to perform a motor task, it is of interest to know the individual muscle force and activation pattern. The problem is that the human muscular system is redundant (i.e., more than one muscle acts to create a torque at a specific joint) and individual muscle forces cannot be determined utilizing standard dynamical methods. To overcome this problem, a lower extremity muscle mechanics model (LEMMM) has been developed which uses non-linear static optimization to predict individual muscle activation patterns and force magnitudes for 43 lower extremity muscles. In addition, the LEMMM provides many other musculoskeletal parameters such as muscle length and velocity, tendon force, fully activated static muscle force and maximum dynamic muscle force which are all useful to researchers in their own right.

With a powerful biomechanics tool such as the LEMMM in their hands, sports biomechanists would be able to predict which muscles are exerting the most force during certain athletic events. Training regimens could then focus on these muscles. Physical therapists could also monitor a patient's "muscular recovery" from a knee surgery or tendon transfer surgery. Rehabilitation engineers could use information from lower extremity muscle force predictions in their designs for prosthetics or orthotics. The LEMMM can be used in any one of the above scenerios provided a kinematic record of the activity (usually obtained

from film analysis) and joint torques (from inverse dynamic analysis or from a machine such as the Cybex® isokinetic testing machine) are supplied.

The LEMMM is written in FORTRAN 77 and requires angular orientation, angular velocity, and joint torques as inputs. Joint torques are not necessary if a researcher does not require individual muscle force predictions for certain analyses. Muscle lengths and muscle shortening/lengthening velocities for each of the 43 muscles are computed, as well as the isometric and dynamic muscle force that could be exerted if fully activated by the neuromuscular system. The last function of the model is to predict the actual individual muscle forces being exerted through the use of non-linear static optimization which allows a solution to the redundant (indeterminate) problem to be found. For human gait analysis, the use of static optimization in this model gives excellent results in terms of correlation between predicted muscle activity and experimental electromyographic (EMG) data. It is important to realize that other features of the LEMMM, mainly its calculation of muscle lengths, muscle velocities, static muscle forces, and dynamic muscle forces are all useful parameters to the rehabilitation engineer, physical therapist, surgeon, or sports biomechanicist.

This thesis begins in Chapter 2 with a review of previous work in the area of human body modeling, musculoskeletal modeling, and individual muscle force prediction. Chapter 3 reviews properties of muscle and tendon, the modeling of these components, and the 43 lower extremity muscles used in this model. The main components of the LEMMM including the musculoskeletal lower-extremity model, the musculotendon actuator model, and the non-linear optimization for individual force prediction; and the computer implementation of the model are discussed in Chapter 4. In Chapter 5, the model is used to predict individual muscle forces during human gait utilizing published kinematic and joint torque data. Chapter 6 offers conclusions and recommendations for future work.

2. LITERATURE REVIEW

While muscle force has been directly measured in the tendons of some animals [1], direct experimental muscle force measurement in humans based on existing technology is currently impossible for ethical reasons. Researchers have instead turned to indirect means, specifically modeling of the musculoskeletal system, in an attempt to predict individual muscle forces. These models include a dynamical model of the human body in conjunction with a model of the muscular system.

The natural chain of kinetic and kinematic events (system dynamics) [2] in the body is as follows: 1) the nervous system excites the muscular system and this electrical excitation is recorded as an electromyogram (EMG); 2) the excitation causes muscular contractions which generate muscle forces that are transferred to the skeleton by tendon creating torques at the joints they cross; and 3) these net muscle torques (joint torques) cause angular accelerations of the joints which, after time, cause the joint angular velocities and joint displacements to change which is manifested as human movement. This change in kinematics is given by the direct multi-joint dynamics of the system which is produced by the net muscle torques. The system dynamics is refered to as *direct* because it refers to the natural sequence in which motion occurs.

Since kinematic information is more easily attained by biomechanics researchers than experimental joint torque data, *inverse dynamics* is often utilized to study human motion and muscular function. In this method, the angular orientations are recorded, usually by some sort of film analysis, and the angular velocities and angular accelerations are found by differentiating the angular orientation versus time data. Once the kinematic inputs are known, the joint torques that developed the observed motion are solved for by inverting the direct system dynamics [3, 4]. Reaction force data is often combined with limited kinematic data to calculate net joint torques. External reaction forces are measured by a force plate and

represent the sum of the inertia forces on the body segments which eliminates having to model the entire body and increases accuracy [5].

Once the joint torques are calculated, it is not easy to calculate the individual muscle forces because more than one muscle contributes to the singular joint torque which constitutes an indeterminate problem. Thus researchers are forced to employ different strategies to determine the individual muscle forces when the number of active muscles is greater than the number of joint torques.

2.1. Individual Muscle Force Prediction Models

Many approaches have been tried by researchers to overcome the indeterminate muscle force distribution problem. In general, these strategies can be grouped in three different categories: reduction-elimination method, static optimization, and dynamic programming (optimal control theory).

Since one goal of this thesis is to provide a general muscle force prediction model that can be used in a variety of different applications, only force prediction methods associated with the inverse dynamics method are reviewed here. While dynamic programming may be the most powerful method for predicting muscle forces in human movement [2], solution techniques are complex and it is not general enough to facilitate modeling of a variety of activities due to the task-specific performance criterion. For example, maximizing the vertical displacement of the body's center of gravity for vertical jumping would not be a very feasible performance criteria for kicking a soccer ball. For a general review of dynamic programming for muscle force modeling, see Zajac and Gordon [2].

2.1.1. Reduction-Elimination Method

Early attempts to find individual muscle forces involved "reducing" the number of active muscles acting at a joint or by "eliminating" a degree of freedom at a specific joint.

Typically, reduction and elimination were combined in what was called the reduction-elimination method. In early work concerning hip muscles during locomotion [6], several simplifying assumptions were made to change the problem from indeterminate to determinate: 1) hip muscles were grouped into six muscle groups; 2) no bi-articular muscles (muscles that cross two joints) were allowed; 3) internal and external motion was eliminated (constrained); and 4) EMG data was used to eliminate antagonistic muscles showing little activity.

Morrison [7] used a similiar method to determine muscle and articular contact forces at the knee joint. Cruciate ligaments constrain the knee eliminating anterior-posterior tibial shear and collateral ligaments constrain varus-valgus displacement thus confining the muscle activity to flexion/extension. Bi-articular muscles were again eliminated and previously measured EMG activity was used to eliminate certain muscles during specific portions of the gait cycle. This method was also applied to predict individual muscle forces in the hand during static conditions [8].

Since the reduction-elimination method's assumptions greatly simplify complex anatomy and function, it can only be considered feasible for actual joint anatomy that is known to be considerably simple. Furthermore, failure to include bi-articular muscles and synergist-antagonist muscle pairs (synergists exert force to produce the same motion as a companion muscle while antagonists exert force to cause the opposite motion than a companion muscle) renders this method useless from a practical standpoint because the interplay between the synergistic and antagonistic muscles aids in motor coordination.

2.1.2. Static Optimization

Static optimization is a method to resolve the indeterminancy problem without requiring simplifications of the functional anatomy or elimination of specific muscles. In this approach, a cost function is defined at every time point of the data and represents what the body is trying to optimize in mathematical terms. Well established algorithms are then used to

minimize this cost function, subject to constraints, yielding a solution for the individual muscle forces. This minimization of the cost function is repeated at each time step of the movement.

Static optimization has been used for a variety of athletic, clinical, and every day motor tasks which have been reviewed elsewhere [2]. The selection of a cost function, both on a physiological basis and a mathematical basis (linear or non-linear), determines what type of the activation patterns and force magnitudes are possible in the muscles.

The availiability of linear programming algorithms and subroutine libraries promoted early use of linear objective functions. One of the first objective functions was derived from the principle of minimal total muscular force [9] which postulates that only the total muscular force that is necessary to perform a motion will be used. The cost function selected to be minimized was the sum of the muscle forces. Seirig and Arvikar [10] used this criterion in analyzing muscle forces of the lower extremity while Penrod *et al.* [11] utilized it in studying the wrist. The minimal total muscular force was also utilized by many researchers studying gait [12, 13, 14].

Minimization of muscle stresses was used by Crowinshield [15]. He defined muscle stress as muscle force divided by physiological cross-sectional area (PCSA) and formulated the linear objective function. Hardt [12] developed a minimum energy consumption objective function that was also linear. Linear optimization had several drawbacks including that the number of non-zero muscle forces could be no greater than the number of equations for resolution [12] and that EMG experiments [16] showed that many more muscles were active than predicted by linear programming.

To overcome the limitations of linear optimization, researchers turned to non-linear objective functions. Pedotti *et al.* [13] and Pederson *et al.* [17] utilized several linear and non-linear objective functions to predict muscle forces during human locomotion and compared the results. Non-linear techniques predicted much greater synergistic and

antagonistic muscle activity than linear techniques [17]. Non-linear techiniques also produced more temporally correct force predictions (when compared to EMG data) than the linear techniques.

Pedotti *et al.* [13] found the non-linear objective function $\Phi = \sum_{i=1}^{n}\left(\frac{F_i}{F_i^{\max}}\right)^2$ to be the most feasible based on temporal comparisons with EMG data. It takes into account the instantaneous state of each muscle because $F_i^{\max}$ depends on the instantaneous length and velocity of the muscle. Pedotti *et al.* [13] claim that by squaring the function, the activation for each muscle is kept as low as possible which allows for a more efficient use (as compared to linear objective functions) of the muscles by the objective function.

While many objective functions may have some physiological basis, none have direct physiological meaning. Currently, there is no rationale for a "correct" objective function and temporal validation by comparison to EMG patterns is the only means of analyzing the feasibility of the selected objective function. There is no method for experimentally validating muscle force magnitudes. However, models that utilize maximum muscle force objective functions require the calculation of parameters such as muscle lengths, muscle velocities, isometric muscle force (fully activated), and dynamic muscle forces, each of which is extremely useful in its own right.

2.2. Musculoskeletal Models

Important to any model of the musculoskeletal system, whether it be an indirect dynamics model or a direct dynamics model, is accurate definition of musculoskeletal geometry. The term "musculoskeletal geometry" is used in this thesis to describe the lengths of the musculotendon actuators (the muscle-tendon complex) and the muscle moment arms.

Straight line approaches, i.e., the muscle is modeled as a straight line extending from muscle origin to muscle insertion, have been used by researchers to calculate the lengths of each muscle-tendon complex [18, 19, 20]. Jensen and Davy [21] tested the straight-line approach by modeling the muscle as a curve passing through transverse cross-sectional centroids of the muscles. The force transmitted by the muscle at any point along the muscle is tangent to the centroid line at that point. This method was found to be cumbersome and did not signficantly affect force prediction values [19].

More recently, Hoy *et al.* [22] modified the Brand *et al.* [20] origin and insertion coordinates for their musculoskeletal model. An accurate model of musculoskeletal geometry may be found in a recent computer model for surgical simulation by Delp [23]. He modeled the muscles as a series of straight line segments and utilized sophisticated computer graphics to define "wrapping points" which constrain the muscle paths to "wrap" around anatomical structures instead of unrealistically passing through them. This multi-segment geometry more accurately predicts musculotendon actuator length and also muscle moment arms.

3. MUSCLE AND TENDON

A brief overview of skeletal muscle anatomy and physiology is presented here. More detailed anatomy discussions can be found in Martini [24]. Detailed discussions concerning muscle contraction, muscle excitation, and related physiology can be found in Ganong [25]. A reader who is familiar with basic skeletal muscle anatomy and physiology may skip to Section 3.3 for a discussion of specific muscle and tendon properties without a loss of continuity. Since this discussion is specific to skeletal muscle and does not include smooth or cardiac muscle, henceforth the term "muscle" refers exclusively to "skeletal muscle."

3.1. Skeletal Muscle Morphology

The functional unit of force generation in muscle is the sarcomere. The sarcomere averages 2.6 μm in length and about 1 μm in diameter. Each sarcomere consists of two distinct types of myofilaments--thin and thick. Thin filaments are composed of the protein actin and are found at either end of the sarcomere. Thick filaments are composed of the protein myosin and are found in the central portion of the sarcomere. The thin filaments surround the thick filaments in the zone of overlap where subsequent interactions cause sarcomere contraction.

The sarcomeres are connected in series in long units called myofibrils. There may be as many as 10,000 sarcomeres in series in the myofibrils and all the sarcomeres shorten at the same time when the muscle cell contracts.

Parallel bundles of myofibrils are arranged in the sarcoplasm (cytoplasm of the muscle cell) which span the length of the cell. Thus, there are many (hundreds to thousands) myofibrils contained in a typical muscle cell commonly referred to as a muscle fiber. The muscle cell is multi-nucleated and contains typical cell structures. Cell structures in muscle cells have different names than the typical structures. For instance, the cytoplasm is called the

sarcoplasm; the plasmalemma is called the sarcolemma; the endoplasmic reticulum is the sarcoplasmic reticulum, and microfilaments are known as myofilaments, as mentioned above.

Muscle fibers form bundles known as muscle fasciculi. Within the fasciculus, muscle fibers are surrounded by connective tissue called endomysium. Fine branches of the nerves and blood vessels permeate the endomysium to reach the muscle cells. Typically, all the muscle fibers in a fasciculus are the same length.

Bundles of fasciculi in parallel, separated by connective tissue called perimysium, form a whole muscle. The muscle itself is surrounded by a dense layer of collagen tissue known as epimysium. The epimysium is continuous with the connective tissue of the tendon. The physiological cross-sectional area (PCSA) is a measure of the number of muscle fibers in parallel in a specific muscle and is proportional to the maximum isometric force that the muscle can produce.

3.2. Skeletal Muscle Physiology

The mechanics of sarcomere contraction are based on the sliding filament theory of contraction. In a resting sarcomere, there is no interaction between the double helix of actin molecules in the thin filament and the helical arrangement of myosin in the thick filaments. This is because the actin active sites on the thin filament are covered by two proteins, tropomyosin and troponin. The thick filaments are golf club-shaped in appearance with the heads facing away from the center of the sarcomere. These heads are called cross-bridges because they bind to active sites on the thin filaments to cause contraction.

When the cross-bridges bind to active sites they pivot and pull the thin filament toward the center of the sarcomere. The cross-bridges do not pull on the same active site constantly; the cross-bridges form, then pivot and pull, and then release. This formation/pivot and pull/release process repeats moving the thin filament along causing the sarcomere to get shorter and shorter. This is known as the sliding filament theory and it explains how

sarcomeres can get shorter without changing any filament lengths. While this theory explains the mechanical aspects of muscular contraction, it does not address how contraction is controlled, or how the energy necessary to pivot cross-bridges is obtained.

The chain of events leading up to muscle contraction is started with an action potential in the muscle. This muscle action potential is created by a motor neuron through the neuromuscular junction. Details of action potentials (both neuronal and muscular) and the neuromuscular junction are described in Ganong [25]. A single action potential is created on the sarcolemma from a single action potential arriving at the synaptic knob, this one-on-one correspondence allows for very fine neuromuscular control.

Once an action potential forms on the sarcolemma, it sweeps along the T tubules which are inward extensions of the sarcolemma that reach every sarcomere in the cell. The action potential changes the membrane permeability of sarcoplasmic reticulum making it extremely permeable to calcium. Since almost all of the calcium ions in the resting muscle cell are stored in the cisternae (reservoirs) of the sarcoplasmic reticulum, calcium ions diffuse out of the sarcoplasmic reticulum and into the sarcoplasma within a millisecond. This 100 fold increase in sarcoplasm calcium concentration activates the contraction mechanism in the sarcomeres.

The calcium in the sarcoplasm interacts with troponin causing a structural change in the troponin-tropomyosin complex which leaves an active actin site exposed. With the active site exposed, the myosin cross-bridge attaches and also gains the ability to convert ATP to ADP which releases energy. This energy is utilized by the myosin heads as they pivot and pull the thin filaments causing the sarcomere to contract. When sarcoplasm calcium levels are reduced to resting levels, the actin active sites are covered, ATP energy is lost, and the muscle contraction ends.

3.3. Muscle and Tendon Properties

3.3.1. Muscle Force-Length (fl) Properties

It is well known that the isometric force produced by a sarcomere changes with sarcomere length. The assumption is made that this sarcomere relationship can be scaled up to muscle fibers and muscle fasciculi and finally to the whole muscle [26]. Thus, the sliding filament theory of contraction for sarcomeres and the accompanying active force-length relationship are valid for whole muscles.

The most thorough work in constructing force-length relationships of active muscle was conducted by Gordon, Huxley, and Julian in 1966 [27]. The representative curve for the relationship was determined by maximally stimulating the muscle at different lengths and measuring the force. The maximum isometric force occurs at the resting fiber length [28] which will be refered to as *optimal fiber length* in this thesis and is denoted as L_O^M. Active muscle force is the force created by the contraction of the sarcomeres, but the force-length relationship also depends on passive muscle force.

Passive muscle force is due to stretching and does not occur as a result of neural stimulation (activation) as do active contractions. The muscle does not develop passive force until the muscle is stretched past the optimal fiber length [28]. While passive tensile force in muscle has long been thought to be a result of interfiber connective tissue such as the epimysium, perimysium, and endomysium, Zajac [26] suggests that this conclusion was incorrectly reached due to the failure of researchers to isolate muscle from tendon effectively. Magid and Law [29] state that intrafiber elasticity may cause the passive tension.

The dimensionless force-length curve of Figure 3.1 is used in the muscle model to represent the static properties of all muscles. The generic curve is scaled for each muscle by that muscle's optimal fiber length (L_O^M) and maximum isometric force (F_O^M). Normalized

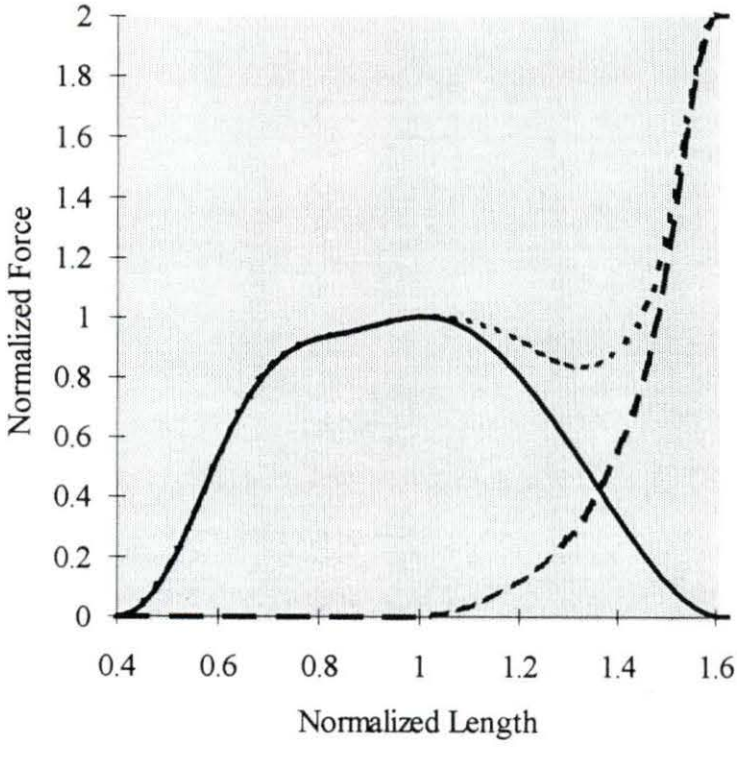
muscle force is the static muscle force divided by the maximum isometric muscle force and the normalized length is the muscle length divided by the optimal fiber length.

$$\tilde{L}^M = \frac{L^M}{L_O^M} \tag{3.1}$$

$$\tilde{F}^M = \frac{F^M}{F_O^M} \tag{3.2}$$

The active and passive muscle curves in Figure 3.1 are obtained by fitting a cubic spline to published control points [23].

The active muscle force is generated in the region $0.4L_O^M \le L^M \le 1.6L_O^M$. When the muscle is shorter than $0.4L_O^M$, the sarcomeres cannot produce force because they cannot contract any



This relationship features the active muscle curve (solid) , the passive muscle curve (long dashes), and the active + passive curve (short dashes). Cubic spline control points for these dimensionless curves are from Delp [23]. Normalized Length is defined as muscle length divided by optimal fiber length ($\tilde{L}^M = L^M / L_O^M$) *and Normalized force is defined as static muscle force divided by maximum isometric muscle force* ($\tilde{F}^M = F^M / F_O^M$).

Figure 3.1 - Muscle force-length (fl) relationship

further, i.e., the thin filaments are "butted" together at the center of the thick filament and cannot be pulled closer together. Muscle cannot produce force at lengths greater than $1.6L_O^M$ because the sarcomere is stretched so much that the thick and thin filaments no longer overlap. Thus, the filaments cannot interact to produce contraction.

3.3.2. Muscle Force-Velocity (fv) Properties

The above description of the force-length relationship of muscle determines the force a muscle can exert isometrically, i.e., it specifies the force when a muscle is at a constant length. When muscle is subjected to a constant pull (tension), it shortens and then stops which is called an isotonic contraction. Isotonic contractions and the relationship between force and velocity were first studied in the 1930's [30, 31]. The length at which muscle stops shortening during isotonic contraction corresponds to the length at which such a force can be sustained statically, i.e., the length value for such a force from the relationship in Figure 3.1 [32].

So during dynamic activity, the relationship depicted in Figure 3.2 determines the muscle force that can be exerted by a fully-activated muscle at optimal fiber length. Shortening velocity (concentric contraction) is defined as positive and lengthening velocity (eccentric contraction) is defined as negative. In Figure 3.2, normalized muscle force is the dynamic muscle force divided by the maximum isometric muscle force and normalized velocity is defined as the muscle velocity divided by the maximum muscle velocity:

$$\tilde{F}^{dyn} = \frac{F^{dyn}}{F_O^M} \qquad (3.3)$$

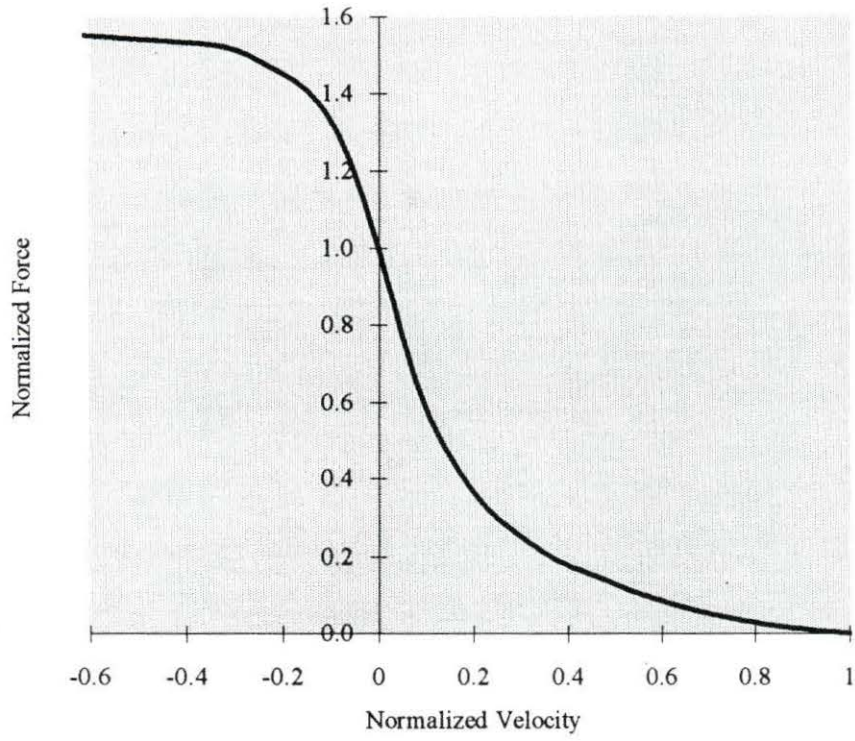
$$\tilde{V}^M = \frac{V^M}{V^{\max}} . \qquad (3.4)$$

Each muscle has a maximum shortening velocity ($V^{\max}$) above which there is negligible force produced. This maximum velocity is approximated by the following empirical relationship [26]:

$$V^{\max} \approx 10.0\left(L_O^M\right)(\sec^{-1}) \qquad (3.5)$$

where L_O^M is in meters and $V^{\max}$ is in meters/sec.

The assumption used in this thesis is that this force-velocity relationship is scaled by the force-length relationship, i.e., the y intercept of Figure 3.2 is less than or greater than 1.0 if the muscle length is not equal to the optimal fiber length. [33, 34]. While many investigators



This relationship is based on a curve found in Zajac [26]. The normalized muscle force is defined as: $\tilde{F}^{dyn} = F^{dyn} / F_O^M$. *The normalized velocity is defined as* $\tilde{V}^M = V^M / V^{\max}$. *The maximum velocity is nominally defined as* $V^{\max} \approx 10 \cdot L_O^M \cdot (\sec^{-1})$. *The muscle can exert no force at maximum velocity (x-intercept) and the y-intercept (isometric force) is scaled by the muscle force-length relationship.*

Figure 3.2 - Muscle force-velocity relationship

have chosen to assume that the muscle length does not affect the force-velocity curve, Zajac [26] points out that to date, the choice of assumptions does not make any difference in muscle coordination studies.

3.3.3. Tendon Properties

The tendon transfers the forces created by the muscle to the skeletal system. Since tendon is viscoelastic [35] and tendon and muscle function together as a "musculotendon

actuator," it is crucial that the properties of tendon not be ignored. Further evidence of the importance of modeling tendon when studying neuromuscular coordination is that tendon and muscle work together with the dynamics of the body segments forming a coupled multiple-input, multiple-output feedback system [26].

Tendon consists of two portions--the internal or *aponeurosis* and the external portion (Figure 3.3). The properties of internal and external tendon have been shown to be similiar [36]. Bundles of the protein collagen form the structure of tendon. Fibrils are the basic load bearing units for the tendon and consist of microfibrils packed into a tetragonal lattice. Microfibrils are bundles of four or five basic molecules which are intertwining molecules of tropocollagen suspended in polysaccharide gel. For a review of tendon and collagenous structures see Elliot [37] and Viidik [38], respectively.

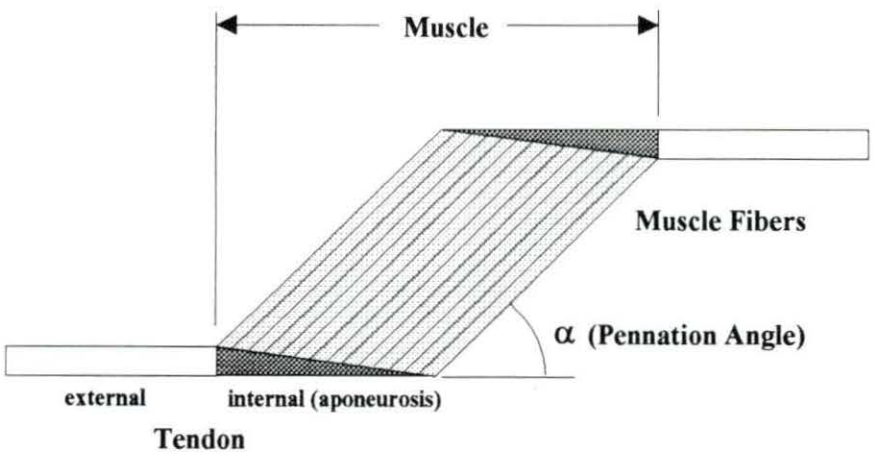


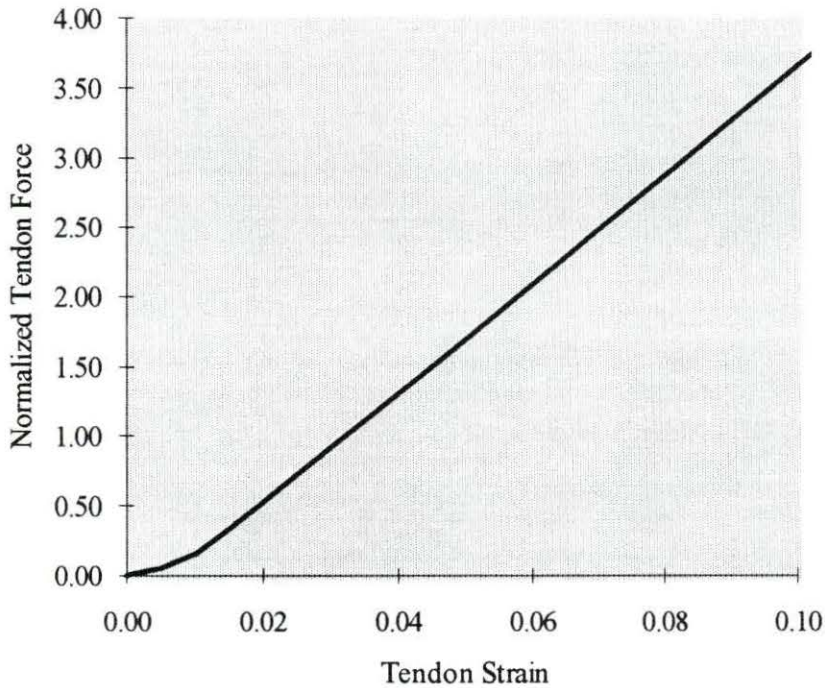
Figure 3.3 - Pennated muscle showing internal and external tendon

The properties of tendon can be summarized in a stress-strain diagram. The tendon tangent modulus of elasticity increases with strain at low strains (toe region) and then is constant (linear region) at higher strains until failure at approximately 10% strain and 100

MPa [26]. The tendon modulus of elasticity (linear region) is taken to be nominally 1.2 GPa and the strain at which the linear region begins is 2 % [26].

Tendon can have a considerable amount of "physiological slack" at resting length. The *tendon slack length* (L_S^T) is defined as the length on elongation at which tendon is capable of developing force. Tendon strain will be used to describe the condition of tendon in this thesis and is defined as:

$$\varepsilon^T = \frac{\Delta L^T}{L_S^T} = \frac{L^T - L_S^T}{L_S^T} . \qquad (3.6)$$



This curve was created by fitting a cubic spline to control points given by Delp [23]. Tendon strain is defined in equation 3.6 and normalized tendon force is defined in Eq. 3.7. This diagram is equivalent to the dimensionless tendon stress-strain diagram.

Figure 3.4 - Tendon force-strain diagram

In this thesis, tendon is assumed to be linearly elastic and is modeled using a generic force-strain curve from Zajac [26]. The actual dimensionless curve shown in Figure 3.4 is a cubic spline fit to control points from Delp [23]. The dimensionless tendon force-strain curve

is scaled by two parameters specific to each muscle--the maximum isometric force normalizes the tendon force and tendon slack length scales the elongation (Eq. 3.6):

$$\tilde{F}^T = \frac{F^T}{F_O^M} . \qquad (3.7)$$

The reader should note that the force-strain diagram shown in Figure 3.4 is equivalent to a dimensionless tendon stress-strain diagram because normalized stress is equal to normalized force:

$$\tilde{\sigma}^T = \frac{\sigma^T}{\sigma_O^T} = \frac{\sigma^T}{\left(F_O^M / A^T\right)} = \frac{F^T}{F_O^M} = \tilde{F}^T . \qquad (3.8)$$

3.4. Lower-Extremity Muscle Descriptions

The 43 lower-extremity muscles featured in this model are taken from Delp [23]. Since each muscle is represented by a straight line or a series of straight lines, it is sometimes necessary to "break apart" muscles with large origin or insertion areas into separate muscle entities. An example of this is the gluteus maximus which is dividing into superior, middle, and inferior components. The muscles featured in this model are illustrated and discussed below. The muscle actions, origins, and insertions are taken from Martini [24]. The numbering system used in this thesis to indentify these muscles is shown in Table 4.4.

Hip Muscles There are twelve muscles in this model acting exclusively about the hip joint.

Gluteus medius — abducts and medially rotates the thigh. It originates on the lateral surface of the anterior iliac crest of ilium and inserts on the greater trochanter of the femur. Represented by three components in the model--the anterior, middle, and posterior compartments.

Gluteus minimus — abducts and medially rotates the thigh. It originates on the lateral surface of the ilium between the inferior and anterior gluteal lines and inserts on the greater trochanter of the femur. Represented by

three components in the model--anterior, middle, and posterior compartments.

Gluteus maximus — extends and laterally rotates the thigh. The largest and most posterior of the gluteal muscles, it originates on the iliac crest of iluim, sacrum, coccyx, and lumbodorsal fascia and inserts on iliotibial tract and gluteal tuberosity of the femur. The model splits the muscle into superior, middle, and inferior compartments.

Adductor magnus — adducts the thigh. The anterior portion flexes the thigh and posterior portion extends the thigh. It originates on the inferior ramus of pubis and ischial tuberosity posterior to the adductor brevis and inserts on the linea aspera of the femur. The model represents this muscle as superior, middle, and inferior components.

Adductor longus — adducts, flexes, and medially rotates the thigh. The muscle originates on the inferior ramus of pubis anterior to the adductor brevis and inserts on the linea aspera of the femur.

Adductor brevis — adducts the thigh. It originates on the inferior ramus of the pubis and inserts on the linea aspera of the femur.

Pectineus — flexes, adducts, and medially rotates the thigh. This muscle originates on the superior surface of the pubis and inserts on the pectineal line inferior to the lesser trochanter of the femur.

Iliacus — flexes the thigh and may also act to flex the lumbar spine. It originates on the iliac fossa of the ilium and inserts on the femur distal to the lesser trochanter. (The tendons of the iliacus and the psoas are fused)

Psoas — flexes the thigh and may also act to flex lumbar spine. It originates on the anterior surfaces of vertebrae T_{12} - L_5 and inserts (in conjunction with the iliacus) on the femur distal to the lesser trochanter.

Quadratus femoris — adducts and rotates the thigh laterally. The muscle originates on the tuberosity of ischium and inserts on the quadrate tubercle of the intertrochanteric crest.

Gemelli — rotates the thigh laterally. Consists of the gemellus inferior and gemellus superior. The gemellus inferior originates on the tuberosity of ischium and the gemellus superior on the spine of ischium. Both insert on the greater trochanter of the femur.

Piriformis laterally rotates and adducts the thigh. It originates on anterolateral surface of the sacrum and inserts on the greater trochanter of the femur.

Hip and Knee Muscles This model contains seven muscles which act about both the hip and the knee joints.

Rectus femoris extends the lower leg and flexes the thigh (one of the four muscles known collectively as the quadriceps femoris). This muscle originates on the anterior inferior iliac spine and inserts on the tibia tuberosity via the patellar ligament.

Semimembranosus flexes and medially rotates the lower leg; extends, adducts, and medially rotates the thigh (one of three muscles known collectively as the hamstrings). It originates on the tuberosity of the ischium and inserts on the posterior surface of the medial condyle of the tibia.

Semitendinosus flexes and medially rotates the lower leg; extends, adducts, and medially rotates the thigh (one of three muscles known collectively as the hamstrings). It originates on the tuberosity of the ischium and inserts on the proximal, posteromedial surface of tibia near insertion of the gracilis.

Biceps femoris (l. head) flexes and laterally rotates the lower leg; extends and adducts the thigh (one of three muscles known collectively as the hamstrings). It originates on the tuberosity of the ischium and inserts on the head of the fibula and lateral condyle of the tibia.

Gracilis flexes and medially rotates the lower leg; adducts the thigh. It originates on the inferior rami of pubis and ischium and inserts on the anterior surface of tibia inferior to medial condyle.

Sartorius flexes and medially rotates the lower leg; flexes and laterally rotates the thigh. It originates on the anterior superior spine of ilium and inserts on the medial surface of tibia near tibial tuberosity.

Tensor fasciae latae flexes, abducts and medially rotates the thigh; tenses the fascia latae which laterally supports the knee. It originates on the iliac crest and surface of ilium between anterior iliac spines and inserts on the iliotibial tract.

Knee Muscles There are four muscles in this model that act exclusively about the knee joint.

Vastus medialis extends the lower leg (one of the four muscles known collectively as the quadriceps femoris). It originates along the entire length of the linea aspera of the femur and inserts on the tibial tuberosity via the patellar ligament.

Vastus intermedius extends the lower leg (one of the four muscles known collectively as the quadriceps femoris). It originates along the anterolateral surface of the femur along the distal half of the linea aspera and inserts on the tibial tuberosity via the patellar ligament.

Vastus lateralis extends the lower leg (one of the four muscles known collectively as the quadriceps femoris). It originates anterior and inferior to the greater trochanter of the femur and along the proximal half of the linea aspera and inserts on the tibial tuberosity via the patellar ligament.

Biceps femoris (s. head) flexes the lower leg. It originates on the linea aspera of the femur and inserts on the head of fibula and lateral condyle of the tibia.

Knee and Ankle Muscles There is only one muscle in this model that acts about the knee and ankle joints.

Gastrocnemius flexes leg; plantar flexes the foot. It originates above the femoral condyles and inserts on the calcaneus via the Achilles tendon. The gastrocnemius is represented by a medial and lateral component in the model.

Ankle Muscles The model includes ten muscles that act exclusively about the ankle joint.

Soleus plantar flexes the foot (primary action at the ankle). It originates on the head and proximal shaft of the fibula and the adjacent posteromedial shaft of the tibia and inserts on the calcaneus via the Achilles tendon.

Tibialis posterior adducts and inverts foot (primary action at the ankle). It originates on the interosseous membrane and adjacent shafts of the tibia and fibula and inserts on the tarsals and metatarsals.

Flexor digitorum longus flexes toes two through five (primary action at the toes). It originates on the posteromedial surface of the tibia and inserts on the inferior surfaces of phalanges two through five.

Flexor hallucis longus flexes the big toe (primary action at the toes). It originates on the posterior surface of the fibula and inserts on the inferior surface of the terminal phalanx of the big toe.

Peroneus brevis everts the foot (primary action at the ankle). It originates at the midlateral margin of the fibula and inserts at the base of the fifth metatarsal.

Peroneus longus everts and plantar flexes the foot; supports longitudinal arch (primary action at the ankle). It originates on the lateral condyle of tibia and head of fibula and inserts at the base of the first metatarsal.

Tibialis anterior dorsiflexes the foot (primary action at the ankle). It originates on the lateral condyle and proximal shaft of tibia and inserts on the base of the first metatarsal.

Peroneus tertius everts and dorsiflexes the foot (primary action at the ankle). It originates on the anterior surface of the fibula and the interosseous membrane and inserts at the base of the fifth metatarsal.

Extensor digitorum longus extends toes two through five (primary action at the toes). It originates on the lateral condyle of the tibia and anterior surface of the fibula and inserts on the superior surfaces of phalanges two through five.

Extensor hallucis longus extends big toe (primary action at the toes). It originates on the anterior surface of the fibula and inserts on the superior surface of the terminal phalanx of the big toe.

4. LOWER-EXTREMITY MUSCLE MECHANICS MODEL

The lower-extremity muscle mechanics model (LEMMM) presented in this thesis consists of three main parts. The first part, the musculoskeletal lower-extremity model, defines the body segments and their associated coordinate systems, joints, generalized coordinates, muscle origins and insertions, and determines the musculoskeletal geometry. This includes the musculotendon actuator length (L^{MT}), musculotendon velocity (V^{MT}), and muscle moment arms. The second part, the musculotendon actuator model, calculates the muscle length (L^{M}), the muscle velocity (V^{M}), the static muscle force (F^{M}), and the dynamic muscle force (F^{dyn}) given the musculotendon actuator length. The last portion of LEMMM is the non-linear optimization procedure to predict the activation patterns and the individual muscle force magnitudes. The last section in this chapter describes the computer implementation of LEMMM.

4.1. The Musculoskeletal Lower-Extremity Model

The lower-extremity model is based on Delp's [23] seven segment model used for surgery simulation and static muscle force prediction. Delp defines coordinate systems, joint rotations and translations, and origins and insertions for 43 muscles. Modifications have been made to the model allowing it to be used in conjunction with film analysis and allowing for dynamic muscle parameters, i.e., musculotendon actuator velocity, muscle velocity, and dynamic muscle forces, to be calculated.

The model of the lower extremity is a nominal (i.e., not scalable with anthropometry) one. It represents a young adult male subject, about 1.8 meters (5 feet, 11 inches) tall. While bones and muscles certainly vary among individuals, much insight into musculoskeletal performance can be gained from a nominal model. A logical next step is to determine how musculoskeletal geometry differences among individuals affect muscular performance. More

specific assumptions concerned with Delp's [23] model and LEMMM will be discussed in later sections.

4.1.1. Coordinate Systems and Joint Kinematics

Delp's lower-extremity model consists of seven rigid-body segments: the pelvis, femur, patella, tibia/fibula, talus, foot (comprising the calcaneus, navicular, cuboid, cuneiforms, and metatarsals), and the phalanges (toes). The body segment reference frames are described in Table 4.1 and shown in Figure 4.1.

Table 4.1 - Body segment reference frames and their locations

Name	Location
Pelvis	Fixed at the midpoint of line connecting the two anterior superior iliac spines
Femur	Fixed at the center of the femoral head
Tibia	Located at the midpoint of the line between the medial and lateral epicondyles
Patella	Located at the most distal point of the patella
Talus	Located at the midpoint of the line between the apices of the medial and lateral maleoli
Calcaneus	Located at the most inferior, lateral point on the posterior surface of the calcaneus
Toe	Located at the base of the second metatarsal

The relative motion (translation and rotation) between the body segments is defined by joint models of the hip, knee, ankle, subtalar, and metatarsophalangeal (MTP) joints. In the LEMMM, the joint translation between the segmental reference frames is defined as a vector between the segmental coordinate system origins whose components are supplied by Delp

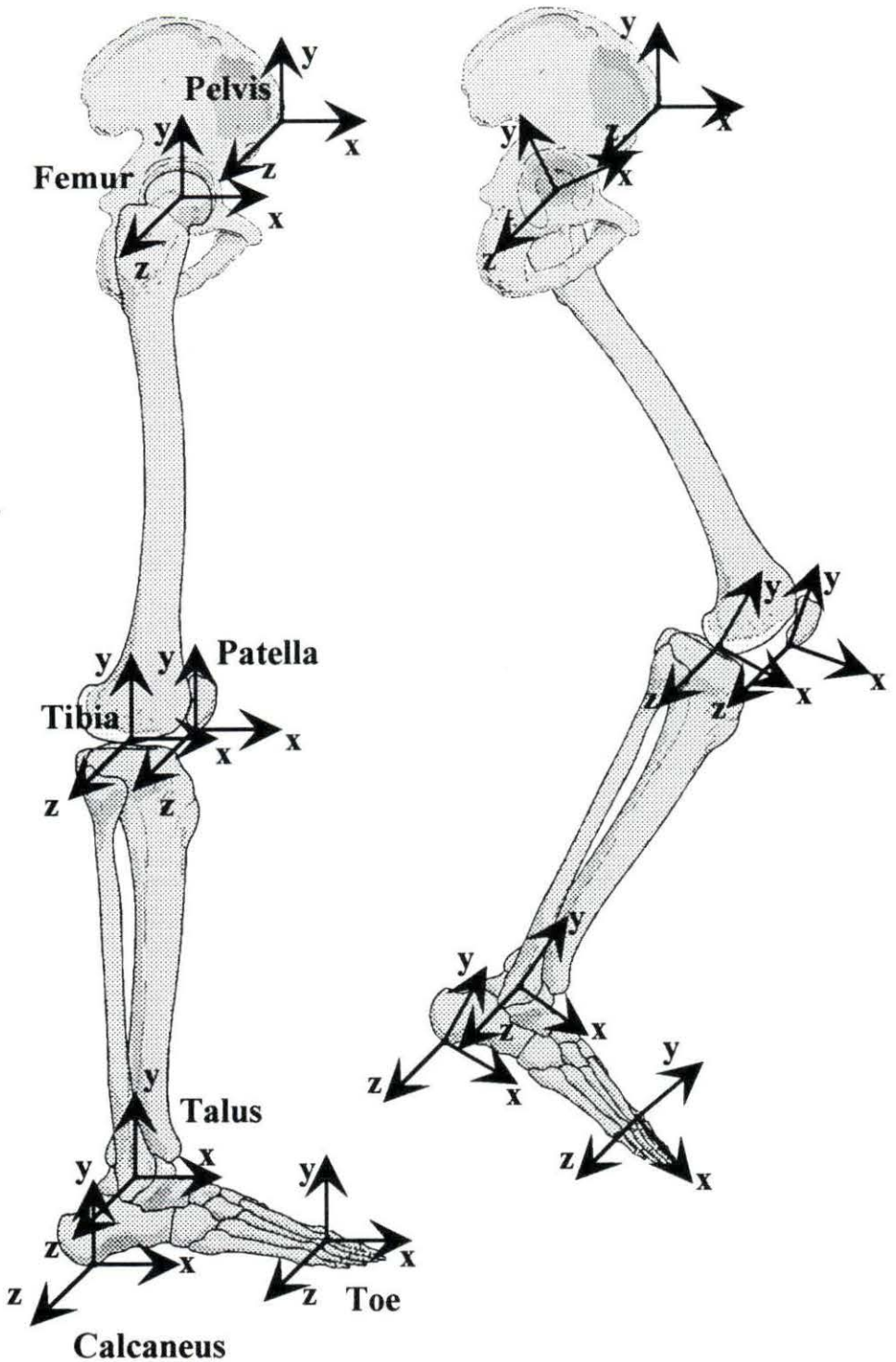


Figure 4.1 - Locations and orientation of the segmental reference frames

[23]. These vectors may be constant (pelvis-femur, tibia-talus, talus-calcaneus, calcaneus-toe) or may vary with a generalized coordinate (femur-tibia, tibia-patella). Table 4.2 shows the translation vectors between the seven body segments. A word on notation--the vector ${}^{A}\vec{r}^{B}$ refers to the vector pointing from the origin of *frame A* to the origin of *frame B* and is expressed in the coordinates of *frame A*.

Table 4.2 - Translation vectors between reference frames

Vector	Coordinates in Frame	X Component (meters)	Y Component (meters)	Z Component (meters)
${}^{p}\vec{r}^{f}$	Pelvis	-0.07070	-0.06610	0.08350
${}^{f}\vec{r}^{t}$	Femur	f(knee angle)	f(knee angle)	0.00000
${}^{t}\vec{r}^{pa}$	Tibia	f(knee angle)	f(knee angle)	0.00240
${}^{t}\vec{r}^{ta}$	Tibia	0.00000	-0.43000	0.00000
${}^{ta}\vec{r}^{c}$	Talus	-0.04877	-0.04195	0.00792
${}^{c}\vec{r}^{to}$	Calcaneus	0.17880	-0.00200	0.00108

The functions shown in Table 4.2 are kinematic functions (translations expressed as a function of the knee angular displacement) for the knee joints--femur/tibia and tibia/patella. The knee model used by Delp is discussed below and graphs of the translation kinematic functions are shown in Appendix A.

The rotational degrees of freedom of the joints form the basis for the seven generalized coordinates of the LEMMM which are shown in Table 4.3. These generalized coordinates and their derivatives must be specified by the user as inputs (see Section 4.1.2.). The rotation axes ($\boldsymbol{n}$) shown in this table are unit vectors aligned with the orthogonal axes of the body segment reference frames. The hat ($\wedge$) denotes a unit vector, the superscript letter represents the reference frame (f for femur, etc.), and the subscript number represents the axis the unit vector is aligned with (1=X, 2=Y, 3=Z), and the primes in the superscript represent

"intermediate" rotation axes described below and in Appendix B. The ankle joints are defined as single revolute joints with a simple rotation about a unit vector (denoted by λ, *the subscript "e" denotes an "Euler" axis, the superscripts still represent the reference frames*). Euler parameters are used to describe the kinematics of these joints. For a description of the λ axes and an overview of Euler parameters, see Appendix C.

Table 4.3 - Generalized coordinates of the LEMMM

Generalized Coordinate	Rotation Axis	Anatomic Description of Motion
θ_1	$\hat{n}_3^p$	hip joint flexion/extension flexion is positive, extension is negative
θ_2	$\hat{n}_1^{f'}$	hip joint adduction/abduction adduction is positive, abduction is negative
θ_3	$\hat{n}_2^{f''}$	hip joint internal/external rotation internal rot. is positive, external rot.is negative
θ_4	$\hat{n}_3^t$	knee joint flexion/extension extension is positive, flexion is negative
θ_5	$\hat{n}_3^{pa}$	patella rotation--known as "patellar levering" kinematic function of the knee angle
θ_6	$\hat{\lambda}_e^{ta}$	ankle joint dorsiflexion/plantarflexion dorsiflexion is positive, plantarflexion is negative
θ_7	$\hat{\lambda}_e^{c}$	subtalar joint inversion/eversion inversion is positive, eversion is negative
θ_8	$\hat{\lambda}_e^{to}$	metatarsophalangeal joint extension/flexion extension is positive, flexion is negative

The hip joint is modeled as a ball and socket joint with three rotational degrees of freedom. The three generalized coordinates are related to flexion/extension, abduction/adduction, and internal/external rotation. The transformation between pelvis and femur reference frames is determined by three successive rotations of the femoral frame about three orthogonal axes fixed in the femoral head. Specifically, the LEMMM uses a 3-1-2 series of dextral rotations to describe the orientation of the femur with respect to the pelvis. That is,

with the two frames originally aligned, the first rotation is θ_1 counterclockwise (ccw) about the pelvis Z axis, the second rotation is θ_2 ccw about the X' femur axis (the original femur X axis rotated by θ_1), and the final rotation is θ_3 ccw about the Y" femur axis (the original femur Y axis rotated by both θ_1 and θ_2). For a comprehensive review of arbitrary orientations and dextral rotations, see Amirouche [39].

LEMMM uses *shifter matrices* [40] to transform vectors in one frame to coordinates in another frame. For calculation of musculoskeletal geometry, vectors are transformed (shifted) into the frame for which the muscle origin is located. For a description of transformation matrices and for the shifter matrices used in the LEMMM, see Appendix B.

The model of the knee joint was constrained to sagittal plane motion and was a modified version of an earlier model used to characterize the knee extensor mechanism [41]. This one degree of freedom (DOF) model is a simplification of the complex motion of the knee which includes varus/valgus rotation, but non-planar motion has been shown to be small compared to sagittal plane motion [41]. The single DOF model includes the flexion/extension of the knee (primary motion), but also determines the kinematics of the tibiofemoral joint (sliding contact point) and the patellofemoral joint (patellar levering mechanism) by using the kinematic functions shown in Appendix A. These kinematic functions allow for accurate prediction of the instantaneous center of rotation between the femur and the tibia and thus accurate determination of moment arms of muscles that insert on the patella.

The ankle, subtalar, and MTP joints were modeled by Delp [23] as frictionless revolutes. The location and orientation of these axes for each of these joints were taken from a previous work [42] and the MTP axis was modified by rotating it 8 degrees clockwise about the vertical axis which prevented dislocation of the toes [23]. While the single revolute axis functions well for the ankle, the subtalar joint is significantly more complex [43] and therefore the single revolute comprises a significant simplification. Since these three axes are expressed

as unit vectors [42, 23] with components in body segment coordinate systems, the LEMMM utilizes Euler parameters to describe the kinematics of these joints and the transformations between the segments of the foot. Euler parameters and the transformations for the foot are described in Appendix C.

4.1.2. User inputs and conventions

The user is required to input the generalized coordinates (θ_1- θ_4) and the derivatives of these generalized coordinates ($\dot{\theta}_1$- $\dot{\theta}_4$) in the format specified in Appendix H. The generalized coordinate (θ_5) and derivative of the generalized coordinate ($\dot{\theta}_5$) for the patellofemmoral joint are calculated by the LEMMM. Generalized coordinates and derivatives are not entered for the ankle, subtalar, and MTP joints, instead these quantities are calculated from standard film plane orientation angles which are inputted by the user. Obtaining time records of these generalized coordinates is straightforward for two-dimensional (2-D) analysis, but three-dimensional (3-D) analysis requires a more sophisticated motion measurement software package that allows the calculation of dextral orientation angles about intermediate axes. Conventions for the generalized coordinate inputs are given in Table 4.3.

Since this model will be used in conjunction with a film analysis and the joints of the foot (ankle, subtalar, and metatarsophalangeal) have complex axes of revolution that are not perpendicular to standard filming planes, standard film plane angular orientations for these three joints can be substituted as inputs. The three generalized coordinates of the foot are then calculated by projecting the film angular orientation on to the plane of rotation define by the single revolute axes. For a 3-D analysis, the user inputs the dorsiflexion/plantarflexion angular orientation as seen by the camera in the sagittal plane, the inversion/eversion angular orientation from the frontal plane, and the toe angular orientation from the sagittal plane.

Given the standard plane film angular orientations, the LEMMM performs the following calculations (see Appendix D):

$$\theta_6 = \left\{\tan^{-1}\left[0.9904 \cdot \tan(\alpha_{ank})\right]\right\} \quad (4.1)$$

$$\theta_7 = \left\{\tan^{-1}\left[1.2517 \cdot \tan(\alpha_{subt})\right]\right\} \quad (4.2)$$

$$\theta_8 = \left\{\tan^{-1}\left[1.2280 \cdot \tan(\alpha_{MTP})\right]\right\} \quad (4.3)$$

where α_{ank} is the ankle dorsiflexion/plantarflexion film orientation angle, α_{subt} is the subtalar inversion/eversion film orientation angle, and α_{MTP} is the metatarsophalageal extension/flexion film orientation angle. The conventions for the standard film plane orientation angles are consistant with the generalized coordinates listed in Table 4.3.

If the user is interested in individual muscle force predictions, joint torques must also be inputted. For 2-D analysis, only the sagittal plane torques are required. The convention for the sagittal plane torques is that a counterclockwise (ccw) torque acting on the segment distal to the joint is positive. Thus, the Z component (femur frame) of the hip torque (T_z^{hip}) is positive for a flexion torque acting on the femur segment, the Z component (tibia frame) is the only component of the knee torque (T_z^{knee}) and it is positive for an extension torque acting on the tibia segment, and the Z component (talus frame) of the ankle torque (T_z^{ank}) is positive for a dorsiflexor moment acting on the talus.

For 3-D analysis, the hip torque components about the X and Y femur axes and the subtalar torque about the X talus axis must be inputted. The convention for these torques is again positive for counterclockwise torques on the distal segments. An adduction torque exerted on the femur segment would thus be a positive X component of the hip torque (T_x^{hip}). The Y component of hip torque (T_y^{hip}) would be positive for an internal rotation torque

exerted on the femur. The subtalar torque about the X talus axis (T_x^{subt}) would be positive for an inversion torque exerted on the calcaneus segment.

4.1.3. Muscle Origins and Insertions

To determine musculoskeletal geometry from relative joint motion (supplied as inputs by user), coordinates for the muscle origins and insertions must be known. If the muscle is represented by a series of segments, the coordinates of the effective insertions, effective origins, and wrapping points must also be known. Coordinates for muscle origins and insertions are expressed in the body segment reference frames.

The coordinates used in the LEMMM are from Delp [23]. He used a 3-D digitizer to digitize the coordinates of muscle origins, insertions, effective origins, and effective insertions from a skeleton. Muscles are represented as a straight line or series of straight lines. He then used a computer graphics visualization system to define "wrapping points" that constrain the muscle to wrap around the bone or anatomical landmark instead of passing through it. This allows for more accurate determination of musculotendon actuator lengths and muscle moment arms. Given the coordinates of the origins and insertions, the musculoskeletal geometry can be determined at any body configuration by using shifter matrices and vector operations.

4.1.4. Musculotendon Actuator Length/Velocity Determination

The musculotendon actuator length (L^{MT}) can be calculated if the coordinates of muscle origins and insertions, the shifter matrices, and the translation vectors between reference frames are known. The musculotendon length must be calculated in order to determine muscle lengths and velocities and muscle forces (static and dynamic).

Consider the generic bi-articular muscle in Figure 4.2 that crosses the knee and ankle joints. The muscle includes an effective origin and effective insertion and is therefore

represented by three segments. The origin and effective origin are expressed in the femur reference frame and the insertion and effective insertion are expressed in the calcaneus frame.

The musculotendon length is the sum of the magnitudes of the individual segment vectors

$$L^{MT} = \left\| \overrightarrow{OO_e} \right\| + \left\| \overrightarrow{O_e I_e} \right\| + \left\| \overrightarrow{I_e I} \right\| . \qquad (4.4)$$

These vectors can be calculated using vector addition and subtraction after all the vectors have been shifted into the frame of reference of the muscle's origin. For the generic muscle in Figure 4.2, this is the femur reference frame. Utilizing the translation vectors and the shifter transformation matrices, the segment vectors are calculated:

$$\overrightarrow{OO_e} = \bar{O}_e - \bar{O} \qquad (4.5)$$

$$\overrightarrow{O_e I_e} = {}^f\bar{r}^t + {}^t\bar{r}^{ta} + {}^{ta}\bar{r}^c + \bar{I}_e - \bar{O}_e \qquad (4.6)$$

$$\overrightarrow{I_e I} = \bar{I} - \bar{I}_e \quad . \qquad (4.7)$$

For calculation of musculotendon actuator velocity (V^{MT}), it is only necessary to find the velocity of one of the muscle segment vectors because the whole muscle must have the same velocity. For the muscle in Figure 4.2, differentiation of Eq. 4.6 is appropriate because this muscle segment crosses two joints. Differentiating Eq. 4.6 with respect to time in the femur reference frame before multiplying by the shifter transformation matrices:

$${}^f\frac{d}{dt}\left(\overrightarrow{O_e I_e}\right) = {}^f\frac{d}{dt}\left({}^f\bar{r}^t\right) + {}^f\frac{d}{dt}\left({}^t\bar{r}^{ta}\right) + {}^f\frac{d}{dt}\left({}^{ta}\bar{r}^c\right) + {}^f\frac{d}{dt}\left(\bar{I}_e\right) - {}^f\frac{d}{dt}\left(\bar{O}_e\right) . \qquad (4.8)$$

Since vector $\bar{O}_e$ does not change with time in the femur reference frame, Eq. 4.8 reduces to:

$$ {}^{f}\frac{d}{dt}\left(\overrightarrow{O_e I_e}\right) = {}^{f}\frac{d}{dt}\left({}^{f}\bar{r}^{t}\right) + {}^{f}\frac{d}{dt}\left({}^{t}\bar{r}^{ta}\right) + {}^{f}\frac{d}{dt}\left({}^{ta}\bar{r}^{c}\right) + {}^{f}\frac{d}{dt}\left(\bar{I}_e\right). \qquad (4.9) $$

To calculate the quantities in Eq. 4.9, the principle of differentiation in two reference frames [44] is utilized. If A and B are two reference frames, the first derivative of any vector $\bar{s}$ in A and B is determined as follows:

$$ \frac{{}^{A}d\bar{s}}{dt} = \frac{{}^{B}d\bar{s}}{dt} + {}^{A}\bar{\omega}^{B} \times \bar{s}\,. \qquad (4.10) $$

where ${}^{A}\bar{\omega}^{B}$ is the angular velocity of B with respect to A. Utilizing this principle and shifter matrices to transform all vectors into the femur frame, Eq. 4.9 becomes:

$$ {}^{f}\frac{d}{dt}\left(\overrightarrow{O_e I_e}\right) = {}^{f}\frac{d}{dt}\left({}^{f}\bar{r}^{t}\right) + \left[{}^{t}\frac{d}{dt}\left({}^{t}\bar{r}^{ta}\right) + {}^{f}\bar{\omega}^{t} \times [SFT]{}^{t}\bar{r}^{ta}\right] $$

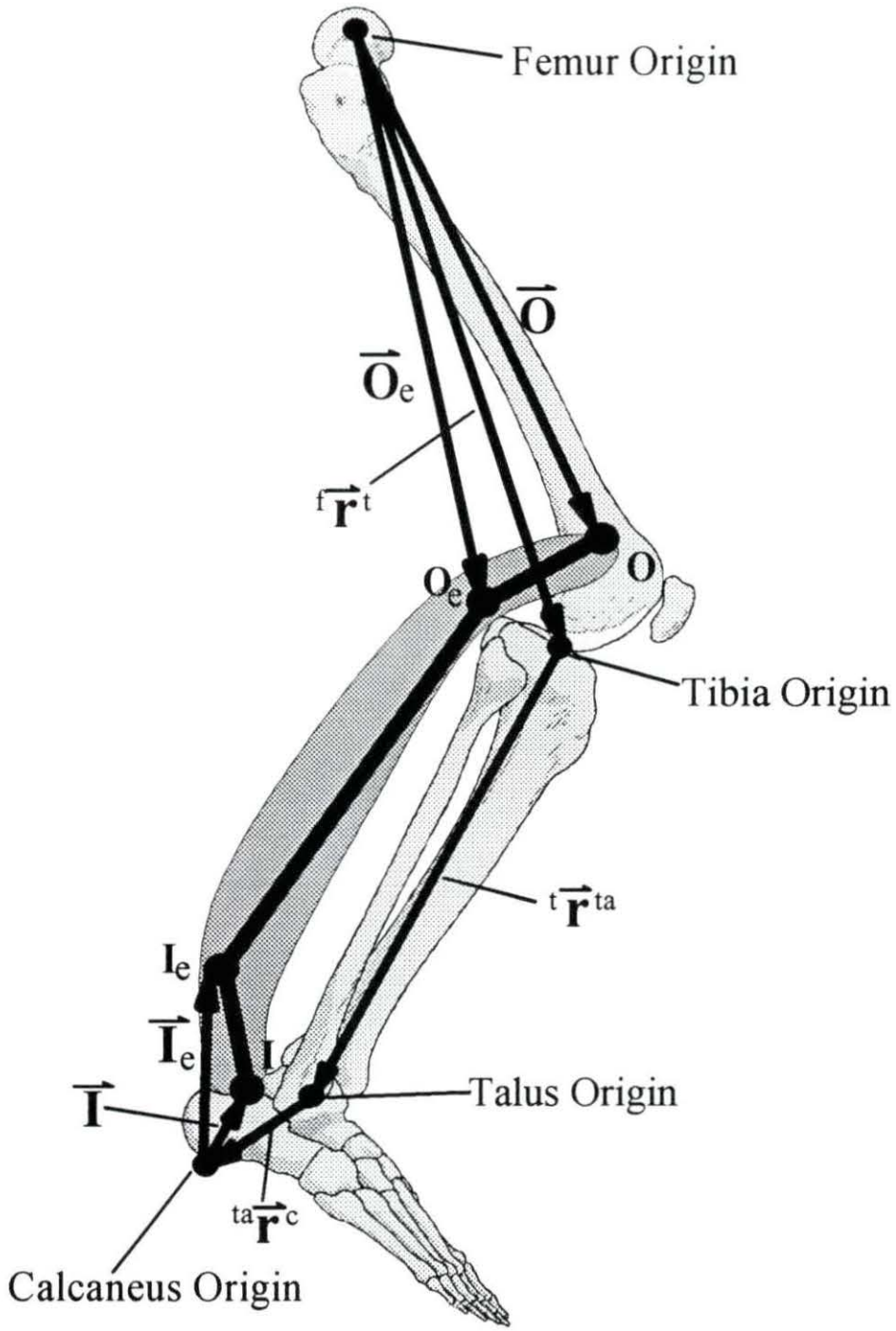
$$ + \left[{}^{ta}\frac{d}{dt}\left({}^{ta}\bar{r}^{c}\right) + {}^{f}\bar{\omega}^{ta} \times [SFTa]{}^{ta}\bar{r}^{c}\right] + \left[{}^{c}\frac{d}{dt}\left(\bar{I}_e\right) + {}^{f}\bar{\omega}^{c} \times [SFC]\bar{I}_e\right]. \qquad (4.11) $$

Noting that vectors ${}^{t}\bar{r}^{ta}$, ${}^{ta}\bar{r}^{c}$, and $\bar{I}_e$ do not change in their respective reference frames, Eq. 4.11 can be futher simplified to

$$ {}^{f}\frac{d}{dt}\left(\overrightarrow{O_e I_e}\right) = {}^{f}\frac{d}{dt}\left({}^{f}\bar{r}^{t}\right) + {}^{f}\bar{\omega}^{t} \times [SFT]{}^{t}\bar{r}^{ta} + {}^{f}\bar{\omega}^{ta} \times [SFTa]{}^{ta}\bar{r}^{c} + {}^{f}\bar{\omega}^{c} \times [SFC]\bar{I}_e\,. \qquad (4.12) $$

Since the translation vector between the femur and tibia reference frames changes with respect to the femur frame, the first term on the right hand side of Eq. 4.12 does not go to zero. Expressing the vector ${}^{f}\bar{r}^{t}$ in terms of its orthogonal components and utilizing the total derivative

$$ {}^{f}\frac{d}{dt}\left({}^{f}r_x^{t}\right) = \overbrace{\frac{\partial {}^{f}r_x^{t}}{\partial x}\frac{\partial x}{\partial \theta_4}}^{\text{slope of Figure A.1}}\frac{d\theta_4}{dt} + \frac{\partial {}^{f}r_x^{t}}{\partial t} \qquad (4.13) $$



This generic multi-segment bi-articular muscle crosses the knee and ankle joints. Origin and effective origin vectors are expressed in femur frame coordinates and the insertion and effective insertion vectors are expressed in calcaneus frame coordinates.

Figure 4.2 - Calculation of musculotendon actuator length

$$ {}^{f}\frac{d}{dt}\left({}^{f}\boldsymbol{r}_{y}^{t}\right)=\overbrace{\frac{\partial {}^{f}\boldsymbol{r}_{y}^{t}}{\partial y}\frac{\partial y}{\partial \theta_4}}^{\text{slope of Figure A.2}}\frac{d\theta_4}{dt}+\frac{\partial {}^{f}\boldsymbol{r}_{y}^{t}}{\partial t} \qquad (4.14) $$

$$ {}^{f}\frac{d}{dt}\left({}^{f}\boldsymbol{r}_{z}^{t}\right)=\frac{\partial {}^{f}\boldsymbol{r}_{z}^{t}}{\partial z}\frac{\partial z}{\partial \theta_4}\frac{d\theta_4}{dt}+\frac{\partial {}^{f}\boldsymbol{r}_{z}^{t}}{\partial t}=0\,. \qquad (4.15) $$

Noting that the z component does not change in the femur reference frame and that the transient terms in Eqs. 4.13, 4.14, and 4.15 are all zero (i.e., the rigid body assumption where there is no time dependent deformation), the first term of Eq. 4.12 can be expressed as:

$$ {}^{f}\frac{d}{dt}\left({}^{f}\bar{\boldsymbol{r}}^{t}\right)=\left(\overbrace{\frac{\partial {}^{f}\boldsymbol{r}_{x}^{t}}{\partial x}\frac{\partial x}{\partial \theta_4}}^{\text{I}}\hat{\boldsymbol{n}}_1^{f}+\overbrace{\frac{\partial {}^{f}\boldsymbol{r}_{y}^{t}}{\partial y}\frac{\partial y}{\partial \theta_4}}^{\text{II}}\hat{\boldsymbol{n}}_2^{f}\right)\frac{d\theta_4}{dt} \qquad (4.16) $$

where I and II are the slopes of the tangent lines at the inputted value of the knee angle (θ_4) of the curves in Figures A.1 and A.2, respectively.

To calculate the remaining three terms on the right hand side of Eq. 4.12, the angular velocities with respect to the femur frame must be computed. This is done through the use of *the addition theorem of angular velocities* [44]. Knowing the relative angular velocities between the tibia and the femur (${}^{f}\bar{\omega}^{t}$), the tibia and the talus (${}^{t}\bar{\omega}^{ta}$), and the talus and calcaneus (${}^{ta}\bar{\omega}^{c}$), the angular velocities of the talus (${}^{f}\bar{\omega}^{ta}$) and calcaneus (${}^{f}\bar{\omega}^{c}$) can be found with respect to the femur reference frame:

$$ {}^{f}\bar{\omega}^{ta}={}^{f}\bar{\omega}^{t}+[\boldsymbol{SFT}]^{t}\bar{\omega}^{ta} \qquad (4.17) $$

$$ {}^{f}\bar{\omega}^{c}={}^{f}\bar{\omega}^{t}+[\boldsymbol{SFT}]^{t}\bar{\omega}^{ta}+[\boldsymbol{SFTa}]^{ta}\bar{\omega}^{c}\,. \qquad (4.18) $$

The relative angular velocities for the LEMMM are found in terms of the derivatives of the generalized coordinates which are inputted by the user. Appendix E outlines the calculation of the relative angular velocities in terms of the generalized coordinate derivatives.

Once the musculotendon actuator length and velocity are calculated, the musculotendon actuator model can be utilized to determine the muscle length and velocity, isometric muscle force, and dynamic muscle force. But one final variable of musculotendon geometry, the muscle moment arms, must be determined before individual muscle forces can be predicted.

4.1.5. Muscle Moment Arm and Muscle Moment Vector Determination

Muscles create joint torques that are the products of the muscle force and the muscle moment arms. These joint torques create angular accelerations that in time cause the angular displacements that we know as human movement. In LEMMM, the muscle moment arms are vectors from the instantaneous joint center to the muscle origin or insertion (or effective origin or effective insertion) whichever is closer to the joint center. Figure 4.3 shows generic moment arms for the hip and the ankle. The insertion and effective insertion are defined in the calcaneus reference frame. The moment arm vector expressed in the talus reference frame is

$$\overrightarrow{AI} = {}^{ta}\bar{r}^{c} + [STaC]\bar{I}_e \,. \qquad (4.19)$$

The *muscle moment vector* is derived from the general torque equation:

$$\bar{T} = \bar{r} \times \bar{F} \qquad (4.20)$$

where $\bar{r}$ is the moment arm vector and $\bar{F}$ is the muscle force vector.

Since the muscle creates force along its muscle path ($\overrightarrow{I_eO_e}$ in Figure 4.3), Eq. 4.20 can be expressed in the following way for the muscle in Figure 4.3:

$$\bar{T} = \overrightarrow{AI} \times \left(\|\bar{F}\| \frac{\overrightarrow{I_eO_e}}{\left\|\overrightarrow{I_eO_e}\right\|} \right). \qquad (4.21)$$

The *muscle moment vector* is then defined as:

$$\overrightarrow{ma^A} = \overrightarrow{AI} \times \frac{\overrightarrow{I_e O_e}}{\left\| \overrightarrow{I_e O_e} \right\|} . \qquad (4.22)$$

The torque equation can then be rewritten in terms of the *muscle moment vector:*

$$\vec{T} = \left\| \vec{F} \right\| \left(\overrightarrow{ma^A} \right) \qquad (4.23)$$

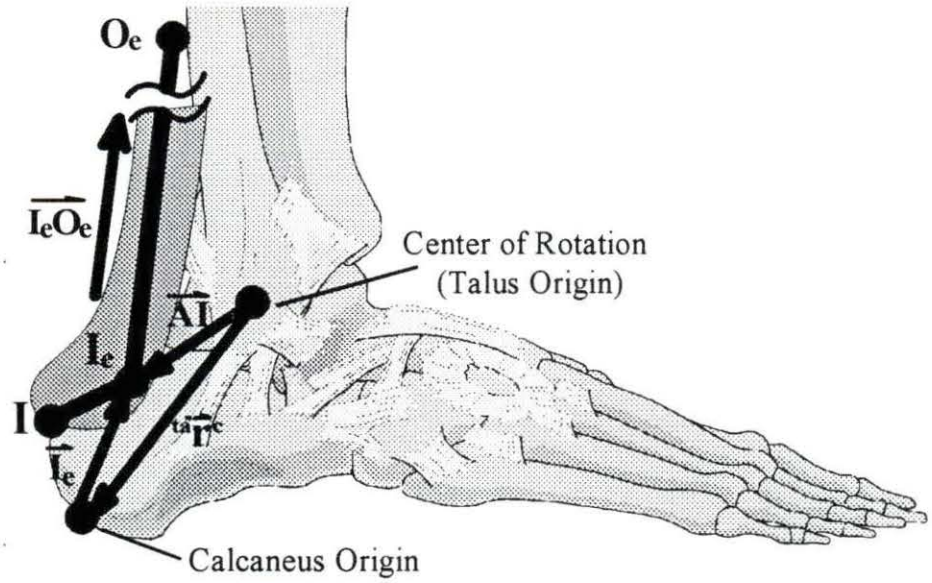
For the single pathline generic hip muscle shown in Figure 4.4, the insertion is in the femur reference frame and the moment arm vector is easily expressed as

$$\vec{HI} = \vec{I} \qquad (4.24)$$

and the *muscle moment vector* is:

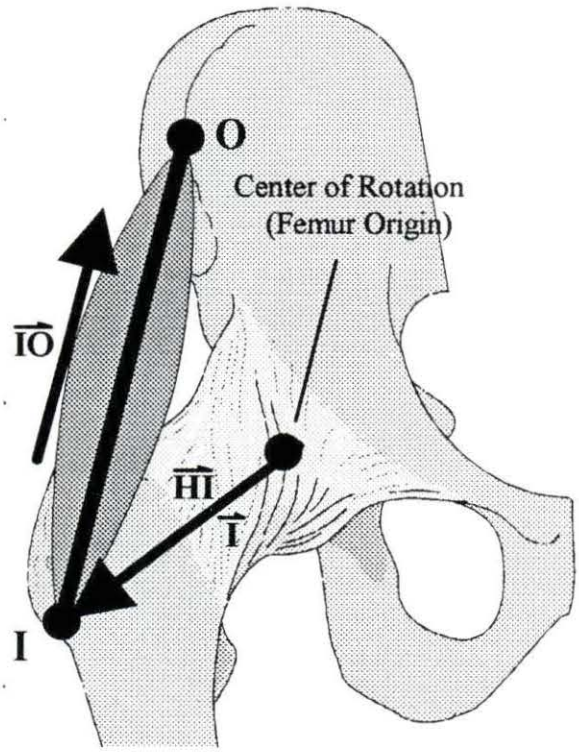
$$\overrightarrow{ma^H} = \overrightarrow{HI} \times \frac{\overrightarrow{IO}}{\left\| \overrightarrow{IO} \right\|} . \qquad (4.25)$$

The muscle moment vector gives an indication of what kind of motion the muscle is capable of producing and also the primary action of the muscle. The primary action of the muscle can be determined by examining the magnitudes and sign of the orthogonal components (X,Y,Z) of the muscle moment vector, i.e., a hip muscle with the largest muscle moment vector component in the negative Z direction would be primarily a hip extensor. However, the reader is cautioned not to rely too heavily on this method to quantify a muscle's function because muscles can have different primary functions depending on the positioning of the limbs. The muscle can also have an action at more than one joint and muscles can even accelerate joints that they do not cross. The muscle moment vector is also utilized for calculating the linear constraints for the optimization routine that predicts the individual muscle forces.



Generic multi-pathline ankle muscle showing the vectors involved in calculating the muscle moment arm vector and also the muscle moment vector. The insertion and effective insertion are defined in the calcaneus reference frame and the center of rotations for the ankle joint and subtalar joint are located at the talus origin.

Figure 4.3 - Vectors required for the calculation of the muscle moment arm and muscle moment vector for a generic ankle muscle



Generic single pathline hip muscle showing the vectors required for the calculation of the muscle moment arm vector and muscle moment vector when the insertion of the muscle and the joint's center of rotation are in the same reference frame. The insertion is in the femur frame and the center of rotation for the hip joint is the femur origin.

Figure 4.4 - Vectors required for the calculation of the muscle moment arm and muscle moment vector for a generic hip muscle

4.2. The Musculotendon Actuator Model

The contraction dynamics model of the muscle-tendon complex (musculotendon actuator) is based on a generic Hill model [26]. The Hill model, shown in Figure 4.5, consists of a *contractile element* (CE) and a *passive element* (PE) both of which contribute to muscle force.

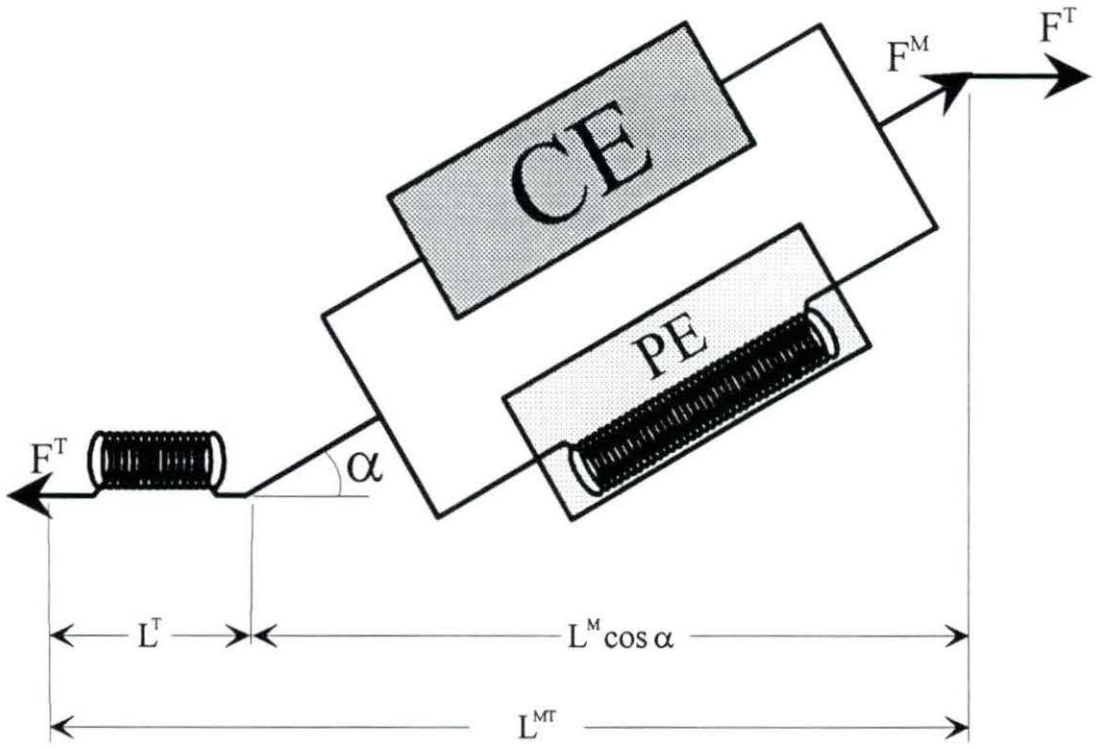


Figure 4.5 - Hill-based model for the musculotendon actuator

The force generated by the contractile element depends on muscle length (Figure 3.1, solid line), muscle velocity (Figure 3.2), and neural activation. The passive element depends on muscle length (Figure 3.1, long dashed line) and can only create force when the muscle is stretched beyond its optimal length. Another element called the series elastic element (SEE) which is used by some researchers [45] and lies in series with the contractile element, is

neglected here. The main reason for neglecting the SEE in this model [26] is that tendon compliance dominates the SEE compliance for most actuators. See Zajac [26] for a review of the controversy surrounding the series elastic element.

4.2.1. Musculotendon Scaling Parameters

The musculotendon actuator model has generic properties and is used for all muscles. To scale the properties for each individual muscle, four scaling parameters are used: 1) the maximum isometric force (F_O^M), 2) the optimum fiber length (L_O^M), 3) the pennation angle at the optimal fiber length (α_O), and 4) the tendon slack length (L_S^T). For the LEMMM, the musculotendon scaling parameters were taken from Delp [23] who calculated the tendon slack lengths and compiled values for the other three parameters from three previous researchers [46,47,48]. Delp's parameters are shown in Table 4.4 along with the muscle numbering convention used in this thesis.

Table 4.4 - Musculotendon scaling parameters

Muscle	Muscle Number	F_O^M (N)	L_O^M (cm)	α_O (deg)	L_S^T (cm)
gluteus medius 1	1	550.0	5.4	8.0	7.8
gluteus medius 2	2	380.0	8.4	0.0	5.3
gluteus medius 3	3	435.0	6.5	19.0	5.3
gluteus minimus 1	4	180.0	6.8	10.0	1.6
gluteus minimus 2	5	190.0	5.6	0.0	2.6
gluteus minimus 3	6	215.0	3.8	1.0	5.1
gluteus maximus 1	7	380.0	14.2	5.0	12.5
gluteus maximus 2	8	550.0	14.7	0.0	12.7
gluteus maximus 3	9	370.0	14.4	5.0	14.5
adductor magnus 1	10	345.0	8.7	5.0	6.0
adductor magnus 2	11	310.0	12.1	3.0	13.0
adductor magnus 3	12	445.0	13.1	5.0	26.0
adductor longus	13	420.0	13.8	6.0	11.0
adductor brevis	14	285.0	13.3	0.0	2.0
pectineus	15	175.0	13.3	0.0	0.1
iliacus	16	430.0	10.0	7.0	9.0

Table 4.4 (cont).

psoas	17	370.0	10.4	8.0	13.0
quadratus femoris	18	255.0	5.4	0.0	2.4
gemelli	19	110.0	2.4	0.0	3.9
piriformis	20	295.0	2.6	10.0	11.5
rectus femoris	21	780.0	8.4	5.0	34.6
semimembranosus	22	1030.0	8.0	15.0	35.9
semitendinosus	23	330.0	20.1	5.0	26.2
biceps femoris (long head)	24	720.0	10.9	0.0	34.1
gracilis	25	110.0	35.2	3.0	14.0
sartorius	26	105.0	57.9	0.0	4.0
tensor fasciae latae	27	155.0	9.5	3.0	42.5
vastus medialis	28	1295.0	8.9	5.0	12.6
vastus intermedius	29	1235.0	8.7	3.0	13.6
vastus lateralis	30	1870.0	8.4	5.0	15.7
biceps femoris (short head)	31	400.0	17.3	23.0	10.0
medial gastrocnemius	32	1115.0	4.5	17.0	40.8
lateral gastrocnemius	33	490.0	6.4	8.0	38.5
soleus	34	2830.0	3.0	25.0	26.8
tibialis posterior	35	1270.0	3.1	12.0	31.0
tibialis anterior	36	600.0	9.8	5.0	22.3
flexor digitorum longus	37	310.0	3.4	7.0	40.0
flexor hallucis longus	38	320.0	4.3	10.0	38.0
peroneus brevis	39	350.0	5.0	5.0	16.1
peroneus longus	40	755.0	4.9	10.0	34.5
peroneus tertius	41	90.0	7.9	13.0	10.0
extensor digitorum longus	42	340.0	10.2	8.0	34.5
extensor hallucis longus	43	110.0	11.1	6.0	30.5

4.2.2. Muscle Length/Velocity Determination

To determine both static muscle force and the dynamic muscle force, the muscle length must be found. The musculotendon actuator length and the muscle and tendon lengths shown in Figure 4.5 are related as follows:

$$L^{MT} = L^{T} + L^{M}\cos\alpha \ . \qquad (4.25)$$

Also from Figure 4.5, the muscle force and tendon force are related by the following expression:

$$F^T = F^M \cos\alpha \ . \qquad (4.26)$$

It should be noted that though Figure 4.5 and Eq. 4.26 show tendon force related to static muscle force (F^M), the tendon force is also related to dynamic muscle force (F^{dyn}) by the same relationship.

The cosine of the pennation angle (α) in Eqs. 4.25 and 4.26 changes with length and it is usually convenient to express this quantity in terms of the pennation angle at optimal fiber length (α_O) and the muscle length. Assuming a constant width for the muscle [45], the following relationship can be found from the muscle geometry:

$$L^M \sin\alpha = L_O^M \sin\alpha_O \ . \qquad (4.27)$$

Using a familiar trigonometric identity, the cosine term in Eqs. 4.25 and 4.26 can then be expressed in terms of the muscle length and the resting pennation angle:

$$\cos\alpha = \sqrt{1-\left(\frac{\sin\alpha_O}{\tilde{L}^M}\right)^2} \qquad (4.28)$$

where ($\tilde{L}^M$) is the normalized muscle length given in Eq. 3.1. Eqs. 4.25 and 4.26 are then normalized (divided) by optimal fiber length and the maximum isometric force, respectively, which yields:

$$\tilde{L}^{MT} = \tilde{L}^T + \tilde{L}^M \sqrt{1-\left(\frac{\sin\alpha_O}{\tilde{L}^M}\right)^2} \qquad (4.29)$$

$$\tilde{F}^T = \tilde{F}^M \sqrt{1-\left(\frac{\sin\alpha_O}{\tilde{L}^M}\right)^2} \qquad (4.30)$$

While the tendon force can be calculated from the muscle force-length relationship as shown in Eq. 4.30, it can also be calculated utilizing the tendon force-strain diagram (Fig. 3.4) and a linearized model of tendon. For a linear model, the tendon force can be found from

$$F^T = k^T \Delta L^T \tag{4.31}$$

where k^T is the tendon stiffness and ΔL^T is the tendon elongation. Normalizing the tendon force by the maximum isometric force and the elongation by the optimal fiber length, the tendon stiffness is subsequently normalized by $\left(L_O^M / F_O^M\right)$ and Eq. 4.31 becomes:

$$\tilde{F}^T = \tilde{k}^T \left(\tilde{L}^T - \tilde{L}_S^T\right) \tag{4.32}$$

It can be shown that the normalized tendon stiffness is represented by the following relationship:

$$\tilde{k}^T = \frac{E^T}{\sigma_O^T} \frac{1}{\tilde{L}_S^T} \tag{4.33}$$

where E^T is the tendon modulus, σ_O^T is the stress in tendon when the static muscle force equals the maximum isometric muscle force, and $\tilde{L}_S^T$ is the tendon slack length normalized by the optimal fiber length of the muscle. Using σ_O^T equal to 32 MPa and E^T equal to 1.2 GPa [26], the normalized tendon strain is conveniently expressed as:

$$\tilde{k}^T = \frac{37.5}{\tilde{L}_S^T}. \tag{4.34}$$

The final expression for normalized tendon force can be found by substituting Eqs. 4.29 and 4.34 into Eq. 4.32:

$$\tilde{F}^T = \frac{37.5}{\tilde{L}_S^T} \left[\left(\tilde{L}^{MT} - \tilde{L}^M \sqrt{1 - \left(\frac{\sin \alpha_O}{\tilde{L}^M} \right)^2} \right) - \tilde{L}_S^T \right]. \tag{4.35}$$

Since Eqs. 4.30 and 4.35 must yield the same normalized tendon force, an iterative procedure can be used to find a muscle length that will satisfy both Eq. 4.30 and Eq. 4.35. Given the musculotendon actuator length calculated by the method described in Section 4.1.3, the following iterative procedure is used to find the correct muscle length.

Iterative Procedure for Determining Muscle Length

1. Guess an initial value for the muscle length.
2. Calculate static muscle force using cubic-spline interpolation of Figure 3.1.
3. Calculate tendon force using Eq. 4.30.
4. Calculate tendon force using Eq. 4.35.
5. Calculate error between the the two tendon force calculations.
6. Adjust muscle length guess until error is within a certain tolerance.

Once the correct muscle length is found, it is differentiated to find the muscle shortening or lengthening velocity (V^M). It should be noted that the muscle velocity is different than the musculotendon actuator velocity (V^{MT}) which was introduced in Section 4.1.3, by the tendon velocity. In fact, in muscles with highly compliant tendons, the musculotendon actuator velocity may be in the opposite direction to the velocity of its muscle fibers, i.e., the origin-to-insertion distance may be lengthening while the muscle fibers are shortening, or vice versa.

4.2.3. Static and Dynamic Muscle Force Determination

Given the muscle length and muscle velocity, the static and dynamic muscle forces may be calculated. The muscle force-length relationship (Figure 3.1) is interpolated with a cubic spline at the normalized muscle length yielding the normalized static muscle force. The static muscle force is then found by multiplying the normalized static muscle force by that muscle's maximum isometric muscle force:

$$F^M = \tilde{F}^M\left(F_0^M\right). \qquad (4.36)$$

The normalized dynamic muscle force is found by cubic spline interpolation of the muscle force-velocity relationship (Figure 3.2) at the normalized muscle velocity (Eq. 3.4). Since the force-velocity relationship is scaled by the force-length relationship, the normalized dynamic force is multiplied by the static muscle force to determine dynamic muscle force:

$$F^{dyn} = \tilde{F}^{dyn}\left(F^{M}\right). \qquad (4.37)$$

The dynamic muscle force represents the maximum force the muscle can exert corresponding to that specific configuration, i.e., muscle length and velocity.

0.1. Non-linear Optimization for Individual Muscle Force Prediction

For the prediction of individual muscle forces, i.e., the activation pattern and force magnitudes, a non-linear static optimization scheme is utilized. To set up the optimization problem, an objective function must be chosen and appropriate constraints and bounds on the variables must be applied. An appropriate algorithm then determines the values of the variables, given the constraints and bounds, that minimizes the objective function.

For the LEMMM, the non-linear objective function of Pedotti [13] was utilized because it showed the best temporal validation with EMG data:

$$\Phi = \sum_{i=1}^{n}\left(\frac{F_i}{F_i^{\max}}\right)^2 \qquad (4.38)$$

where F_i is the individual muscle force of muscle "i" (i=1,43, the muscle numbers given in Table 4.4) and $F_i^{\max}$ is the maximum muscle force of muscle "i" which is equivalent to the dynamic muscle force (F^{dyn}) for that muscle as discussed in the previous section. This objective function takes into account the instantaneous state of each muscle because the maximum muscle force depends on the instantaneous length and velocity of the muscle.

The static optimization problem is solved at each inputted time step. There are 43 variables (the individual muscle forces) in each optimization problem with upper and lower bounds and six linear torque constraints. The specifics of the formulation of optimization problem for the LEMMM are given below. The objective function is

$$\Phi = \sum_{i=1}^{43}\left(\frac{F_i}{F_i^{dyn}}\right)^2 \qquad (4.39)$$

with the following bounds on the variables:

$$0 \le F_i \le F_i^{dyn} \tag{4.40}$$

which states that the individual muscle forces must be tensile and that they must be equal to or less than the maximum force allowable at that instant. Furthermore, the variables are subject to the following six scalar constraints:

$$T_x^{hip} = \sum_{i=1}^{27} \left(ma_x^H \right)_i \cdot F_i \tag{4.41}$$

$$T_y^{hip} = \sum_{i=1}^{27} \left(ma_y^H \right)_i \cdot F_i \tag{4.42}$$

$$T_z^{hip} = \sum_{i=1}^{27} \left(ma_z^H \right)_i \cdot F_i \tag{4.43}$$

$$T_z^{knee} = \sum_{i=21}^{33} \left(ma_z^K \right)_i \cdot F_i \tag{4.44}$$

$$T_x^{subt} = \sum_{i=32}^{43} \left(ma_x^A \right) \cdot F_i \tag{4.45}$$

$$T_z^{ank} = \sum_{i=32}^{43} \left(ma_z^A \right)_i \cdot F_i \tag{4.46}$$

where the T_x, T_y, and T_z are the X, Y, and Z components of the hip, knee, and ankle torques, and the ma_x, ma_y, and ma_z are the components of the muscle moment vectors for the ith muscle (Section 4.1.5). These constraints represent the fact that the sum of all the individual muscle forces crossing a certain joint times their respective muscle moment vector components must equal the component of the torque for that same joint.

4.4. Computer Implementation

The Lower-Extremity Muscle Mechanics Model (LEMMM) is written in FORTRAN and is implemented on the ULTRIX 4.2 operating system on a DECStation 5000. The

program utilizes numerical routines from the Numerical Algorithms Group (NAG) subroutine library. The program is written in standard ANSI 77 FORTRAN and therefore should be portable to other workstations or computers provided the NAG library is available. The full listing of the program code is provided in Appendix F. A user can also substitute routines from available libraries for the NAG routines with minimum effort. The LEMMM produces 16 output files which contain data on specific variables (i.e., muscle length, static muscle force, etc.) at each time step. The first column contains the time value for each datapoint and next 43 columns contain the specific variable values for the 43 muscles. The files are comma-delimited to facilitate easy import into Lotus 1-2-3® or Microsoft Excel®. For a listing and description of the output files, see Appendix G.

The required input to the program is in the form of an ASCII data file. The user must provide the generalized coordinates and their derivatives, the standard plane film angles for the ankle, subtalar, and MTP joints and their derivatives, and the joint torques, as discussed in Section 4.1.2. For the format of the required data file and sample data for the gait analysis, see Appendix H.

5. EXAMPLE -- HUMAN GAIT

One of the more studied movements in biomechanics is human gait. Thus, there is a wealth of kinematic and kinetic data on this movement. Electromyographic (EMG) data for human gait is also readily available allowing for the temporal validation of force prediction models. Since the main focus of this thesis is the development of the LEMMM, an extensive biomechanical analysis of the movements associated with human gait will not be presented in this chapter. Instead the focus will be on temporal validation with available EMG data. The results for 24 muscles are presented. These muscles were chosen because they have significant actions in the sagittal plane and because there is EMG gait data available.

5.1. Kinematic and Joint Moment Inputs

The data used for this gait analysis is from Winter [5]. The motion was filmed at a camera speed of 69.9 frames/sec and was digitized frame by frame to establish landmark coordinates. These coordinates were smoothed, and the joint kinematics were calculated. It should be noted that the convention for joint angles used by Winter [5] is different than the convention used by the LEMMM that being that right-hand rotations between segments are positive in the LEMMM. The signs of the inputs are changed appropriately to match the LEMMM conventions in Table 4.3. The torque conventions for the sagittal plane are consistent between Winter [5] and the LEMMM. The joint angular orientations (in the LEMMM convention) are plotted in Figure 5.1, the joint angular velocities in Figure 5.2, and the joint torques in Figure 5.3, all graphed as a function of gait cycle (GC) percentage. A gait cycle of zero percent corresponds with right foot heel strike, from 0 to 61.83 percent GC represents right foot contact with the floor, 61.83 percent GC is right foot toe off, and 61.83 to 100 percent GC is right leg swing returning to right foot heel strike at 100 percent GC.

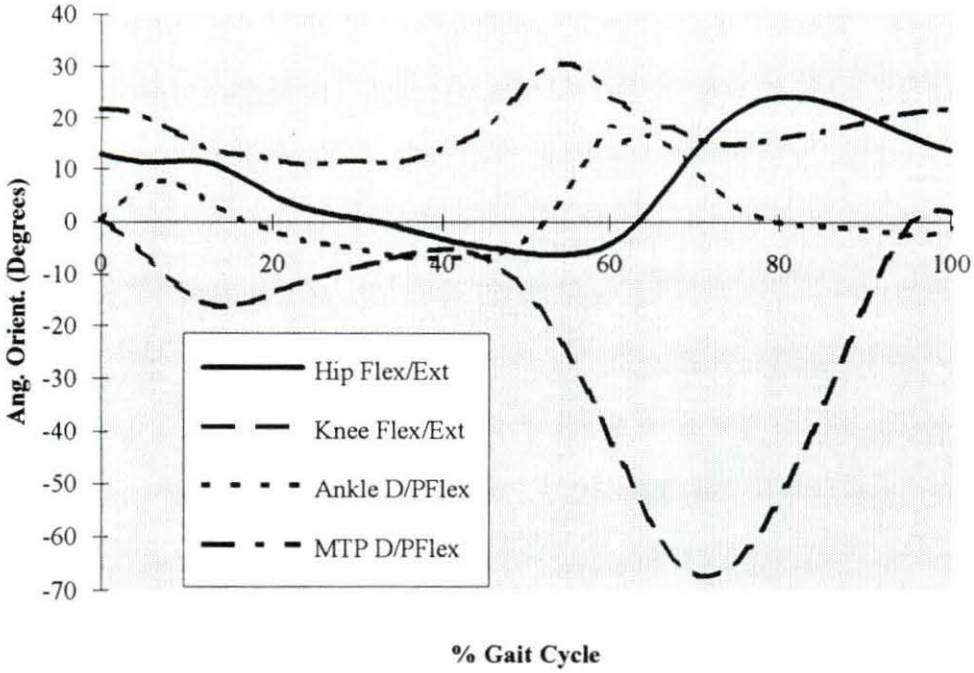


Figure 5.1 - Angular orientation inputs for human gait

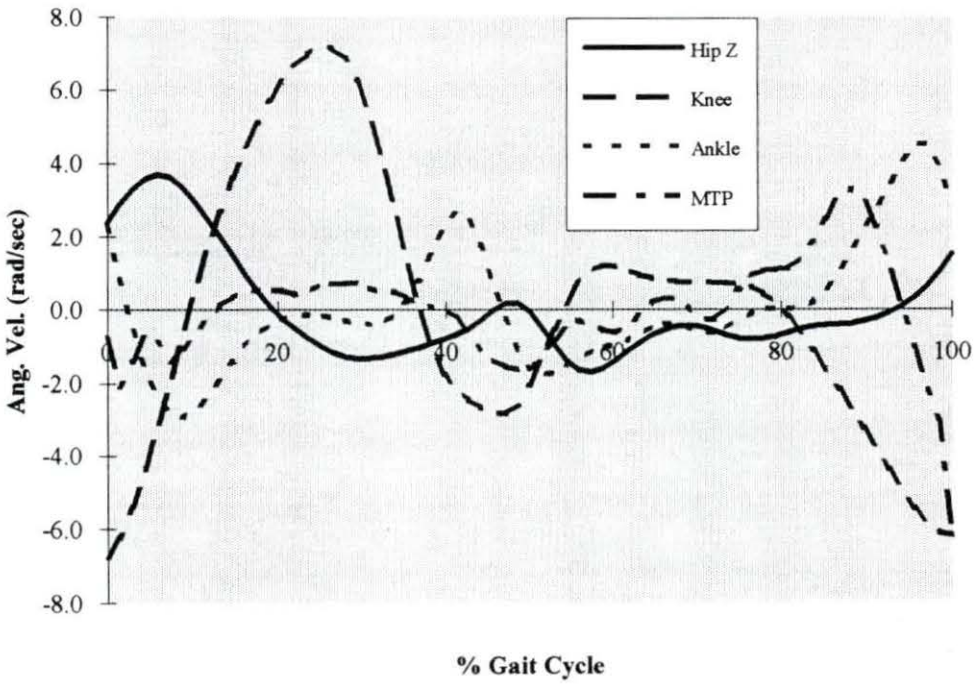


Figure 5.2 - Angular velocity inputs for human gait

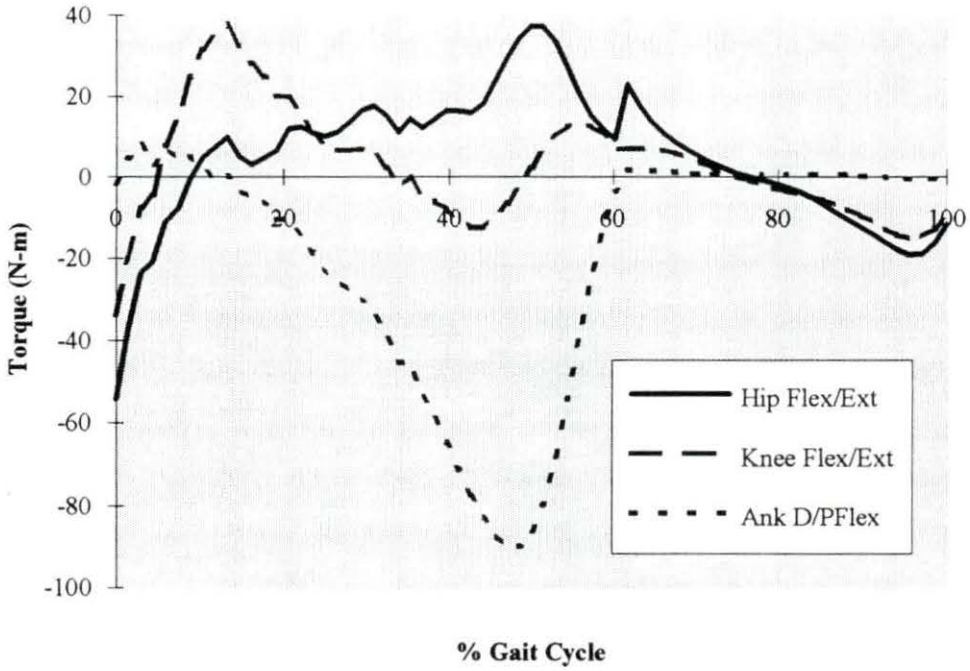


Figure 5.3 - Joint torque inputs for human gait

5.2. Results

The force prediction results of the human gait analysis are presented in nine graphs (Figures 5.4-5.12) showing the individual muscle force predictions for the entire gait cycle. Since the relationship between muscle force and EMG is unknown during dynamic conditions, EMG as understood today cannot be used as a quantitative validation tool, but can be used to temporally validate the muscle force predicitions (i.e., is the muscle firing or resting in a given period of time?). Included on the graphs are shaded regions representing the EMG activity envelopes which allow for temporal comparison between muscle activity and the predicted muscle activation. The EMG envelopes are taken from two sources. One source is from Yamaguchi and Zajac [49] who compiled EMG data from nine previous researchers and averaged these data to form their envelopes. The other source is from Pedersen *et al.* [17]

who defined their envelopes as EMG activity with magnitudes at least 20 percent of the peak EMG voltage recorded.

The force predictions for the primary hip extensors, the gluteus maximus 1 (GMax1, superior component), the gluteus maximus 2 (GMax2, middle component), and the gluteus maximus 3 (GMax3, inferior component) are shown in Figure 5.4 along with EMG envelopes (dark gray zones) from Pedersen *et al.* [17]. The predicted muscle activity temporally matches the EMG envelopes.

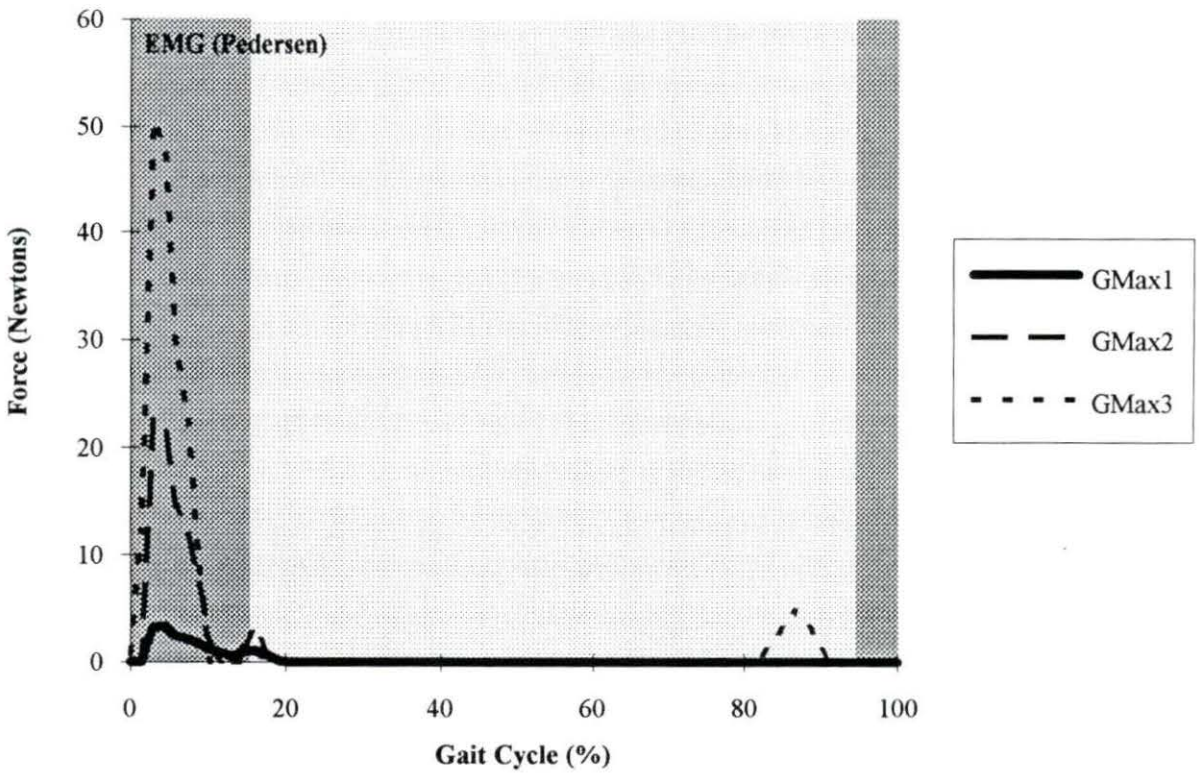


Figure 5.4 - Predicted muscle forces for the gluteus maximus muscles

These hip extensors act concentrically (i.e., muscle has a shortening velocity) on the femur from heel contact (0% GC) through early stance (0-20% GC) to help the quadriceps control the amount of knee flexion. They also act to control the torso from rotating too far forward over the hip joint during this same time period. The GMax3 (inferior component)

acts concentrically during late leg swing to control the amount of hip flexion, though this predicted force is low in magnitude and lies slightly outside the EMG envelope.

The gluteus medius muscle, represented in the LEMMM model as three separate muscles, (GMed1 - anterior component, GMed2 - middle component, and GMed2 - posterior component), is not a primary hip extensor. However, the gluteus medius is active as a synergist with the gluteus maximus during portions of the gait cycle. Figure 5.5 shows the three gluteus medius muscles along with the EMG envelope from Yamaguchi and Zajac [49].

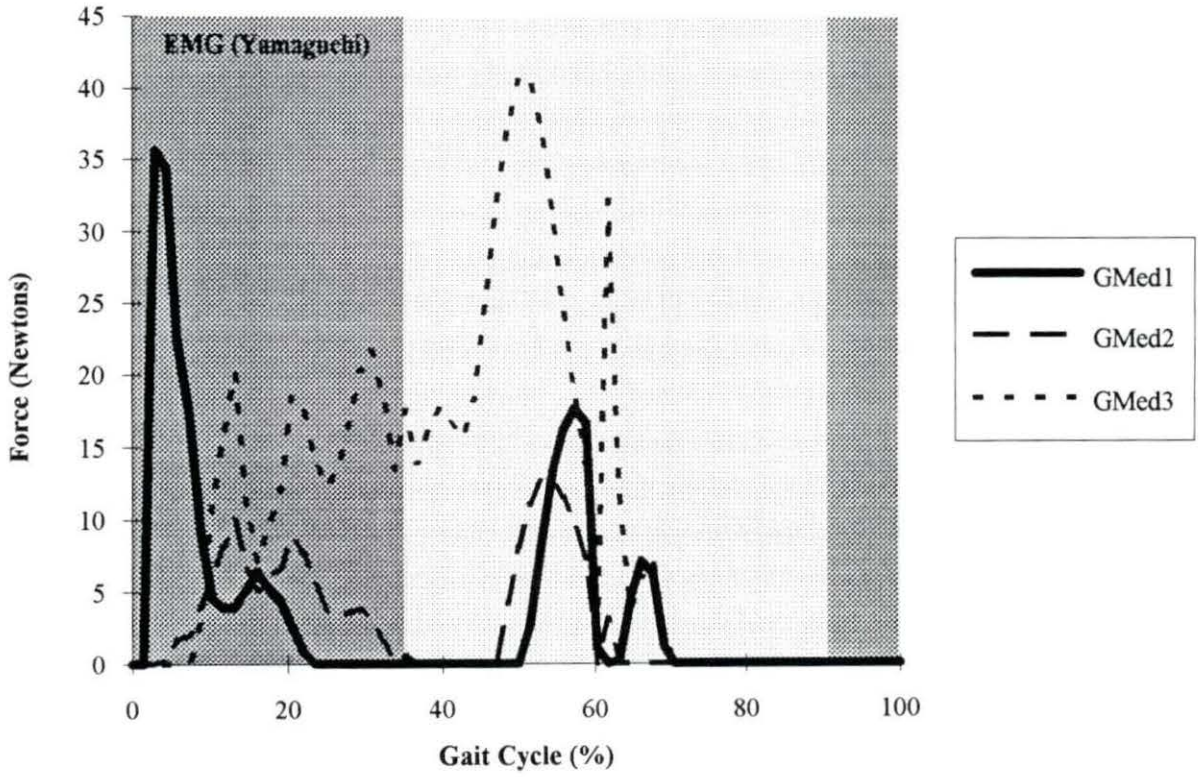


Figure 5.5 - Predicted muscle forces for the gluteus medius muscles

The muscles are concentrically active in the early stage of stance (0-20% GC) aiding the gluteus maximus muscles in controlling the amount of knee flexion and in preventing too much forward rotation of the torso over the legs. Force predictions for the concentric actions

of all three muscles agree well temporally with the EMG envelopes. The model predicts eccentric action (i.e., muscle has a lengthening velocity) in all three muscles leading up to toe off (61.83% GC). This action acts antagonistically against the hip flexors to help limit the amount of hip flexion that is produced as the hip flexors pull the femur upward and forward during the first phase of leg swing. Preceding toe off, a portion of the force in GMed2 and GMed3 is passive, i.e., these muscles are stretched past their optimum fiber lengths. Passive muscle force does not necessarily create EMG activity which may explain the lack of an EMG envelope in this region.

Force predictions for the adductor magnus muscles (AMag1 - superior component, AMag2 - middle component, AMag3 - inferior component) are shown in Figure 5.6. These muscles are primarily adductors of the hip. However, they sometimes act as hip extensors.

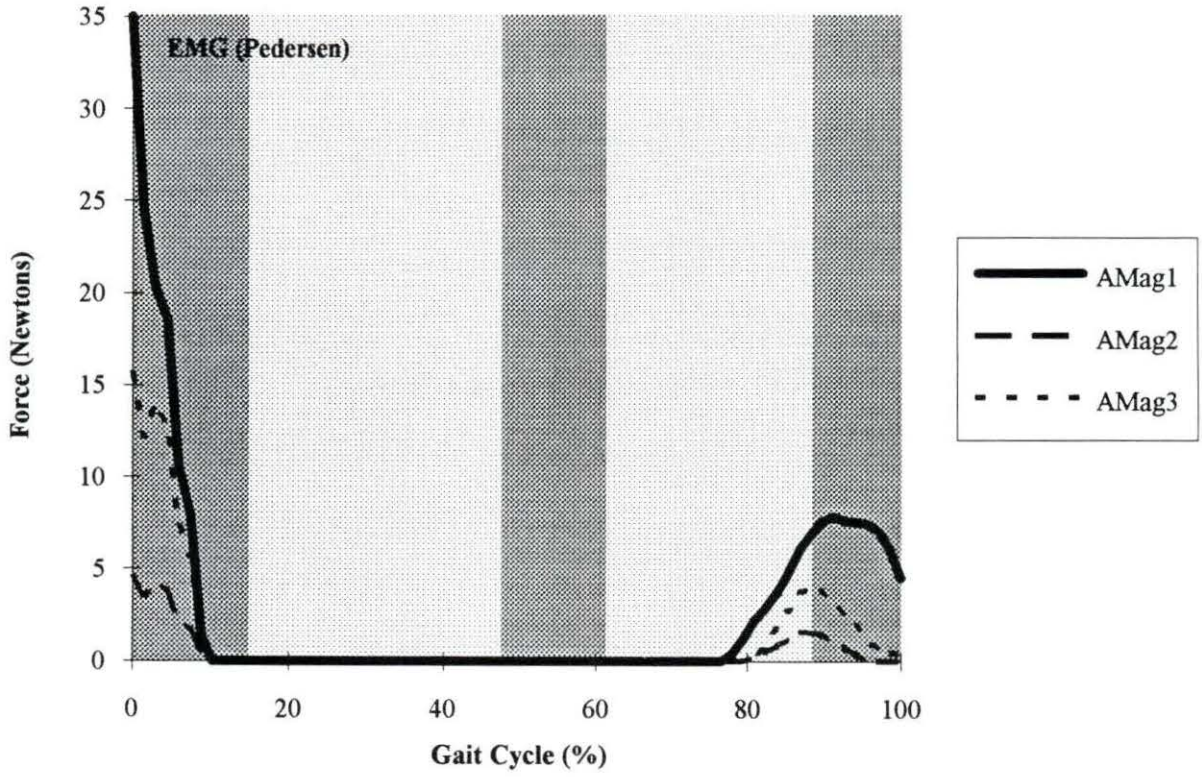


Figure 5.6 - Predicted muscle forces for the adductor magnus muscles

The AMag1 (inferior) muscle can even act as a hip flexor in certain anatomical positions. In gait, all three muscles act as hip extensors, synergistically with the gluteus maximus muscles to limit hip flexion during early stance and during the last half of leg swing (80-100% GC). The predicted muscle activity fits well into the EMG envelopes of Pederson *et al.* [17].

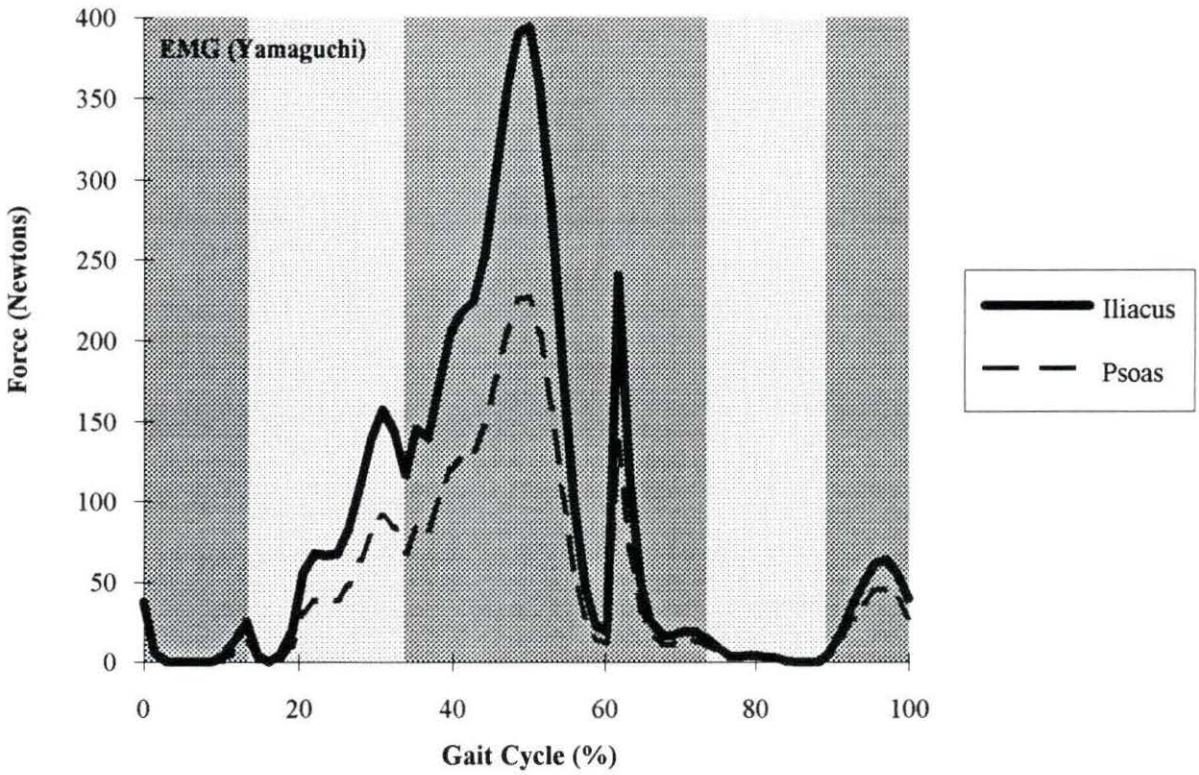


Figure 5.7 - Predicted muscle forces for the iliacus and psoas muscles

The iliacus and psoas muscles shown in Figure 5.7 are flexors of the hip. The predicted muscle activity correlates very well with the EMG envelopes provided by Yamaguchi and Zajac [49]. The initial action of these muscles begins during mid-stance (approximately 30% GC) and is eccentric with the purpose of stopping the backward rotation of the thigh. Concentric action begins to pull the limb upward and forward beginning with

approximately 55% gait cycle, continues through toe-off (61.83% GC), and ends during the first half of leg swing (approximately 75% GC). The muscles are then relaxed during the first part of late leg swing (80-90% GC), but begin to exert force eccentrically in late leg swing to maintain the hip flexion needed for heel contact.

The hamstring muscles (semimembranosus, semitendinosus, and the biceps femoris - long head) are the primary knee flexors. The predicted activity shown in Figure 5.8 excellently matches the EMG envelopes of Yamaguchi and Zajac [49]. The two regions of activity represent two primary concentric actions of these muscles. The first action, during early stance (0-20% GC), creates knee flexion which helps accomodate the weight transfer

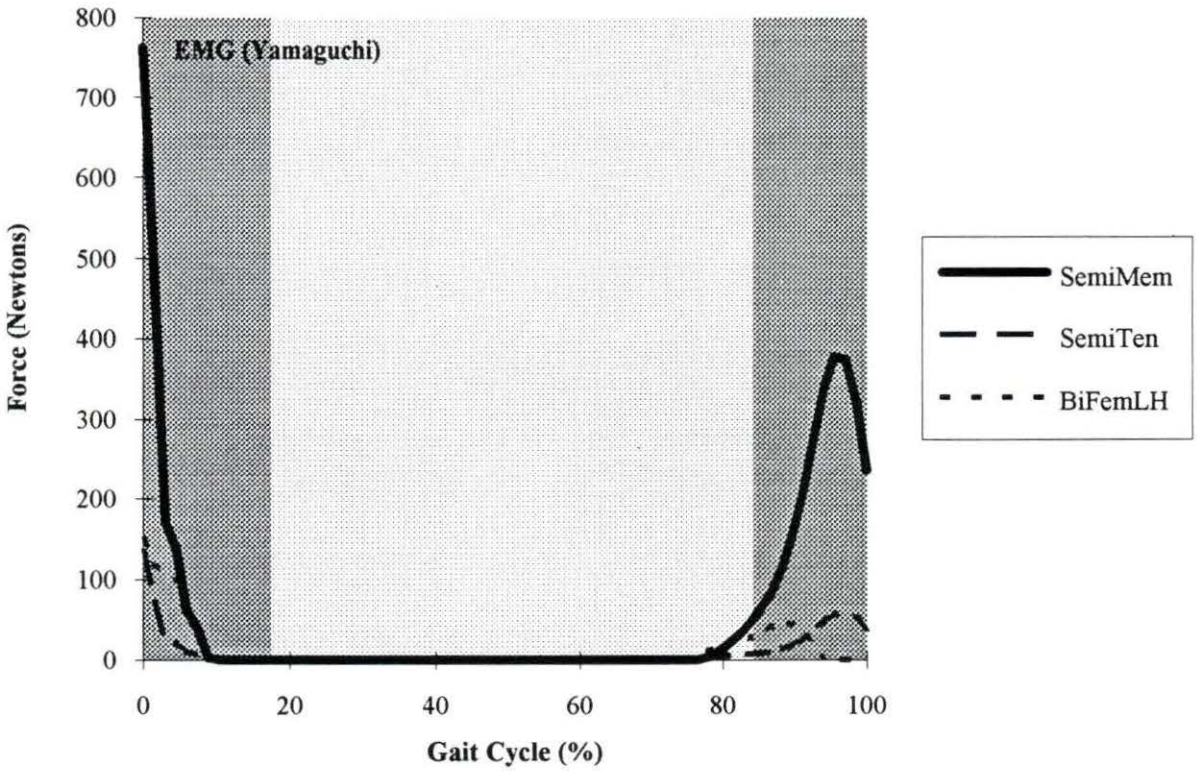


Figure 5.8 - Predicted muscle forces for the hamstring muscles

from the heel to the ball of the foot. The second action, occurring during late leg swing (80-100% GC), creates knee flexion to prepare the lower leg for heel contact.

The quadriceps muscle group includes the vastus medialis (VasMed), the vastus intermedius (VasInt), the vastus lateralis (VasLat), and the rectus femoris (RF, shown separately in Figure 5.10), which are the primary extensors of the knee. The three vasti muscles act only about the knee and are shown in Figure 5.9 along with EMG envelopes from Yamaguchi and Zajac [49]. The vasti muscle activity correlates well temporally with the EMG envelopes.

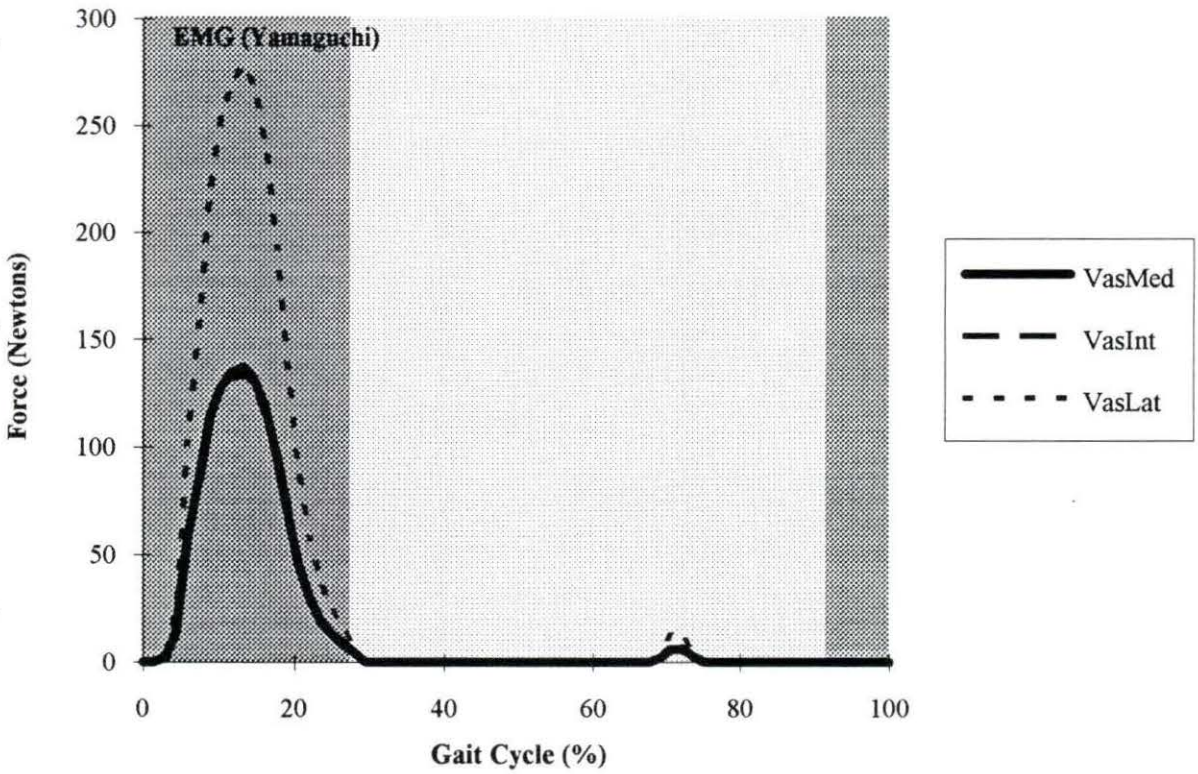


Figure 5.9 - Predicted muscle forces for the vasti muscles

The activity in early stance is an eccentric action by all three of the muscles that control the amount of knee flexion. This antagonistic activity aids walking coordination by balancing the flexion created by the hamstrings.

The fourth quadricep muscle, the rectus femoris, is bi-articular and acts to create flexion at the hip and extension at the knee. This muscle is shown in Figure 5.10 with accompanying EMG envelope from Pedersen *et al.* [17]. The rectus femoris (RF) eccentrically acts as an antagonist to the hamstrings in early stance (0-20% GC) to control the amount of knee flexion. This eccentric activity is also creating hip flexion, though the amount of flexion is decreasing in this period of the gait cycle. The predicted activity for the rectus femoris shows concentric action during late stance through early leg swing (50-75% GC) when the muscle is working as a synergist with the iliacus and psoas to flex the hip as the femur is lifted upward and forward to initiate leg swing.

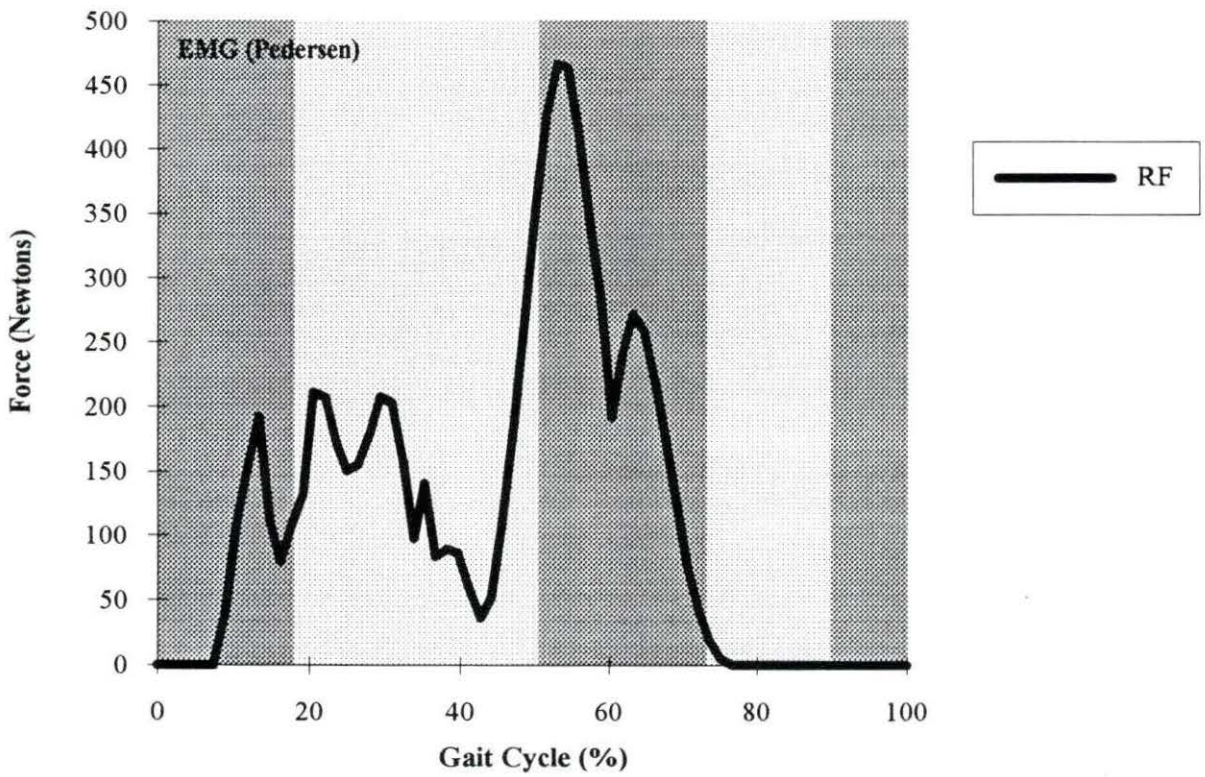


Figure 5.10 - Predicted muscle force for the rectus femoris

The plantarflexors of the ankle including the medial and lateral heads of the gastrocnemius (MedGas and LatGas, respectively) and the soleus (Sol) are among the strongest muscles in the body. The predicted activity for these muscles is shown in Figure 5.11. Two EMG envelopes [49] are shown, the larger soleus EMG envelope is shown in the lighter gray while the smaller gastrocnemius EMG envelope is shown in darker gray. The timing of the force predictions for both the soleus and both heads of the gastrocnemius match the EMG envelopes very well.

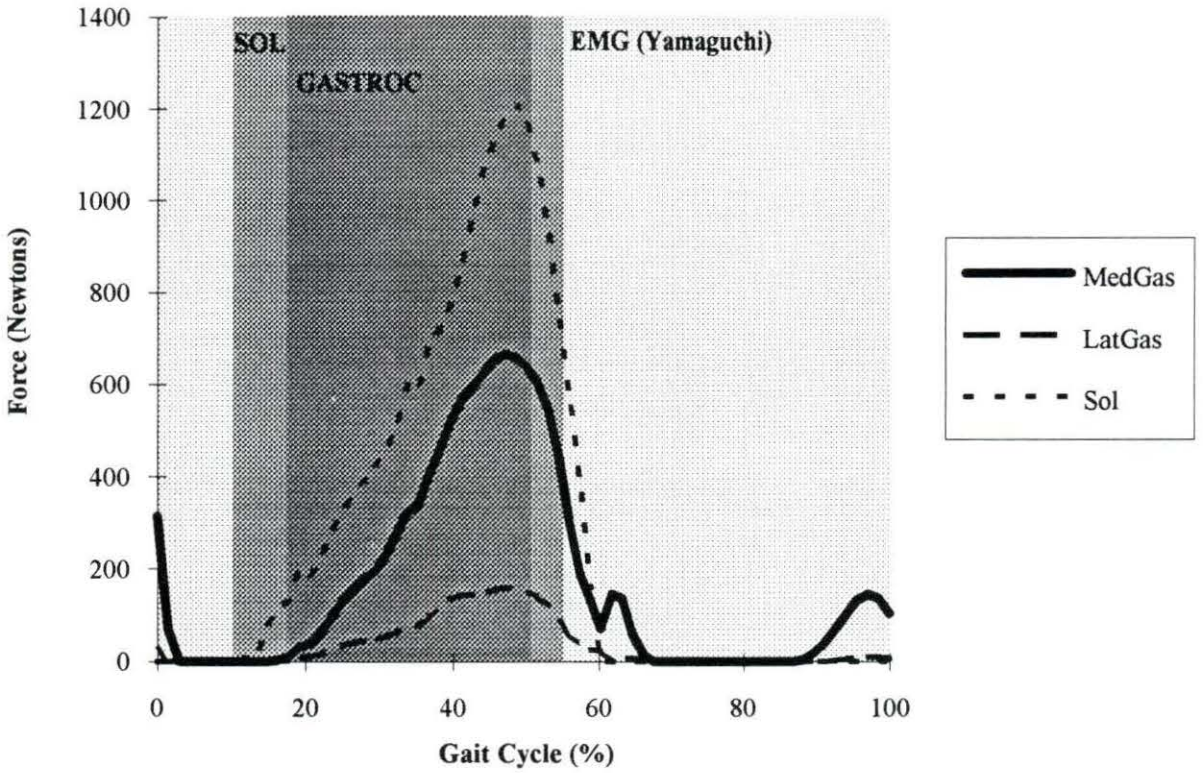


Figure 5.11 - Predicted muscle forces for the ankle plantarflexors

During midstance (20-40 % GC), the plantarflexors increase their concentric activity to control the forward rotation of the leg over the foot which is flat on the ground. In late stance (40-60% GC), there is a eccentric pre-stretching of the plantarflexors followed by an

enormous concentric contraction (note the force magnitudes in Figure 5.11) that causes the foot to push off the ground. The plantarflexor activity then drops to zero as the foot completely clears the ground (approximately 60-63% GC).

The ankle dorsiflexors including the tibialis anterior (TibAnt), the extensor digitorum longus (ExtDig), and the extensor hallucis longus (ExtHal) are shown in Figure 5.12. The EMG envelopes [49] and the predicted muscle activity very closely match. The dorsiflexors act concentrically to position the foot for heel contact near the end of leg swing (see the TibAnt and ExtDig peaks at 95% GC). After heel contact, the dorsiflexors act eccentrically with great force to lower the foot to the ground--this activity is completed by 15% GC. The dorsiflexors also act as antagonists against the plantarflexors to control the reverse rotation of the foot as the foot swings back immediately after toe off (61.83% GC).

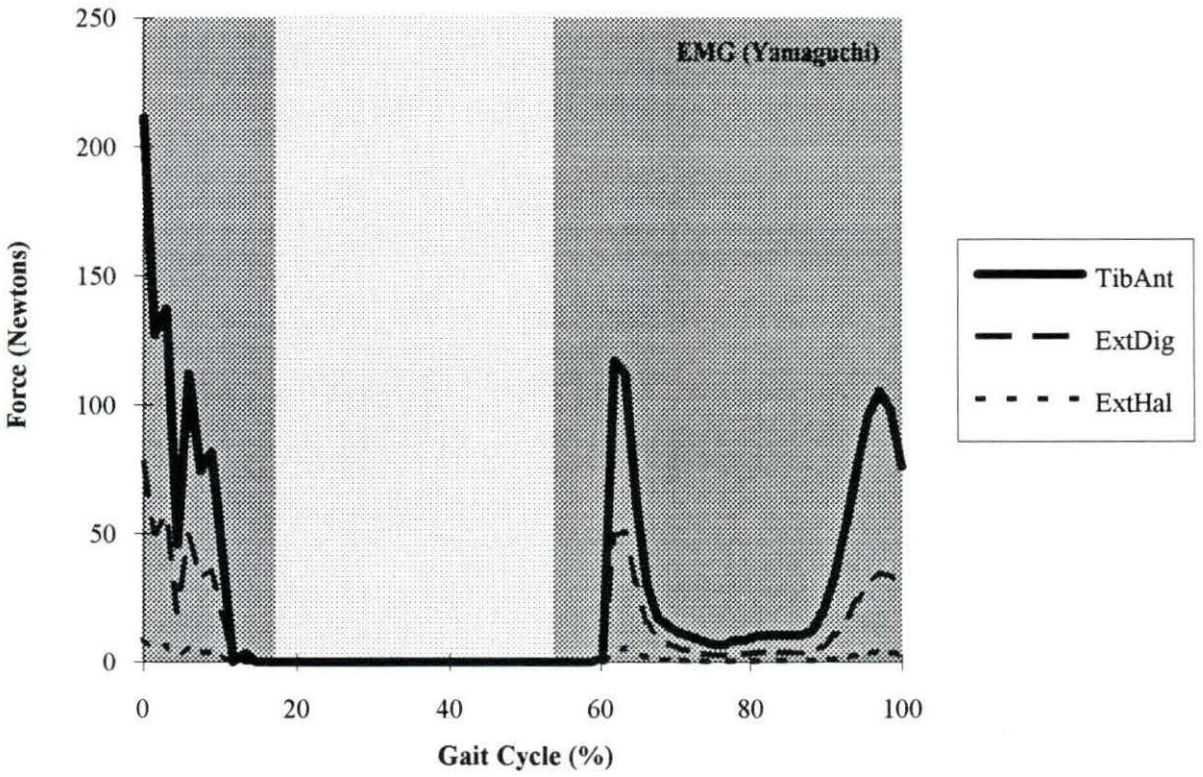


Figure 5.12 - Predicted muscle forces for the ankle dorsiflexors

5. CONCLUSIONS

This thesis has developed a lower extremity muscle mechanics model (LEMMM) to be utilized in a variety of fields which could include rehabilitation engineering, physical therapy, and sports biomechanics. Given the joint kinematics and the joint torques, the model can predict the activation pattern and accompanying force magnitudes for 43 lower extremity muscles. The muscle activation levels and individual muscle forces are predicted through the use of static non-linear optimization.

The model will also provide useful information on the musculoskeletal geometry (musculotendon actuator lengths and velocities, and moment arms), the muscle lengths and velocities, and the static and dynamic muscle forces for full activation. For instance a physical therapist working with a knee surgery patient on a Cybex® isokinetic testing machine may not be interested in individual force predictions, but instead the maximum dynamic force in the vastus medialis for a certain knee angle and velocity. Given the knee kinematics and torques from the isokinetic test, the LEMMM could calculate this dynamic muscle force for the therapist. The therapist could then chart the progress of the patient to determine strength recovery not just in terms of muscle torques, but in terms of predicted muscle forces.

The timing of the individual muscle force predictions has been shown to correlate well with EMG data for human gait. While there is no quantative validation of the magnitude of the forces predicted, i.e., the muscles activation level, the predicted forces are bounded by the calculated maximum muscle force (dynamic muscle force) which is an indicator of the instantaneous muscle state. Care should be taken in supplying inputs to the LEMMM; muscle calculations and force predictions are sensitive to both kinematic and torque inputs. The kinematic inputs should be smoothed to remove unwanted noise and the joint torques should be calculated utilizing force plate data to establish the highest level of accuracy. Every

attempt should be made to obtain EMG data from the activity to use for temporal validation. This is especially important when the activity being analyzed is novel or complicated.

While the model in its present form supplies the user with a wealth of information on several muscle parameters, there are additions and improvements that could be implemented in a second generation model. The power of the model as a biomechanics analysis tool would be greatly enhanced by the use of three-dimensional computer graphics for post-processing. Currently, the output data is supplied to the user in the form of output files to be imported into a spreadsheet for analysis. While numerical analysis is certainly important, visual data presentation of the bones and muscles complete with animation and a color coding scheme for each muscle based on force magnitude would certainly enhance understanding of the event being studied.

Additional improvements would focus on the user interface. With the age of computer windowing systems and the graphical user interface (GUI), commercial computer software packages do not require a user to respond to prompts by entering text or to input data to a program by writing a data file using a ASCII text editor. To develop a GUI for the LEMMM, a windows toolkit on a workstation could be utilized and an interface could be written in C to link the GUI and the FORTRAN code of the LEMMM. Another way to establish a GUI would be to port the entire program to C and write the interface as an integral portion of the program. However, this approach would require several numerical routines to be written in C to replace the FORTRAN subroutines from the NAG subroutine library.

While improving the user interface and the post processing of the computer program are important, future model improvements should focus on the individual muscle force prediction. While the musculoskeletal model can continue to be improved (i.e., scale the muscle insertions and origins for different anthropometric measurements, develop and study differences in musculotendon scaling parameters for men, women, and children, etc.) to

continually improve the model, the method utilized to activate the muscles must be continually scrutinized. While it is not currently possible to utilize optimal control theory (dynamic programming) for a generic muscle prediction model, it has been used for specific models. In time, the static optimization may need to be replaced with a dynamic optimization scheme which would allow the user to input an objective function thereby retaining the general applicability of the model. This modification could be accomodated into the structure of the current LEMMM due to the modular structure of the model, i.e., the lower extremity model and musculotendon actuator models are separate from the individual force prediction method.

BIBLIOGRAPHY

1. W. C. Whiting, R. J. Gregor, R. R. Roy, and V. R. Edgerton. "A Technique for Estimating Mechanical Work of Individual Muscles in the Cat during Treadmill Locomotion." *Journal of Biomechanics*. 17 (1984): 685-694.

2. F. E. Zajac and M. E. Gordon. "Determining Muscle's Force and Action in Multi-Articular Movement." *Exercise and Sport Science Review*. 17 (1989): 187-230.

3. A. Cappozzo, T. Leo, and A. Pedotti. "A General Computational Method for the Analysis of Human Locomotion." *Journal of Biomechanics*. 8 (1975): 307-320.

4. F. M. L. Amirouche, S. K. Ider, and J. Trimble. "Analytical Method for the Analysis and Simulation of Human Locomotion." *Journal of Biomech. Engr.* 112 (1990): 379-386.

5. D. A. Winter. *Biomechanics and Motor Control of Human Movement*. 2nd ed. New York: Wiley, 1990.

6. J. P. Paul. "Bio-Engineering Studies of the Forces Transmitted by Joints: II. Engineering Analysis." In *Biomechanics and Related Bio-Engineering Topics*. Ed. R. M. Kenedi. Oxford: Pergamon Press, 1965.

7. J. B. Morrison. *The Force Transmitted by the Human Knee Joint during Activity*. Ph.D. Dissertation. Univ. of Strathclyde, U.K., 1967.

8. E. Y. Chao, J. D. Opgrande, and F. E. Axmear. "Three-dimensional Force Analysis of Finger Joints in Selected Isometric Hand Functions." *Journal of Biomechanics*. 9 (1976): 387-396.

9. M. A. MacConaill. "The Ergonomic Aspects of Articular Mechanics." In *Studies on the Anatomy and Functions of Bones and Joints*. Ed. F.G. Evans. Berlin: Springer, 1967.

10. A. Seireg and R. J. Arvikar. "A Mathematical Model for Evaluation of Forces in Lower Extremities of the Musculoskeletal System." *Journal of Biomechanics*. 8 (1973): 313-326.

11. D. D. Penrod, D. T. Davy, and D. P. Singh. "An Optimization Approach to Tendon Force Analysis." *Journal of Biomechanics*. 7 (1974): 123-129.

12. D. E. Hardt. "Determining Muscle Forces in the Leg during Normal Human Walking -- An Application and Evaluation of Optimization Methods. *Journal of Biomech. Engr.* 100 (1978): 72-78.

13. A. Pedotti, V. V. Krishnan and L. Stark. "Optimization of Muscle Force Sequencing in Human Locomotion." *Mathematical Biosciences.* 38 (1974): 123-129.

14. A.G. Patriarco, R.W. Mann, S.R. Simon, and J.M. Mansour. "An Evaluation of the Approaches of Optimization Models in the Prediction of Muscle Forces During Human Gait." *Journal of Biomechanics.* 14 (1981): 513-525.

15. R.D. Crowinshield. "Use of Optimization Techniques to Predict Muscle Forces." *Journal of Biomech. Engr.* 100 (1978): 88-92.

16. R. Shiavi. "Electromyographic Patterns in Adult Locomotion: A Comprehensive Review." *Journal of Rehabilitation Research and Development.* 22 (1985): 85-98.

17. D.R. Pederson, R.A. Brand, C. Cheng, and J.S. Arora. "Direct Comparison of Muscle Force Predictions Using Linear and NonLinear Programming." *Journal of Biomech. Engr.* 109 (1987): 192-199.

18. R.D. Crowinshield, R.C. Johnston, J.G. Andrews and R.A. Brand. "A Biomechanical Investigation of the Human Hip." *Journal of Biomechanics.* 11 (1978): 75-85.

19. W.F. Dostal and J.G. Andrews. "A Three Dimensional Biomechanical Model of Hip Musculature." *Journal of Biomechanics.* 14, No. 11 (1981): 803-812.

20. R.A. Brand, R.D. Crowinshield, C.E. Wittstock, D.R. Pederson, C.R. Clark, and F.M. van Krieken. "A Model of Lower Extremity Muscular Anatomy." *Journal of Biomech. Engr.* 104 (1982): 304-310.

21. R. H. Jenson and D. T. Davy. "An Investigation of Muscle Lines of Action about the Hip: A Centroid Line Approach vs the Straight Line Approach." *Journal of Biomechanics.* 8 (1975): 103-110.

22. M.G. Hoy, F.E. Zajac, M.E. Gordon. "A Musculoskeletal Model of the Human Lower Extremity: The Effect of Muscle, Tendon, and Moment Arm on the Moment-Angle Relationship of Musculotendon Actuators at the Hip, Knee, and Ankle." *Journal of Biomechanics.* 23, No. 2 (1990): 157-169.

23. S. L. Delp. *Surgery Simulation: A Computer Graphics System to Analyze and Design Musculoskeletal Reconstructions of the Lower Limb.* Ph.D. Dissertation. Stanford University, 1990.

24. F. Martini. *Fundamentals of Anatomy and Physiology*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

25. W. F. Ganong. *Review of Medical Physiology*. 14th ed. Norwalk, CN: Appleton & Lange, 1989.

26. F. E. Zajac. "Muscle and Tendon: Properties, Models, Scaling, and Application to Biomechanics and Motor Control." *Critical Reviews in Biomedical Engineering*. 17, No. 4 (1989): 359-411.

27. A. M. Gordon, A. F. Huxley, and F. J. Julian. "The Variation in Isometric Tension with Sarcomere Length in Vertebrate Muscle Fibres." *Journal of Physiology*. 184 (1966): 170-192.

28. R. I. Close. "Dynamic Properties of Mammalian Skeletal Muscles." *Physiological Reviews*. 52, No. 1 (Jan. 1972): 129-197.

29. A. Magid and D.J. Law. "Myofibrils Bear Most of the Resting Tension in Frog Skeletal Muscle." *Science*. 230 (1985): 1280.

30. W.O. Fenn and B. S. Marsh. "Muscular Force at Different Speeds of Shortening." *Journal of Physiology*. 85 (1935): 277-297.

31. A. V. Hill. "The Heat of Shortening and the Dynamic Constants of Muscle." *Proc. Royal Soc. London Ser. B*. 126 (1938): 136-195.

32. B. C. Abbott and D.R. Wilkie. "The Relation between Velocity of Shortening and The Tension-Length Curve of Skeletal Muscle." *Journal of Physiology*. 120 (1953): 214-223.

33. M. R. Clark and L. Stark. "Control of Human Eye Movements. I. Modelling of Extraocular muscle." *Mathematical Biosciences*. 20 (1974): 191-198.

34. A. L. Hof and J. W. Van den Berg. "EMG to Force Processing I: An Electrical Analogue of Hill Muscle Model. *Journal of Biomechanics*. 14 (1981): 747-752.

35. P. M. H. Rack and D. R. Westbury. "Elastic Properties of the Cat Soleus Tendon and their Functional Importance." *Journal of Physiology, London*. 347 (1984): 479-495.

36. U. Proske and D. L. Morgan. "Tendon Stiffness: Methods of Measurement and Significance for the control of movement. A Review." *Journal of Biomechanics*. 20 No. 1 (1987): 75-82.

37. D. H. Elliot. "Structure and Function of Mammalian Tendon." *Biol. Rev.* 40 (1965): 392-421.

38. A. Viidik. "Functional Properties of Collagenous Tissues." *Int. Rev. Connect. Tiss. Res.* 6 (1973): 127-215.

39. F.M.L. Amirouche. *Computational Methods in Multibody Dynamics*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

40. R.L. Huston, C.E. Passerello, and M.W. Harlow. "Dynamics of Multirigid-Body Systems." *Journal of Applied Mechanics.* 45 (1978): 889-894.

41. G.T. Yamaguchi and F.E. Zajac. "A Planar Model of the Knee Joint to Characterize the Knee Extensor Mechanism." *Journal of Biomechanics.* 22 (1989): 1-10.

42. V. T. Inman. *The Joints of the Ankle.* Baltimore, MD: Williams & Wilkins, 1981.

43. S. Siegler, J. Chen, and C.D. Schneck. "The Three-Dimensional Kinematics and Flexibility Characteristics of the Human Ankle and Subtalar Joints -- Part I. Kinematics." *Journal of Biomech. Engr.* 110 (1988): 364-373.

44. T. R. Kane and D. A. Levinson. *Dynamics: Theory and Applications.* New York: McGraw-Hill, 1985.

45. M. R. Pierrynowski and J. B. Morrison. "A Physiological Model for the Evaluation of Muscular Forces in Human Locomotion: Theoretical Aspects." *Math. Biosci.* 75 (1985): 69-101.

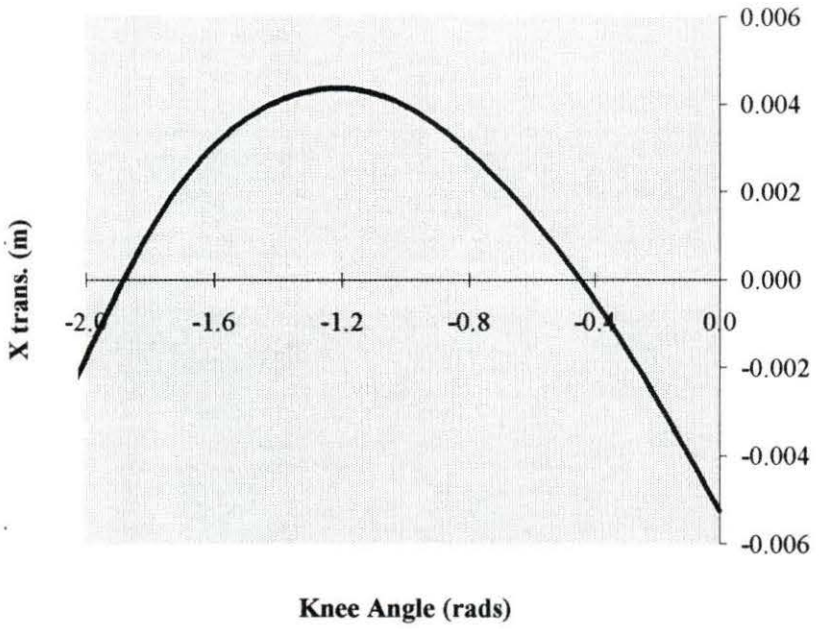
46. R. A. Brand, D. R. Pedersen, and J.A. Friederich. "The Sensitivity of Muscle Force Predictions to Changes in Physiological Cross-Sectional Area." *Journal of Biomechanics.* 104 (1982): 304-310.

47. J. A. Friederich and R. A. Brand. "Muscle Fiber Architecture in the Human Lower Limb." *Journal of Biomechanics*, 23 (1990): 91-95.

48. T. L. Wickiewicz, R.R. Roy, P.L. Powell, and V.R. Edgerton. "Muscle Architecture of the Human Lower Limb." *Clin. Orthop. Rel. Res.* 179 (1983): 275-283.

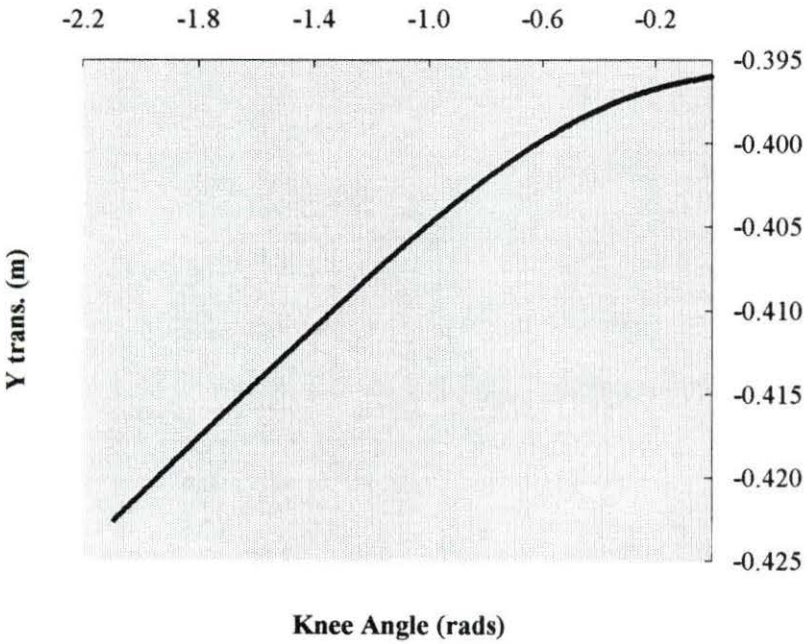
49. G. T. Yamaguchi and F. E. Zajac. "Restoring Natural Gait to Paraplegic Through Functional Neuromuscular Stimulation: A Feasibility Study." In *Issues in the Modeling and Control of Biomechanical Systems.* Ed. J.L. Stein, J.A. Ashton-Miller, and M.G. Pandy. New York: ASME, 1989.

APPENDIX A - KINEMATIC FUNCTIONS FOR THE KNEE



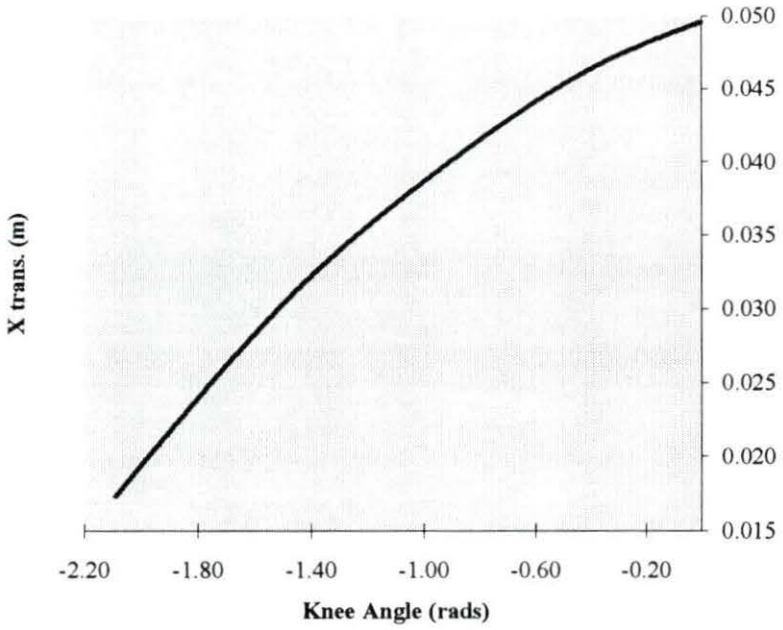
The X translation curve was created by fitting a cubic spline to control points from Delp [23]. Knee angle ranges from 120 degrees flexion to 0 degrees extension. This is the x component of the ${}^{f}\vec{r}^{t}$ vector. The z component of ${}^{f}\vec{r}^{t}$ is zero.

Figure A.1 - X translation from femur reference frame to tibia reference frame



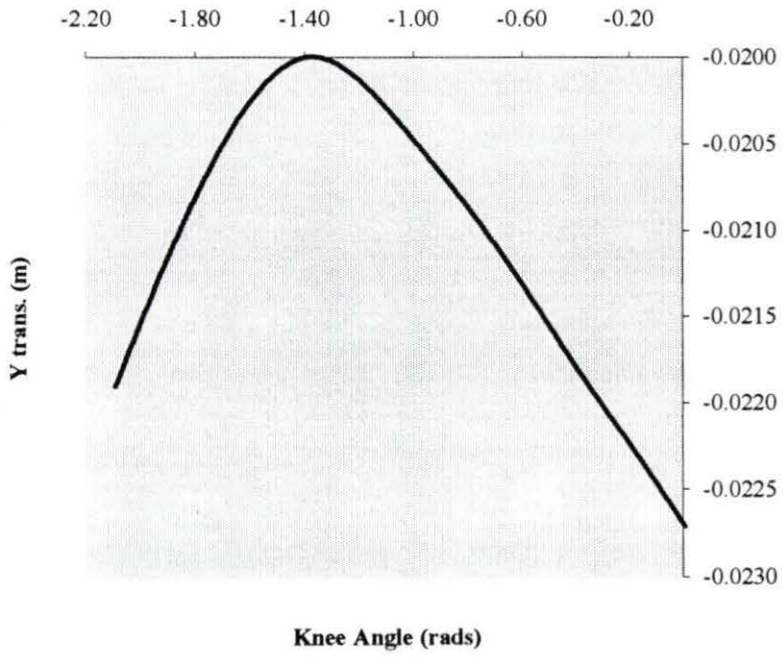
The Y translation curve was created by fitting a cubic spline to control points from Delp [23]. Knee angle ranges from 120 degrees flexion to 0 degrees extension. This is the x component of the ${}^{f}\vec{r}^{t}$ vector. The z component of ${}^{f}\vec{r}^{t}$ is zero.

Figure A.2 - Y translation from femur reference frame to tibia reference frame



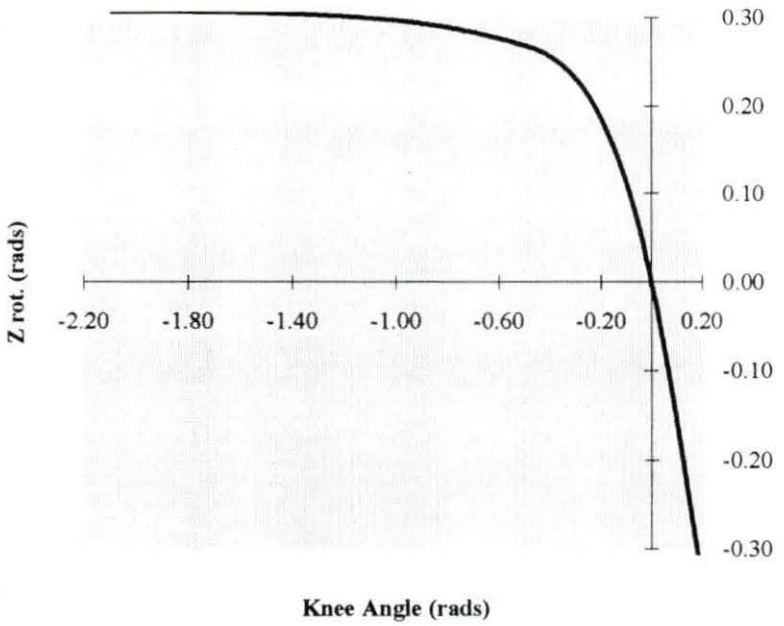
The X translation curve was created by fitting a cubic spline to control points from Delp [23]. Knee angle ranges from 120 degrees flexion to 0 degrees extension. This is the x component of the ${}^{t}\bar{r}^{pa}$ vector. The z component of ${}^{t}\bar{r}^{pa}$ is constant and equal to 0.0024.

Figure A.3 - X translation from tibia reference frame to patella reference frame



The Y translation curve was created by fitting a cubic spline to control points from Delp [23]. Knee angle ranges from 120 degrees flexion to 0 degrees extension. This is the x component of the ${}^{t}\bar{r}^{pa}$ vector. The z component of ${}^{t}\bar{r}^{pa}$ is constant and equal to 0.0024.

Figure A.4 - Y translation from tibia reference frame to patella reference frame



The Z rotation curve was created by fitting a cubic spline to control points from Delp [23]. Knee angle ranges from 120 degrees flexion to 0 degrees extension. This rotation is the generalized coordinate θ_5 in LEMMM.

Figure A.5 - Z axis rotation between tibia reference frame and the patella reference frame

APPENDIX B - TRANSFORMATION MATRICES (SHIFTERS)

Transformation matrices, called "shifters" in this thesis, are utilized to transform vectors expressed in one frame of reference to vectors expressed in another frame of reference. For an example, consider a hip muscle whose origin vector is given in pelvis frame coordinates and whose insertion vector is given in femur coordinates (these frames are originally aligned). To find the length of the vector connecting the origin and insertion, one need only to subtract the vectors provided no rotations and translations between the pelvis and femur frames have occurred. If rotations have occurred, the femur frame is no longer aligned with the pelvis and the insertion vector expressed in femur coordinates must be *shifted* into the pelvis frame before the vector subtraction can be performed.

Using column matrices to denote the three (η) components of a vector consider the following properties of shifter matrices. For two bodies, J and K with reference frames embedded in each body:

$$\begin{bmatrix} \eta_1^J \\ \eta_2^J \\ \eta_3^J \end{bmatrix} = [\boldsymbol{SJK}] \begin{bmatrix} \eta_1^K \\ \eta_2^K \\ \eta_3^K \end{bmatrix}. \qquad (B.1)$$

Where the shifter [***SJK***] is a 3x3 matrix and the name "***SJK***" means the shifter from body K to body J, i.e. the shifter transforms vectors expressed in the body K frame to vectors expressed in the body J frame. The nine scalar elements of [***SJK***] are found by the following identity:

$$\boldsymbol{SJK}_{ij} = \hat{\boldsymbol{n}}_i^J \cdot \hat{\boldsymbol{n}}_j^K \qquad \boldsymbol{i, j=1, 3} \qquad (B.2)$$

where the $\hat{\boldsymbol{n}}$'s are the orthogonal unit vectors in each frame. As an example, the elements of [***SJK***] for a single right hand rotation β of the K frame about the Z axis (common to both frames J and K) are obtained as follows:

$$\boldsymbol{SJK}_{11} = \hat{\boldsymbol{n}}_1^J \cdot \hat{\boldsymbol{n}}_1^K = \hat{\boldsymbol{n}}_1^J \cdot \left(\cos\beta \hat{\boldsymbol{n}}_1^J + \sin\beta \hat{\boldsymbol{n}}_2^J\right) = \cos\beta \tag{B.3}$$

$$\boldsymbol{SJK}_{12} = \hat{\boldsymbol{n}}_1^J \cdot \hat{\boldsymbol{n}}_2^K = \hat{\boldsymbol{n}}_1^J \cdot \left(-\sin\beta \hat{\boldsymbol{n}}_1^J + \cos\beta \hat{\boldsymbol{n}}_2^J\right) = -\sin\beta \tag{B.4}$$

$$\boldsymbol{SJK}_{13} = \hat{\boldsymbol{n}}_1^J \cdot \hat{\boldsymbol{n}}_3^K = \hat{\boldsymbol{n}}_1^J \cdot \left(\hat{\boldsymbol{n}}_3^J\right) = 0 \tag{B.5}$$

$$\boldsymbol{SJK}_{21} = \hat{\boldsymbol{n}}_2^J \cdot \hat{\boldsymbol{n}}_1^K = \hat{\boldsymbol{n}}_2^J \cdot \left(\cos\beta \hat{\boldsymbol{n}}_1^J + \sin\beta \hat{\boldsymbol{n}}_2^J\right) = \sin\beta \tag{B.6}$$

$$\boldsymbol{SJK}_{22} = \hat{\boldsymbol{n}}_2^J \cdot \hat{\boldsymbol{n}}_2^K = \hat{\boldsymbol{n}}_2^J \cdot \left(-\sin\beta \hat{\boldsymbol{n}}_1^J + \cos\beta \hat{\boldsymbol{n}}_2^J\right) = \cos\beta \tag{B.7}$$

$$\boldsymbol{SJK}_{23} = \hat{\boldsymbol{n}}_2^J \cdot \hat{\boldsymbol{n}}_3^K = \hat{\boldsymbol{n}}_2^J \cdot \left(\hat{\boldsymbol{n}}_3^J\right) = 0 \tag{B.8}$$

$$\boldsymbol{SJK}_{31} = \hat{\boldsymbol{n}}_3^J \cdot \hat{\boldsymbol{n}}_1^K = \hat{\boldsymbol{n}}_3^J \cdot \left(\cos\beta \hat{\boldsymbol{n}}_1^J + \sin\beta \hat{\boldsymbol{n}}_2^J\right) = 0 \tag{B.9}$$

$$\boldsymbol{SJK}_{32} = \hat{\boldsymbol{n}}_3^J \cdot \hat{\boldsymbol{n}}_2^K = \hat{\boldsymbol{n}}_3^J \cdot \left(-\sin\beta \hat{\boldsymbol{n}}_1^J + \cos\beta \hat{\boldsymbol{n}}_2^J\right) = 0 \tag{B.10}$$

$$\boldsymbol{SJK}_{33} = \hat{\boldsymbol{n}}_3^J \cdot \hat{\boldsymbol{n}}_3^K = \hat{\boldsymbol{n}}_3^J \cdot \left(\hat{\boldsymbol{n}}_3^J\right) = 1 \tag{B.11}$$

or written in matrix form as:

$$\left[\boldsymbol{SJK}\right] = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{B.12}$$

Shifter matrices have two important properties: 1) they are orthogonal and 2) they obey the chain rule of transformation matrices. The first property is convenient because an orthogonal matrix has an inverse equal to the transpose of the matrix:

$$[\boldsymbol{SJK}]^{-1} = [\boldsymbol{SJK}]^{T} = [\boldsymbol{SKJ}]. \qquad (B.13)$$

Thus, the components of body J can be shifted into body K coordinates by utilizing the [***SKJ***] shifter which is the transpose of the [***SJK***] shifter:

$$\begin{bmatrix} \eta_1^K \\ \eta_2^K \\ \eta_3^K \end{bmatrix} = [\boldsymbol{SKJ}] \begin{bmatrix} \eta_1^J \\ \eta_2^J \\ \eta_3^J \end{bmatrix}. \qquad (B.14)$$

The second property is very useful with multi-body dynamical systems to conveniently express vectors in body local reference frames in terms of the fixed global frame. Consider a four body system J, K, L, M and fixed frame R, knowing the relative rotations between each body and between body J and the fixed frame R, the individual [***SRJ***], [***SJK***], [***SKL***], and [***SLM***] shifters can be formulated. To express a vector in body M in the coordinates of the fixed frame R, one needs to formulate the [***SRM***] shifter. Knowing the individual shifters above, [***SRM***] can be formulated by using pre-multiplication known as the chain rule:

$$[\boldsymbol{SRM}] = [\boldsymbol{SRJ}][\boldsymbol{SJK}][\boldsymbol{SKL}][\boldsymbol{SLM}]. \qquad (B.15)$$

The shifters utilized in the LEMMM are listed below, the chain rule described above was utilized extensively. The tibia/talus, talus/calcaneus, and calcaneus/toes shifters were formulated utilizing Euler parameters which are described in Appendix C.

Shifter from Femur to Pelvis:

$$\begin{bmatrix} \eta_1^P \\ \eta_2^P \\ \eta_3^P \end{bmatrix} = [\boldsymbol{SPF}] \begin{bmatrix} \eta_1^F \\ \eta_2^F \\ \eta_3^F \end{bmatrix} \tag{B.16}$$

$$[\boldsymbol{SPF}] = \begin{bmatrix} \cos\theta_1 \cos\theta_3 - \sin\theta_1 \sin\theta_2 \sin\theta_3 & -\sin\theta_1 \cos\theta_2 & \cos\theta_1 \sin\theta_3 + \sin\theta_1 \sin\theta_2 \cos\theta_3 \\ \sin\theta_1 \cos\theta_3 + \cos\theta_1 \sin\theta_2 \sin\theta_3 & \cos\theta_1 \cos\theta_2 & \sin\theta_1 \sin\theta_3 - \cos\theta_1 \sin\theta_2 \cos\theta_3 \\ -\cos\theta_2 \sin\theta_3 & \sin\theta_2 & \cos\theta_2 \cos\theta_3 \end{bmatrix}$$

(B.17)

Shifter from Tibia to Femur:

$$\begin{bmatrix} \eta_1^F \\ \eta_2^F \\ \eta_3^F \end{bmatrix} = [\boldsymbol{SFT}] \begin{bmatrix} \eta_1^T \\ \eta_2^T \\ \eta_3^T \end{bmatrix} \tag{B.18}$$

$$[\boldsymbol{SFT}] = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.19}$$

Shifter from Patella to Tibia:

$$\begin{bmatrix} \eta_1^T \\ \eta_2^T \\ \eta_3^T \end{bmatrix} = [\boldsymbol{STPa}] \begin{bmatrix} \eta_1^{Pa} \\ \eta_2^{Pa} \\ \eta_3^{Pa} \end{bmatrix} \tag{B.20}$$

$$[\boldsymbol{STPa}] = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.21}$$

APPENDIX C - EULER PARAMETERS AND ANKLE KINEMATICS

For single revolute joints where the joint axis is complex (i.e. does not coincide with one of the three orthogonal axes of the reference system of the body), it is convenient to express joint kinematics in terms of Euler Parameters.

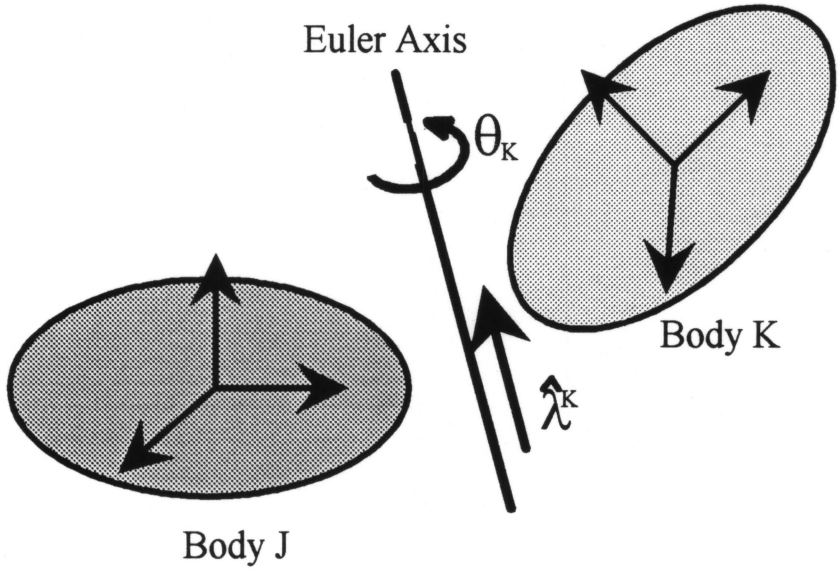


Figure C.1 - General orientation between two bodies using Euler axis and a simple rotation

Simply stated, the *Euler Theorem on Rotations* says that a body K can be brought into a general orientation with respect to body J by utilizing a simple right-hand rotation about a single axis. Let θ_K represent the rotation and specify $\hat{\lambda}^K$ as the unit vector directed along the single axis with the direction determined by the right-hand rule (See Figure C.1).

The four Euler parameters are defined as:

$$\varepsilon_i^K = \lambda_i^K \sin\left(\frac{\theta_K}{2}\right) \qquad i=1,2,3 \qquad (C.1)$$

$$\varepsilon_4^K = \cos\left(\frac{\theta_K}{2}\right) \qquad (C.2)$$

where λ_i^K are the components of the $\hat{\lambda}^K$ unit vector

$$\hat{\lambda}^K = \lambda_1^K \hat{n}_1^J + \lambda_2^K \hat{n}_2^J + \lambda_3^K \hat{n}_3^J \ . \qquad (C.3)$$

The four Euler Parameters are not independent; they are constrained by the following relationship

$$\left(\varepsilon_1^K\right)^2 + \left(\varepsilon_2^K\right)^2 + \left(\varepsilon_3^K\right)^2 + \left(\varepsilon_4^K\right)^2 = 1 \ . \qquad (C.4)$$

The transformation shifter matrices (see Appendix B) can be formulated using Euler parameters instead of dextral angles of orientation. The general shifter to transform vectors from reference frame K back to reference frame J, i.e., [***SJK***], is formulated from Euler parameters [39] and utilized for the ankle, subtalar, and toe joints:

$$[SJK] = \begin{bmatrix} \left(\varepsilon_1^K\right)^2 - \left(\varepsilon_2^K\right)^2 - \left(\varepsilon_3^K\right)^2 + \left(\varepsilon_4^K\right)^2 & 2\left(\varepsilon_1^K\varepsilon_2^K - \varepsilon_3^K\varepsilon_4^K\right) & 2\left(\varepsilon_1^K\varepsilon_3^K + \varepsilon_2^K\varepsilon_4^K\right) \\ 2\left(\varepsilon_1^K\varepsilon_2^K + \varepsilon_3^K\varepsilon_4^K\right) & -\left(\varepsilon_1^K\right)^2 + \left(\varepsilon_2^K\right)^2 - \left(\varepsilon_3^K\right)^2 + \left(\varepsilon_4^K\right)^2 & 2\left(\varepsilon_2^K\varepsilon_3^K - \varepsilon_1^K\varepsilon_4^K\right) \\ 2\left(\varepsilon_1^K\varepsilon_3^K - \varepsilon_2^K\varepsilon_4^K\right) & 2\left(\varepsilon_2^K\varepsilon_3^K + \varepsilon_1^K\varepsilon_4^K\right) & -\left(\varepsilon_1^K\right)^2 - \left(\varepsilon_2^K\right)^2 + \left(\varepsilon_3^K\right)^2 + \left(\varepsilon_4^K\right)^2 \end{bmatrix} . \qquad (C.5)$$

The Euler parameters describing the ankle, subtalar and toe joints are given below. The unit vectors for the Euler axes are given by Inman [42] and Delp [23].

Ankle Joint connecting Tibia and Talus:

Euler Axis: $$\hat{\lambda}^{tal} = -0.105\hat{n}_1^t - 0.174\hat{n}_2^t + 0.979\hat{n}_3^t \qquad (C.6)$$

Euler Parameters: $$\varepsilon_1^{tal} = -0.105 \sin\left(\frac{\theta_6}{2}\right) \qquad (C.7)$$

$$\varepsilon_2^{tal} = -0.174 \sin\left(\frac{\theta_6}{2}\right) \qquad (C.8)$$

$$\varepsilon_3^{tal} = 0.979 \sin\left(\frac{\theta_6}{2}\right) \qquad (C.9)$$

$$\varepsilon_4^{tal} = \cos\left(\frac{\theta_6}{2}\right) \qquad (C.10)$$

Subtalar Joint connecting Talus and Calcaneus:

Euler Axis: $\hat{\lambda}^c = 0.781\hat{n}_1^{tal} + 0.600\hat{n}_2^{tal} - 0.120\hat{n}_3^{tal}$ *(C.11)*

Euler Parameters: $\varepsilon_1^c = 0.781\sin\left(\frac{\theta_7}{2}\right)$ *(C.12)*

$$\varepsilon_2^c = 0.600\sin\left(\frac{\theta_7}{2}\right) \qquad (C.13)$$

$$\varepsilon_3^c = -0.120\sin\left(\frac{\theta_7}{2}\right) \qquad (C.14)$$

$$\varepsilon_4^c = \cos\left(\frac{\theta_7}{2}\right) \qquad (C.15)$$

MTP Joint connecting Calcaneus and Toes:

Note: The direction of the Euler axis was changed to facilitate use with standard film analysis. The original axis was given by Delp [23] as $\hat{\lambda}^{toe} = 0.581\hat{n}_1^c - 0.814\hat{n}_3^c$.

Euler Axis: $\hat{\lambda}^{toe} = -0.581\hat{n}_1^c + 0.814\hat{n}_3^c$ *(C.16)*

Euler Parameters: $\varepsilon_1^{toe} = -0.581\sin\left(\frac{\theta_8}{2}\right)$ *(C.17)*

$$\varepsilon_2^{toe} = 0.0 \qquad (C.18)$$

$$\varepsilon_3^{toe} = 0.814\sin\left(\frac{\theta_8}{2}\right) \qquad (C.19)$$

$$\varepsilon_4^{toe} = \cos\left(\frac{\theta_8}{2}\right) \qquad (C.20)$$

APPENDIX D - CONVERSION OF FILM ANGLES

The Euler axes for ankle, subtalar, and MTP rotations are not conveniently aligned with any of the segment reference frame orthogonal axes. Since it is convenient to record these motions (i.e., film analysis) in standard camera planes (sagittal plane for ankle and MTP motion and the frontal plane for subtalar motion), the LEMMM accepts kinematic angular displacement and velocity in these planes and transforms the data into planes perpendicular to the Euler axis for the joint. Given the angular kinematics in the sagittal and frontal planes in the format specified in Appendix H, θ_6, θ_7, and θ_8 can be calculated.

For the tibia-talus joint, the components of the unit vector aligned with the Euler axis ($\hat{\lambda}^{tal}$) are used to find the dextral rotation ($\beta = 6.122°$) around the Y tibia axis and the rotation ($\gamma = 10.02°$) about the intermediate tibia X' axis (i.e., the original X tibia axis rotated by $\beta = 6.122°$). Using these angles to establish the plane of rotation for θ_6, the camera angle (α_{ank} in the sagittal plane) is transformed into the θ_6 plane:

$$\tan\theta_6 = \frac{\cos 10.02^\circ}{\cos 6.122^\circ}\left(\tan\alpha_{ank}\right) \qquad (D.1)$$

$$\theta_6 = \tan^{-1}\left(0.9904\tan\left(\alpha_{ank}\right)\right). \qquad (D.2)$$

For the talus-calcaneus joint, the components of the unit vector aligned with the Euler axis ($\hat{\lambda}^{c}$) are used to find the dextral rotation ($\beta = 37.53°$) around the Z talus axis and the rotation ($\gamma = 6.947°$) about the intermediate talus Y' axis (i.e. the original Y talus axis rotated by 37.53°). Using these angles to establish the plane of rotation for θ_7, (α_{subt} in the frontal plane) is transformed into the θ_7 plane:

$$\tan\theta_7 = \frac{\cos 6.947^\circ}{\cos 37.53^\circ}\left(\tan\alpha_{subt}\right) \qquad (D.3)$$

$$\theta_7 = \tan^{-1}\left(1.2517\tan(\alpha_{subt})\right). \qquad (D.4)$$

For the calcaneus-toe joint, the components of the unit vector aligned with the Euler axis ($\hat{\lambda}^{toe}$) are used to find the dextral rotation ($\beta = 35.51°$) around the negative Y calcaneus axis. Using this angle to establish the plane of rotation for θ_8, (α_{MTP} in the sagittal plane) is transformed into the θ_8 plane:

$$\tan\theta_8 = \frac{1}{\cos 35.51^\circ}\left(\tan\alpha_{MTP}\right) \qquad (D.5)$$

$$\theta_8 = \tan^{-1}\left(1.2280\tan(\alpha_{MTP})\right). \qquad (D.6)$$

APPENDIX E - ANGULAR VELOCITIES

The relative angular velocities for the LEMMM are calculated from the derivatives of the generalized coordinates which are inputted by the user. The exception is for the ankle, subtalar, and MTP joints where the derivatives of the film angles (see Section 4.1.2) are supplied by the user.

Hip joint connecting pelvis and femur:

For the three degree-of-freedom (DOF) hip joint, the angular velocity of the femur relative to the pelvis is

$$ {}^{p}\vec{\omega}^{f} = \dot{\theta}_1 \hat{n}_3^{p} + \dot{\theta}_2 \hat{n}_1^{f'} + \dot{\theta}_3 \hat{n}_2^{f''} \qquad (E.1) $$

where the intermediate reference frames can be expressed in terms of the pelvis reference frame orthogonal unit vectors:

$$ \hat{n}_1^{f'} = \cos\theta_1 \hat{n}_1^{p} + \sin\theta_1 \hat{n}_2^{p} \qquad (E.2) $$

$$ \hat{n}_2^{f''} = -\sin\theta_1 \cos\theta_2 \hat{n}_1^{p} + \cos\theta_1 \cos\theta_2 \hat{n}_2^{p} + \sin\theta_2 \hat{n}_3^{p} \,. \qquad (E.3) $$

Substituting Eqs. E.2 and E.3 into Eq. E.1 and simplifying, yields the final equation for the angular velocity of the femur relative to the pelvis:

$$ \begin{aligned} {}^{p}\vec{\omega}^{f} = &\left(\dot{\theta}_2 \cos\theta_1 - \dot{\theta}_3 \sin\theta_1 \cos\theta_2\right)\hat{n}_1^{p} + \left(\dot{\theta}_2 \sin\theta_1 + \dot{\theta}_3 \cos\theta_1 \cos\theta_2\right)\hat{n}_2^{p} \\ &+\left(\dot{\theta}_1 + \dot{\theta}_3 \sin\theta_2\right)\hat{n}_3^{p} . \end{aligned} \qquad (E.4) $$

Knee joint connecting femur and tibia:

For this simple hinge knee joint (1 DOF), the angular velocity of the tibia relative to the femur is a simple angular velocity defined below:

$$ {}^{f}\vec{\omega}^{t} = \dot{\theta}_4 \hat{n}_3^{f} = \dot{\theta}_4 \hat{n}_3^{t} \,. \qquad (E.5) $$

Patellofemoral joint connecting femur and patella:

This joint is also a simple hinge joint (1 DOF) and the angular velocity of the patella with respect to the tibia is a simple angular velocity:

$$ {}^{t}\vec{\omega}^{\,pa} = \dot{\theta}_5 \hat{\boldsymbol{n}}_3^{t} = \dot{\theta}_5 \hat{\boldsymbol{n}}_3^{pa} \,. \qquad (E.6) $$

The angular orientation of the patella with respect to the tibia (θ_5) is a function of the knee angle (θ_4) and is described by the kinematic function shown in Figure A.5. Since θ_5 is a function of knee angle (θ_4), the derivative of θ_5 can be found:

$$ \dot{\theta}_5 = \frac{\partial \theta_5}{\partial \theta_4} \frac{d\theta_4}{dt} = \frac{\partial \theta_5}{\partial \theta_4} \dot{\theta}_4 \qquad (E.7) $$

where $\frac{\partial \theta_5}{\partial \theta_4}$ is the slope of the tangent line of the curve in Figure A.5 at the given value of the knee angle (θ_4).

Ankle joint connecting the tibia and the talus:

This single revolute joint is defined using Euler parameters (see Appendix C). Since the Euler axis does not change with time, the relative angular velocity is a simple angular velocity [39] expressed as:

$$ {}^{t}\vec{\omega}^{\,ta} = \dot{\theta}_6 \hat{\lambda}^{ta} \qquad (E.8) $$

where $\hat{\lambda}^{tal} = -0.105\hat{\boldsymbol{n}}_1^{t} - 0.174\hat{\boldsymbol{n}}_2^{t} + 0.979\hat{\boldsymbol{n}}_3^{t}$ (Eq. C.6). The derivative, $\dot{\theta}_6$, is determined by differentiating Eq. 4.1 which yields:

$$ \dot{\theta}_6 = \left[\frac{0.9904 \sec^2(\alpha_{ank})}{1.0 + 0.9904 \tan(\alpha_{ank})} \right] \dot{\alpha}_{ank} \qquad (E.9) $$

where $\dot{\alpha}_{ank}$ is the derivative of the sagittal plane dorsiflexion/plantarflexion film angle inputted by the user.

Subtalar joint connecting the talus and the calcaneus:

This single revolute joint is also described by Euler parameters and has the following simple angular velocity:

$$^{ta}\vec{\omega}^{c} = \dot{\theta}_7 \hat{\lambda}^{c} \qquad (E.10)$$

where $\hat{\lambda}^{c} = 0.781\hat{n}_1^{tal} + 0.600\hat{n}_2^{tal} - 0.120\hat{n}_3^{tal}$ (Eq. C.11). The derivative, $\dot{\theta}_7$, is determined by differentiating Eq. 4.2 which yields:

$$\dot{\theta}_7 = \left[\frac{1.2517\sec^2(\alpha_{subt})}{1.0 + 1.2517\tan(\alpha_{subt})}\right]\dot{\alpha}_{subt} \qquad (E.11)$$

where $\dot{\alpha}_{subt}$ is the derivative of the transverse plane inversion/eversion film angle which is supplied by the user.

Metatarsophalangeal joint connecting the calcaneus and the toes:

This single revolute joint is also described by Euler parameters and has the following simple angular velocity:

$$^{c}\vec{\omega}^{to} = \dot{\theta}_8 \hat{\lambda}^{toe} \qquad (E.12)$$

where $\hat{\lambda}^{toe} = -0.581\hat{n}_1^{c} + 0.814\hat{n}_3^{c}$ (Eq. C.16). The derivative, $\dot{\theta}_8$, is determined by differentiating Eq. 4.2 which yields:

$$\dot{\theta}_8 = \left[\frac{1.2280\sec^2(\alpha_{MTP})}{1.0 + 1.2280\tan(\alpha_{MTP})}\right]\dot{\alpha}_{MTP} \qquad (E.13)$$

where $\dot{\alpha}_{MTP}$ is the derivative of the transverse plane inversion/eversion film angle which is supplied by the user.

APPENDIX F - FORTRAN CODE FOR LEMMM

```
      Program Main

c     There are 43 lower extremity muscles included in this model.
c     The array index for each muscle is defined below.
c
c          Muscle Index #                    Muscle
c     ------------------------          -------------------------
c     Hip Muscles
c               1                         gluteus medius1
c               2                         gluteus medius2
c               3                         gluteus medius3
c               4                         gluteus minimus1
c               5                         gluteus minimus2
c               6                         gluteus minimus3
c               7                         gluteus maximus1
c               8                         gluteus maximus2
c               9                         gluteus maximus3
c               10                        adductor magnus1
c               11                        adductor magnus2
c               12                        adductor magnus3
c               13                        adductor longus
c               14                        adductor brevis
c               15                        pectineus
c               16                        iliacus
c               17                        psoas
c               18                        quadratus femoris
c               19                        gemelli
c               20                        piriformis
c     Hip and Knee Muscles
c               21                        rectus femoris
c               22                        semimembranosus
c               23                        semitendinosus
c               24                        biceps femoris(long head)
c               25                        gracilis
c               26                        sartorius
c               27                        tensor fasciae latae
c     Knee Muscles
c               28                        vastus medialus
c               29                        vastus intermedius
c               30                        vastus lateralis
c               31                        biceps femoris(short head)
c     Knee and Ankle Muscles
c               32                        medial gastrocnemius
c               33                        lateral gastrocnemius
c     Ankle Muscles
c               34                        soleus
```

```
c               35                          tibilias posterior
c               36                          tibilias anterior
c               37                          flexor digitorum longus
c               38                          flexor hallucis longus
c               39                          peroneus brevis
c               40                          peroneus longus
c               41                          peroneus tertius
c               42                          extensor digitorum longus
c               43                          extensor hallucis longus

      Real*8 Lmt(1000,43), Lm(1000,43), Lt(1000,43), LmtDot(1000,43)
      Real*8 Strain(1000,43), Fm(1000,43), Ft(1000,43), Vm(1000,43)
      Real*8 Fdyn(1000,43), MForc(1000,43), theta(1000,8), thdot(1000,8)
      Real*8 MAhip(1000,27,3), MAknee(1000,13), MAank(1000,12,2)
      Real*8 Time(1000), Hipang(1000,3), Kneeang(1000), Ankang(1000,3)
      Real*8 Hipvel(1000,3), Kneevel(1000), Ankvel(1000,3)
      Real*8 Hipmom(1000,3), Kneemom(1000), Ankmom(1000,3), LmE(1000,43)

      Call MuscConv
      Call CreateSplines
      Call Input(npts, time, Hipang, Kneeang, Ankang, Hipvel, Kneevel,
     &           Ankvel, Hipmom, Kneemom, Ankmom)

      Do 50 i=1,npts
         theta(i,1)  = Hipang(i,1)
         theta(i,2)  = Hipang(i,2)
         theta(i,3)  = Hipang(i,3)
         theta(i,4)  = Kneeang(i)
         theta(i,5)  = 0.0D0
         theta(i,6)  = ATAN(0.9904*dTAN(Ankang(i,1)))
         theta(i,7)  = ATAN(1.2517*dTAN(Ankang(i,2)))
         theta(i,8)  = ATAN(1.2280*dTAN(Ankang(i,3)))
         thdot(i,1)  = Hipvel(i,1)
         thdot(i,2)  = Hipvel(i,2)
         thdot(i,3)  = Hipvel(i,3)
         thdot(i,4)  = Kneevel(i)
         thdot(i,5)  = 0.0D0
         thdot(i,6)  = ((0.9904*(1.0/dcos(Ankang(i,1)))**2)/
     &                  (1.0+0.9904*dtan(Ankang(i,1))))*Ankvel(i,1)
         thdot(i,7)  = ((1.2517*(1.0/dcos(Ankang(i,2)))**2)/
     &                  (1.0+1.2517*dtan(Ankang(i,2))))*Ankvel(i,2)
         thdot(i,8)  = ((1.2280*(1.0/dcos(Ankang(i,3)))**2)/
     &                  (1.0+1.2280*dtan(Ankang(i,3))))*Ankvel(i,3)
 50   Continue

      Call MuscModel(npts,theta,thdot,Lmt,Lm,Lt,Strain,LmtDot,MAhip,
     &               MAknee,MAank,Fm,Ft)
      Call LmtoVm(npts,Time,Lm,LmE,Vm)
      Call DynaMusc(npts, Vm, Fm, Fdyn)
      Call IndForcPred(npts,Hipmom,Kneemom,Ankmom,MAhip,MAknee,MAank,
     &                 Fdyn,MForc)
```

```
      Call Output(npts,time,theta,Lmt,LmtDot,Lm,LmE,Lt,Vm,Fm,Ft,Fdyn,
     &           MAhip,MAknee,MAank,MForc)
      Stop
      End

c-----------------------------------------------------------------------

      Block Data
      Real*8 actlen(17), actforc(17), passlen(13), passforc(13)
      Real*8 tendstrain(17), tendforc(17)
      Real*8 Lmo(43), Fmo(43), Lts(43), Alphao(43)
      Real*8 FemTibtx(8), FTtxang(8), FemTibty(6), FTtyang(6)
      Real*8 TPtxang(7), TibPattx(7), TPtyang(8), TibPatty(8)
      Real*8 TPang(6), TibPatrz(6), vel(18), velforc(18)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /MuscData/ actlen, actforc, passlen, passforc
      Common /Muscles/ Lmo, Fmo, Lts, Alphao
      Common /MuscVel/ vel, velforc
      Common /TendData/ tendstrain, tendforc
      Common /Fibers/ SO, FO, FG
      Common /FemTb/ FemTibtx, FTtxang, FemTibty, FTtyang
      Common /TibPt/ TPtxang,TibPattx,TPtyang,TibPatty,TPang,TibPatrz
      Common /rvec/ rpf, rtta, rtac, rcto

c     The active, passive, and tendon curves are dimensionless

      Data tendstrain /-10.0, -0.002, -0.001, 0.000, 0.0013, 0.0028,
     &                 0.0043, 0.0058, 0.0073, 0.0088, 0.0103, 0.0118,
     &                 0.0123, 9.2000, 9.2010, 9.2020, 20.000/
      Data tendforc   /0.00, 0.00, 0.00, 0.000, 0.011, 0.026, 0.043,
     &                 0.065, 0.091, 0.123, 0.161, 0.208, 0.227, 345.0,
     &                 345.0, 345.0, 345.0/
      Data actlen     /-5.000, 0.000, 0.401, 0.402, 0.403, 0.522,
     &                 0.628, 0.718, 0.861, 1.045, 1.217, 1.438, 1.618,
     &                 1.620, 1.621, 2.200, 5.000/
      Data actforc    /0.000, 0.000, 0.000, 0.000, 0.000, 0.226, 0.636,
     &                 0.856, 0.950, 0.993, 0.770, 0.246, 0.000, 0.000,
     &                 0.000, 0.000, 0.000/
      Data passlen    /-5.00, 0.998, 0.999, 1.000, 1.100, 1.200, 1.300,
     &                 1.400, 1.500, 1.600, 1.601, 1.602, 5.000/
      Data passforc   /0.000, 0.000, 0.000, 0.000, 0.035, 0.120, 0.260,
     &                 0.550, 1.170, 2.000, 2.000, 2.000, 2.000/

c     The force-velocity curve is also dimensionless

      Data vel        /-10.0,-0.37879,-0.30303,-0.22727,-0.15152,
     &                 -0.07576,0.00,0.07576,0.15152,0.22727,0.30303,
     &                 0.37879,0.45450,0.50000,1.00,2.00,5.00,10.0/
      Data velforc    /1.80,1.53030,1.51512,1.46970,1.40909,1.27273,
     &                 1.00,0.66667,0.45455,0.32576,0.25000,0.18939,
     &                 0.15152,0.12879,0.00,0.00,0.00,0.00/
```

```
c     The units for the 4 specific muscle parameters are as follows:
c           maximum isometric force (Fmo) - Newtons
c           optimal fiber length (Lmo)    - cm
c           tendon slack length (Lts)     - cm
c           pennation angle (Alphao)      - degrees

      Data Fmo     /550.0, 380.0, 435.0, 180.0, 190.0, 215.0, 380.0,
     &              550.0, 370.0, 345.0, 310.0, 445.0, 420.0, 285.0,
     &              175.0, 430.0, 370.0, 255.0, 110.0, 295.0, 780.0,
     &              1030.0, 330.0, 720.0, 110.0, 105.0, 155.0, 1295.0,
     &              1235.0, 1870.0, 400.0, 1115.0, 490.0, 2830.0,
     &              1270.0, 600.0, 310.0, 320.0, 350.0, 755.0, 90.0,
     &              340.0, 110.0/
      Data Lmo     /5.4, 8.4, 6.5, 6.8, 5.6, 3.8, 14.2, 14.7, 14.4, 8.7,
     &              12.1, 13.1, 13.8, 13.3, 13.3, 10.0, 10.4, 5.4, 2.4,
     &              2.6, 8.4, 8.0, 20.1, 10.9, 35.2, 57.9, 9.5, 8.9, 8.7
     &              ,8.4, 17.3, 4.5, 6.4, 3.0, 3.1, 9.8, 3.4, 4.3, 5.0,
     &              4.9, 7.9, 10.2, 11.1/
      Data Lts     /7.8, 5.3, 5.3, 1.6, 2.6, 5.1, 12.5, 12.7, 14.5,
     &              6.0, 13.0, 26.0, 11.0, 2.0, 0.1, 9.0, 13.0, 2.4,
     &              3.9, 11.5, 34.6, 35.9, 26.2, 34.1, 14.0, 4.0, 42.5,
     &              12.6, 13.6, 15.7, 10.0, 40.8, 38.5, 26.8, 31.0,
     &              22.3, 40.0, 38.0, 16.1, 34.5, 10.0, 34.5, 30.5/
      Data Alphao  /8.0, 0.0, 19.0, 10.0, 0.0, 1.0, 5.0, 0.0, 5.0,
     &              5.0, 3.0, 5.0, 6.0, 0.0, 0.0, 7.0, 8.0, 0.0, 0.0,
     &              10.0, 5.0, 15.0, 5.0, 0.0, 3.0, 0.0, 3.0, 5.0,
     &              3.0, 5.0, 23.0, 17.0, 8.0, 25.0, 12.0, 5.0, 7.0,
     &              10.0, 5.0, 10.0, 13.0, 8.0, 6.0/

c     Femur-Tibia Kinematics -- the following data from Delp (1990) is
c     used to construct cubic splines for x and y translation of the
c     tibia reference frame from the femur reference frame relative to
c     knee angle. Knee angles are in radians and translations are in
c     meters.

      Data FTtxang  /-2.09, -1.74, -1.39, -1.04, -0.69, -0.35,
     &               -0.17, 0.0/
      Data FemTibtx /-0.0032, 0.00179, 0.00411, 0.00410, 0.00212,
     &               -0.0010, -0.0031, -0.00525/
      Data FTtyang  /-2.09, -1.22, -0.52, -0.35, -0.17, 0.0/
      Data FemTibty /-0.4226, -0.4082, -0.3990, -0.3976, -0.3966,
     &               -0.3960/

c     Tibia-Patella Kinematics  -- the following data from Delp (1990) is
c     used to construct cubic splines for x translation, y translation,
c     and z rotation of the patellar reference frame relative to the tib-
c     ial frame. These translations and rotations are a function of knee
c     angle. Knee angles are in radians and translations are in meters.

      Data TPtxang  /-2.09, -1.39, -1.04, -0.69, -0.35, -0.17, 0.0/
```

```
      Data TibPattx /0.0173, 0.0324, 0.0381, 0.0430, 0.0469, 0.0484,
     &               0.0496/
      Data TPtyang  /-2.09, -1.57, -1.39, -1.04, -0.69, -0.35,
     &               -0.17, 0.005/
      Data TibPatty /-0.0219, -0.0202, -0.0200, -0.0204, -0.0211,
     &               -0.0219, -0.0223, -0.0227/
      Data TPang    /-2.09, -2.00, -1.45, -0.52, 0.027, 0.170/
      Data TibPatrz /0.308, 0.308, 0.306, 0.270, -0.036, -0.280/

c     The vectors representing the constant distance from reference frame
c     origin to reference frame origin

      Data rpf  /-0.0707, -0.0661, 0.0835/
      Data rtta /0.0D0, -0.430, 0.0D0/
      Data rtac /-0.04877, -0.04195, 0.00792/
      Data rcto /0.1788, -0.0020, 0.00108/

      End

c-----------------------------------------------------------------------

      Subroutine Input(npts, time, Hipang, Kneeang, Ankang, Hipvel,
     &                 Kneevel, Ankvel, Hipmom, Kneemom, Ankmom)

      Real*8 Time(1000), Hipang(1000,3), Kneeang(1000), Ankang(1000,3)
      Real*8 Hipvel(1000,3), Kneevel(1000), Ankvel(1000,3)
      Real*8 Hipmom(1000,3), Kneemom(1000), Ankmom(1000,3)
      Character Filename*20
c     Print the Title Screen
      Print 145, 'Welcome to LEMMM'
      Print 150, 'LEMMM - Lower Extremity Muscle Mechanics Model'
      Print 155, 'Written by:'
      Print 160, 'Mike Sellberg'
      Print 160, 'Biomedical Engineering'
      Print 160, 'Engineering Mechanics'
      Print 160, 'Iowa State University'
      Print 160, '2068 Black Engineering'
      Print 160, 'Ames, IA 50010'
      Print 160, '(515)294-0083'
      Print 160, 'EMAIL:  sellberg@iastate.edu'
      Print 170
      Print*, 'Enter name of datafile including extension'
      Read(5,100) Filename
      Open (Unit=15, file=Filename, status='unknown')
      Rewind 15
      Read(15,110) npts
      Read(15,120) (Time(i), i=1,npts)
      Read(15,130) ((Hipang(j,k), k=1,3), j=1,npts)
      Read(15,140) (Kneeang(i), i=1,npts)
      Read(15,130) ((Ankang(j,k), k=1,3), j=1,npts)
      Read(15,130) ((Hipvel(j,k), k=1,3), j=1,npts)
```

```
      Read(15,140) (Kneevel(i), i=1,npts)
      Read(15,130) ((Ankvel(j,k), k=1,3), j=1,npts)
      Read(15,130) ((Hipmom(j,k), k=1,3), j=1,npts)
      Read(15,140) (Kneemom(i), i=1,npts)
      Read(15,130) ((Ankmom(j,k), k=1,3), j=1,npts)
      Print*, 'Finished reading input data'
  100 Format(A20)
  110 Format(I4)
  120 Format(F6.4)
  130 Format(3F13.8)
  140 Format(F13.8)
  145 Format(T15,A /)
  150 Format(A /)
  155 Format(T5,A /)
  160 Format(T18,A)
  170 Format(////)
      Return
      End

C-----------------------------------------------------------------------

      Subroutine Output(npts,time,theta,Lmt,LmtDot,Lm,LmE,Lt,Vm,Fm,Ft,
     &                  Fdyn,MAhip,MAknee,MAank,MForc)
      Real*8 time(1000), Lmt(1000,43), LmtDot(1000,43), Lm(1000,43)
      Real*8 LmE(1000,43), Lt(1000,43), Vm(1000,43),theta(1000,8)
      Real*8 Fm(1000,43), Ft(1000,43), Fdyn(1000,43), MForc(1000,43)
      Real*8 MAhip(1000,27,3), MAknee(1000,13), MAank(1000,12,2)
      Open (unit = 10,   file = 'Lmt.out',     status = 'unknown')
      Open (unit = 20,   file = 'Vmt.out',     status = 'unknown')
      Open (unit = 30,   file = 'Lm.out',      status = 'unknown')
      Open (unit = 50,   file = 'Lt.out',      status = 'unknown')
      Open (unit = 60,   file = 'Vm.out',      status = 'unknown')
      Open (unit = 70,   file = 'theta.out',   status = 'unknown')
      Open (unit = 80,   file = 'Fm.out',      status = 'unknown')
      Open (unit = 90,   file = 'Ft.out',      status = 'unknown')
      Open (unit = 100,  file = 'Fdyn.out',    status = 'unknown')
      Open (unit = 110,  file = 'MAhipx.out',  status = 'unknown')
      Open (unit = 120,  file = 'MAhipy.out',  status = 'unknown')
      Open (unit = 130,  file = 'MAhipz.out',  status = 'unknown')
      Open (unit = 140,  file = 'MAknee.out',  status = 'unknown')
      Open (unit = 150,  file = 'MAankz.out',  status = 'unknown')
      Open (unit = 160,  file = 'MAankx.out',  status = 'unknown')
      Open (unit = 170,  file = 'MForc.out',   status = 'unknown')
      Rewind 10
      Rewind 20
      Rewind 30
      Rewind 50
      Rewind 60
      Rewind 70
      Rewind 80
      Rewind 90
```

```
      Rewind 100
      Rewind 110
      Rewind 120
      Rewind 130
      Rewind 140
      Rewind 150
      Rewind 160
      Rewind 170
      Write(10, 200)  (time(i), (Lmt    (i,j),  j=1,43), i=1,npts)
      Write(20, 210)  (time(i), (LmtDot(i,j),  j=1,43), i=1,npts)
      Write(30, 200)  (time(i), (Lm     (i,j),  j=1,43), i=1,npts)
      Write(50, 200)  (time(i), (Lt     (i,j),  j=1,43), i=1,npts)
      Write(60, 200)  (time(i), (Vm     (i,j),  j=1,43), i=1,npts)
      Write(70, 220)  (time(i), (theta  (i,j),  j=1,8 ), i=1,npts)
      Write(80, 260)  (time(i), (Fm     (i,j),  j=1,43), i=1,npts)
      Write(90, 260)  (time(i), (Ft     (i,j),  j=1,43), i=1,npts)
      Write(100,260)  (time(i), (Fdyn   (i,j),  j=1,43), i=1,npts)
      Write(110,230)  (time(i), (MAhip (i,j,1),j=1,27), i=1,npts)
      Write(120,230)  (time(i), (MAhip (i,j,2),j=1,27), i=1,npts)
      Write(130,230)  (time(i), (MAhip (i,j,3),j=1,27), i=1,npts)
      Write(140,240)  (time(i), (MAknee(i,j),  j=1,13), i=1,npts)
      Write(150,250)  (time(i), (MAank (i,j,1),j=1,12), i=1,npts)
      Write(160,250)  (time(i), (MAank (i,j,2),j=1,12), i=1,npts)
      Write(170,260)  (time(i), (MForc (i,j),  j=1,43), i=1,npts)
      Close (10)
      Close (20)
      Close (30)
      Close (50)
      Close (60)
      Close (70)
      Close (80)
      Close (90)
      Close (100)
      Close (110)
      Close (120)
      Close (130)
      Close (140)
      Close (150)
      Close (160)
      Close (170)
  200 Format(44(F8.6,','))
  210 Format(44(F10.6,','))
  220 Format(9(F10.6,','))
  230 Format(28(F10.6,','))
  240 Format(14(F10.6,','))
  250 Format(13(F10.6,','))
  260 Format(44(F12.6,','))
      Return
      End

C-----------------------------------------------------------------------
```

```
      Subroutine LmtoVm(npts, Time, Lm, LmE, Vm)
      Real*8 Time(1000), Lm(1000,43), LmE(1000,43), Vm(1000,43)
      Real*8 A(9), X(1000), Y(1000), REF
      EXTERNAL E02ACF

      M1 = 9
      Do 100 i=1,43
         Do 10 j=1,npts
            X(j) = Time(j)
            Y(j) = Lm(j,i)
10       Continue
         Call E02ACF(X,Y,npts,A,M1,REF)
         Do 20 j=1,npts
            LmE(j,i) =  A(1)+X(j)*(A(2)+X(j)*(A(3)+X(j)*(A(4)+X(j)*
     &                 (A(5)+X(j)*(A(6)+X(j)*(A(7)+X(j)*(A(8)+X(j)*
     &                 (A(9)))))))))
            Vm(j,i)  =  A(2)+X(j)*(2.0*A(3)+X(j)*(3.0*A(4)+X(j)*
     &                 (4.0*A(5)+X(j)*(5.0*A(6)+X(j)*(6.0*A(7)+X(j)*
     &                 (7.0*A(8)+X(j)*(8.0*A(9)))))))))
            Vm(j,i)  = -Vm(j,i)  !Shortening Velocity is positive
20       Continue
100   Continue
300   Format(f6.4,1x,f12.6)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine MuscModel(npts,theta,thdot,Lmt,Lm,Lt,Strain,LmtDot,
     &                     MAhip,MAknee,MAank,Fm,Ft)
      Real*8 theta(1000,8), thdot(1000,8), rft(1000,3), rtpa(1000,3)
      Real*8 Lmt(1000,43), Lm(1000,43), Lt(1000,43), LmtDot(1000,43)
      Real*8 Strain(1000,43), Fm(1000,43), Ft(1000,43), Vm(1000,43)
      Real*8 MAhip(1000,27,3), MAknee(1000,13), MAank(1000,12,2)
      Real*8 STC(1000,3,3), STTo(1000,3,3), SPF(1000,3,3)
      Real*8 SFT(1000,3,3), STPa(1000,3,3), STTa(1000,3,3)
      Real*8 STaC(1000,3,3), SCTo(1000,3,3), SPT(1000,3,3)
      Real*8 SFPa(1000,3,3), SFTa(1000,3,3), SFC(1000,3,3)
      Real*8 SPPa(1000,3,3), Wtto(1000,3), kneez(1000)
      Real*8 Wpf(1000,3), Wft(1000,3), Wtpa(1000,3), Wtta(1000,3)
      Real*8 Wtac(1000,3), Wcto(1000,3), Wpt(1000,3), Wfpa(1000,3)
      Real*8 Wfta(1000,3), Wfc(1000,3), Wtc(1000,3), Wppa(1000,3)
      Real*8 slrft(1000,3), slrtpa(1000,3)
      Call LEMSetup(npts,theta,thdot,rft,rtpa,slrft,slrtpa,SPF,SFT,STPa,
     &              SPPa,STTa,STaC,SCTo,SPT,SFPa,SFTa,SFC,STC,STTo,Wpf,
     &              Wft,Wtpa,Wtta,Wtac,Wcto,Wpt,Wfpa,Wfta,Wfc,Wtc,Wppa,
     &              Wtto)
      Do 50 i=1,npts
         Kneez(i) = theta(i,4)
50    Continue
```

```
      Call LowExtModel(npts,rft,rtpa,slrft,slrtpa,SPF,SFT,STPa,SPPa,
     &                 STTa,STaC,SCTo,SPT,SFPa,SFTa,SFC,STC,STTo,Wpf,
     &                 Wft,Wtpa,Wtta,Wtac,Wcto,Wpt,Wfpa,Wfta,Wfc,Wtc,
     &                 Wppa,Wtto,Lmt,LmtDot,MAhip,MAknee,MAank,kneez)
      Print*, 'Finished with Lower Extremity Model'
      Call StatMusc(npts, Lmt, Lm, Lt, Strain, Fm, Ft)
      Print*, 'Finished with Static Muscle Forces'
      Return
      End

c-----------------------------------------------------------------------

c     To calculate transformation shifters, femur-tibia and tibia-patellar
c     kinematics, and relative angular velocities for input into the Lower
c     Extremity Model.

      Subroutine LEMSetup(npts,theta,thdot,rft,rtpa,slrft,slrtpa,SPF,
     &                    SFT,STPa,SPPa,STTa,STaC,SCTo,SPT,SFPa,SFTa,
     &                    SFC,STC,STTo,Wpf,Wft,Wtpa,Wtta,Wtac,Wcto,Wpt,
     &                    Wfpa,Wfta,Wfc,Wtc,Wppa,Wtto)
      Real*8 theta(1000,8), thdot(1000,8), rft(1000,3), rtpa(1000,3)
      Real*8 STC(1000,3,3), STTo(1000,3,3), SPF(1000,3,3)
      Real*8 SFT(1000,3,3), STPa(1000,3,3), STTa(1000,3,3)
      Real*8 STaC(1000,3,3), SCTo(1000,3,3), SPT(1000,3,3)
      Real*8 SFPa(1000,3,3), SFTa(1000,3,3), SFC(1000,3,3)
      Real*8 SPPa(1000,3,3), Wtto(1000,3)
      Real*8 Wpf(1000,3), Wft(1000,3), Wtpa(1000,3), Wtta(1000,3)
      Real*8 Wtac(1000,3), Wcto(1000,3), Wpt(1000,3), Wfpa(1000,3)
      Real*8 Wfta(1000,3), Wfc(1000,3), Wtc(1000,3), Wppa(1000,3)
      Real*8 Wfts(3), Wtpas1(3), Wtpas2(3), Wttas(3), Wtacs1(3)
      Real*8 Wtacs2(3), Wctos(3)
      Real*8 angle1(3), trans1(3), trans2(3), S1(3,3), S2(3,3), S3(3,3)
      Real*8 S4(3,3), S5(3,3), S6(3,3), S7(3,3), S8(3,3), S9(3,3)
      Real*8 S10(3,3), S11(3,3), S12(3,3), S13(3,3)
      Real*8 W1(3), W2(3), W3(3), W4(3), W5(3), rzslope
      Real*8 slrft(1000,3), slrtpa(1000,3), slope1(3), slope2(3)
      Do 100 i=1,npts
c        Determine the femur-tibia kinematics and the tibia-patella kine-
c        matics from the knee angle.
         Call FemTib(theta(i,4),trans1,slope1)
         Call TibPat(theta(i,4),trans2,slope2,theta(i,5),rzslope)
         Do 10 j=1,3
            rft(i,j)   = trans1(j)
            rtpa(i,j)  = trans2(j)
            slrft(i,j)  = slope1(j)
            slrtpa(i,j) = slope2(j)
 10      Continue
         thdot(i,5) = rzslope*thdot(i,4)
c        Create the shifter transformation matrices for each body refer-
c        ence frame.
         Do 20 j=1,3
```

```
            angle1(j) = theta(i,j)
 20       Continue
          Call Shift1(angle1,S1)
          Call Shift2(theta(i,4),S2)
          Call Shift2(theta(i,5),S3)
          Call TaTShft(theta(i,6),S4)
          Call CTaShft(theta(i,7),S5)
          Call ToCaShft(theta(i,8),S6)
          Call Matmult2(S1,S2,S7)
          Call Matmult2(S2,S3,S8)
          Call Matmult2(S7,S3,S9)
          Call Matmult2(S2,S4,S10)
          Call Matmult2(S10,S5,S11)
          Call Matmult2(S4,S5,S12)
          Call Matmult2(S12,S6,S13)
          Do 30 j=1,3
             Do 25 k=1,3
                SPF (i,j,k) = S1 (j,k)
                SFT (i,j,k) = S2 (j,k)
                STPa(i,j,k) = S3 (j,k)
                STTa(i,j,k) = S4 (j,k)
                STaC(i,j,k) = S5 (j,k)
                SCTo(i,j,k) = S6 (j,k)
                SPT (i,j,k) = S7 (j,k)
                SFPa(i,j,k) = S8 (j,k)
                SPPa(i,j,k) = S9 (j,k)
                SFTa(i,j,k) = S10(j,k)
                SFC (i,j,k) = S11(j,k)
                STC (i,j,k) = S12(j,k)
                STTo(i,j,k) = S13(j,k)
 25          Continue
 30       Continue
c         Calculate the relative angular velocities between reference frames
c         Between femur and pelvis frame, fixed in pelvis frame
          Wpf(i,1) = (thdot(i,2)*dcos(theta(i,1))) -
     &               (thdot(i,3)*dsin(theta(i,1))*dcos(theta(i,2)))
          Wpf(i,2) = (thdot(i,2)*dsin(theta(i,1))) +
     &               (thdot(i,3)*dcos(theta(i,1))*dcos(theta(i,2)))
          Wpf(i,3) = thdot(i,1) + thdot(i,3)*dsin(theta(i,2))
c         Between tibia and femur frame, fixed in femur frame
          Wft(i,1) = 0.0D0
          Wft(i,2) = 0.0D0
          Wft(i,3) = thdot(i,4)
c         Between patella and tibia frame, fixed in tibia frame
          Wtpa(i,1) = 0.0D0
          Wtpa(i,2) = 0.0D0
          Wtpa(i,3) = thdot(i,5)
c         Between talus and tibia frame, fixed in tibia frame
          Wtta(i,1) = -0.105*thdot(i,6)
          Wtta(i,2) = -0.174*thdot(i,6)
          Wtta(i,3) =  0.979*thdot(i,6)
```

```
c        Between calcanus and talus frame, fixed in talus frame
         Wtac(i,1) =  0.787*thdot(i,7)
         Wtac(i,2) =  0.600*thdot(i,7)
         Wtac(i,3) = -0.120*thdot(i,7)
c        Between toe and calcanus frame, fixed in calcanus frame
         Wcto(i,1) = -0.581*thdot(i,8)
         Wcto(i,2) =  0.0D0
         Wcto(i,3) =  0.814*thdot(i,8)
c        Use addition theory of angular velocity to calculate composite
c        angular velocities between frames
         Do 40 j=1,3
            W1(j) = Wft (i,j)
            W2(j) = Wtpa(i,j)
            W3(j) = Wtta(i,j)
            W4(j) = Wtac(i,j)
            W5(j) = Wcto(i,j)
 40      Continue
         Call Matmult(S1, W1,Wfts)
         Call Matmult(S2, W2,Wtpas1)
         Call Matmult(S9, W2,Wtpas2)
         Call Matmult(S2, W3,Wttas)
         Call Matmult(S10,W4,Wtacs1)
         Call Matmult(S4 ,W4,Wtacs2)
         Call Matmult(S12,W5,Wctos)
         Do 50, j=1,3
            Wpt (i,j) = Wpf (i,j) + Wfts(j)
            Wfpa(i,j) = Wft (i,j) + Wtpas1(j)
            Wppa(i,j) = Wpt (i,j) + Wtpas2(j)
            Wfta(i,j) = Wft (i,j) + Wttas(j)
            Wfc (i,j) = Wfta(i,j) + Wtacs1(j)
            Wtc (i,j) = Wtta(i,j) + Wtacs2(j)
            Wtto(i,j) = Wtc (i,j) + Wctos(j)
 50      Continue
 100  Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine LowExtModel(npts,rft,rtpa,slrft,slrtpa,SPF,SFT,STPa,
     &                       SPPa,STTa,STaC,SCTo,SPT,SFPa,SFTa,SFC,STC,
     &                       STTo,Wpf,Wft,Wtpa,Wtta,Wtac,Wcto,Wpt,Wfpa,
     &                       Wfta,Wfc,Wtc,Wppa,Wtto,Lmt,LmtDot,MAhip,
     &                       MAknee,MAank,Kneez)
      Real*8 rft(1000,3), rtpa(1000,3), Kneez(1000)
      Real*8 STC(1000,3,3), STTo(1000,3,3), SPF(1000,3,3)
      Real*8 SFT(1000,3,3), STPa(1000,3,3), STTa(1000,3,3)
      Real*8 STaC(1000,3,3), SCTo(1000,3,3), SPT(1000,3,3)
      Real*8 SFPa(1000,3,3), SFTa(1000,3,3), SFC(1000,3,3)
      Real*8 SPPa(1000,3,3), Wtto(1000,3), sl1(3), sl2(3)
      Real*8 Wpf(1000,3), Wft(1000,3), Wtpa(1000,3), Wtta(1000,3)
```

```
      Real*8 Wtac(1000,3), Wcto(1000,3), Wpt(1000,3), Wfpa(1000,3)
      Real*8 Wfta(1000,3), Wfc(1000,3), Wtc(1000,3), Wppa(1000,3)
      Real*8 Lmt(1000,43), LmtDot(1000,43), MAhip(1000,27,3)
      Real*8 MAknee(1000,13), MAank(1000,12,2), LmtTest(1001,43)
      Real*8 S1(3,3), S2(3,3), S3(3,3), S4(3,3), S5(3,3), S6(3,3)
      Real*8 S7(3,3), S8(3,3), S9(3,3), S10(3,3), S11(3,3), S12(3,3)
      Real*8 S13(3,3), W1(3), W2(3), W3(3), W4(3), W5(3), W6(3), W7(3)
      Real*8 W8(3), W9(3), r1(3), r2(3), slrft(1000,3), slrtpa(1000,3)
      Do 10 l=1,43
         LmtTest(1,l)=0.0D0
 10   Continue
      Do 50 i=1,npts
         Do 30 j=1,3
            Do 20 k=1,3
               S1(j,k)  = SPF (i,j,k)
               S2(j,k)  = SFT (i,j,k)
               S3(j,k)  = STPa(i,j,k)
               S4(j,k)  = STTa(i,j,k)
               S5(j,k)  = STaC(i,j,k)
               S6(j,k)  = SCTo(i,j,k)
               S7(j,k)  = SPT (i,j,k)
               S8(j,k)  = SFPa(i,j,k)
               S9(j,k)  = SPPa(i,j,k)
               S10(j,k) = SFTa(i,j,k)
               S11(j,k) = SFC (i,j,k)
               S12(j,k) = STC (i,j,k)
               S13(j,k) = STTo(i,j,k)
 20         Continue
            W1(j)  = Wpf (i,j)
            W2(j)  = Wft (i,j)
            W3(j)  = Wtpa(i,j)
            W4(j)  = Wppa(i,j)
            W5(j)  = Wpt (i,j)
            W6(j)  = Wfpa(i,j)
            W7(j)  = Wtta(i,j)
            W8(j)  = Wfc (i,j)
            W9(j)  = Wtc (i,j)
            r1(j)  = rft (i,j)
            r2(j)  = rtpa(i,j)
            sl1(j) = slrft(i,j)
            sl2(j) = slrtpa(i,j)
 30      Continue
c        Monoarticular Muscles of the Hip
         Call GMed1  (i,S1,W1,Lmt(i,1), LmtTest(i,1), LmtDot(i,1),
     &                MAhip(i,1,1),MAhip(i,1,2),MAhip(i,1,3))
         Call GMed2  (i,S1,W1,Lmt(i,2), LmtTest(i,2), LmtDot(i,2),
     &                MAhip(i,2,1),MAhip(i,2,2),MAhip(i,2,3))
         Call GMed3  (i,S1,W1,Lmt(i,3), LmtTest(i,3), LmtDot(i,3),
     &                MAhip(i,3,1),MAhip(i,3,2),MAhip(i,3,3))
         Call GMin1  (i,S1,W1,Lmt(i,4), LmtTest(i,4), LmtDot(i,4),
     &                MAhip(i,4,1),MAhip(i,4,2),MAhip(i,4,3))
```

```
      Call GMin2  (i,S1,W1,Lmt(i,5), LmtTest(i,5), LmtDot(i,5),
     &             MAhip(i,5,1),MAhip(i,5,2),MAhip(i,5,3))
      Call GMin3  (i,S1,W1,Lmt(i,6), LmtTest(i,6), LmtDot(i,6),
     &             MAhip(i,6,1),MAhip(i,6,2),MAhip(i,6,3))
      Call GMax1  (i,S1,W1,Lmt(i,7), LmtTest(i,7), LmtDot(i,7),
     &             MAhip(i,7,1),MAhip(i,7,2),MAhip(i,7,3))
      Call GMax2  (i,S1,W1,Lmt(i,8), LmtTest(i,8), LmtDot(i,8),
     &             MAhip(i,8,1),MAhip(i,8,2),MAhip(i,8,3))
      Call GMax3  (i,S1,W1,Lmt(i,9), LmtTest(i,9), LmtDot(i,9),
     &             MAhip(i,9,1),MAhip(i,9,2),MAhip(i,9,3))
      Call AMag1  (i,S1,W1,Lmt(i,10),LmtTest(i,10),LmtDot(i,10),
     &             MAhip(i,10,1),MAhip(i,10,2),MAhip(i,10,3))
      Call AMag2  (i,S1,W1,Lmt(i,11),LmtTest(i,11),LmtDot(i,11),
     &             MAhip(i,11,1),MAhip(i,11,2),MAhip(i,11,3))
      Call AMag3  (i,S1,W1,Lmt(i,12),LmtTest(i,12),LmtDot(i,12),
     &             MAhip(i,12,1),MAhip(i,12,2),MAhip(i,12,3))
      Call AddLong(i,S1,W1,Lmt(i,13),LmtTest(i,13),LmtDot(i,13),
     &             MAhip(i,13,1),MAhip(i,13,2),MAhip(i,13,3))
      Call AddBrev(i,S1,W1,Lmt(i,14),LmtTest(i,14),LmtDot(i,14),
     &             MAhip(i,14,1),MAhip(i,14,2),MAhip(i,14,3))
      Call Pect   (i,S1,W1,Lmt(i,15),LmtTest(i,15),LmtDot(i,15),
     &             MAhip(i,15,1),MAhip(i,15,2),MAhip(i,15,3))
      Call Iliacus(i,S1,W1,Lmt(i,16),LmtTest(i,16),LmtDot(i,16),
     &             MAhip(i,16,1),MAhip(i,16,2),MAhip(i,16,3))
      Call Psoas  (i,S1,W1,Lmt(i,17),LmtTest(i,17),LmtDot(i,17),
     &             MAhip(i,17,1),MAhip(i,17,2),MAhip(i,17,3))
      Call QuadFem(i,S1,W1,Lmt(i,18),LmtTest(i,18),LmtDot(i,18),
     &             MAhip(i,18,1),MAhip(i,18,2),MAhip(i,18,3))
      Call Gem    (i,S1,W1,Lmt(i,19),LmtTest(i,19),LmtDot(i,19),
     &             MAhip(i,19,1),MAhip(i,19,2),MAhip(i,19,3))
      Call Piri   (i,S1,W1,Lmt(i,20),LmtTest(i,20),LmtDot(i,20),
     &             MAhip(i,20,1),MAhip(i,20,2),MAhip(i,20,3))
C     Biarticular Muscles of the Hip and Knee
      Call RF     (i,Kneez(i),r1,r2,sl1,sl2,S1,S7,S9,W2,W1,W3,W4,
     &             Lmt(i,21),LmtTest(i,21),LmtDot(i,21),
     &             MAhip(i,21,1),MAhip(i,21,2),MAhip(i,21,3),
     &             MAknee(i,1))
      Call Semimem(i,r1,sl1,S1,S7,W2,W1,W5,Lmt(i,22),LmtTest(i,22),
     &             LmtDot(i,22),MAhip(i,22,1),MAhip(i,22,2),
     &             MAhip(i,22,3),MAknee(i,2))
      Call Semiten(i,r1,sl1,S1,S7,W2,W1,W5,Lmt(i,23),LmtTest(i,23),
     &             LmtDot(i,23),MAhip(i,23,1),MAhip(i,23,2),
     &             MAhip(i,23,3),MAknee(i,3))
      Call BiFemLH(i,r1,sl1,S1,S7,W2,W1,W5,Lmt(i,24),LmtTest(i,24),
     &             LmtDot(i,24),MAhip(i,24,1),MAhip(i,24,2),
     &             MAhip(i,24,3),MAknee(i,4))
      Call Gra    (i,r1,sl1,S1,S7,W2,W1,W5,Lmt(i,25),LmtTest(i,25),
     &             LmtDot(i,25),MAhip(i,25,1),MAhip(i,25,2),
     &             MAhip(i,25,3),MAknee(i,5))
      Call Sar    (i,r1,sl1,S1,S7,W2,W1,W5,Lmt(i,26),LmtTest(i,26),
     &             LmtDot(i,26),MAhip(i,26,1),MAhip(i,26,2),
```

```
     &                MAhip(i,26,3),MAknee(i,6))
          Call TFL    (i,r1,sl1,S1,S7,W2,W1,W5,Lmt(i,27),LmtTest(i,27),
     &                LmtDot(i,27),MAhip(i,27,1),MAhip(i,27,2),
     &                MAhip(i,27,3),MAknee(i,7))
c         Monoarticular Muscles of the Knee
          Call VasMed (i,Kneez(i),r1,r2,sl1,sl2,S2,S8,W3,W2,W6,Lmt(i,28),
     &                LmtTest(i,28),LmtDot(i,28),MAknee(i,8))
          Call VasInt (i,Kneez(i),r1,r2,sl1,sl2,S2,S8,W3,W2,W6,Lmt(i,29),
     &                LmtTest(i,29),LmtDot(i,29),MAknee(i,9))
          Call VasLat (i,Kneez(i),r1,r2,sl1,sl2,S2,S8,W3,W2,W6,Lmt(i,30),
     &                LmtTest(i,30),LmtDot(i,30),MAknee(i,10))
          Call BiFemSH(i,r1,sl1,S2,W2,Lmt(i,31),LmtTest(i,31),
     &                LmtDot(i,31),MAknee(i,11))
c         Biarticular Muscles of the Knee and Ankle
          Call MedGas (i,Kneez(i),r1,S2,S10,S11,W2,W7,W8,Lmt(i,32),
     &                LmtTest(i,32),LmtDot(i,32),MAknee(i,12),
     &                MAank(i,1,1),MAank(i,1,2))
          Call LatGas (i,Kneez(i),r1,S2,S10,S11,W2,W7,W8,Lmt(i,33),
     &                LmtTest(i,33),LmtDot(i,33),MAknee(i,13),
     &                MAank(i,2,1),MAank(i,2,2))
c         Monoarticular Muscles of the Ankle
          Call Sol    (i,S4,S12,    W7,W9,Lmt(i,34),LmtTest(i,34),
     &                LmtDot(i,34),MAank(i, 3,1), MAank(i, 3,2))
          Call TibPost(i,S4,S12,    W7,W9,Lmt(i,35),LmtTest(i,35),
     &                LmtDot(i,35),MAank(i, 4,1), MAank(i, 4,2))
          Call TibAnt (i,S4,S12,    W7,W9,Lmt(i,36),LmtTest(i,36),
     &                LmtDot(i,36),MAank(i, 5,1), MAank(i, 5,2))
          Call FlexDig(i,S4,S12,S13,W7,W9,Lmt(i,37),LmtTest(i,37),
     &                LmtDot(i,37),MAank(i, 6,1), MAank(i, 6,2))
          Call FlexHal(i,S4,S12,S13,W7,W9,Lmt(i,38),LmtTest(i,38),
     &                LmtDot(i,38),MAank(i, 7,1), MAank(i, 7,2))
          Call PerBrev(i,S4,S12,    W7,W9,Lmt(i,39),LmtTest(i,39),
     &                LmtDot(i,39),MAank(i, 8,1), MAank(i, 8,2))
          Call PerLong(i,S4,S12,    W7,W9,Lmt(i,40),LmtTest(i,40),
     &                LmtDot(i,40),MAank(i, 9,1), MAank(i, 9,2))
          Call PerTert(i,S4,S12,    W7,W9,Lmt(i,41),LmtTest(i,41),
     &                LmtDot(i,41),MAank(i,10,1), MAank(i,10,2))
          Call ExtDig (i,S4,S12,S13,W7,W9,Lmt(i,42),LmtTest(i,42),
     &                LmtDot(i,42),MAank(i,11,1), MAank(i,11,2))
          Call ExtHal (i,S4,S12,S13,W7,W9,Lmt(i,43),LmtTest(i,43),
     &                LmtDot(i,43),MAank(i,12,1), MAank(i,12,2))
          Do 40 j=1,43
             LmtTest(i+1,j) = Lmt(i,j)
 40       Continue
 50    Continue
       Return
       End

c-------------------------------------------------------------------------

c     This subroutine uses a nonlinear optimization technique to solve for
```

```
c     the individual muscle forces.
      Subroutine IndForcPred(npts, Hipmom, Kneemom, Ankmom, MAhip,
     &                       MAknee, MAank, Fdyn, MForc)
      Integer M, N, NCLIN, NCNLN, LDA, LDCJ, LDFJ, LDR, ITER
      Integer ISTATE(49), IWORK(2500), LIWORK, LWORK, IUSER(1), IFAIL
      Real*8  Hipmom(1000,3), Kneemom(1000), Ankmom(1000,3)
      Real*8  MAhip(1000,27,3), MAknee(1000,13), MAank(1000,12,2)
      Real*8  Fdyn(1000,43), phi, MForc(1000,43), CLAMDA(49), R(43,43)
      Real*8  WORK(10000), A(6,43), BL(49), BU(49), C(1), CJAC(1,1)
      Real*8  F(43), FJAC(43,43), IMF(43), Fmax(43), Weight(43)

      External OBJFN, E04UPF, E04UDM, E04URF

c     Set the problem parameters
      M = 43          !Number of subfunctions
      N = 43          !Number of variables
      NCLIN = 6       !Number of linear constraints
      NCNLN = 0       !Number of nonlinear constraints
      LDA = 6
      LDCJ = 1
      LDFJ = 43
      LDR = 43
      LIWORK = 2500
      LWORK = 10000

      Do 100 i=1,npts
         Write(6,200) i
         IFAIL = 0    !Terminate program for a NAG error
c        If dynamic muscle force is zero, utilize IVEC to modify the
c        the subfunction and Jacobian calculation in routine OBJFN
         Do 20 j=1,43
            If (Fdyn(i,j).eq.0.0) then
              Weight(j) = 0.0
              Goto 12
            End If
            Weight(j) = 1.0D0/Fdyn(i,j)
 12         Continue
c        Set up the linear constraint coefficient matrix
           If (j.le.27) then
              A(1,j) = MAhip(i,j,1)
              A(2,j) = MAhip(i,j,2)
              A(3,j) = MAhip(i,j,3)
           Else
              A(1,j) = 0.0D0
              A(2,j) = 0.0D0
              A(3,j) = 0.0D0
           End If
           If (j.ge.21.and.j.le.33) then
              A(4,j) = MAknee(i,j-20)
           Else
              A(4,j) = 0.0D0
```

```
            End If
            If (j.ge.32) then
               A(5,j) = MAank(i,j-31,2)
               A(6,j) = MAank(i,j-31,1)
            Else
               A(5,j) = 0.0D0
               A(6,j) = 0.0D0
            End If
 20      Continue

c        Set the upper and lower bounds for the constraints.
c        The first 43 values are lower and upper bounds for the variables
c        The last 6 values are the bounds for the linear constraints
         Do 30 k=1,43
            BL(k) = 0.0D0      !Muscle forces must be tensile
            BU(k) = Fdyn(i,k) !The maximum muscle force producible is Fdyn
 30      Continue
         BL(44) = Hipmom(i,1) !The constraints must sum to equal the
         BL(45) = Hipmom(i,2) !appropriate joint torques
         BL(46) = Hipmom(i,3)
         BL(47) = Kneemom(i)
         BL(48) = Ankmom(i,2)
         BL(49) = Ankmom(i,1)
         BU(44) = BL(44)       !The upper bound equals the lower bound
         BU(45) = BL(45)       !an equality constraint
         BU(46) = BL(46)
         BU(47) = BL(47)
         BU(48) = BL(48)
         BU(49) = BL(49)

c        Initialize variables to an initial guess
         Do 40 l=1,43
            IMF(l)=0.500*Fdyn(i,l)
 40      Continue

c        Set addtional options for subroutine E04UPF
         Call E04URF('Major Print level = 10')
         Call E04URF('Minor Print level = 10')
         Call E04URF('Derivative level = 3')
         Call E04URF('Verify level = 3')
c         Call E04URF('Linear Feasibility Tolerance = 5.00D-04')
c         Call E04URF('Optimality Tolerance = 3.00D-05')
         Call E04URF('Minor Iteration Limit = 250')
         Call E04URF('Linesearch Tolerance = 3.00D-01')
c        Call the nonlinear optimization routine
         Call E04UPF(M,N,NCLIN,NCNLN,LDA,LDCJ,LDFJ,LDR,A,BL,BU,E04UDM,
     &              OBJFN,ITER,ISTATE,C,CJAC,F,FJAC,CLAMDA,phi,R,
     &              IMF,IWORK,LIWORK,WORK,LWORK,IUSER,Weight,IFAIL)

         Do 50 k=1,43
            MForc(i,k) = IMF(k)
```

```
 50      Continue
 100  Continue
 200  Format('Performing Nonlinear optimization -- Time Pt. ',I4)
      Return
      End

c-----------------------------------------------------------------------

c     Subroutine which evaluates the objective function and Jacobian for
c     the nonlinear optimization routine E04UPF

      Subroutine OBJFN(MODE,M,N,LDFJ,X,F,FJAC,NSTATE,IUSER,WEIGHT)
      Integer MODE, M, N, LDFJ, NSTATE, IVEC(1)
      Real*8 X(N), F(M), FJAC(LDFJ,N), WEIGHT(43)

c     Calculate the ith subfunction F and the Jacobian
         Do 20 i=1,43
            F(i) = WEIGHT(i)*X(i)
            Do 15 j=1,43
               If (i.eq.j) then
                  FJAC(i,j) = WEIGHT(i)
               Else If (i.ne.j) then
                  FJAC(i,j) = 0.0D0
               End If
 15         Continue
 20      Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine GMed1(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin     /-0.0408, 0.0304, 0.1209/   ! x,y,z in pelvis coord.
      Data Insertion  /-0.0218, -0.0117, 0.0555/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine GMed2(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin     /-0.0855, 0.04450, 0.0766/  ! x,y,z in pelvis coord.
      Data Insertion  /-0.0258, -0.0058, 0.0527/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
```

```
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine GMed3(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin     /-0.1223, 0.0105, 0.0648/   ! x,y,z in pelvis coord.
      Data Insertion  /-0.0309, -0.0047, 0.0518/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine GMin1(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin     /-0.0467, -0.0080, 0.1056/  ! x,y,z in pelvis coord.
      Data Insertion  /-0.0072, -0.0104, 0.0560/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine GMin2(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin     /-0.0633, -0.0065, 0.0991/  ! x,y,z in pelvis coord.
      Data Insertion  /-0.0096, -0.0104, 0.0560/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine GMin3(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin    /-0.0834, -0.0063, 0.0856/   ! x,y,z in pelvis coord.
      Data Insertion /-0.0135, -0.0083, 0.0550/   ! x,y,z in femur coord.
```

```
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine GMax1(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3), EffOrig(3), EffIns(3)
c     All dimensions in meters
      Data Origin    /-0.1195, 0.0612, 0.0700/   ! x,y,z in pelvis coord.
      Data EffOrig   /-0.1291, 0.0012, 0.0886/   ! x,y,x in pelvis coord.
      Data EffIns    /-0.0457, -0.0248, 0.0392/  ! x,y,z in femur coord.
      Data Insertion /-0.0277, -0.0566, 0.0470/  ! x,y,z in femur coord.
      Call MArMPH(num, SPF, Wpf, Origin, EffOrig, Insertion, EffIns,
     &            Lmt, LmtTest, LmtDot, ma1, ma2, ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine GMax2(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3), EffOrig(3), EffIns(3)
c     All dimensions in meters
      Data Origin    /-0.1349, 0.0176, 0.0563/   ! x,y,z in pelvis coord.
      Data EffOrig   /-0.1376, -0.0520, 0.0914/  ! x,y,z in pelvis coord.
      Data EffIns    /-0.0426, -0.0530, 0.0293/  ! x,y,z in femur coord.
      Data Insertion /-0.0156, -0.1016, 0.0419/  ! x,y,z in femur coord.
      Call MArMPH(num, SPF, Wpf, Origin, EffOrig, Insertion, EffIns,
     &            Lmt, LmtTest, LmtDot, ma1, ma2, ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine GMax3(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3), EffOrig(3), EffIns(3)
c     All dimensions in meters
      Data Origin    /-0.1556, -0.0314, 0.0058/  ! x,y,z in pelvis coord.
      Data EffOrig   /-0.1529, -0.1052, 0.0403/  ! x,y,x in pelvis coord.
      Data EffIns    /-0.0299, -0.1041, 0.0135/  ! x,y,z in femur coord.
      Data Insertion /-0.0060, -0.1419, 0.0411/  ! x,y,z in femur coord.
      Call MArMPH(num, SPF, Wpf, Origin, EffOrig, Insertion, EffIns,
     &            Lmt, LmtTest, LmtDot, ma1, ma2, ma3)
      Return
      End

c-----------------------------------------------------------------------
```

```
      Subroutine AMag1(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin    /-0.0732, -0.1174, 0.0255/  ! x,y,z in pelvis coord.
      Data Insertion /-0.0045, -0.1211, 0.0339/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine AMag2(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin    /-0.0831, -0.1192, 0.0308/  ! x,y,z in pelvis coord.
      Data Insertion /0.0054, -0.2285, 0.0227/   ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine AMag3(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin     /-0.0771, -0.1181, 0.0276/  ! x,y,z in pelvis coord.
      Data Insertion  /0.0070, -0.3837, -0.0266/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine AddLong(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin    /-0.0316, -0.0836, 0.0169/  ! x,y,z in pelvis coord.
      Data Insertion /0.0050, -0.2111, 0.0234/   ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End
```

```
c-----------------------------------------------------------------------

      Subroutine AddBrev(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin     /-0.0587, -0.0915, 0.0164/  ! x,y,z in pelvis coord.
      Data Insertion  /0.0009, -0.1196, 0.0294/   ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Pect(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin    /-0.0431, -0.0768, 0.0451/  ! x,y,z in pelvis coord.
      Data Insertion /-0.0122, -0.0822, 0.0253/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Iliacus(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3), EffOrig(3), EffIns(3)
c     All dimensions in meters
      Data Origin    /-0.0674, 0.0365, 0.0854/    ! x,y,z in pelvis coord.
      Data EffOrig   /-0.0218, -0.0550, 0.0851/   ! x,y,x in pelvis coord.
      Data EffIns    /0.0017, -0.0543, 0.0057/    ! x,y,z in femur coord.
      Data Insertion /-0.0193, -0.0621, 0.0129/   ! x,y,z in femur coord.
      Call MArMPH(num, SPF, Wpf, Origin, EffOrig, Insertion, EffIns,
     &            Lmt, LmtTest, LmtDot, ma1, ma2, ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Psoas(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3), EffOrig(3), EffIns(3)
c     All dimensions in meters
      Data Origin    /-0.0647, 0.0887, 0.0289/    ! x,y,z in pelvis coord.
      Data EffOrig   /-0.0238, -0.0570, 0.0759/   ! x,y,x in pelvis coord.
      Data EffIns    /0.0016, -0.0507, 0.0038/    ! x,y,z in femur coord.
      Data Insertion /-0.0188, -0.0597, 0.0104/   ! x,y,z in femur coord.
```

```
      Call MArMPH(num, SPF, Wpf, Origin, EffOrig, Insertion, EffIns,
     &            Lmt, LmtTest, LmtDot, ma1, ma2, ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine QuadFem(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin    /-0.1143, -0.1151, 0.0520/  ! x,y,z in pelvis coord.
      Data Insertion /-0.0381, -0.0359, 0.0366/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Gem(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin    /-0.1133, -0.0820, 0.0714/  ! x,y,z in pelvis coord.
      Data Insertion /-0.0142, -0.0033, 0.0443/  ! x,y,z in femur coord.
      Call MArSPH(num,SPF,Wpf,Origin,Insertion,Lmt,LmtTest,LmtDot,
     &            ma1,ma2,ma3)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Piri(num,SPF,Wpf,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), Wpf(3), Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 O(3), I(3), Oe(3), OeIdot(3), dOeImag, IOe(3), HI(3)
      Real*8 OOe(3), Is(3), OeI(3), HOe(3), mavec(3), OeImag
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
c     All dimensions in meters
      Data O  /-0.1396, 0.0003, 0.0235/   ! x,y,z in pelvis coord.
      Data Oe /-0.1193, -0.0276, 0.0657/  ! x,y,x in pelvis coord.
      Data I  /-0.0148, -0.0036, 0.0437/  ! x,y,z in femur coord.
      Call Matmult(SPF, I, Is)
      Do 20 j=1,3
         OOe(j) = Oe(j) - O(j)
         OeI(j) = rpf(j) + Is(j) - Oe(j)
         IOe(j) = - OeI(j)
         HOe(j) = Oe(j) - rpf(j)
         HI(j)  = Is(j)
 20   Continue
```

```
      OeImag = dsqrt(OeI(1)**2+OeI(2)**2+OeI(3)**2)
      Lmt = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2) + OeImag
      Call Cross(HI, IOe, mavec)
      ma1 = mavec(1)/OeImag
      ma2 = mavec(2)/OeImag
      ma3 = mavec(3)/OeImag
      Call Cross(Wpf, Is, OeIdot)
      dOeImag = dsqrt(OeIdot(1)**2+OeIdot(2)**2+OeIdot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0 !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOeImag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOeImag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
         LmtDot = 0.0D0     !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine RF(num,Kneez,rft,rtpa,slrft,slrtpa,SPF,SPT,SPPa,Wft,
     &              Wpf,Wtpa,Wppa,Lmt,LmtTest,LmtDot,mah1,mah2,mah3,
     &              mak)
      Real*8 Lmt, mah1, mah2, mah3, mak, Kneez, LmtTest, LmtDot
      Real*8 SPF(3,3), SPT(3,3), SPPa(3,3), slrft(3), slrtpa(3)
      Real*8 Wft(3), Wpf(3), Wtpa(3), Wppa(3)
      Real*8 rft(3), rfts(3), rtpa(3), rtpas(3)
      Real*8 O(3), I(3), Wrp(3), Is(3), Wrps(3), OI(3), IO(3)
      Real*8 OWrp(3), WrpI(3), IWrp(3), HO(3), KI(3)
      Real*8 OWrpmag, IWrpmag, mahvec(3), makvec(3)
      Real*8 dvec1(3), dvec2(3), dvec3(3), dvec4(3), dvec5(3), dvec6(3)
      Real*8 WrpIdot(3), dWrpImag, OIdot(3), dOImag
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
c     All dimensions in meters
      Data O  /-0.0295, -0.0311, 0.0968/ ! Origin (x,y,z) in pelvis
      Data Wrp /0.0334, -0.4030, 0.0019/ ! Wrap Pt (x,y,z) in femur
      Data I  /0.0121, 0.0437, -0.0010/  ! Insertion (x,y,z) in patella
c     Calculate necessary vectors to determine moment arms and musculo-
c     tendonlengths for the wrapping point and no wrapping point config-
c     urations. For knee flexion angles between 3.00 and 1.46, a wrapping
c     point is defined by Delp (1990).  This wrapping point is to cons-
c     train the muscle path so it "wraps" around the bone instead projects
c     through it which gives a more accurate musculotendon unit length.
      Call Matmult(SPF, rft, rfts)
      Call Matmult(SPT, rtpa, rtpas)
```

```
      Call Matmult(SPPa, I, Is)
      Call Matmult(SPF, Wrp, Wrps)
      If ((Kneez.gt.-3.0000).and.(Kneez.lt.-1.4600)) then
c        The wrapping point condition
         Do 20 j=1,3
            OWrp(j) = rpf(j) + Wrps(j) - O(j)
            WrpI(j) = rfts(j) + rtpas(j) + Is(j) - Wrps(j)
            IWrp(j) = -WrpI(j)
            HO(j) = O(j) - rpf(j)
            KI(j) = rtpas(j) + Is(j)
 20      Continue
         OWrpmag = dsqrt(OWrp(1)**2+OWrp(2)**2+OWrp(3)**2)
         IWrpmag = dsqrt(IWrp(1)**2+IWrp(2)**2+IWrp(3)**2)
         Lmt = OWrpmag + IWrpmag
         Call Cross(HO, OWrp, mahvec)
         mah1 =  mahvec(1)/OWrpmag
         mah2 =  mahvec(2)/OWrpmag
         mah3 = -mahvec(3)/OWrpmag
         Call Cross(KI, IWrp, makvec)
         mak =  makvec(3)/IWrpmag
c        Calculate the time rate of change of WrpI vector
         Call Cross(Wpf, rfts, dvec2)
         Call Cross(Wppa, rtpas, dvec4)
         Call Cross(Wppa, Is, dvec5)
         Call Cross(Wpf, Wrps, dvec6)
         Do 25 j=1,2
            dvec1(j) = slrft(j)*Wft(3)
            dvec3(j) = slrtpa(j)*Wtpa(3)
 25      Continue
         dvec1(3) = 0.0D0  ! The z slope is zero
         dvec3(3) = 0.0D0  ! The z slope is zero
         Do 30 j=1,3
            WrpIdot(j) = dvec1(j) + dvec2(j) + dvec3(j) + dvec4(j) +
     &                   dvec5(j) - dvec6(j)
 30      Continue
         dWrpImag = dsqrt(WrpIdot(1)**2+WrpIdot(2)**2+WrpIdot(3)**2)
         If (num.eq.1) then
            LmtDot = 0.0D0 !Can't determine velocity sign for time step 1
            Goto 100
         End If
         If (Lmt.lt.LmtTest) then
            LmtDot = dWrpImag
         Else If (Lmt.gt.LmtTest) then
            LmtDot = -dWrpImag
         Else If (Lmt.eq.LmtTest) then
            LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
         End If
      Else
c        The no wrapping point condition
         Do 50 k=1,3
            OI(k) = rpf(k) + rfts(k) + rtpas(k) + Is(k) - O(k)
```

```
            IO(k) = -OI(k)
            HO(k) = O(k) - rpf(k)
            KI(k) = rtpas(k) + Is(k)
 50       Continue
          Lmt = dsqrt(OI(1)**2+OI(2)**2+OI(3)**2)
          Call Cross(HO, OI, mahvec)
          mah1 =  mahvec(1)/Lmt
          mah2 =  mahvec(2)/Lmt
          mah3 = -mahvec(3)/Lmt
          Call Cross(KI, IO, makvec)
          mak = makvec(3)/Lmt
c         Calculate the time rate of change of OI vector
          Call Cross(Wpf, rfts, dvec2)
          Call Cross(Wppa, rtpas, dvec4)
          Call Cross(Wppa, Is, dvec5)
          Do 55 k=1,2
             dvec1(k) = slrft(k)*Wft(3)
             dvec3(k) = slrtpa(k)*Wtpa(3)
 55       Continue
          dvec1(3) = 0.0D0
          dvec3(3) = 0.0D0
          Do 60 k=1,3
             OIdot(k)=dvec1(k)+dvec2(k)+dvec3(k)+dvec4(k)+dvec5(k)
 60       Continue
          dOImag = dsqrt(OIdot(1)**2+OIdot(2)**2+OIdot(3)**2)
          If (num.eq.1) then
             LmtDot = 0.0D0
             Goto 100
          End If
          If (Lmt.lt.LmtTest) then
             LmtDot = dOImag
          Else If (Lmt.gt.LmtTest) then
             LmtDot = -dOImag
          Else If (Lmt.eq.LmtTest) then
             LmtDot = 0.0D0
          End If
       End If
 100   Continue
       Return
       End

c-------------------------------------------------------------------------

       Subroutine Semimem(num,rft,slrft,SPF,SPT,Wft,Wpf,Wpt,Lmt,LmtTest,
      &                   LmtDot,mah1,mah2,mah3,mak)
       Real*8 rft(3), SPF(3,3), SPT(3,3), Wft(3), Wpf(3), Wpt(3)
       Real*8 Lmt, LmtTest, LmtDot, mah1, mah2, mah3, mak, slrft(3)
       Real*8 Origin(3), Insertion(3)
c      All dimensions in meters
       Data Origin     /-0.1192, -0.1015, 0.0695/  ! x,y,z in pelvis coord.
       Data Insertion  /-0.0243, -0.0536, -0.0194/ ! x,y,z in tibia coord.
```

```
      Call BArSPHK(num,rft,slrft,SPF,SPT,Wft,Wpf,Wpt,Origin,Insertion,
     &             Lmt,LmtTest,LmtDot,mah1,mah2,mah3,mak)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Semiten(num,rft,slrft,SPF,SPT,Wft,Wpf,Wpt,Lmt,LmtTest,
     &                   LmtDot,mah1,mah2,mah3,mak)
      Real*8 Lmt, LmtTest, LmtDot, mah1, mah2, mah3, mak, slrft(3)
      Real*8 O(3), I(3), Ie1(3), Ie2(3)
      Real*8 SPF(3,3), SPT(3,3), Wft(3), Wpf(3), Wpt(3)
      Real*8 rft(3), rfts(3), mahvec(3), makvec(3)
      Real*8 OIe2(3), Ie2O(3), HO(3), KIe2(3), Ie2s(3)
      Real*8 Ie2Ie1(3), Ie1I(3), OIe2mag, OIe2dot(3), dOIe2mag
      Real*8 dvec1(3), dvec2(3), dvec3(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
c     All dimensions in meters
      Data O   /-0.1237, -0.1043, 0.0603/  ! x,y,z in pelvis coord.
      Data Ie2 /-0.0314, -0.0545, -0.0146/ ! x,y,z in tibia coord.
      Data Ie1 /-0.0113, -0.0746, -0.0245/ ! x,y,z in tibia coord.
      Data I   /0.0027, -0.0956, -0.0193/  ! x,y,z in tibia coord.
c     Transform the rft from femur to pelvis coord. and transform the
c     effective insertion vector from tibia to pelvis coord.
      Call Matmult(SPF, rft, rfts)
      Call Matmult(SPT, Ie2, Ie2s)
c     Find the necessary vectors for calculation of the scalar moment arms
c     and musculotendon length.  See notes for vector details.
      Do 20 j=1,3
         Ie2Ie1(j) = Ie1(j) - Ie2(j)
         Ie1I(j)   = I(j) - Ie1(j)
         KIe2(j)   = Ie2s(j)
         OIe2(j)   = rpf(j) + rfts(j) + Ie2s(j) - O(j)
         Ie2O(j)   = -OIe2(j)
         HO(j)     = O(j) - rpf(j)
 20   Continue
c     Calculate the length of musculotendon unit
      OIe2mag = dsqrt(OIe2(1)**2+OIe2(2)**2+OIe2(3)**2)
      Lmt     = dsqrt(Ie2Ie1(1)**2+Ie2Ie1(2)**2+Ie2Ie1(3)**2)
     &        + dsqrt(Ie1I(1)**2+Ie1I(2)**2+Ie1I(3)**2)
     &        + OIe2mag
c     Calculate the scalar moment arm for the hip and knee associated with
c     this muscle position.  See notes for derivation of this quantity
      Call Cross(HO, OIe2, mahvec)
      mah1 =  mahvec(1)/OIe2mag
      mah2 =  mahvec(2)/OIe2mag
      mah3 = -mahvec(3)/OIe2mag
      Call Cross(KIe2, Ie2O, makvec)
      mak = makvec(3)/OIe2mag
c     Calculate the time rate of change of OIe2 vector
```

```
      Call Cross(Wpf, rfts, dvec2)
      Call Cross(Wpt, Ie2s, dvec3)
      Do 25 j=1,2
         dvec1(j) = slrft(j)*Wft(3)
 25   Continue
      dvec1(3) = 0.0D0  ! The z slope is zero
      Do 30 j=1,3
         OIe2dot(j)=dvec1(j)+dvec2(j)+dvec3(j)
 30   Continue
      dOIe2mag = dsqrt(OIe2dot(1)**2+OIe2dot(2)**2+OIe2dot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOIe2mag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOIe2mag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
         LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine BiFemLH(num,rft,slrft,SPF,SPT,Wft,Wpf,Wpt,Lmt,LmtTest,
     &                   LmtDot,mah1,mah2,mah3,mak)
      Real*8 rft(3), SPF(3,3), SPT(3,3), Wft(3), Wpf(3), Wpt(3)
      Real*8 Lmt, LmtTest, LmtDot, mah1, mah2, mah3, mak, slrft(3)
      Real*8 Origin(3), Insertion(3)
c     All dimensions in meters
      Data Origin     /-0.1244, -0.1001, 0.0666/  ! x,y,z in pelvis coord.
      Data Insertion  /-0.0081, -0.0729, 0.0423/  ! x,y,z in tibia coord.
      Call BArSPHK(num,rft,slrft,SPF,SPT,Wft,Wpf,Wpt,Origin,Insertion,
     &             Lmt,LmtTest,LmtDot,mah1,mah2,mah3,mak)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Gra(num,rft,slrft,SPF,SPT,Wft,Wpf,Wpt,Lmt,LmtTest,
     &               LmtDot,mah1,mah2,mah3,mak)
      Real*8 Lmt, LmtTest, LmtDot, mah1, mah2, mah3, mak
      Real*8 Wft(3), Wpf(3), Wpt(3), dvec2(3)
      Real*8 O(3), I(3), Ie(3), SPF(3,3), SPT(3,3), slrft(3)
      Real*8 rft(3), rfts(3), mahvec(3), makvec(3)
      Real*8 OIe(3), IeO(3), IeI(3), KIe(3), HO(3), dvec1(3)
      Real*8 Ies(3), OIemag, OIedot(3), dOIemag, dvec3(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
```

```
      Common /rvec/ rpf, rtta, rtac, rcto
c     All dimensions in meters
      Data O  /-0.0563, -0.1038, 0.0079/  ! x,y,z in pelvis coord.
      Data Ie /-0.0154, -0.0475, -0.0358/ ! x,y,z in tibia coord.
      Data I  /0.0060, -0.0836, -0.0228/  ! x,y,z in tibia coord.
      Call Matmult(SPF, rft, rfts)
      Call Matmult(SPT, Ie, Ies)
      Do 20 j=1,3
         OIe(j) = rpf(j) + rfts(j) + Ies(j) - O(j)
         IeO(j) = -OIe(j)
         IeI(j) = I(j) - Ie(j)
         KIe(j) = Ies(j)
         HO(j)  = O(j) - rpf(j)
 20   Continue
      OIemag = dsqrt(OIe(1)**2+OIe(2)**2+OIe(3)**2)
      Lmt    = dsqrt(IeI(1)**2+IeI(2)**2+IeI(3)**2) + OIemag
      Call Cross(HO, OIe, mahvec)
      mah1 = mahvec(1)/OIemag
      mah2 = mahvec(2)/OIemag
      mah3 = mahvec(3)/OIemag
      Call Cross(KIe, IeO, makvec)
      mak = makvec(3)/OIemag
c     Calculate the time rate of change of OIe vector
      Call Cross(Wpf, rfts, dvec2)
      Call Cross(Wpt, Ies, dvec3)
      Do 25 j=1,2
         dvec1(j) = slrft(j)*Wft(3)
 25   Continue
      dvec1(3) = 0.0D0  ! The z slope is zero
      Do 30 j=1,3
         OIedot(j)=dvec1(j)+dvec2(j)+dvec3(j)
 30   Continue
      dOIemag = dsqrt(OIedot(1)**2+OIedot(2)**2+OIedot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOIemag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOIemag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
         LmtDot = 0.0D0     !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Sar(num,rft,slrft,SPF,SPT,Wft,Wpf,Wpt,Lmt,LmtTest,
```

```
     &                LmtDot,mah1,mah2,mah3,mak)
      Real*8 Lmt, LmtTest, LmtDot, mah1, mah2, mah3, mak
      Real*8 Wft(3), Wpf(3), Wpt(3), dOeIe2mag
      Real*8 O(3), Oe(3), I(3), Ie1(3), Ie2(3), SPF(3,3), SPT(3,3)
      Real*8 rft(3), rfts(3), mahvec(3), makvec(3), OeIe2dot(3)
      Real*8 OeIe2(3), Ie2Oe(3), HO(3), KIe2(3), Oes(3), Ie2s(3)
      Real*8 OOe(3), Ie2Ie1(3), Ie1I(3), OOemag, OeIe2mag
      Real*8 dvec1(3), dvec2(3), dvec3(3), dvec4(3), slrft(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
c     All dimensions in meters
      Data O   /-0.0153, -0.0013, 0.1242/  ! x,y,z in pelvis coord.
      Data Oe  /-0.0030, -0.3568, -0.0421/ ! x,y,z in femur coord.
      Data Ie2 /-0.0056, -0.0419, -0.0399/ ! x,y,z in tibia coord.
      Data Ie1 /0.0060, -0.0589, -0.0383/  ! x,y,z in tibia coord.
      Data I   /0.0243, -0.0840, -0.0252/  ! x,y,z in tibia coord.
      Call Matmult(SPF, Oe, Oes)
      Call Matmult(SPF, rft, rfts)
      Call Matmult(SPT, Ie2, Ie2s)
      Do 20 j=1,3
         Ie2Ie1(j) = Ie1(j) - Ie2(j)
         Ie1I(j)   = I(j) - Ie1(j)
         KIe2(j)   = Ie2s(j)
         OOe(j)    = rpf(j) + Oes(j) - O(j)
         OeIe2(j)  = rfts(j) + Ie2s(j) - Oes(j)
         Ie2Oe(j)  = -OeIe2(j)
         HO(j)     = O(j) - rpf(j)
 20   Continue
      OeIe2mag = dsqrt(OeIe2(1)**2+OeIe2(2)**2+OeIe2(3)**2)
      OOemag   = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
      Lmt      = dsqrt(Ie2Ie1(1)**2+Ie2Ie1(2)**2+Ie2Ie1(3)**2)
     &         + dsqrt(Ie1I(1)**2+Ie1I(2)**2+Ie1I(3)**2)
     &         + OOemag + OeIe2mag
      Call Cross(HO, OOe, mahvec)
      mah1 = mahvec(1)/OOemag
      mah2 = mahvec(2)/OOemag
      mah3 = mahvec(3)/OOemag
      Call Cross(KIe2, Ie2Oe, makvec)
      mak = makvec(3)/OeIe2mag
c     Calculate the time rate of change of OeIe2 vector
      Call Cross(Wpf, rfts, dvec2)
      Call Cross(Wpt, Ie2s, dvec3)
      Call Cross(Wpf, Oes, dvec4)
      Do 25 j=1,2
         dvec1(j) = slrft(j)*Wft(3)
 25   Continue
      dvec1(3) = 0.0D0  ! The z slope is zero
      Do 30 j=1,3
         OeIe2dot(j)=dvec1(j)+dvec2(j)+dvec3(j)-dvec4(j)
 30   Continue
      dOeIe2mag = dsqrt(OeIe2dot(1)**2+OeIe2dot(2)**2+OeIe2dot(3)**2)
```

```
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOeIe2mag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOeIe2mag !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
         LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

C-----------------------------------------------------------------------

      Subroutine TFL(num,rft,slrft,SPF,SPT,Wft,Wpf,Wpt,Lmt,LmtTest,
     &               LmtDot,mah1,mah2,mah3,mak)
      Real*8 Lmt, LmtTest, LmtDot, mah1, mah2, mah3, mak
      Real*8 Wft(3), Wpf(3), Wpt(3)
      Real*8 O(3), Oe(3), I(3), Ie(3), SPF(3,3), SPT(3,3)
      Real*8 rft(3), rfts(3), mahvec(3), makvec(3), slrft(3)
      Real*8 OOe(3), OeIe(3), IIe(3), IeI(3), KI(3), HO(3)
      Real*8 Oes(3), Ies(3), Is(3), OeIemag, IeIdot(3), dIeImag
      Real*8 dvec1(3), dvec2(3), dvec3(3), dvec4(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
C     All dimensions in meters
      Data O  /-0.0311, 0.0214, 0.1241/  ! x,y,z in pelvis coord.
      Data Oe /0.0294, -0.0995, 0.0597/  ! x,y,z in femur coord.
      Data Ie /0.0054, -0.4049, 0.0357/  ! x,y,z in femur coord.
      Data I  /0.0060, -0.0487, 0.0297/  ! x,y,z in tibia coord.
      Call Matmult(SPF, Oe, Oes)
      Call Matmult(SPF, Ie, Ies)
      Call Matmult(SPF, rft, rfts)
      Call Matmult(SPT, I, Is)
      Do 20 j=1,3
         OOe(j)  = rpf(j) + Oes(j) - O(j)
         OeIe(j) = Ies(j) - Oes(j)
         IeI(j)  = rfts(j) + Is(j) - Ies(j)
         IIe(j)  = -IeI(j)
         KI(j)   = Is(j)
         HO(j)   = rpf(j) - O(j)
 20   Continue
      OeIemag = dsqrt(OeIe(1)**2+OeIe(2)**2+OeIe(3)**2)
      OOemag  = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
      Lmt     = dsqrt(IeI(1)**2+IeI(2)**2+IeI(3)**2) + OOemag + OeIemag
      Call Cross(HO, OOe, mahvec)
      mah1 = mahvec(1)/OOemag
      mah2 = mahvec(2)/OOemag
```

```
      mah3 = mahvec(3)/OOemag
      Call Cross(KI, IIe, makvec)
      mak = makvec(3)/OeIemag
c     Calculate the time rate of change of IeI vector
      Call Cross(Wpf, rfts, dvec2)
      Call Cross(Wpt, Is, dvec3)
      Call Cross(Wpf, Ies, dvec4)
      Do 25 j=1,2
         dvec1(j) = slrft(j)*Wft(3)
 25   Continue
      dvec1(3) = 0.0D0  ! The z slope is zero
      Do 30 j=1,3
         IeIdot(j)=dvec1(j)+dvec2(j)+dvec3(j)-dvec4(j)
 30   Continue
      dIeImag = dsqrt(IeIdot(1)**2+IeIdot(2)**2+IeIdot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dIeImag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dIeImag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
         LmtDot = 0.0D0     !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine VasMed(num,Kneez,rft,rtpa,slrft,slrtpa,SFT,SFPa,Wtpa,
     &                  Wft,Wfpa,Lmt,LmtTest,LmtDot,ma)
      Real*8 Kneez, Lmt, LmtTest, LmtDot, ma
      Real*8 SFT(3,3), SFPa(3,3), Wtpa(3), Wft(3), Wfpa(3)
      Real*8 rft(3), rtpa(3), rtpas(3), slrft(3), slrtpa(3)
      Real*8 O(3), Oe(3), I(3), Wrp1(3), Wrp2(3), Is(3)
      Real*8 OOe(3), OeI(3), IOe(3), KI(3), IOemag, mavec(3)
      Real*8 OeWrp1(3), Wrp1I(3), IWrp1(3), Wrp1Imag
      Real*8 Wrp1Wrp2(3), Wrp2I(3), IWrp2(3), Wrp2Imag
      Real*8 vecdot(3), dvecmag, dvec1(3), dvec2(3), dvec3(3), dvec4(3)
c     All dimensions in meters
      Data O    /0.0140, -0.2099, 0.0188/  ! Origin (x,y,z) in femur
      Data Oe   /0.0356, -0.2769, 0.0009/  ! Eff. Origin (x,y,z) in femur
      Data Wrp1 /0.0370, -0.4048, -0.0125/ ! Wrap Pt 1 (x,y,z) in femur
      Data Wrp2 /0.0274, -0.4255, -0.0131/ ! Wrap Pt 2 (x,y,z) in femur
      Data I    /0.0063, 0.0445, -0.0170/  ! Insertion (x,y,z) in patella
      Call Matmult(SFT, rtpa, rtpas)
      Call Matmult(SFPa, I, Is)
      If ((Kneez.ge.-1.7800).and.(Kneez.le.-1.2100)) then
```

```
c        The 1 wrapping point condition
c        All vectors in the femur frame
         Do 20 j=1,3
            OOe(j)    = Oe(j) - O(j)
            OeWrp1(j) = Wrp1(j) - Oe(j)
            Wrp1I(j)  = rft(j) + rtpas(j) + Is(j) - Wrp1(j)
            IWrp1(j)  = -Wrp1I(j)
            KI(j)     = rtpas(j) + Is(j)
 20      Continue
         Wrp1Imag = dsqrt(Wrp1I(1)**2+Wrp1I(2)**2+Wrp1I(3)**2)
         Lmt      = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
     &            + dsqrt(OeWrp1(1)**2+OeWrp1(2)**2+OeWrp1(3)**2)
     &            + Wrp1Imag
         Call Cross(KI, IWrp1, mavec)
         ma = mavec(3)/Wrp1Imag
      Else If ((Kneez.gt.-3.0000).and.(Kneez.lt.-1.7800)) then
c        The 2 wrapping point condition
c        All vectors in the femur frame
         Do 30 l=1,3
            OOe(l)     = Oe(l) - O(l)
            OeWrp1(l)  = Wrp1(l) - Oe(l)
            Wrp1Wrp2(l) = Wrp2(l) - Wrp1(l)
            Wrp2I(l)   = rft(l) + rtpas(l) + Is(l) - Wrp2(l)
            IWrp2(l)   = -Wrp2I(l)
            KI(l)      = rtpas(l) + Is(l)
 30      Continue
         Wrp2Imag = dsqrt(Wrp2I(1)**2+Wrp2I(2)**2+Wrp2I(3)**2)
         Lmt      = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
     &            + dsqrt(OeWrp1(1)**2+OeWrp1(2)**2+OeWrp1(3)**2)
     &            + dsqrt(Wrp1Wrp2(1)**2+Wrp1Wrp2(2)**2+Wrp1Wrp2(3)**2)
     &            + Wrp2Imag
         Call Cross(KI, IWrp2, mavec)
         ma = mavec(3)/Wrp2Imag
      Else
c        The no wrapping point condition
c        All vectors in the femur frame
         Do 50 k=1,3
            OOe(k) = Oe(k) - O(k)
            OeI(k) = rft(k) + rtpas(k) + Is(k) - Oe(k)
            IOe(k) = -OeI(k)
            KI(k)  = rtpas(k) + Is(k)
 50      Continue
         IOemag = dsqrt(OeI(1)**2+OeI(2)**2+OeI(3)**2)
         Lmt    = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2) + IOemag
         Call Cross(KI, IOe, mavec)
         ma = mavec(3)/IOemag
      End If
c     Calculate the time rate of change of the Wrp1I, Wrp2I,and OeI vector
c     Note the time derivatives of these vectors are the same since the
c     the 2 wrapping points and the effective origin are all in the femur
c     frame.  Call this derivative vecdot.
```

```
      Call Cross(Wfpa, rtpas, dvec3)
      Call Cross(Wfpa, Is, dvec4)
      Do 65 k=1,2
         dvec1(k) = slrft(k)*Wft(3)
         dvec2(k) = slrtpa(k)*Wtpa(3)
 65   Continue
      dvec1(3) = 0.0D0  ! The z slopes are zero
      dvec2(3) = 0.0D0
      Do 70 k=1,3
         vecdot(k)=dvec1(k)+dvec2(k)+dvec3(k)+dvec4(k)
 70   Continue
      dvecmag = dsqrt(vecdot(1)**2+vecdot(2)**2+vecdot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 100
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dvecmag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dvecmag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
        LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
      End If
 100  Continue
 200  Format(3(F10.6,2X))
 210  Format('IWrp1:',3(F10.6,2X))
 220  Format('IWrp2:',3(F10.6,2X))
 230  Format('IOe:  ',3(F10.6,2X))
      Return
      End

C-------------------------------------------------------------------------

      Subroutine VasInt(num,Kneez,rft,rtpa,slrft,slrtpa,SFT,SFPa,Wtpa,
     &                  Wft,Wfpa,Lmt,LmtTest,LmtDot,ma)
      Real*8 Kneez, Lmt, LmtTest, LmtDot, ma
      Real*8 SFT(3,3), SFPa(3,3), Wtpa(3), Wft(3), Wfpa(3)
      Real*8 rft(3), rtpa(3), rtpas(3), slrft(3), slrtpa(3)
      Real*8 O(3), Oe(3), I(3), Wrp(3), Is(3)
      Real*8 OOe(3), OeI(3), IOe(3), KI(3), IOemag, mavec(3)
      Real*8 OeWrp(3), WrpI(3), IWrp(3), WrpImag
      Real*8 vecdot(3), dvecmag, dvec1(3), dvec2(3), dvec3(3), dvec4(3)
C     All dimensions in meters
      Data O   /0.0290, -0.1924, 0.0310/ ! Origin (x,y,z) in femur
      Data Oe  /0.0335, -0.2084, 0.0285/ ! Eff. Origin (x,y,z) in femur
      Data Wrp /0.0343, -0.4030, 0.0055/ ! Wrap Pt (x,y,z) in femur
      Data I   /0.0058, 0.0480, -0.0006/ ! Insertion (x,y,z) in patella
      Call Matmult(SFT, rtpa, rtpas)
      Call Matmult(SFPa, I, Is)
      If ((Kneez.ge.-3.0000).and.(Kneez.le.-1.4200)) then
         Do 20 j=1,3
```

```
            OOe(j)  = Oe(j) - O(j)
            OeWrp(j) = Wrp(j) - Oe(j)
            WrpI(j)  = rft(j) + rtpas(j) + Is(j) - Wrp(j)
            IWrp(j)  = -WrpI(j)
            KI(j)    = rtpas(j) + Is(j)
 20       Continue
          WrpImag  = dsqrt(WrpI(1)**2+WrpI(2)**2+WrpI(3)**2)
          Lmt      = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
     &             + dsqrt(OeWrp(1)**2+OeWrp(2)**2+OeWrp(3)**2)
     &             + WrpImag
          Call Cross(KI, IWrp, mavec)
          ma = mavec(3)/WrpImag
       Else
          Do 50 k=1,3
             OOe(k) = Oe(k) - O(k)
             OeI(k) = rft(k) + rtpas(k) + Is(k) - Oe(k)
             IOe(k) = -OeI(k)
             KI(k)  = rtpas(k) + Is(k)
 50       Continue
          IOemag = dsqrt(OeI(1)**2+OeI(2)**2+OeI(3)**2)
          Lmt    = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2) + IOemag
          Call Cross(KI, IOe, mavec)
          ma = mavec(3)/IOemag
       End If
c      Calculate the time rate of change of the WrpI and OeI vectors.
c      Note the time derivatives of these vectors are the same since the
c      the wrapping point and the effective origin are both in the femur
c      frame.  Call this derivative vecdot.
       Call Cross(Wfpa, rtpas, dvec3)
       Call Cross(Wfpa, Is, dvec4)
       Do 65 k=1,2
          dvec1(k) = slrft(k)*Wft(3)
          dvec2(k) = slrtpa(k)*Wtpa(3)
 65    Continue
       dvec1(3) = 0.0D0  ! The z slopes are zero
       dvec2(3) = 0.0D0
       Do 70 k=1,3
          vecdot(k)=dvec1(k)+dvec2(k)+dvec3(k)+dvec4(k)
 70    Continue
       dvecmag = dsqrt(vecdot(1)**2+vecdot(2)**2+vecdot(3)**2)
       If (num.eq.1) then
          LmtDot = 0.0D0  !Can't determine velocity sign for first time step
          Goto 100
       End If
       If (Lmt.lt.LmtTest) then
          LmtDot = dvecmag   !Positive velocity means muscle is shortening
       Else If (Lmt.gt.LmtTest) then
          LmtDot = -dvecmag  !Negative velocity means muscle is lengthening
       Else If (Lmt.eq.LmtTest) then
         LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
       End If
```

```
  100  Continue
       Return
       End

c-------------------------------------------------------------------------

       Subroutine VasLat(num,Kneez,rft,rtpa,slrft,slrtpa,SFT,SFPa,Wtpa,
      &                  Wft,Wfpa,Lmt,LmtTest,LmtDot,ma)
       Real*8 Kneez, Lmt, LmtTest, LmtDot, ma
       Real*8 SFT(3,3), SFPa(3,3), Wtpa(3), Wft(3), Wfpa(3)
       Real*8 rft(3), rtpa(3), rtpas(3), slrft(3), slrtpa(3)
       Real*8 O(3), Oe(3), I(3), Wrp1(3), Wrp2(3), Is(3)
       Real*8 OOe(3), OeI(3), IOe(3), KI(3), IOemag, mavec(3)
       Real*8 OeWrp1(3), Wrp1I(3), IWrp1(3), Wrp1Imag
       Real*8 Wrp1Wrp2(3), Wrp2I(3), IWrp2(3), Wrp2Imag
       Real*8 vecdot(3), dvecmag, dvec1(3), dvec2(3), dvec3(3), dvec4(3)
c      All dimensions in meters
       Data O    /0.0048, -0.1854, 0.0349/ ! Origin (x,y,z) in femur
       Data Oe   /0.0269, -0.2591, 0.0409/ ! Eff. Origin (x,y,z) in femur
       Data Wrp1 /0.0361, -0.4030, 0.0205/ ! Wrap Pt 1 (x,y,z) in femur
       Data Wrp2 /0.0253, -0.4243, 0.0184/ ! Wrap Pt 2 (x,y,z) in femur
       Data I    /0.0103, 0.0423, 0.0141/  ! Insertion (x,y,z) in patella
       Call Matmult(SFT, rtpa, rtpas)
       Call Matmult(SFPa, I, Is)
       If ((Kneez.ge.-1.9200).and.(Kneez.le.-1.2100)) then
          Do 20 j=1,3
             OOe(j)    = Oe(j) - O(j)
             OeWrp1(j) = Wrp1(j) - Oe(j)
             Wrp1I(j)  = rft(j) + rtpas(j) + Is(j) - Wrp1(j)
             IWrp1(j)  = -Wrp1I(j)
             KI(j)     = rtpas(j) + Is(j)
  20      Continue
          Wrp1Imag = dsqrt(Wrp1I(1)**2+Wrp1I(2)**2+Wrp1I(3)**2)
          Lmt      = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
      &            + dsqrt(OeWrp1(1)**2+OeWrp1(2)**2+OeWrp1(3)**2)
      &            + Wrp1Imag
          Call Cross(KI, IWrp1, mavec)
          ma = mavec(3)/Wrp1Imag
       Else If ((Kneez.gt.-3.0000).and.(Kneez.lt.-1.9200)) then
          Do 30 l=1,3
             OOe(l)     = Oe(l) - O(l)
             OeWrp1(l)  = Wrp1(l) - Oe(l)
             Wrp1Wrp2(l) = Wrp2(l) - Wrp1(l)
             Wrp2I(l)   = rft(l) + rtpas(l) + Is(l) - Wrp2(l)
             IWrp2(l)   = -Wrp2I(l)
             KI(l)      = rtpas(l) + Is(l)
  30      Continue
          Wrp2Imag = dsqrt(Wrp2I(1)**2+Wrp2I(2)**2+Wrp2I(3)**2)
          Lmt      = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
      &            + dsqrt(OeWrp1(1)**2+OeWrp1(2)**2+OeWrp1(3)**2)
      &            + dsqrt(Wrp1Wrp2(1)**2+Wrp1Wrp2(2)**2+Wrp1Wrp2(3)**2)
```

```
     &            + Wrp2Imag
          Call Cross(KI, IWrp2, mavec)
          ma = mavec(3)/Wrp2Imag
        Else
          Do 50 k=1,3
            OOe(k) = Oe(k) - O(k)
            OeI(k) = rft(k) + rtpas(k) + Is(k) - Oe(k)
            IOe(k) = -OeI(k)
            KI(k)  = rtpas(k) + Is(k)
 50       Continue
          IOemag = dsqrt(OeI(1)**2+OeI(2)**2+OeI(3)**2)
          Lmt    = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2) + IOemag
          Call Cross(KI, IOe, mavec)
          ma = mavec(3)/IOemag
        End If
c       Calculate the time rate of change of the Wrp1I, Wrp2I,and OeI vector
c       Note the time derivatives of these vectors are the same since the
c       the 2 wrapping points and the effective origin are all in the femur
c       frame.  Call this derivative vecdot.
        Call Cross(Wfpa, rtpas, dvec3)
        Call Cross(Wfpa, Is, dvec4)
        Do 65 k=1,2
          dvec1(k) = slrft(k)*Wft(3)
          dvec2(k) = slrtpa(k)*Wtpa(3)
 65     Continue
        dvec1(3) = 0.0D0  ! The z slopes are zero
        dvec2(3) = 0.0D0
        Do 70 k=1,3
          vecdot(k)=dvec1(k)+dvec2(k)+dvec3(k)+dvec4(k)
 70     Continue
        dvecmag = dsqrt(vecdot(1)**2+vecdot(2)**2+vecdot(3)**2)
        If (num.eq.1) then
          LmtDot = 0.0D0  !Can't determine velocity sign for first time step
          Goto 100
        End If
        If (Lmt.lt.LmtTest) then
          LmtDot = dvecmag   !Positive velocity means muscle is shortening
        Else If (Lmt.gt.LmtTest) then
          LmtDot = -dvecmag  !Negative velocity means muscle is lengthening
        Else If (Lmt.eq.LmtTest) then
          LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
        End If
 100    Continue
        Return
        End

c-------------------------------------------------------------------------

        Subroutine BiFemSH(num,rft,slrft,SFT,Wft,Lmt,LmtTest,LmtDot,ma)
        Real*8 Lmt, LmtTest, LmtDot, ma, slrft(3)
        Real*8 SFT(3,3), Wft(3), rft(3), O(3), I(3), Is(3)
```

```
      Real*8 IO(3), OI(3), KI(3), mavec(3), dvec1(3), dvec2(3)
      Real*8 OIdot(3), dOImag
c     All dimensions in meters
      Data O /0.0050, -0.2111, 0.0234/  ! Origin (x,y,z) in femur
      Data I /-0.0101, -0.0725, 0.0406/ ! Insertion (x,y,z) in tibia
      Call Matmult(SFT, I, Is)
      Do 20 j=1,3
         OI(j) = rft(j) + Is(j) - O(j)
         IO(j) = -OI(j)
         KI(j) = Is(j)
 20   Continue
      Lmt = dsqrt(OI(1)**2+OI(2)**2+OI(3)**2)
      Call Cross(KI, IO, mavec)
      ma = mavec(3)/Lmt
c     Calculate the time rate of change of the OI vector
      Call Cross(Wft, Is, dvec2)
      Do 30 j=1,3
         dvec1(j) = slrft(j)*Wft(3)
         dvec1(3) = 0.0D0  ! The z coordinate of the tibia is constant
         OIdot(j) = dvec1(j) + dvec2(j)
 30   Continue
      dOImag = dsqrt(OIdot(1)**2+OIdot(2)**2+OIdot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOImag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOImag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
        LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine MedGas(num,Kneez,rft,SFT,SFTa,SFC,Wft,Wtta,Wfc,
     &                  Lmt,LmtTest,LmtDot,mak,maa1,maa2)
      Real*8 Kneez, Lmt, LmtTest, LmtDot, mak, maa1, maa2
      Real*8 SFT(3,3), SFTa(3,3), SFC(3,3), Wft(3), Wtta(3), Wfc(3)
      Real*8 rft(3), O(3), Oe(3), Wrp(3), I(3)
c     All dimensions in meters
      Data O   /-0.0127, -0.3929, -0.0235/ ! Origin (x,y,z) in femur
      Data Oe  /-0.0217, -0.0487, -0.0295/ ! Eff. Origin (x,y,z) in tibia
      Data Wrp /-0.0239, -0.4022, -0.0258/ ! Wrap Pt  (x,y,z) in femur
      Data I   /0.0044, 0.0310, -0.0053/   ! Insertion (x,y,z) in calc.
      Call BiArMPKA(num,Kneez,rft,SFT,SFTa,SFC,Wft,Wtta,Wfc,O,Oe,
     &              Wrp,I,Lmt,LmtTest,LmtDot,mak,maa1,maa2)
```

```
      Return
      End

c-----------------------------------------------------------------------

      Subroutine LatGas(num,Kneez,rft,SFT,SFTa,SFC,Wft,Wtta,Wfc,
     &                  Lmt,LmtTest,LmtDot,mak,maa1,maa2)
      Real*8 Kneez, Lmt, LmtTest, LmtDot, mak, maa1, maa2
      Real*8 SFT(3,3), SFTa(3,3), SFC(3,3), Wft(3), Wtta(3), Wfc(3)
      Real*8 rft(3), O(3), Oe(3), Wrp(3), I(3)
c     All dimensions in meters
      Data O   /-0.0155, -0.3946, 0.0272/ ! Origin (x,y,z) in femur
      Data Oe  /-0.0242, -0.0481, 0.0235/ ! Eff. Origin (x,y,z) in tibia
      Data Wrp /-0.0254, -0.4018, 0.0274/ ! Wrap Pt  (x,y,z) in femur
      Data I   /0.0044, 0.0310, -0.0053/  ! Insertion (x,y,z) in calc.
      Call BiArMPKA(num,Kneez,rft,SFT,SFTa,SFC,Wft,Wtta,Wfc,O,Oe,
     &              Wrp,I,Lmt,LmtTest,LmtDot,mak,maa1,maa2)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Sol(num,STTa,STC,Wtta,Wtc,Lmt,LmtTest,LmtDot,ma1,ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2, OIdot(3), dOImag
      Real*8 STC(3,3), STTa(3,3), rtacs(3), Wtta(3), Wtc(3)
      Real*8 O(3), I(3), Is(3), OI(3), IO(3), AI(3), mavec(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3), dvec1(3), dvec2(3)
      Common /rvec/ rpf, rtta, rtac, rcto
c     All Dimensions in meters
      Data O /-0.0024, -0.1533, 0.0071/ ! Origin (x,y,z) in tibia
      Data I /0.0044, 0.0310, -0.0053/  ! Insertion (x,y,z) in calcaneus
      Call Matmult(STTa, rtac, rtacs)
      Call Matmult(STC, I, Is)
      Do 20 j=1,3
         OI(j) = rtta(j) + rtacs(j) + Is(j) - O(j)
         IO(j) = -OI(j)
         AI(j) = rtacs(j) + Is(j)
 20   Continue
      Lmt = dsqrt(OI(1)**2+OI(2)**2+OI(3)**2)
      Call Cross(AI, IO, mavec)
      ma1 = mavec(3)/Lmt
      ma2 = mavec(1)/Lmt
c     Calculate the time rate of change of the OI vector
      Call Cross(Wtta, rtacs, dvec1)
      Call Cross(Wtc, Is, dvec2)
      Do 30 j=1,3
         OIdot(j) = dvec1(j) + dvec2(j)
 30   Continue
      dOImag = dsqrt(OIdot(1)**2+OIdot(2)**2+OIdot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
```

```
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOImag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOImag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
        LmtDot = 0.0D0     !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine TibPost(num,STTa,STC,Wtta,Wtc,Lmt,LmtTest,LmtDot,
     &                   ma1,ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2
      Real*8 STC(3,3), STTa(3,3), rtacs(3), Wtta(3), Wtc(3)
      Real*8 O(3), Oe(3), Ie(3), I(3), Ies(3), Is(3)
      Real*8 OOe(3), OeIe(3), IeOe(3), IeI(3), AIe(3)
      Real*8 OeIemag, mavec(3), OeIedot(3), dOeIemag, dvec1(3), dvec2(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
      Data O  /-0.0094, -0.1348, 0.0019/  ! Origin (x,y,z) in tibia
      Data Oe /-0.0144, -0.4051, -0.0229/ ! Eff. Origin (x,y,z) in tibia
      Data Ie /0.0417, 0.0334, -0.0286/   ! Eff. Ins. (x,y,z) in calcaneus
      Data I  /0.0772, 0.0159, -0.0281/   ! Insertion (x,y,z) in calcaneus
      Call Matmult(STTa, rtac, rtacs)
      Call Matmult(STC, Ie, Ies)
      Call Matmult(STC, I, Is)
      Do 20 j=1,3
         OOe(j)  = Oe(j) - O(j)
         OeIe(j) = rtta(j) + rtacs(j) + Ies(j) - Oe(j)
         IeOe(j) = -OeIe(j)
         IeI(j)  = Is(j) - Ies(j)
         AIe(j)   = rtacs(j) + Ies(j)
 20   Continue
      OeIemag = dsqrt(OeIe(1)**2+OeIe(2)**2+OeIe(3)**2)
      Lmt = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
     &    + dsqrt(IeI(1)**2+IeI(2)**2+IeI(3)**2)
     &    + OeIemag
      Call Cross(AIe, IeOe, mavec)
      ma1 = mavec(3)/OeIemag
      ma2 = mavec(1)/OeIemag
c     Calculate the time rate of change of the OI vector
      Call Cross(Wtta, rtacs, dvec1)
      Call Cross(Wtc, Ies, dvec2)
      Do 30 j=1,3
         OeIedot(j) = dvec1(j) + dvec2(j)
 30   Continue
```

```
      dOeIemag = dsqrt(OeIedot(1)**2+OeIedot(2)**2+OeIedot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOeIemag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOeIemag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
        LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

C-----------------------------------------------------------------------

      Subroutine TibAnt(num,STTa,STC,Wtta,Wtc,Lmt,LmtTest,LmtDot,
     &                  ma1,ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2
      Real*8 STC(3,3), STTa(3,3), Wtta(3), Wtc(3)
      Real*8 O(3), Oe(3), I(3)
      Data O  /0.0179, -0.1624, 0.0115/  ! Origin (x,y,z) in tibia
      Data Oe /0.0329, -0.3951, -0.0177/ ! Eff. Origin (x,y,z) in tibia
      Data I  /0.1166, 0.0178, -0.0305/  ! Insertion (x,y,z) in calcaneus
      Call MArMPA2(num,STTa,STC,Wtta,Wtc,O,Oe,I,Lmt,LmtTest,LmtDot,
     &             ma1,ma2)
      Return
      End

C-----------------------------------------------------------------------

      Subroutine FlexDig(num,STTa,STC,STTo,Wtta,Wtc,Lmt,LmtTest,
     &                   LmtDot,ma1,ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2, rtacs(3), rctos(3)
      Real*8 STC(3,3), STTa(3,3), STTo(3,3), Wtta(3), Wtc(3)
      Real*8 O(3), Oe(3), H1(3), H2(3), H3(3), Ie2(3), Ie1(3), I(3)
      Real*8 H1s(3), H2s(3), H3s(3), Ie2s(3), Ie1s(3), Is(3)
      Real*8 OOe(3),OeH1(3),H1H2(3),H2H3(3),H3Ie2(3),Ie2Ie1(3),Ie1I(3)
      Real*8 H1Oe(3), AH1(3), OeH1mag, mavec(3), OeH1dot(3), dOeH1mag
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3), dvec1(3), dvec2(3)
      Common /rvec/ rpf, rtta, rtac, rcto
      Data O   /-0.0083, -0.2046, -0.0018/ ! Origin (x,y,z) in tibia
      Data Oe  /-0.0154, -0.4051, -0.0196/ ! Eff. Origin (x,y,z) in tibia
      Data H1  /0.0436, 0.0315, -0.0280/   ! Heel point 1 (x,y,z) in calc.
      Data H2  /0.0708, 0.0176, -0.0263/   ! Heel point 2 (x,y,z) in calc.
      Data H3  /0.1658, -0.0081, 0.0116/   ! Heel point 3 (x,y,z) in calc.
      Data Ie2 /-0.0019, -0.0078, 0.0147/  ! Eff. Ins. 2 (x,y,z) in toes
      Data Ie1 /0.0285, -0.0071, 0.0215/   ! Eff. Ins. 1 (x,y,z) in toes
      Data I   /0.0441, -0.0060, 0.0242/   ! Insertion (x,y,z) in toes
```

```
      Call Matmult(STTa, rtac, rtacs)
      Call Matmult(STC, rcto, rctos)
      Call Matmult(STC, H1, H1s)
      Call Matmult(STC, H2, H2s)
      Call Matmult(STC, H3, H3s)
      Call Matmult(STTo, Ie2, Ie2s)
      Call Matmult(STTo, Ie1, Ie1s)
      Call Matmult(STTo, I, Is)
      Do 20 j=1,3
         OOe(j)     = Oe(j) - O(j)
         OeH1(j)    = rtta(j) + rtacs(j) + H1s(j) - Oe(j)
         H1H2(j)    = H2s(j) - H1s(j)
         H2H3(j)    = H3s(j) - H2s(j)
         H3Ie2(j)   = rctos(j) + Ie2s(j) - H3s(j)
         Ie2Ie1(j)  = Ie1s(j) - Ie2s(j)
         Ie1I(j)    = Is(j) - Ie1s(j)
         H1Oe(j)    = -OeH1(j)
         AH1(j)     = rtacs(j) + H1s(j)
 20   Continue
      OeH1mag = dsqrt(OeH1(1)**2+OeH1(2)**2+OeH1(3)**2)
      Lmt = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
     &     + dsqrt(H1H2(1)**2+H1H2(2)**2+H1H2(3)**2)
     &     + dsqrt(H2H3(1)**2+H2H3(2)**2+H2H3(3)**2)
     &     + dsqrt(H3Ie2(1)**2+H3Ie2(2)**2+H3Ie2(3)**2)
     &     + dsqrt(Ie2Ie1(1)**2+Ie2Ie1(2)**2+Ie2Ie1(3)**2)
     &     + dsqrt(Ie1I(1)**2+Ie1I(2)**2+Ie1I(3)**2)
     &     + OeH1mag
      Call Cross(AH1, H1Oe, mavec)
      ma1 = mavec(3)/OeH1mag
      ma2 = mavec(1)/OeH1mag
c     Calculate the time rate of change of the OeH1 vector
      Call Cross(Wtta, rtacs, dvec1)
      Call Cross(Wtc, H1s, dvec2)
      Do 30 j=1,3
         OeH1dot(j) = dvec1(j) + dvec2(j)
 30   Continue
      dOeH1mag = dsqrt(OeH1dot(1)**2+OeH1dot(2)**2+OeH1dot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOeH1mag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOeH1mag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
        LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End
```

```
C-----------------------------------------------------------------------

      Subroutine FlexHal(num,STTa,STC,STTo,Wtta,Wtc,Lmt,LmtTest,
     &                   LmtDot,ma1,ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2
      Real*8 STC(3,3), STTa(3,3), STTo(3,3), Wtta(3), Wtc(3)
      Real*8 O(3), Oe(3), H1(3), H2(3), H3(3), Ie(3), I(3)
      Data O  /-0.0079, -0.2334, 0.0244/ ! Origin (x,y,z) in tibia
      Data Oe /-0.0186, -0.4079, -0.0174/! Eff. Origin (x,y,z) in tibia
      Data H1 /0.0374, 0.0276, -0.0241/  ! Heel point 1 (x,y,z) in calc.
      Data H2 /0.1038, 0.0068, -0.0256/  ! Heel point 2 (x,y,z) in calc.
      Data H3 /0.1726, -0.0053, -0.0269/ ! Heel point 3 (x,y,z) in calc.
      Data Ie /0.0155, -0.0064, -0.0265/ ! Eff. Insertion (x,y,z) in toes
      Data I  /0.0562, -0.0102, -0.0181/ ! Insertion (x,y,z) in toes
      Call MArMPA1(num,STTa,STC,STTo,Wtta,Wtc,O,Oe,H1,H2,H3,Ie,I,Lmt,
     &             LmtTest,LmtDot,ma1,ma2)
      Return
      End

C-----------------------------------------------------------------------

      Subroutine PerBrev(num,STTa,STC,Wtta,Wtc,Lmt,LmtTest,LmtDot,
     &                   ma1,ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2, mavec(3)
      Real*8 STC(3,3), STTa(3,3), rtacs(3), Wtta(3), Wtc(3)
      Real*8 O(3), Oe1(3), Oe2(3), Ie(3), I(3), Ies(3), Is(3)
      Real*8 OOe1(3), Oe1Oe2(3), Oe2Ie(3), IeOe2(3), IeI(3), AIe(3)
      Real*8 Oe2Iemag, Oe2Iedot(3), dOe2Iemag, dvec1(3), dvec2(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
      Data O   /-0.0070, -0.2646, 0.0325/ ! Origin (x,y,z) in tibia
      Data Oe1 /-0.0198, -0.4184, 0.0283/ ! Eff. Origin 1(x,y,z) in tibia
      Data Oe2 /-0.0144, -0.4295, 0.0289/ ! Eff. Origin 2(x,y,z) in tibia
      Data Ie  /0.0471, 0.0270, 0.0233/   ! Eff. Ins. (x,y,z) in calcaneus
      Data I   /0.0677, 0.0219, 0.0343/   ! Insertion (x,y,z) in calcaneus
      Call Matmult(STTa, rtac, rtacs)
      Call Matmult(STC, Ie, Ies)
      Call Matmult(STC, I, Is)
      Do 20 j=1,3
         OOe1(j)   = Oe1(j) - O(j)
         Oe1Oe2(j) = Oe2(j) - Oe1(j)
         Oe2Ie(j)  = rtta(j) + rtacs(j) + Ies(j) - Oe2(j)
         IeOe2(j)  = -Oe2Ie(j)
         IeI(j)    = Is(j) - Ies(j)
         AIe(j)     = rtacs(j) + Ies(j)
 20   Continue
      Oe2Iemag = dsqrt(Oe2Ie(1)**2+Oe2Ie(2)**2+Oe2Ie(3)**2)
      Lmt = dsqrt(OOe1(1)**2+OOe1(2)**2+OOe1(3)**2)
     &    + dsqrt(Oe1Oe2(1)**2+Oe1Oe2(2)**2+Oe1Oe2(3)**2)
     &    + dsqrt(IeI(1)**2+IeI(2)**2+IeI(3)**2)
```

```
     &     + Oe2Iemag
           Call Cross(AIe, IeOe2, mavec)
           ma1 = mavec(3)/Oe2Iemag
           ma2 = mavec(1)/Oe2Iemag
      c    Calculate the time rate of change of the Oe2Ie vector
           Call Cross(Wtta, rtacs, dvec1)
           Call Cross(Wtc, Ies, dvec2)
           Do 30 j=1,3
              Oe2Iedot(j) = dvec1(j) + dvec2(j)
       30  Continue
           dOe2Iemag = dsqrt(Oe2Iedot(1)**2+Oe2Iedot(2)**2+Oe2Iedot(3)**2)
           If (num.eq.1) then
              LmtDot = 0.0D0  !Can't determine velocity sign for first time step
              Goto 50
           End If
           If (Lmt.lt.LmtTest) then
              LmtDot = dOe2Iemag   !Positive velocity means muscle is shortening
           Else If (Lmt.gt.LmtTest) then
              LmtDot = -dOe2Iemag !Negative velocity means muscle is lengthening
           Else If (Lmt.eq.LmtTest) then
             LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
           End If
       50  Continue
           Return
           End

      c--------------------------------------------------------------------

           Subroutine PerLong(num,STTa,STC,Wtta,Wtc,Lmt,LmtTest,LmtDot,
     &                      ma1, ma2)
           Real*8 Lmt, LmtTest, LmtDot, ma1, ma2, mavec(3)
           Real*8 STC(3,3), STTa(3,3), rtacs(3), Wtta(3), Wtc(3)
           Real*8 O(3), Oe1(3), Oe2(3), Ie3(3), Ie2(3), Ie1(3), I(3)
           Real*8 Ie3s(3), Ie2s(3), Ie1s(3), Is(3), Oe2Ie3dot(3), dvec1(3)
           Real*8 OOe1(3),Oe1Oe2(3),Oe2Ie3(3),Ie3Ie2(3),Ie2Ie1(3),Ie1I(3)
           Real*8 Ie3Oe2(3), AIe3(3), Oe2Ie3mag, dOe2Ie3mag, dvec2(3)
           Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
           Common /rvec/ rpf, rtta, rtac, rcto
           Data O   /0.0005, -0.1568, 0.0362/  ! Origin (x,y,z) in tibia
           Data Oe1 /-0.0207, -0.4205, 0.0286/ ! Eff. Origin 1(x,y,z) in tibia
           Data Oe2 /-0.0162, -0.4319, 0.0289/ ! Eff. Origin 2(x,y,z) in tibia
           Data Ie3 /0.0438, 0.0230, 0.0221/   ! Eff. Ins. 3 (x,y,z) in calc.
           Data Ie2 /0.0681, 0.0106, 0.0284/   ! Eff. Ins. 2 (x,y,z) in calc.
           Data Ie1 /0.0852, 0.0069, 0.0118/   ! Eff. Ins. 1 (x,y,z) in calc.
           Data I   /0.1203, 0.0085, -0.0184/  ! Insertion (x,y,z) in calcaneus
           Call Matmult(STTa, rtac, rtacs)
           Call Matmult(STC, Ie3, Ie3s)
           Call Matmult(STC, Ie2, Ie2s)
           Call Matmult(STC, Ie1, Ie1s)
           Call Matmult(STC, I, Is)
           Do 20 j=1,3
```

```
          OOe1(j)    = Oe1(j) - O(j)
          Oe1Oe2(j) = Oe2(j) - Oe1(j)
          Oe2Ie3(j) = rtta(j) + rtacs(j) + Ie3s(j) - Oe2(j)
          Ie3Ie2(j) = Ie2s(j) - Ie3s(j)
          Ie2Ie1(j) = Ie1s(j) - Ie2s(j)
          Ie1I(j)    = Is(j) - Ie1s(j)
          Ie3Oe2(j) = -Oe2Ie3(j)
          AIe3(j)    = rtacs(j) + Ie3s(j)
 20    Continue
       Oe2Ie3mag = dsqrt(Oe2Ie3(1)**2+Oe2Ie3(2)**2+Oe2Ie3(3)**2)
       Lmt = dsqrt(OOe1(1)**2+OOe1(2)**2+OOe1(3)**2)
      &     + dsqrt(Oe1Oe2(1)**2+Oe1Oe2(2)**2+Oe1Oe2(3)**2)
      &     + dsqrt(Ie3Ie2(1)**2+Ie3Ie2(2)**2+Ie3Ie2(3)**2)
      &     + dsqrt(Ie2Ie1(1)**2+Ie2Ie1(2)**2+Ie2Ie1(3)**2)
      &     + dsqrt(Ie1I(1)**2+Ie1I(2)**2+Ie1I(3)**2)
      &     + Oe2Ie3mag
       Call Cross(AIe3, Ie3Oe2, mavec)
       ma1 = mavec(3)/Oe2Ie3mag
       ma2 = mavec(1)/Oe2Ie3mag
 c     Calculate the time rate of change of the Oe2Ie3 vector
       Call Cross(Wtta, rtacs, dvec1)
       Call Cross(Wtc, Ie3s, dvec2)
       Do 30 j=1,3
          Oe2Ie3dot(j) = dvec1(j) + dvec2(j)
 30    Continue
       dOe2Ie3mag = dsqrt(Oe2Ie3dot(1)**2+Oe2Ie3dot(2)**2+
      &                   Oe2Ie3dot(3)**2)
       If (num.eq.1) then
          LmtDot = 0.0D0
          Goto 50
       End If
       If (Lmt.lt.LmtTest) then
          LmtDot = dOe2Ie3mag
       Else If (Lmt.gt.LmtTest) then
          LmtDot = -dOe2Ie3mag
       Else If (Lmt.eq.LmtTest) then
         LmtDot = 0.0D0
       End If
 50    Continue
       Return
       End

 c-----------------------------------------------------------------------

       Subroutine PerTert(num,STTa,STC,Wtta,Wtc,Lmt,LmtTest,LmtDot,
      &                   ma1,ma2)
       Real*8 Lmt, LmtTest, LmtDot, ma1, ma2
       Real*8 STC(3,3), STTa(3,3), Wtta(3), Wtc(3)
       Real*8 O(3), Oe(3), I(3)
       Data O  /0.0010, -0.2804,  0.0231/ ! Origin (x,y,z) in tibia
       Data Oe /0.0229, -0.4069, 0.0159/  ! Eff. Origin (x,y,z) in tibia
```

```
      Data I  /0.0857, 0.0228, 0.0299/   ! Insertion (x,y,z) in calcaneus
      Call MArMPA2(num,STTa,STC,Wtta,Wtc,O,Oe,I,Lmt,LmtTest,LmtDot,
     &             ma1,ma2)
      Return
      End

C-----------------------------------------------------------------------

      Subroutine ExtDig(num, STTa, STC, STTo, Wtta, Wtc, Lmt, LmtTest,
     &                  LmtDot, ma1, ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2, rtacs(3), rctos(3)
      Real*8 STC(3,3), STTa(3,3), STTo(3,3), Wtta(3), Wtc(3)
      Real*8 O(3), Oe(3), H1(3), H2(3), Ie(3), I(3)
      Real*8 H1s(3), H2s(3), Ies(3), Is(3), dvec1(3), dvec2(3)
      Real*8 OOe(3), OeH1(3), H1H2(3), H2Ie(3), IeI(3)
      Real*8 H1Oe(3), AH1(3), OeH1mag, mavec(3), OeH1dot(3), dOeH1mag
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
      Data O  /0.0032, -0.1381, 0.0276/ ! Origin (x,y,z) in tibia
      Data Oe /0.0289, -0.4007, 0.0072/ ! Eff. Origin (x,y,z) in tibia
      Data H1 /0.0922, 0.0388, -0.0001/ ! Heel point 1 (x,y,z) in calc.
      Data H2 /0.1616, 0.0055, 0.0130/  ! Heel point 2 (x,y,z) in calc.
      Data Ie /0.0003, 0.0047, 0.0153/  ! Eff. Insertion (x,y,z) in toes
      Data I  /0.0443, -0.0004, 0.0250/ ! Insertion (x,y,z) in toes
      Call Matmult(STTa, rtac, rtacs)
      Call Matmult(STC, rcto, rctos)
      Call Matmult(STC, H1, H1s)
      Call Matmult(STC, H2, H2s)
      Call Matmult(STTo, Ie, Ies)
      Call Matmult(STTo, I, Is)
      Do 20 j=1,3
         OOe(j)    = Oe(j) - O(j)
         OeH1(j)   = rtta(j) + rtacs(j) + H1s(j) - Oe(j)
         H1H2(j)   = H2s(j) - H1s(j)
         H2Ie(j)  = rctos(j) + Ies(j) - H2s(j)
         IeI(j)   = Is(j) - Ies(j)
         H1Oe(j)   = -OeH1(j)
         AH1(j)    = rtacs(j) + H1s(j)
20    Continue
      OeH1mag = dsqrt(OeH1(1)**2+OeH1(2)**2+OeH1(3)**2)
      Lmt = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
     &    + dsqrt(H1H2(1)**2+H1H2(2)**2+H1H2(3)**2)
     &    + dsqrt(H2Ie(1)**2+H2Ie(2)**2+H2Ie(3)**2)
     &    + dsqrt(IeI(1)**2+IeI(2)**2+IeI(3)**2)
     &    + OeH1mag
      Call Cross(AH1, H1Oe, mavec)
      ma1 = mavec(3)/OeH1mag
      ma2 = mavec(1)/OeH1mag
c     Calculate the time rate of change of the OeH1 vector
      Call Cross(Wtta, rtacs, dvec1)
      Call Cross(Wtc, H1s, dvec2)
```

```
      Do 30 j=1,3
         OeH1dot(j) = dvec1(j) + dvec2(j)
 30   Continue
      dOeH1mag = dsqrt(OeH1dot(1)**2+OeH1dot(2)**2+OeH1dot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOeH1mag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOeH1mag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
        LmtDot = 0.0D0     !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine ExtHal(num, STTa, STC, STTo, Wtta, Wtc, Lmt, LmtTest,
     &                  LmtDot, ma1, ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2
      Real*8 STC(3,3), STTa(3,3), STTo(3,3), Wtta(3), Wtc(3)
      Real*8 O(3), Oe(3), H1(3), H2(3), H3(3), Ie(3), I(3)
      Data O  /0.0012, -0.1767, 0.0228/  ! Origin (x,y,z) in tibia
      Data Oe /0.0326, -0.3985, -0.0085/ ! Eff. Origin (x,y,z) in tibia
      Data H1 /0.0970, 0.0389, -0.0211/  ! Heel point 1 (x,y,z) in calc.
      Data H2 /0.1293, 0.0309, -0.0257/  ! Heel point 2 (x,y,z) in calc.
      Data H3 /0.1734, 0.0139, -0.0280/  ! Heel point 3 (x,y,z) in calc.
      Data Ie /0.0298, 0.0041, -0.0245/  ! Eff. Insertion (x,y,z) in toes
      Data I  /0.0563, 0.0034, -0.0186/  ! Insertion (x,y,z) in toes
      Call MArMPA1(num,STTa,STC,STTo,Wtta,Wtc,O,Oe,H1,H2,H3,Ie,I,Lmt,
     &             LmtTest,LmtDot,ma1,ma2)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine MArMPA1(num,STTa,STC,STTo,Wtta,Wtc,O,Oe,H1,H2,H3,
     &                   Ie,I,Lmt,LmtTest,LmtDot,ma1,ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2, rtacs(3), rctos(3)
      Real*8 STC(3,3), STTa(3,3), STTo(3,3), Wtta(3), Wtc(3)
      Real*8 O(3), Oe(3), H1(3), H2(3), H3(3), Ie(3), I(3)
      Real*8 H1s(3), H2s(3), H3s(3), Ies(3), Is(3)
      Real*8 OOe(3), OeH1(3), H1H2(3), H2H3(3), H3Ie(3), IeI(3)
      Real*8 H1Oe(3), AH1(3), OeH1mag, mavec(3), OeH1dot(3), dvec1(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3), dOeH1mag, dvec2(3)
      Common /rvec/ rpf, rtta, rtac, rcto
      Call Matmult(STTa, rtac, rtacs)
```

```
      Call Matmult(STC, rcto, rctos)
      Call Matmult(STC, H1, H1s)
      Call Matmult(STC, H2, H2s)
      Call Matmult(STC, H3, H3s)
      Call Matmult(STTo, Ie, Ies)
      Call Matmult(STTo, I, Is)
      Do 20 j=1,3
         OOe(j)     = Oe(j) - O(j)
         OeH1(j)    = rtta(j) + rtacs(j) + H1s(j) - Oe(j)
         H1H2(j)    = H2s(j) - H1s(j)
         H2H3(j)    = H3s(j) - H2s(j)
         H3Ie(j)  = rctos(j) + Ies(j) - H3s(j)
         IeI(j)    = Is(j) - Ies(j)
         H1Oe(j)    = -OeH1(j)
         AH1(j)     = rtacs(j) + H1s(j)
 20   Continue
      OeH1mag = dsqrt(OeH1(1)**2+OeH1(2)**2+OeH1(3)**2)
      Lmt = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
     &     + dsqrt(H1H2(1)**2+H1H2(2)**2+H1H2(3)**2)
     &     + dsqrt(H2H3(1)**2+H2H3(2)**2+H2H3(3)**2)
     &     + dsqrt(H3Ie(1)**2+H3Ie(2)**2+H3Ie(3)**2)
     &     + dsqrt(IeI(1)**2+IeI(2)**2+IeI(3)**2)
     &     + OeH1mag
      Call Cross(AH1, H1Oe, mavec)
      ma1 = mavec(3)/OeH1mag
      ma2 = mavec(1)/OeH1mag
c     Calculate the time rate of change of the OeH1 vector
      Call Cross(Wtta, rtacs, dvec1)
      Call Cross(Wtc, H1s, dvec2)
      Do 30 j=1,3
         OeH1dot(j) = dvec1(j) + dvec2(j)
 30   Continue
      dOeH1mag = dsqrt(OeH1dot(1)**2+OeH1dot(2)**2+OeH1dot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOeH1mag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOeH1mag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
        LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine MArMPA2(num,STTa,STC,Wtta,Wtc,O,Oe,I,Lmt,LmtTest,
```

```
     &                    LmtDot,ma1,ma2)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2
      Real*8 STC(3,3), STTa(3,3), rtacs(3), Wtta(3), Wtc(3)
      Real*8 O(3), Oe(3), I(3), Is(3), OeIdot(3), dOeImag
      Real*8 OOe(3), OeI(3), IOe(3), AI(3)
      Real*8 OeImag, mavec(3), dvec1(3), dvec2(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
      Call Matmult(STTa, rtac, rtacs)
      Call Matmult(STC, I, Is)
      Do 20 j=1,3
         OOe(j)  = Oe(j) - O(j)
         OeI(j) = rtta(j) + rtacs(j) + Is(j) - Oe(j)
         IOe(j) = -OeI(j)
         AI(j)   = rtacs(j) + Is(j)
 20   Continue
      OeImag = dsqrt(OeI(1)**2+OeI(2)**2+OeI(3)**2)
      Lmt = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2) + OeImag
      Call Cross(AI, IOe, mavec)
      ma1 = mavec(3)/OeImag
      ma2 = mavec(1)/OeImag
c     Calculate the time rate of change of the OI vector
      Call Cross(Wtta, rtacs, dvec1)
      Call Cross(Wtc, Is, dvec2)
      Do 30 j=1,3
         OeIdot(j) = dvec1(j) + dvec2(j)
 30   Continue
      dOeImag = dsqrt(OeIdot(1)**2+OeIdot(2)**2+OeIdot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOeImag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOeImag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
        LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine MArSPH(num, SPF,W,O,I,Lmt,LmtTest,LmtDot,ma1,ma2,ma3)
      Real*8 SPF(3,3), W(3), Lmt, LmtDot, LmtTest, ma1, ma2, ma3
      Real*8 O(3), I(3), dOImag, OIdot(3)
      Real*8 Is(3), OI(3), HO(3), HI(3), IO(3), mavec(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
```

```
      Call Matmult(SPF, I, Is)
      Do 20 j=1,3
         OI(j) = rpf(j) + Is(j) - O(j)
         IO(j) = -OI(j)
         HO(j) = O(j) - rpf(j)
         HI(j) = Is(j)
 20   Continue
      Lmt = dsqrt(OI(1)**2+OI(2)**2+OI(3)**2)
      Call Cross(HI, IO, mavec)
      ma1 = mavec(1)/Lmt
      ma2 = mavec(2)/Lmt
      ma3 = mavec(3)/Lmt
      Call Cross(W, Is, OIdot)
      dOImag = dsqrt(OIdot(1)**2+OIdot(2)**2+OIdot(3)**2)
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 50
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOImag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOImag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
         LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
      End If
 50   Continue
      Return
      End

c-----------------------------------------------------------------------

      Subroutine MArMPH(num,SPF,W,O,Oe,I,Ie,Lmt,LmtTest,LmtDot,
     &                  ma1,ma2,ma3)
      Real*8 Lmt, LmtTest, LmtDot, ma1, ma2, ma3
      Real*8 O(3), I(3), Oe(3), Ie(3), W(3), mavec(3)
      Real*8 OOe(3), IIe(3), Ies(3), SPF(3,3), OeIe(3), HOe(3)
      Real*8 OeIemag, dOeIemag, OeIedot(3), IeOe(3), HIe(3)
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
      Common /rvec/ rpf, rtta, rtac, rcto
c     Multiply the insertion vector by the transformation shifter matrix
      Call Matmult(SPF, Ie, Ies)
c     Find the vector connecting the effective origin and the effective
c     insertion points (vector OeIe). This vector represents the line of
c     action of the muscle force vector.  Also find vector HOe which is
c     the moment arm vector from the femur reference frame origin to the
c     the effective origin of the muscle.
      Do 20 j=1,3
         OOe(j)  = Oe(j) - O(j)
         IIe(j)  = I(j) - Ie(j)
         OeIe(j) = rpf(j) + Ies(j) - Oe(j)
         IeOe(j) = -OeIe(j)
```

```
         HOe(j)  = Oe(j) - rpf(j)
         HIe(j)  = Ies(j)
20     Continue
c      Calculate the length of musculotendon unit
       OeIemag = dsqrt(OeIe(1)**2+OeIe(2)**2+OeIe(3)**2)
       Lmt = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
      &     + dsqrt(IIe(1)**2+IIe(2)**2+IIe(3)**2)
      &     + OeIemag
       Call Cross(HIe, IeOe, mavec)
       ma1 = mavec(1)/OeIemag
       ma2 = mavec(2)/OeIemag
       ma3 = mavec(3)/OeIemag
c      Calculate time rate of change of the origin-insertion vector OeIe
       Call Cross(W, Ies, OeIedot)
       dOeIemag = dsqrt(OeIedot(1)**2+OeIedot(2)**2+OeIedot(3)**2)
c      Determine if muscle action is concentric or eccentric
       If (num.eq.1) then
          LmtDot = 0.0D0  !Can't determine velocity sign for first time step
          Goto 50
       End If
       If (Lmt.lt.LmtTest) then
          LmtDot = dOeIemag   !Positive velocity means muscle is shortening
       Else If (Lmt.gt.LmtTest) then
          LmtDot = -dOeIemag  !Negative velocity means muscle is lengthening
       Else If (Lmt.eq.LmtTest) then
          LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
       End If
 50    Continue
       Return
       End

c-------------------------------------------------------------------------

       Subroutine BArSPHK(num,rft,slrft,SPF,SPT,Wft,Wpf,Wpt,O,I,Lmt,
      &                   LmtTest,LmtDot,mah1,mah2,mah3,mak)
       Real*8 O(3), I(3), Lmt, LmtTest, LmtDot, mah1, mah2, mah3, mak
       Real*8 Is(3), OI(3), IO(3), HO(3), KI(3), slrft(3)
       Real*8 rft(3), rfts(3), mahvec(3), makvec(3)
       Real*8 SPT(3,3), SPF(3,3), Wft(3), Wpf(3), Wpt(3)
       Real*8 OIdot(3), dOImag, dvec1(3), dvec2(3), dvec3(3)
       Real*8 rpf(3), rtta(3), rtac(3), rcto(3)
       Common /rvec/ rpf, rtta, rtac, rcto
c      Transform the rft from femur to pelvis frame and transform the
c      insertion vector from tibia to pelvis frames
       Call Matmult(SPF, rft, rfts)
       Call Matmult(SPT, I, Is)
c      Find the necessary vectors for calculation of the scalar moment arms
c      and musculotendon length.  See notes for vector details.
       Do 20 j=1,3
          KI(j) = Is(j)
          OI(j) = rpf(j) + rfts(j) + Is(j) - O(j)
```

```
         IO(j) = -OI(j)
         HO(j) = O(j) - rpf(j)
 20    Continue
c      Calculate the magnitude (length of musculotendon unit) of OI vector
       Lmt = dsqrt(OI(1)**2+OI(2)**2+OI(3)**2)
c      Calculate the scalar moment arm for the hip and knee associated with
c      this muscle position.  See notes for derivation of this quantity
       Call Cross(HO, OI, mahvec)
       mah1 =  mahvec(1)/Lmt
       mah2 =  mahvec(2)/Lmt
       mah3 = -mahvec(3)/Lmt
       Call Cross(KI, IO, makvec)
       mak = makvec(3)/Lmt
c      Calculate time rate of change of the origin-insertion vector OI
       Call Cross(Wpf, rfts, dvec2)
       Call Cross(Wpt, Is, dvec3)
       Do 25 j=1,2
          dvec1(j) = slrft(j)*Wft(3)
 25    Continue
       dvec1(3) = 0.0D0  !The z slope of rft is zero
       Do 30 j=1,3
          OIdot(j) = dvec1(j) + dvec2(j) + dvec3(j)
 30    Continue
       dOImag = dsqrt(OIdot(1)**2+OIdot(2)**2+OIdot(3)**2)
c      Determine if muscle action is concentric or eccentric
       If (num.eq.1) then
          LmtDot = 0.0D0  !Can't determine velocity sign for first time step
          Goto 50
       End If
       If (Lmt.lt.LmtTest) then
          LmtDot = dOImag   !Positive velocity means muscle is shortening
       Else If (Lmt.gt.LmtTest) then
          LmtDot = -dOImag  !Negative velocity means muscle is lengthening
       Else If (Lmt.eq.LmtTest) then
          LmtDot = 0.0D0    !Muscle hasn't shortened or lengthened
       End If
 50    Continue
       Return
       End

c-----------------------------------------------------------------------

       Subroutine BiArMPKA(num,Kneez,rft,SFT,SFTa,SFC,Wft,Wtta,Wfc,O,
      &                    Oe,Wrp,I,Lmt,LmtTest,LmtDot,mak,maa1,maa2)
       Real*8 Kneez, Lmt, LmtTest, LmtDot, mak, maa1, maa2
       Real*8 SFT(3,3), SFTa(3,3), SFC(3,3), Wft(3), Wtta(3), Wfc(3)
       Real*8 rft(3), rttas(3), rtacs(3)
       Real*8 O(3), Oe(3), Wrp(3), I(3), Oes(3), Is(3), OeIdot(3)
       Real*8 OOe(3), OeI(3), IOe(3), KO(3), KWrp(3), AI(3), dOeImag
       Real*8 OWrp(3), WrpOe(3), OeImag, makvec(3), maavec(3)
       Real*8 dvec1(3), dvec2(3), dvec3(3), dvec4(3), WrpOemag
```

```
      Real*8 rpf(3), rtta(3), rtac(3), rcto(3), OOemag
      Common /rvec/ rpf, rtta, rtac, rcto
      Call Matmult(SFT, Oe, Oes)
      Call Matmult(SFT, rtta, rttas)
      Call Matmult(SFTa, rtac, rtacs)
      Call Matmult(SFC, I, Is)
      If ((Kneez.ge.-0.7700).and.(Kneez.le.0.1000)) then
c     The wrapping point condition
         Do 20 j=1,3
            OWrp(j)  = Wrp(j) - O(j)
            WrpOe(j) = rft(j) + Oes(j) - Wrp(j)
            OeI(j)   = rttas(j) + rtacs(j) + Is(j) - Oes(j)
            IOe(j)   = -OeI(j)
            KWrp(j)  = Wrp(j) - rft(j)
            AI(j)    = rtacs(j) + Is(j)
 20      Continue
         OeImag   = dsqrt(OeI(1)**2+OeI(2)**2+OeI(3)**2)
         WrpOemag = dsqrt(WrpOe(1)**2+WrpOe(2)**2+WrpOe(3)**2)
         Lmt = dsqrt(OWrp(1)**2+OWrp(2)**2+OWrp(3)**2) +
     &         OeImag + WrpOemag
         Call Cross(KO, WrpOe, makvec)
         Call Cross(AI, IOe, maavec)
         mak  = -makvec(3)/WrpOemag
         maa1 =  maavec(3)/OeImag
         maa2 =  maavec(1)/OeImag
      Else
c        The no wrapping point condition
         Do 50 k=1,3
            OOe(k) = rft(k) + Oes(k) - O(k)
            OeI(k) = rttas(k) + rtacs(k) + Is(k) - Oes(k)
            IOe(k) = -OeI(k)
            KO(k)  = O(k) - rft(k)
            AI(k)  = rtacs(k) + Is(k)
 50      Continue
         OeImag = dsqrt(OeI(1)**2+OeI(2)**2+OeI(3)**2)
         OOemag = dsqrt(OOe(1)**2+OOe(2)**2+OOe(3)**2)
         Lmt = OeImag + OOemag
         Call Cross(KO, OOe, makvec)
         Call Cross(AI, IOe, maavec)
         mak  = -makvec(3)/OOemag
         maa1 =  maavec(3)/OeImag
         maa2 =  maavec(1)/OeImag
      End If
c     Calculate the time derivative of OeI
      Call Cross(Wft, rttas, dvec1)
      Call Cross(Wtta, rtacs, dvec2)
      Call Cross(Wfc, Is, dvec3)
      Call Cross(Wft, Oes, dvec4)
      Do 70 k=1,3
         OeIdot(k) = dvec1(k) + dvec2(k) + dvec3(k) - dvec4(k)
 70   Continue
```

```
      dOeImag = dsqrt(OeIdot(1)**2+OeIdot(2)**2+OeIdot(3)**2)
c     Determine if muscle action is concentric or eccentric
      If (num.eq.1) then
         LmtDot = 0.0D0  !Can't determine velocity sign for first time step
         Goto 100
      End If
      If (Lmt.lt.LmtTest) then
         LmtDot = dOeImag   !Positive velocity means muscle is shortening
      Else If (Lmt.gt.LmtTest) then
         LmtDot = -dOeImag  !Negative velocity means muscle is lengthening
      Else If (Lmt.eq.LmtTest) then
         LmtDot = 0.0D0     !Muscle hasn't shortened or lengthened
      End If
  100 Continue
      Return
      End

c-----------------------------------------------------------------------

c     This routine calculates the femur-tibia kinematics (x,y,z trans-
c     lations of the tibia relative to the femur) and the slopes of these
c     functions based on knee angle.
      Subroutine FemTib(Kneeang, trans, slope)
      Real*8 Kneeang, trans(3), slope(3)
      Real*8 FemTibtx(8), FTtxang(8), FemTibty(6), FTtyang(6)
      Real*8 FTtxder(8), FTtyder(6), XHi, XLo, YHi, YLo
      Real*8 KneeHi, KneeLo
      Common /FemTb/    FemTibtx, FTtxang, FemTibty, FTtyang
      Common /FemTibDer/ FTtxder, FTtyder
c      Print*, 'Entering Subroutine FemTib'
      m=6
      n=8
      Call Splint(FTtxang,FemTibtx,FTtxder,n,Kneeang,trans(1))
      Call Splint(FTtyang,FemTibty,FTtyder,m,Kneeang,trans(2))
      trans(3) = 0.0D0
      KneeLo = Kneeang - 0.001
      KneeHi = Kneeang + 0.001
      Call Splint(FTtxang,FemTibtx,FTtxder,n,KneeLo,XLo)
      Call Splint(FTtxang,FemTibtx,FTtxder,n,KneeHi,XHi)
      Call Splint(FTtyang,FemTibty,FTtyder,m,KneeLo,YLo)
      Call Splint(FTtyang,FemTibty,FTtyder,m,KneeHi,YHi)
      Slope(1) = (XHi - XLo)/0.002
      Slope(2) = (YHi - YLo)/0.002
      slope(3) = 0.0D0
      Return
      End

c-----------------------------------------------------------------------

c     This routine calculates tibia-patella kinematics (x,y,z translations
c     and z rotation of the patella relative to the tibia) based on knee
```

```
c     angle.
      Subroutine TibPat(kneeang, trans, slope, rz, angslope)
      Integer n, m, o
      Real*8 kneeang, trans(3), rz, slope(3), angslope
      Real*8 TPtxang(7), TibPattx(7), TPtyang(8), TibPatty(8)
      Real*8 TPang(6), TibPatrz(6)
      Real*8 TPtxder(7), TPtyder(8), TPrzder(6)
      Real*8 KneeHi, KneeLo, XHi, XLo, YHi, YLo, rzHi, rzLo
      Common /TibPt/ TPtxang,TibPattx,TPtyang,TibPatty,TPang,TibPatrz
      Common /TibPatDer/ TPtxder, TPtyder, TPrzder
c      Print*, 'Entering Subroutine TibPat'
      n=6
      m=7
      o=8
      Call Splint(TPtxang,TibPattx,TPtxder,m,Kneeang,trans(1))
      Call Splint(TPtyang,TibPatty,TPtyder,o,Kneeang,trans(2))
      trans(3) = 0.00240
      Call Splint(TPang,TibPatrz,TPrzder,n,Kneeang,rz)
      KneeLo = Kneeang - 0.001
      KneeHi = Kneeang + 0.001
      Call Splint(TPtxang,TibPattx,TPtxder,m,KneeLo,XLo)
      Call Splint(TPtxang,TibPattx,TPtxder,m,KneeHi,XHi)
      Call Splint(TPtyang,TibPatty,TPtyder,o,KneeLo,YLo)
      Call Splint(TPtyang,TibPatty,TPtyder,o,KneeHi,YHi)
      Call Splint(TPang,TibPatrz,TPrzder,n,KneeLo,rzLo)
      Call Splint(TPang,TibPatrz,TPrzder,n,KneeHi,rzHi)
      Slope(1) = (XHi - XLo)/0.002
      Slope(2) = (YHi - YLo)/0.002
      slope(3) = 0.0D0
      angslope = (rzHi - rzLo)/0.002
      Return
      End

c-----------------------------------------------------------------------

c     This subroutine develops the transformation shifter matrix for a
c     3-1-2 series of dextral right hand rotations
      Subroutine Shift1(angle, shifter)
      Real*8 angle(3), shifter(3,3), C(3), S(3)
      Do 10, i=1,3
         C(i) = dcos(angle(i))
         S(i) = dsin(angle(i))
 10   Continue
      shifter(1,1) = (c(1)*c(3)) - (s(1)*s(2)*s(3))
      shifter(1,2) = -s(1)*c(2)
      shifter(1,3) = (c(1)*s(3)) + (s(1)*s(2)*c(3))
      shifter(2,1) = (s(1)*c(3)) + (c(1)*s(2)*s(3))
      shifter(2,2) = c(1)*c(2)
      shifter(2,3) = (s(1)*s(3)) - (c(1)*s(2)*c(3))
      shifter(3,1) = -c(2)*s(3)
      shifter(3,2) = s(2)
```

```
      shifter(3,3) = c(2)*c(3)
      Return
      End

c-----------------------------------------------------------------------

c     This subroutine develops the transformation shifter matrix for a
c     planar right hand rotation around the z axis
      Subroutine Shift2(angle, shifter)
      Real*8 angle, shifter(3,3), C, S
      Do 10, i=1,3
         C = dcos(angle)
         S = dsin(angle)
 10   Continue
      shifter(1,1) = C
      shifter(1,2) = -S
      shifter(1,3) = 0.0D0
      shifter(2,1) = S
      shifter(2,2) = C
      shifter(2,3) = 0.0D0
      shifter(3,1) = 0.0D0
      shifter(3,2) = 0.0D0
      shifter(3,3) = 1.000
      Return
      End

c-----------------------------------------------------------------------

c     This subroutine develops the special transformation shifter matrix
c     for the toes-calcaneus joint.  The input angle is the metatarsal-
c     phalangeal(MTP) angle.

      Subroutine ToCaShft(angle, shifter)
      Real*8 angle, shifter(3,3), e(4)
c     Calculate the Euler Parameters
      e(1) = -0.581*dsin(angle/2.0)
      e(2) =  0.0D0
      e(3) =  0.814*dsin(angle/2.0)
      e(4) =  dcos(angle/2.0)
      Call EulerShft(e,Shifter)
      Return
      End

c-----------------------------------------------------------------------

c     This subroutine develops the special transformation shifter matrix
c     for the talus-tibia joint.  The input angle is the ankle angle.

      Subroutine TaTShft(angle, shifter)
      Real*8 angle, shifter(3,3), e(4)
c     Calculate the Euler Parameters
```

```
      e(1) = -0.105*dsin(angle/2.0)
      e(2) = -0.174*dsin(angle/2.0)
      e(3) =  0.979*dsin(angle/2.0)
      e(4) =  dcos(angle/2.0)
      Call EulerShft(e,Shifter)
      Return
      End

c-----------------------------------------------------------------------

c     This subroutine develops the special transformation shifter matrix
c     for the calcaneus-talus joint.  The input angle is the inversion/
c     eversion angle.

      Subroutine CTaShft(angle, shifter)
      Real*8 angle, shifter(3,3), e(4)
c     Calculate the Euler Parameters
      e(1) =  0.781*dsin(angle/2.0)
      e(2) =  0.600*dsin(angle/2.0)
      e(3) = -0.120*dsin(angle/2.0)
      e(4) =  dcos(angle/2.0)
      Call EulerShft(e,Shifter)
      Return
      End

c-----------------------------------------------------------------------

c     Routine to calculate the 3x3 transformation matrix given the 4 Euler
c     parameters.

      Subroutine EulerShft(e, shifter)
      Real*8 e(4), shifter(3,3)
      shifter(1,1) =  e(1)**2 - e(2)**2 - e(3)**2 + e(4)**2
      shifter(1,2) =  2.0*(e(1)*e(2)-e(3)*e(4))
      shifter(1,3) =  2.0*(e(1)*e(3)+e(2)*e(4))
      shifter(2,1) =  2.0*(e(1)*e(2)+e(3)*e(4))
      shifter(2,2) = -e(1)**2 + e(2)**2 - e(3)**2 + e(4)**2
      shifter(2,3) =  2.0*(e(2)*e(3)-e(1)*e(4))
      shifter(3,1) =  2.0*(e(1)*e(3)-e(2)*e(4))
      shifter(3,2) =  2.0*(e(2)*e(3)+e(1)*e(4))
      shifter(3,3) = -e(1)**2 - e(2)**2 + e(3)**2 + e(4)**2
      Return
      End

c-----------------------------------------------------------------------

c     This routine performs the matrix product C = AB where A is a 3x3
c     matrix and both B and C are 3x1 vectors.
      Subroutine Matmult(A, B, C)
      Real*8 A(3,3), B(3), C(3), DSUM
      Do 100 i=1,3
```

```
         DSUM = 0.0D0
         Do 50 j=1,3
            DSUM = DSUM + A(i,j)*B(j)
50       Continue
         C(i) = DSUM
100   Continue
      Return
      End

c-----------------------------------------------------------------------

c     This routine performs the matrix product C = AB where A, B, and C
c     are all 3x3 matrices
      Subroutine Matmult2(A, B, C)
      Real*8 A(3,3), B(3,3), C(3,3), DSUM
      Do 150 i=1,3
         Do 100 j=1,3
            DSUM = 0.0D0
            Do 50 k=1,3
               DSUM = DSUM + A(i,k)*B(k,j)
 50         Continue
            C(i,j) = DSUM
 100     Continue
 150  Continue
      Return
      End

c-----------------------------------------------------------------------

c     This routine subroutine calculates the cross product of two vectors
      Subroutine Cross(X,Y,D)
      Real*8 X(3), Y(3), D(3)
      D(1) = X(2)*Y(3)-Y(2)*X(3)
      D(2) = X(3)*Y(1)-Y(3)*X(1)
      D(3) = X(1)*Y(2)-Y(1)*X(2)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine StatMusc(npts, Lmt, Lm, Lt, Strain, Fm, Ft)
      Real*8 Lmt(1000,43), Lm(1000,43), Lt(1000,43), Strain(1000,43)
      Real*8 Fm(1000,43), Ft(1000,43), FinErr(1000,43)
      Real*8 actlen(17), actforc(17), passlen(13), passforc(13)
      Real*8 tendstrain(17), tendforc(17), ErrPer(1000,43)
      Real*8 actder2(17), passder2(13), tendder2(17)
      Real*8 Lmo(43), Fmo(43), Lts(43), Alphao(43)
      Common /MuscData/  actlen, actforc, passlen, passforc
      Common /Muscles/   Lmo, Fmo, Lts, Alphao
      Common /SplineDer/ actder2, passder2, tendder2
      Common /TendData/  tendstrain, tendforc
```

```
      Call Iterate(npts, Lmt, Lm, Lt, Fm, Ft, FinErr)

      Do 80 j=1,npts
         Write(6,210) j
         Do 50 k=1,43
            Strain(j,k) = (Lt(j,k)-Lts(k))/Lts(k)
            ErrPer(j,k) = FinErr(j,k)*100.0
            Print*, ' '
            Write(6,100) k, Lmt(j,k)
            Write(6,140) k, Lm(j,k)
            Write(6,110) k, Lmo(k)
            Write(6,150) k, Lt(j,k)
            Write(6,120) k, Lts(k)
            Write(6,200) k, Strain(j,k)
            Write(6,190) k, Alphao(k)
            Write(6,130) k, Fmo(k)
            Write(6,160) k, Fm(j,k)
            Write(6,170) k, Ft(j,k)
            Write(6,180) k, ErrPer(j,k)
            If (Strain(j,k).ge.0.1000) then
               Print*, ' '
               Print*, 'WARNING'
               Print*, 'Exceeded Limit for Tendon Strain'
            End If
 50      Continue
 80   Continue
 100  Format('Lmt(',i2,')          = ',f13.8)
 110  Format('Lmo(',i2,')          = ',f13.8)
 120  Format('Lts(',i2,')          = ',f13.8)
 130  Format('Fmo(',i2,')          = ',f13.8)
 140  Format('Lm(',i2,')           = ',f13.8)
 150  Format('Lt(',i2,')           = ',f13.8)
 160  Format('Fm(',i2,')           = ',f13.8)
 170  Format('Ft(',i2,')           = ',f13.8)
 180  Format('Final Error(',i2,') = ',f13.8,' %')
 190  Format('Alphao(',i2,')       = ',f13.8)
 200  Format('Strain(',i2,')       = ',f13.8)
 210  Format(/'Time Point #',i4)
      Return
      End

c-----------------------------------------------------------------------

      Subroutine Iterate(npts, Lmt, Lm, Lt, Fm, Ft, FinErr)
      Real*8 Lmt(1000,43), Lm(1000,43), Lt(1000,43)
      Real*8 Fm(1000,43), Ft(1000,43), FinErr(1000,43)
      Real*8 Lmo(43), Fmo(43), Lts(43), Alphao(43), FinE(43)
      Real*8 actlen(17), actforc(17), passlen(13), passforc(13)
      Real*8 tendstrain(17), tendforc(17)
      Real*8 actder2(17), passder2(13), tendder2(17)
```

```
      Real*8 Norml(43), NMuscFor(43), NTendFor(43), NLmt(43)
      Common /MuscData/  actlen, actforc, passlen, passforc
      Common /Muscles/   Lmo, Fmo, Lts, Alphao
      Common /SplineDer/ actder2, passder2, tendder2
      Common /TendData/  tendstrain, tendforc
      Do 100 i=1,npts
         Write(6,200) i
         Do 50 j=1,43
c           Determine if the musculotendon actuator is stretched enough
c           to produce force
            If (Lmt(i,j).le.(Lts(j)+0.4*(Lmo(j)*dcos(alphao(j))))) then
               Write(6,210) j
               Fm(i,j) = 0.0D0
               Ft(i,j) = 0.0D0
               Lm(i,j) = (Lmt(i,j)-Lts(j))/dcos(Alphao(j))
               Lt(i,j) = Lts(j)
               FinErr(i,j) = 0.0D0
               If (Lmt(i,j).lt.Lts(j)) then
                  Write(6,230) j
                  Lm(i,j) = 0.0D0
               End If
               Goto 40
            Else If (Lmt(i,j).ge.(Lts(j) +
     &                           1.6*(Lmo(j)*dcos(alphao(j))))) then
               Write(6,220) j
               Fm(i,j) = 0.0D0
               Ft(i,j) = 0.0D0
               Lm(i,j) = (Lmt(i,j)-Lts(j))/dcos(Alphao(j))
               Lt(i,j) = Lts(j)
               FinErr(i,j) = 0.0D0
               Goto 40
            End If

c           Calculate the normalized musculotendon length
            NLmt(j)=Lmt(i,j)/Lmo(j)
c           Determine which portion of the FL curve the muscle is on and
c           call the appropriate subroutine
            If (Lmt(i,j).eq.(Lmo(j)*dcos(alphao(j))+Lts(j)*(1.0 +
     &                0.027*dcos(alphao(j))))) then
               Fm(i,j) = Fmo(j)
               Ft(i,j) = Fmo(j)
               Lm(i,j) = Lmo(j)
               Lt(i,j) = Lmt(i,j) - Lmo(j)*dcos(alphao(j))
               FinErr(i,j) = 0.0D0
               Goto 40
            Else If (Lmt(i,j).lt.(Lmo(j)*dcos(alphao(j))+Lts(j)*(1.0 +
     &                           0.027*dcos(alphao(j))))) then
               Norml(j)=0.700
               Call AscendFL(NLmt(j), Norml(j), NMuscFor(j),
     &                       NTendFor(j), FinE(j), j)
            Else If (Lmt(i,j).gt.(Lmo(j)*dcos(alphao(j))+Lts(j)*(1.0 +
```

```
     &                              0.027*dcos(alphao(j))))) then
                Call DescendFL(NLmt(j), Norml(j), NMuscFor(j),
     &                         NTendFor(j), FinE(j), j)
             End If
c            After the iteration calculate the final muscle quantities
             Lm(i,j)=norml(j)*Lmo(j)
             Lt(i,j)=(NLmt(j)-norml(j)*dsqrt(1.0-(dsin(alphao(j))/
     &                norml(j))**2))*Lmo(j)
             Fm(i,j)=NMuscFor(j)*Fmo(j)
             Ft(i,j)=NTendFor(j)*Fmo(j)
             FinErr(i,j) = FinE(j)
             If (ABS(FinErr(i,j)).gt.0.0010) then
                Write(6,240) j
             End If
 40          Continue
 50       Continue
 100   Continue
 200   Format('Finding Static Muscle Force .... Iterating time pt.',I4)
 210   Format('Muscle(',I2,') not stretched enough to produce force')
 220   Format('Muscle(',I2,') stretched too much to produce force')
 230   Format('Muscle(',I2,') has Lmt shorter than Lts')
 240   Format('Muscle(',I2,') -- Iteration Error greater than 0.108')
       Return
       End

c-------------------------------------------------------------------------

       Subroutine AscendFl(NLmt, Norml, Nmf, Ntf, FinErr, i)
       Integer i
       Real*8 NLmt, Norml, Nmf, Ntf, Ntf1, Ntf2, FinErr
       Real*8 Lmo(43), Fmo(43), Lts(43), Alphao(43)
       Real*8 actlen(17), actforc(17), passlen(13), passforc(13)
       Real*8 tendstrain(17), tendforc(17)
       Real*8 actder2(17), passder2(13), tendder2(17)
       Real*8 NLts, NKt, NrmlMF(100)
       Real*8 Error, Err(100), Error2, Err2(100), Error3
       Real*8 Incrm, Incr(100), Incr2(100)
       Real*8 NE1, NN1, NE2, NN2
       Real*8 Lt, Strain
       Common /MuscData/  actlen, actforc, passlen, passforc
       Common /Muscles/   Lmo, Fmo, Lts, Alphao
       Common /SplineDer/ actder2, passder2, tendder2
       Common /TendData/  tendstrain, tendforc
       Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Error, i)
       If (dabs(Error).le.0.000001) goto 190
       Incrm = 0.3000/2.0000
       Do 100 j=1,15
          If (j.eq.1) then
             Incr(j)=Incrm
             If (Error.lt.0.0) then
                Norml=Norml+Incr(j)
```

```
            Else If (Error.gt.0.0) then
               Norml=Norml-Incr(j)
            End If
            Goto 30
         End If
         Incr(j)=Incr(j-1)/2.0000
         If (Err(j-1).lt.0.0) then
            Norml=Norml+Incr(j)
         Else If (Err(j-1).gt.0.0) then
            Norml=Norml-Incr(j)
         End If
 30      Continue
         NrmlMF(j)=Norml
         Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Err(j), i)
         If (dabs(Err(j)).le.0.000001) goto 190
         If (j.eq.1) then
            If (((Err(j).gt.0.0).and.(Error.lt.0.0)).or.
     &        ((Err(j).lt.0.0).and.(Error.gt.0.0))) then
              goto 120
            End If
            Goto 90
         End If
         If (((Err(j).gt.0.0).and.(Err(j-1).lt.0.0)).or.
     &        ((Err(j).lt.0.0).and.(Err(j-1).gt.0.0))) goto 120
 90      Continue
 100     Continue
         Goto 190
 120     Continue
c        Second stage of iteration
         If (j.eq.1) then
            Norml=(0.700+NrmlMF(j))/2.0000
            Incrm=dabs((((0.7000+NrmlMF(j))/2.0000)-0.7000)/2.0000)
         Else If (j.ne.1) then
            Norml=(NrmlMF(j)+NrmlMF(j-1))/2.0000
            Incrm=dabs((((NrmlMF(j)+NrmlMF(j-1))/2.0)-NrmlMF(j))/2.0)
         End If
         Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Error2, i)
         If (dabs(Error2).le.0.000001) goto 190
       Do 150 k=1,25
         If (k.eq.1) then
            Incr2(k)=Incrm
            If (Error2.lt.0.0) then
               Norml=Norml+Incr2(k)
            Else If (Error2.gt.0.0) then
               Norml=Norml-Incr2(k)
            End If
            Goto 130
         End If
         Incr2(k)=Incr2(k-1)/2.0000
         If (Err2(k-1).lt.0.0) then
            Norml=Norml+Incr2(k)
```

```
         Else If (Err2(k-1).gt.0.0) then
            Norml=Norml-Incr2(k)
         End If
 130     Continue
         Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Err2(k), i)
         If (dabs(Err2(k)).le.0.000001) goto 190
         If (Err2(k).lt.0.0) then
            NN1 = Norml
            NE1 = Err2(k)
         Else If (Err2(k).gt.0.0) then
            PN1 = Norml
            PE1 = Err2(k)
         End If
 150  Continue
      If (dabs(NE1).ge.PE1) then
         Norml = PN1
      Else If (dabs(NE1).lt.PE1) then
         Norml = NN1
      End If
      Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Error3, i)
 190  Continue
      Ntf = Ntf1
      FinErr = Ntf1 - Ntf2
      Return
      End

c-----------------------------------------------------------------------

      Subroutine DescendFl(NLmt, Norml, Nmf, Ntf, FinErr, i)
      Integer i, Test
      Real*8 NLmt, Norml, Nmf, Ntf, Ntf1, Ntf2, FinErr
      Real*8 Lmo(43), Fmo(43), Lts(43), Alphao(43)
      Real*8 actlen(17), actforc(17), passlen(13), passforc(13)
      Real*8 tendstrain(17), tendforc(17)
      Real*8 actder2(17), passder2(13), tendder2(17)
      Real*8 NLts, NKt, NrmlMF(100)
      Real*8 Error, Err(100), Error2, Err2(100), Error3
      Real*8 Incrm, Incr(100), Incr2(100)
      Real*8 Lt, Strain
      Real*8 NE1, NN1, PE1, PN1
      Common /MuscData/  actlen, actforc, passlen, passforc
      Common /Muscles/   Lmo, Fmo, Lts, Alphao
      Common /SplineDer/ actder2, passder2, tendder2
      Common /TendData/  tendstrain, tendforc
c     Test to see which portion of the descending curve the muscle is
c     operating
      Norml = 1.330
      Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Error, i)
      If (dabs(Error).le.0.000001) goto 190
      If (Ntf2.lt.0.0) then
         Norml = 1.165
```

```
            Incrm = 0.1650/2.0000
            Test = 0
         Else If (Ntf2.gt.0.0) then
            If (Error.gt.0.0) then
               Norml = 1.165
               Incrm = 0.1650/2.0000
               Test = 0
            Else If (Error.lt.0.0) then
               Norml = 1.465
               Incrm = 0.1350/2.0000
               Test = 1
            End If
         End If
c        Find Correct Tendon Force
         Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Error, i)
         If (dabs(Error).le.0.000001) goto 190
         Do 100 j=1,15
            If (j.eq.1) then
               Incr(j)=Incrm
               If (Error.lt.0.0) then
                  Norml=Norml+Incr(j)
               Else If (Error.gt.0.0) then
                  Norml=Norml-Incr(j)
               End If
               Goto 30
            End If
            Incr(j)=Incr(j-1)/2.0000
            If (Err(j-1).lt.0.0) then
               Norml=Norml+Incr(j)
            Else If (Err(j-1).gt.0.0) then
               Norml=Norml-Incr(j)
            End If
 30         Continue
            NrmlMF(j)=Norml
            Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Err(j), i)
            If (dabs(Err(j)).le.0.000001) goto 190
            If (j.eq.1) then
               If (((Err(j).gt.0.0).and.(Error.lt.0.0)).or.
     &             ((Err(j).lt.0.0).and.(Error.gt.0.0))) then
                  goto 120
               End If
               Goto 90
            End If
            If (((Err(j).gt.0.0).and.(Err(j-1).lt.0.0)).or.
     &          ((Err(j).lt.0.0).and.(Err(j-1).gt.0.0))) goto 120
 90         Continue
 100        Continue
            Goto 190
 120        Continue
c           Second stage of iteration
            If (j.eq.1) then
```

```
c            Norml=(0.7000+NrmlMF(j))/2.0000
c            Incrm=dabs((((0.7000+NrmlMF(j))/2.0000)-0.7000)/2.0000)
             If (Test.eq.0) then
                Norml=(1.1650+NrmlMF(j))/2.0000
                Incrm=dabs((((1.1650+NrmlMF(j))/2.0000)-1.1650)/2.0000)
             Else If (Test.eq.1) then
                Norml=(1.4650+NrmlMF(j))/2.0000
                Incrm=dabs((((1.4650+NrmlMF(j))/2.0000)-1.4650)/2.0000)
             End If
          Else If (j.ne.1) then
             Norml=(NrmlMF(j)+NrmlMF(j-1))/2.0000
             Incrm=dabs((((NrmlMF(j)+NrmlMF(j-1))/2.0)-NrmlMF(j))/2.0)
          End If
          Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Error2, i)
          If (dabs(Error2).le.0.000001) goto 190
       Do 150 k=1,25
          If (k.eq.1) then
             Incr2(k)=Incrm
             If (Error2.lt.0.0) then
                Norml=Norml+Incr2(k)
             Else If (Error2.gt.0.0) then
                Norml=Norml-Incr2(k)
             End If
             Goto 130
          End If
          Incr2(k)=Incr2(k-1)/2.0000
          If (Err2(k-1).lt.0.0) then
             Norml=Norml+Incr2(k)
          Else If (Err2(k-1).gt.0.0) then
             Norml=Norml-Incr2(k)
          End If
  130     Continue
          Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Err2(k), i)
          If (dabs(Err2(k)).le.0.000001) goto 190
          If (Err2(k).lt.0.0) then
             NN1 = Norml
             NE1 = Err2(k)
          Else If (Err2(k).gt.0.0) then
             PN1 = Norml
             PE1 = Err2(k)
          End If
  150  Continue
       If (dabs(NE1).ge.PE1) then
          Norml = PN1
       Else If (dabs(NE1).lt.PE1) then
          Norml = NN1
       End If
       Call CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Error3, i)
  190  Continue
       Ntf = Ntf1
       FinErr = Ntf1 - Ntf2
```

```
      Return
      End

c-----------------------------------------------------------------------

      Subroutine CalcForc(NLmt, Norml, Nmf, Ntf1, Ntf2, Error, i)
      Integer i
      Real*8 NLmt, Norml, Nmf, Namf, Npmf, Ntf, Ntf1, Ntf2, Error
      Real*8 Lmo(43), Fmo(43), Lts(43), Alphao(43)
      Real*8 actlen(17), actforc(17), passlen(13), passforc(13)
      Real*8 tendstrain(17), tendforc(17)
      Real*8 actder2(17), passder2(13), tendder2(17)
      Real*8 NLts, NKt
      Common /MuscData/  actlen, actforc, passlen, passforc
      Common /Muscles/   Lmo, Fmo, Lts, Alphao
      Common /SplineDer/ actder2, passder2, tendder2
      Common /TendData/  tendstrain, tendforc
      n=17
      m=13
c     Calculate normalized muscle force
      Call Splint(actlen,actforc,actder2,n,Norml,Namf)
      Call Splint(passlen,passforc,passder2,m,Norml,Npmf)
      Nmf = Namf + Npmf
c     Calculate normalized tendon force by the first method
      Ntf1=Nmf*dsqrt(1.0-(dsin(alphao(i))/Norml)**2)
c     Calculate normalized tendon force by the second method
      NLts = Lts(i)/Lmo(i)
      NKt =  37.5/NLts
      Ntf2= Nkt*(NLmt-Norml*dsqrt(1.0-(dsin(alphao(i))/Norml)**2)-NLts)
      Error = Ntf1 - Ntf2
      Return
      End

c-----------------------------------------------------------------------

c     This subroutine calculates the second derivatives of the interpolat-
c     ing  functions - active muscle force-length, passive muscle force-
c     length, and tendon force-strain.  The control points for these func-
c     tions are taken from Delp (1990).  The second derivatives are used
c     later by the spline interpolation program to calculate a Y for a
c     given X.

      Subroutine CreateSplines
      Integer m, n, p, o, q, r
      Real*8 yp1, ypn
      Real*8 actlen(17), actforc(17), passlen(13), passforc(13)
      Real*8 tendstrain(17), tendforc(17)
      Real*8 actder2(17), passder2(13), tendder2(17)
      Real*8 FemTibtx(8), FTtxang(8), FemTibty(6), FTtyang(6)
      Real*8 FTtxder(8), FTtyder(6), DerVel(18)
      Real*8 TPtxang(7), TibPattx(7), TPtyang(8), TibPatty(8)
```

```
      Real*8 TPang(6), TibPatrz(6), vel(18), velforc(18)
      Real*8 TPtxder(7), TPtyder(8), TPrzder(6)
      Common /MuscData/  actlen, actforc, passlen, passforc
      Common /SplineDer/ actder2, passder2, tendder2
      Common /TendData/  tendstrain, tendforc
      Common /MuscVel/ vel, velforc
      Common /MVelDer/ DerVel
      Common /FemTb/    FemTibtx, FTtxang, FemTibty, FTtyang
      Common /FemTibDer/ FTtxder, FTtyder
      Common /TibPt/ TPtxang,TibPattx,TPtyang,TibPatty,TPang,TibPatrz
      Common /TibPatDer/ TPtxder, TPtyder, TPrzder
c     Set variables yp1 and ypn to values greater than 1e30 to signal the
c     the subroutine spline to set the corresponding boundary condition
c     for a natural cubic spline with zero second derivatives on that
c     boundary
      yp1 = 1.1e30
      ypn = 1.1e30
c     Set the size of arrays
      n=17
      r=18
      m=13
      o=6
      p=8
      q=7
c     Call the subroutine to create the cubic splines
      Call Spline(actlen,     actforc,  n, yp1, ypn, actder2)
      Call Spline(passlen,    passforc, m, yp1, ypn, passder2)
      Call Spline(tendstrain, tendforc, n, yp1, ypn, tendder2)
      Call Spline(vel,        velforc,  r, yp1, ypn, DerVel)
      Call Spline(FTtxang,    FemTibtx, p, yp1, ypn, FTtxder)
      Call Spline(TPtxang,    TibPattx, q, yp1, ypn, TPtxder)
      Call Spline(TPtyang,    TibPatty, p, yp1, ypn, TPtyder)
      Call Spline(TPang,      TibPatrz, o, yp1, ypn, TPrzder)
      Return
      End

c-----------------------------------------------------------------------

c     Subroutine MuscConv - converts the muscle parameters given by Delp
c     (1990) into appropriate units for calculations

      Subroutine MuscConv
      Parameter (pi=3.141593)
      Real*8 Lmo(43), Fmo(43), Lts(43), Alphao(43)
      Common /Muscles/  Lmo, Fmo, Lts, Alphao
c     Convert the pennation angle from degrees to radians and the lengths
c     from centimeters to meters
      Do 11 i=1,43
         Alphao(i)=Alphao(i)*pi/180.0
         Lmo(i)=Lmo(i)/100.0
         Lts(i)=Lts(i)/100.0
```

```
 11   Continue
      Return
      End

c-----------------------------------------------------------------------

c     Subroutine SPLINT - From "Numerical Recipes - The Art of Scientific
c     Computing" by W.H. Press et. al.  The algorithm originated in
c     "Computer Methods for Mathematical Computations" by George E.
c     Forsythe, et. al.
c     Purpose:  Given arrays XA, YA of length N, which tabulate a function
c     (with XA's in order), and given the array Y2A, which is the output
c     from Subroutine SPLINE, and given a value of X, this routine returns
c     a cubic-spline interpolated value Y.

      SUBROUTINE SPLINT(XA,YA,Y2A,N,X,Y)
      IMPLICIT REAL*8 (A-H, O-Z)
      DIMENSION  XA(N),YA(N),Y2A(N)
      KLO=1
      KHI=N
   1   IF (KHI-KLO.GT.1) THEN
          K=(KHI+KLO)/2
          IF(XA(K).GT.X)THEN
           KHI=K
        ELSE
           KLO=K
        ENDIF
      GOTO 1
      ENDIF
      H=XA(KHI)-XA(KLO)
      IF (H.EQ.0.) PAUSE 'Bad XA input.'
      A=(XA(KHI)-X)/H
      B=(X-XA(KLO))/H
      Y=A*YA(KLO)+B*YA(KHI)+
     &             ((A**3-A)*Y2A(KLO)+(B**3-B)*Y2A(KHI))*(H**2)/6.
      RETURN
      END

c-----------------------------------------------------------------------

c     Subroutine SPLINE - From "Numerical Recipes - The Art of Scientific
c     Computing" by W.H. Press et. al.  The algorithm originated in
c     "Computer Methods for Mathematical Computations" by George E.
c     Forsythe, et. al.
c     Purpose:  Given arrays X and Y of length N containing a tabulated
c     function, i.e. Y=f(x) with X1<X2<...<Xn, and given YP1 and YPN for
c     the first derivative of the interpolating function at points 1 and N
c     respectively, this routine returns an array Y2 of length N which
c     contains the second derivatives of the interpolating function at the
c     tabulated points X.  If YP1 and/or YPN are equal 1E30 or larger, the
c     routine is signalled to set the corresponding boundary condition for
```

```
c     a natural cubic spline, with zero second derivatives on that
c     boundary.

      SUBROUTINE SPLINE(X,Y,N,YP1,YPN,Y2)
      IMPLICIT REAL*8 (A-H, O-Z)
      PARAMETER (NMAX=20)
      DIMENSION X(N), Y(N), Y2(N), U(NMAX)
      IF (YP1.GT..99E30) THEN
          Y2(1)=0.
          U(1)=0.
      ELSE
          Y2(1)=-0.5
          U(1)=(3./(X(2)-X(1)))*((Y(2)-Y(1))/(X(2)-X(1))-YP1)
      ENDIF
      DO 11 I=2,N-1
          SIG=(X(I)-X(I-1))/(X(I+1)-X(I-1))
          P=SIG*Y2(I-1)+2.0
          Y2(I)=(SIG-1.0)/P
          U(I)=(6.0*((Y(I+1)-Y(I))/(X(I+1)-X(I))-(Y(I)-Y(I-1))
     &          /(X(I)-X(I-1)))/(X(I+1)-X(I-1))-SIG*U(I-1))/P
   11 CONTINUE
      IF (YPN.GT..99E30) THEN
          QN=0.
          UN=0.
      ELSE
          QN=0.5
          UN=(3./(X(N)-(N-1)))*(YPN-(Y(N)-Y(N-1))/(X(N)-X(N-1)))
      ENDIF
      Y2(N)=(UN-QN*U(N-1))/(QN*Y2(N-1)+1.)
      DO 12 K=N-1,1,-1
          Y2(K)=Y2(K)*Y2(K+1)+U(K)
   12 CONTINUE
      RETURN
      END

c-----------------------------------------------------------------------

c     This subroutine calculates the dynamic muscle force (maximum muscle
c     force for a given state) for each of the 43 lower extremity muscles
c     at each time step.

      Subroutine DynaMusc(npts, Vm, Fm, Fdyn)
      Real*8 Vm(1000,43), Fm(1000,43), Fdyn(1000,43)
      Real*8 Lmo(43), Fmo(43), Lts(43), Alphao(43)
      Real*8 vel(18), velforc(18), DerVel(18)
      Real*8 Vmax, NVm, NFdyn
      Common /Muscles/ Lmo, Fmo, Lts, Alphao
      Common /MuscVel/ vel, velforc
      Common /MVelDer/ DerVel
      n=18
      Do 100 i=1,npts
```

```
          Do 50 j=1,43
             Vmax = 10.0*Lmo(j)
             If (Vm(i,j).ge.Vmax) then
                Fdyn(i,j) = 0.0D0  !Muscle velocity is too high for force
                Write(6,200) i,j
                Goto 40
             End If
             NVm = Vm(i,j)/Vmax
             Call Splint(vel,velforc,DerVel,n,NVm,NFdyn)
c            The force-velocity curve is scaled by the force length curve
             Fdyn(i,j) = NFdyn*Fm(i,j)
 40          Continue
 50       Continue
100   Continue
      Print*, 'Finished with Dynamic Muscle Forces'
200   Format('Time Pt.',I4,' -- Muscle(',I2,') velocity is too high')
      Return
      End

c-------------------------------------------------------------------------------
```

APPENDIX G - OUTPUT FILES

The LEMMM creates 16 comma-delimited ASCII output files for input into popular spreadsheets. The first column of each of these files is the time record of the activity with each row being a different time point, the next 43 columns are the values of the specific parameter for each lower extremity muscle with the exception of the output file for the generalized coordinates which contains 8 columns of data after the time point column. For example, for a film analysis of an activity which takes 100 seconds and the time step is 1 second (i.e. 100 time points), then each output file would be 44 columns by 100 rows. The names of the ouput files and a brief description are given in Table G.1 below.

Table G.1 - Description of output files

Filename	Description
Lmt.out	musculotendon actuator length
Vmt.out	musculotendon actuator velocity
Lm.out	muscle length
Lt.out	tendon length
Vm.out	muscle velocity
theta.out	generalized coordinates
Fm.out	static muscle force
Ft.out	tendon force
Fdyn.out	dynamic muscle force
MForc.out	individual muscle force predictions
MAhipx.out	hip unit moments in the X direction
MAhipy.out	hip unit moments in the Y direction
MAhipz.out	hip unit moments in the Z direction
MAknee.out	knee unit moments in the Z direction
MAankz.out	ankle unit moments in the Z direction
MAankx.out	ankle unit moments in the X direction

APPENDIX H - EXAMPLE INPUT FILE FOR GAIT ANALYSIS

At the beginning of program execution for the LEMMM, the user is prompted to enter a name of the required input data file. A sample input file for the gait analysis is given below. Descriptions and FORTRAN format information are shown in the body of the input file. These comments are demarked by a /* and are in the Times-Roman font common to the entire thesis while the actual data is presented in Courier font. Comments should not be entered as part of the input file.

```
  69          /* Number of datapoints, Format (I4)
0.0000        /* Lines 2-70, time point values, Format (F6.4)
0.0140
0.0290
0.0430
0.0570
0.0720
0.0860
0.1000
0.1140
0.1290
0.1430
0.1570
0.1720
0.1860
0.2000
0.2150
0.2290
0.2430
0.2570
0.2720
0.2860
0.3000
0.3150
0.3290
0.3430
0.3570
0.3720
0.3860
0.4000
0.4150
0.4290
0.4430
```

```
0.4580
0.4720
0.4860
0.5000
0.5150
0.5290
0.5430
0.5580
0.5720
0.5860
0.6010
0.6150
0.6290
0.6430
0.6580
0.6720
0.6860
0.7010
0.7150
0.7290
0.7440
0.7580
0.7720
0.7860
0.8010
0.8150
0.8290
0.8440
0.8580
0.8720
0.8870
0.9010
0.9150
0.9290
0.9440
0.9580
0.9720
  -0.04189000    0.00000000    0.00000000  /* Lines 71-139, Format (3F13.8)
  -0.00349000    0.00000000    0.00000000  /* theta1,theta2,theta3
   0.04014300    0.00000000    0.00000000
   0.09075700    0.00000000    0.00000000
   0.14311700    0.00000000    0.00000000
   0.19722200    0.00000000    0.00000000
   0.24783700    0.00000000    0.00000000
   0.29321000    0.00000000    0.00000000
   0.33161300    0.00000000    0.00000000
   0.36302800    0.00000000    0.00000000
   0.38746300    0.00000000    0.00000000
   0.40666200    0.00000000    0.00000000
   0.41538800    0.00000000    0.00000000
```

0.42062400	0.00000000	0.00000000
0.41887900	0.00000000	0.00000000
0.41364300	0.00000000	0.00000000
0.40317100	0.00000000	0.00000000
0.39095400	0.00000000	0.00000000
0.37524600	0.00000000	0.00000000
0.35779300	0.00000000	0.00000000
0.33859400	0.00000000	0.00000000
0.31939500	0.00000000	0.00000000
0.30019700	0.00000000	0.00000000
0.28274300	0.00000000	0.00000000
0.26529000	0.00000000	0.00000000
0.24958200	0.00000000	0.00000000
0.23736500	0.00000000	0.00000000
0.22340200	0.00000000	0.00000000
0.21293000	0.00000000	0.00000000
0.20420400	0.00000000	0.00000000
0.19896800	0.00000000	0.00000000
0.19547700	0.00000000	0.00000000
0.19722200	0.00000000	0.00000000
0.20071300	0.00000000	0.00000000
0.20245800	0.00000000	0.00000000
0.19896800	0.00000000	0.00000000
0.18849600	0.00000000	0.00000000
0.17104200	0.00000000	0.00000000
0.15009800	0.00000000	0.00000000
0.12566400	0.00000000	0.00000000
0.10122900	0.00000000	0.00000000
0.07854000	0.00000000	0.00000000
0.06108700	0.00000000	0.00000000
0.04712400	0.00000000	0.00000000
0.03490700	0.00000000	0.00000000
0.02618000	0.00000000	0.00000000
0.01919900	0.00000000	0.00000000
0.01396300	0.00000000	0.00000000
0.00698100	0.00000000	0.00000000
0.00000000	0.00000000	0.00000000
-0.00873000	0.00000000	0.00000000
-0.02094000	0.00000000	0.00000000
-0.03142000	0.00000000	0.00000000
-0.04363000	0.00000000	0.00000000
-0.05411000	0.00000000	0.00000000
-0.06458000	0.00000000	0.00000000
-0.07300000	0.00000000	0.00000000
-0.08029000	0.00000000	0.00000000
-0.08552000	0.00000000	0.00000000
-0.09076000	0.00000000	0.00000000
-0.09774000	0.00000000	0.00000000
-0.10297000	0.00000000	0.00000000
-0.10647000	0.00000000	0.00000000
-0.10821000	0.00000000	0.00000000

```
-0.10821000    0.00000000    0.00000000
-0.10472000    0.00000000    0.00000000
-0.09948000    0.00000000    0.00000000
-0.08727000    0.00000000    0.00000000
-0.06981000    0.00000000    0.00000000
-0.81506900      /* Lines 140-208, theta4, Format (F13.8)
-0.90931700
-0.99309200
-1.06465100
-1.11875600
-1.15540800
-1.17460700
-1.17460700
-1.15889900
-1.12748300
-1.08210400
-1.02450800
-0.95644000
-0.87790100
-0.79238000
-0.69987700
-0.60213900
-0.50091000
-0.39793500
-0.29496100
-0.20071300
-0.11693700
-0.04712400
 0.00175000
 0.03142000
 0.04014000
 0.03142000
 0.01047000
-0.01919900
-0.05236000
-0.08901200
-0.12915400
-0.17104200
-0.21118500
-0.24434600
-0.26878100
-0.28274300
-0.28448900
-0.27750700
-0.26529000
-0.24783700
-0.23038300
-0.21467600
-0.19896800
-0.18675000
```

```
-0.17627800
-0.16406100
-0.15358900
-0.14311700
-0.13264500
-0.12217300
-0.11170100
-0.10297400
-0.09424800
-0.09075700
-0.08901200
-0.09599300
-0.10821000
-0.12740900
-0.15533400
-0.19024100
-0.23212900
-0.28274300
-0.34208500
-0.41015200
-0.48520200
-0.56548700
-0.65275300
-0.74176500
 0.26529000   0.00000000   0.39554206   /* Lines 209-277, Format (3F13.8)
 0.28623000   0.00000000   0.35941097   /* D/Pflex, Inv/Eversion, MTP
 0.28798000   0.00000000   0.33612369   /* Film angles
 0.27227000   0.00000000   0.31917765
 0.24260000   0.00000000   0.31704941
 0.20420000   0.00000000   0.29324485
 0.16232000   0.00000000   0.27685961
 0.12043000   0.00000000   0.26310003
 0.08378000   0.00000000   0.25771460
 0.05236000   0.00000000   0.25663360
 0.02967000   0.00000000   0.26087345
 0.01222000   0.00000000   0.26614478
 0.00175000   0.00000000   0.27509628
-0.00523600   0.00000000   0.28167664
-0.01047200   0.00000000   0.29043720
-0.01221700   0.00000000   0.29651483
-0.01570800   0.00000000   0.30239847
-0.01745300   0.00000000   0.31129728
-0.01919900   0.00000000   0.32122731
-0.02268900   0.00000000   0.33131960
-0.02792500   0.00000000   0.34191267
-0.03316100   0.00000000   0.35258568
-0.03839700   0.00000000   0.35874955
-0.04363300   0.00000000   0.36611033
-0.04363300   0.00000000   0.37254543
-0.03665200   0.00000000   0.37573265
```

```
-0.02094400   0.00000000   0.37718151
 0.00698000   0.00000000   0.37771134
 0.04363000   0.00000000   0.37779615
 0.08203000   0.00000000   0.37185090
 0.11345000   0.00000000   0.35978408
 0.13265000   0.00000000   0.34190960
 0.13614000   0.00000000   0.32049955
 0.12566000   0.00000000   0.29750563
 0.10472000   0.00000000   0.27357133
 0.08029000   0.00000000   0.25112862
 0.05411000   0.00000000   0.23760143
 0.03142000   0.00000000   0.22715067
 0.01047000   0.00000000   0.22318359
-0.00698100   0.00000000   0.21892496
-0.02094400   0.00000000   0.21052495
-0.03665200   0.00000000   0.20268022
-0.04886900   0.00000000   0.19262554
-0.05934100   0.00000000   0.18811914
-0.06806800   0.00000000   0.18876827
-0.07330400   0.00000000   0.19365485
-0.08028500   0.00000000   0.19803566
-0.08552100   0.00000000   0.20167980
-0.09250200   0.00000000   0.19946696
-0.10297400   0.00000000   0.19473132
-0.11170100   0.00000000   0.19150374
-0.11693700   0.00000000   0.19595171
-0.12042800   0.00000000   0.20712545
-0.12042800   0.00000000   0.22227666
-0.12042800   0.00000000   0.23722615
-0.11868200   0.00000000   0.25332402
-0.11868200   0.00000000   0.26935441
-0.11868200   0.00000000   0.29400084
-0.11344600   0.00000000   0.32241387
-0.10297400   0.00000000   0.36495832
-0.08377600   0.00000000   0.41215838
-0.05585100   0.00000000   0.45930428
-0.02094400   0.00000000   0.49910392
 0.02443000   0.00000000   0.52388550
 0.07679000   0.00000000   0.52867685
 0.20595000   0.00000000   0.50490523
 0.26878000   0.00000000   0.46542912
 0.31940000   0.00000000   0.41748952
 2.39000000   0.00000000   0.00000000   /* Lines 278-346, Format (3F13.8)
 2.89000000   0.00000000   0.00000000   /* thdot1, thdot2, thdot3
 3.30000000   0.00000000   0.00000000
 3.58000000   0.00000000   0.00000000
 3.70000000   0.00000000   0.00000000
 3.62000000   0.00000000   0.00000000
 3.35000000   0.00000000   0.00000000
 2.95000000   0.00000000   0.00000000
```

2.48000000	0.00000000	0.00000000
1.99000000	0.00000000	0.00000000
1.48000000	0.00000000	0.00000000
0.98000000	0.00000000	0.00000000
0.50000000	0.00000000	0.00000000
0.09000000	0.00000000	0.00000000
-0.25000000	0.00000000	0.00000000
-0.53000000	0.00000000	0.00000000
-0.77000000	0.00000000	0.00000000
-0.99000000	0.00000000	0.00000000
-1.16000000	0.00000000	0.00000000
-1.28000000	0.00000000	0.00000000
-1.34000000	0.00000000	0.00000000
-1.33000000	0.00000000	0.00000000
-1.28000000	0.00000000	0.00000000
-1.21000000	0.00000000	0.00000000
-1.13000000	0.00000000	0.00000000
-1.03000000	0.00000000	0.00000000
-0.92000000	0.00000000	0.00000000
-0.81000000	0.00000000	0.00000000
-0.69000000	0.00000000	0.00000000
-0.52000000	0.00000000	0.00000000
-0.30000000	0.00000000	0.00000000
-0.04000000	0.00000000	0.00000000
0.16000000	0.00000000	0.00000000
0.18000000	0.00000000	0.00000000
-0.05000000	0.00000000	0.00000000
-0.47000000	0.00000000	0.00000000
-0.96000000	0.00000000	0.00000000
-1.38000000	0.00000000	0.00000000
-1.64000000	0.00000000	0.00000000
-1.71000000	0.00000000	0.00000000
-1.61000000	0.00000000	0.00000000
-1.40000000	0.00000000	0.00000000
-1.13000000	0.00000000	0.00000000
-0.89000000	0.00000000	0.00000000
-0.70000000	0.00000000	0.00000000
-0.56000000	0.00000000	0.00000000
-0.46000000	0.00000000	0.00000000
-0.43000000	0.00000000	0.00000000
-0.47000000	0.00000000	0.00000000
-0.58000000	0.00000000	0.00000000
-0.70000000	0.00000000	0.00000000
-0.79000000	0.00000000	0.00000000
-0.81000000	0.00000000	0.00000000
-0.79000000	0.00000000	0.00000000
-0.73000000	0.00000000	0.00000000
-0.63000000	0.00000000	0.00000000
-0.53000000	0.00000000	0.00000000
-0.45000000	0.00000000	0.00000000
-0.41000000	0.00000000	0.00000000

```
-0.40000000    0.00000000    0.00000000
-0.38000000    0.00000000    0.00000000
-0.32000000    0.00000000    0.00000000
-0.22000000    0.00000000    0.00000000
-0.08000000    0.00000000    0.00000000
 0.10000000    0.00000000    0.00000000
 0.33000000    0.00000000    0.00000000
 0.64000000    0.00000000    0.00000000
 1.04000000    0.00000000    0.00000000
 1.53000000    0.00000000    0.00000000
-6.74000000      /* Lines 347-415, thdot4, Format (F13.8)
-6.23000000
-5.41000000
-4.37000000
-3.19000000
-1.94000000
-0.68000000
 0.53000000
 1.66000000
 2.68000000
 3.60000000
 4.41000000
 5.11000000
 5.72000000
 6.23000000
 6.66000000
 6.99000000
 7.17000000
 7.16000000
 6.87000000
 6.27000000
 5.34000000
 4.13000000
 2.75000000
 1.33000000
 0.02000000
-1.04000000
-1.76000000
-2.20000000
-2.46000000
-2.67000000
-2.83000000
-2.84000000
-2.60000000
-2.07000000
-1.34000000
-0.55000000
 0.17000000
 0.72000000
 1.07000000
```

```
 1.21000000
 1.18000000
 1.06000000
 0.94000000
 0.85000000
 0.80000000
 0.76000000
 0.74000000
 0.73000000
 0.73000000
 0.72000000
 0.69000000
 0.61000000
 0.44000000
 0.18000000
-0.19000000
-0.63000000
-1.13000000
-1.65000000
-2.18000000
-2.71000000
-3.26000000
-3.83000000
-4.41000000
-4.97000000
-5.48000000
-5.90000000
-6.15000000
-6.21000000
 2.29000000   0.00000000  -0.76686001  /* Lines 416-484, Format (3F13.8)
 0.82000000   0.00000000  -2.07669425  /* D/PFlex, Inv/Eversion, MTP
-0.54000000   0.00000000  -1.40616918  /* Film angular velocities
-1.60000000   0.00000000  -0.66665250
-2.34000000   0.00000000  -0.90636158
-2.78000000   0.00000000  -1.40464842
-2.92000000   0.00000000  -1.05357230
-2.76000000   0.00000000  -0.66912526
-2.38000000   0.00000000  -0.22600372
-1.90000000   0.00000000   0.11040254
-1.41000000   0.00000000   0.33241901
-0.98000000   0.00000000   0.49709317
-0.64000000   0.00000000   0.54284453
-0.40000000   0.00000000   0.53617096
-0.25000000   0.00000000   0.51860017
-0.17000000   0.00000000   0.41805121
-0.14000000   0.00000000   0.51665133
-0.15000000   0.00000000   0.65807486
-0.21000000   0.00000000   0.69978791
-0.29000000   0.00000000   0.72296047
-0.36000000   0.00000000   0.74325711
```

```
-0.40000000   0.00000000   0.58845526
-0.34000000   0.00000000   0.47269157
-0.16000000   0.00000000   0.48217118
 0.21000000   0.00000000   0.33630419
 0.79000000   0.00000000   0.16203268
 1.54000000   0.00000000   0.06915510
 2.24000000   0.00000000   0.02148200
 2.63000000   0.00000000  -0.20482378
 2.47000000   0.00000000  -0.62952876
 1.77000000   0.00000000  -1.04646015
 0.77000000   0.00000000  -1.37300873
-0.25000000   0.00000000  -1.55193520
-1.08000000   0.00000000  -1.64015782
-1.59000000   0.00000000  -1.62089336
-1.77000000   0.00000000  -1.25716174
-1.71000000   0.00000000  -0.83803821
-1.52000000   0.00000000  -0.50390893
-1.31000000   0.00000000  -0.28749159
-1.14000000   0.00000000  -0.44242418
-1.04000000   0.00000000  -0.56775963
-0.96000000   0.00000000  -0.62559098
-0.84000000   0.00000000  -0.50891483
-0.66000000   0.00000000  -0.13481301
-0.49000000   0.00000000   0.19347499
-0.39000000   0.00000000   0.32389870
-0.40000000   0.00000000   0.28047481
-0.49000000   0.00000000   0.05002451
-0.60000000   0.00000000  -0.24285163
-0.63000000   0.00000000  -0.27831766
-0.51000000   0.00000000   0.04265309
-0.30000000   0.00000000   0.54598492
-0.10000000   0.00000000   0.92006648
 0.01000000   0.00000000   1.05203021
 0.02000000   0.00000000   1.08511722
 0.01000000   0.00000000   1.12289453
 0.05000000   0.00000000   1.42166924
 0.22000000   0.00000000   1.85444772
 0.55000000   0.00000000   2.47999024
 1.04000000   0.00000000   3.13660383
 1.62000000   0.00000000   3.29742599
 2.21000000   0.00000000   3.03877950
 2.79000000   0.00000000   2.25713730
 3.40000000   0.00000000   1.21786940
 4.03000000   0.00000000   0.16746047
 4.50000000   0.00000000  -1.01511216
 4.54000000   0.00000000  -2.21053195
 3.97000000   0.00000000  -3.05521083
 2.84000000   0.00000000  -6.26692009
 0.00000000   0.00000000  22.90000000  /* Lines 485-553, Format (3F13.8)
 0.00000000   0.00000000  17.80000000  /* Hip torques:  X, Y, Z
```

```
0.00000000   0.00000000  13.80000000
0.00000000   0.00000000  10.80000000
0.00000000   0.00000000   8.50000000
0.00000000   0.00000000   6.70000000
0.00000000   0.00000000   5.00000000
0.00000000   0.00000000   3.40000000
0.00000000   0.00000000   2.00000000
0.00000000   0.00000000   0.90000000
0.00000000   0.00000000   0.00000000
0.00000000   0.00000000  -0.80000000
0.00000000   0.00000000  -1.50000000
0.00000000   0.00000000  -2.50000000
0.00000000   0.00000000  -3.60000000
0.00000000   0.00000000  -5.00000000
0.00000000   0.00000000  -6.70000000
0.00000000   0.00000000  -8.70000000
0.00000000   0.00000000 -10.50000000
0.00000000   0.00000000 -12.20000000
0.00000000   0.00000000 -14.00000000
0.00000000   0.00000000 -16.00000000
0.00000000   0.00000000 -17.90000000
0.00000000   0.00000000 -19.10000000
0.00000000   0.00000000 -18.70000000
0.00000000   0.00000000 -16.10000000
0.00000000   0.00000000 -11.70000000
0.00000000   0.00000000 -54.40000000
0.00000000   0.00000000 -37.60000000
0.00000000   0.00000000 -23.50000000
0.00000000   0.00000000 -20.80000000
0.00000000   0.00000000 -11.10000000
0.00000000   0.00000000  -7.80000000
0.00000000   0.00000000  -0.40000000
0.00000000   0.00000000   4.70000000
0.00000000   0.00000000   7.30000000
0.00000000   0.00000000   9.90000000
0.00000000   0.00000000   5.00000000
0.00000000   0.00000000   3.00000000
0.00000000   0.00000000   4.70000000
0.00000000   0.00000000   6.50000000
0.00000000   0.00000000  12.00000000
0.00000000   0.00000000  12.50000000
0.00000000   0.00000000  10.90000000
0.00000000   0.00000000  10.10000000
0.00000000   0.00000000  11.20000000
0.00000000   0.00000000  13.80000000
0.00000000   0.00000000  16.70000000
0.00000000   0.00000000  17.70000000
0.00000000   0.00000000  15.20000000
0.00000000   0.00000000  11.20000000
0.00000000   0.00000000  14.60000000
0.00000000   0.00000000  12.20000000
```

```
 0.00000000   0.00000000  14.70000000
 0.00000000   0.00000000  16.60000000
 0.00000000   0.00000000  16.50000000
 0.00000000   0.00000000  16.10000000
 0.00000000   0.00000000  18.30000000
 0.00000000   0.00000000  23.60000000
 0.00000000   0.00000000  29.50000000
 0.00000000   0.00000000  34.40000000
 0.00000000   0.00000000  37.30000000
 0.00000000   0.00000000  37.20000000
 0.00000000   0.00000000  33.60000000
 0.00000000   0.00000000  27.50000000
 0.00000000   0.00000000  20.70000000
 0.00000000   0.00000000  15.20000000
 0.00000000   0.00000000  11.90000000
 0.00000000   0.00000000   9.30000000
 7.00000000      /* Lines 554-622, Knee torque, Format (F13.8)
 7.00000000
 6.90000000
 6.50000000
 5.80000000
 4.80000000
 3.60000000
 2.30000000
 1.00000000
-0.10000000
-1.00000000
-1.90000000
-2.70000000
-3.50000000
-4.20000000
-4.90000000
-5.80000000
-6.80000000
-8.00000000
-9.40000000
-11.10000000
-12.80000000
-14.30000000
-15.10000000
-14.60000000
-12.70000000
 -9.50000000
-33.80000000
-19.90000000
 -7.60000000
 -4.50000000
  7.80000000
 14.50000000
 24.80000000
```

```
  31.60000000
  35.00000000
  37.80000000
  32.50000000
  28.10000000
  24.80000000
  20.20000000
  19.80000000
  15.80000000
  10.90000000
   7.80000000
   6.60000000
   6.60000000
   7.00000000
   5.90000000
   2.30000000
  -2.30000000
  -0.10000000
  -5.00000000
  -6.20000000
  -7.90000000
 -10.40000000
 -12.50000000
 -12.40000000
 -10.00000000
  -6.40000000
  -2.20000000
   2.30000000
   6.50000000
  10.10000000
  12.50000000
  13.30000000
  12.50000000
  10.60000000
   6.80000000
   1.60000000   0.0000000   0.00000000   /* Lines 623-691, Format (3F13.8)
   1.60000000   0.0000000   0.00000000   /* D/PFlex, Inv/Eversion, MTP
   1.50000000   0.0000000   0.00000000   /* Torques
   1.30000000   0.0000000   0.00000000
   1.10000000   0.0000000   0.00000000
   0.90000000   0.0000000   0.00000000
   0.70000000   0.0000000   0.00000000
   0.60000000   0.0000000   0.00000000
   0.50000000   0.0000000   0.00000000
   0.40000000   0.0000000   0.00000000
   0.40000000   0.0000000   0.00000000
   0.50000000   0.0000000   0.00000000
   0.50000000   0.0000000   0.00000000
   0.60000000   0.0000000   0.00000000
   0.60000000   0.0000000   0.00000000
```

0.60000000	0.0000000	0.00000000
0.60000000	0.0000000	0.00000000
0.60000000	0.0000000	0.00000000
0.50000000	0.0000000	0.00000000
0.30000000	0.0000000	0.00000000
0.10000000	0.0000000	0.00000000
-0.10000000	0.0000000	0.00000000
-0.30000000	0.0000000	0.00000000
-0.50000000	0.0000000	0.00000000
-0.50000000	0.0000000	0.00000000
-0.50000000	0.0000000	0.00000000
-0.30000000	0.0000000	0.00000000
-1.70000000	0.0000000	0.00000000
4.60000000	0.0000000	0.00000000
8.30000000	0.0000000	0.00000000
2.80000000	0.0000000	0.00000000
6.90000000	0.0000000	0.00000000
4.60000000	0.0000000	0.00000000
5.00000000	0.0000000	0.00000000
2.60000000	0.0000000	0.00000000
-0.50000000	0.0000000	0.00000000
0.20000000	0.0000000	0.00000000
-3.50000000	0.0000000	0.00000000
-5.10000000	0.0000000	0.00000000
-6.60000000	0.0000000	0.00000000
-10.50000000	0.0000000	0.00000000
-10.60000000	0.0000000	0.00000000
-14.10000000	0.0000000	0.00000000
-18.70000000	0.0000000	0.00000000
-22.50000000	0.0000000	0.00000000
-25.30000000	0.0000000	0.00000000
-27.90000000	0.0000000	0.00000000
-30.30000000	0.0000000	0.00000000
-33.50000000	0.0000000	0.00000000
-38.80000000	0.0000000	0.00000000
-45.10000000	0.0000000	0.00000000
-45.40000000	0.0000000	0.00000000
-53.50000000	0.0000000	0.00000000
-58.80000000	0.0000000	0.00000000
-64.80000000	0.0000000	0.00000000
-71.20000000	0.0000000	0.00000000
-77.30000000	0.0000000	0.00000000
-82.70000000	0.0000000	0.00000000
-87.00000000	0.0000000	0.00000000
-89.70000000	0.0000000	0.00000000
-89.80000000	0.0000000	0.00000000
-86.70000000	0.0000000	0.00000000
-79.70000000	0.0000000	0.00000000
-68.60000000	0.0000000	0.00000000
-54.30000000	0.0000000	0.00000000
-38.80000000	0.0000000	0.00000000

```
-24.20000000    0.0000000    0.00000000
-12.30000000    0.0000000    0.00000000
 -3.60000000    0.0000000    0.00000000
```