

1A

# A Reproduced Copy 1430

OF



Reproduced for NASA

*by the*

**NASA Scientific and Technical Information Facility**

(NASA-CR-104970) TRANSLATED MARKING	100-24457
RELATES TO: KINEMATIC AND DYNAMIC	
ANALYSIS AND SIMULATION Advanced Design	
Program (our in Inst. of Tech.) 1970	uncles
OSCI 157 95/51	0700224

The George W. Woodruff  
School of Mechanical Engineering



**Georgia Institute  
of Technology**

Atlanta, Georgia 30332

---

**NASA/USRA  
University Advanced Design Program**

**Three Legged Walking Mobile Platform  
Kinematic & Dynamic  
Analysis and Simulation**

**June 1988**

**Gary V. McMurray  
Brice K. MacLaren**

## A THREE LEGGED WALKER

Georgia Institute of Technology  
Atlanta, Georgia

The three legged walker is proposed as a mobile work platform for numerous tasks associated with Lunar Base site preparation and construction. It is seen as one of several forms of surface transportation, each of which will be best suited for its respective tasks.

Utilizing the principle of dynamic stability and taking advantage of the Moon's Gravity, it appears to be capable of walking in any radial direction and rotating about a point. Typical curved path walking could involve some combination of the radial and rotational movements.

Comprised mainly of a body, six actuators, and six moving parts, it is mechanically quite simple. Each leg connects to the body at a hip joint and has a femur, a knee joint, and a tibia that terminates at a foot.

Also capable of enabling or enhancing the dexterity of a series of implements, the walker concept provides a mechanically simple and weight efficient means of drilling, digging, mining, and transporting cargo, and performing other like tasks.

A proof of principle machine has demonstrated the feasibility of the walking concept.

## Table of Contents

I. Abstract .....	1
II. Introduction .....	1
III. Motion .....	3
A. Lean .....	3
B. Leap .....	5
C. Crutch Walk .....	5
D. Squat .....	8
E. Pivoting .....	8
F. Self-Righting Mode .....	9
IV. Dynamics and Kinematics .....	10
A. Jump Mode .....	10
B. Lean Mode .....	15
V. Dynamic Simulation .....	21
A. Computer Model .....	21
B. Results .....	23
VI. Proof of Principle Model .....	30
VII. Conclusion .....	31
Appendix A - Figures	
Appendix B - SKIT_3-D Graphic Simulation, Figures, and Program Listing	
Appendix C - SKITTER Dynamics and Kinematics Program Listing	

### Introduction

The Georgia Institute of Technology has been involved in the design of machinery for the construction of the Lunar Base for several years. Because of the unique and stringent constraints imposed upon any piece of machinery that will operate in the Lunar environment, a three-legged walking robot, called SKITTER, is being designed and developed. While SKITTER has been initially designed for Lunar applications, it is not solely limited to the moon. Some terrestrial applications include hazardous environments, military reconnaissance, underwater operations, etc. The purpose of this paper is to discuss and detail some of the initial work leading to the development of the theory of a three-legged walker (gaits, modes of operation, kinematics, and dynamics) and the proof of principle model.

The task of designing automated machinery for the Lunar environment is very difficult. Besides the intense temperatures and lack of an atmosphere, the one-sixth gravity complicates the task of moving machinery and cargo. The reduced gravity, while making any payload lighter, also reduces the normal force at the ground to such a low point that it becomes extremely difficult to be able to do even the basic task of scraping soil at a construction site. Even though the weight has been reduced by one-sixth, the inertia has not been similarly reduced. Thus, it is not a sufficient solution to the problem to add more mass to the vehicle to generate the required normal force because the power requirements for acceleration and deceleration would then rise sharply. Since any construction machine must be able to traverse all terrain that it might encounter, the decision to use legged locomotion over wheeled vehicles was justified.

SKITTER is a very simple device from a mechanical point of view. It consists of only six actuators, six moving parts, and a central body (figs. 1 and 2). The legs are located radially

through the centerline of the body at 120 degrees apart from the other. The upper part of the leg, or femur, connects to the body at a hip joint. Connected at the other end of the femur is the lower leg, or tibia, which terminates as a foot at its end. To move the legs, two actuators per leg are used. The first actuator rotates the femur about the hinge line formed by the union of the femur and the central body, while the second actuator rotates the tibia about the hinge line created by the union of the femur and tibia. In this way, each leg operates in its own plane. The central body serves a connect point for various implements that may be attached to the walker and as host for the electronic hardware and power supply.

This mechanical simplicity does, however, have its disadvantage in the fact that more complex controls are needed for machine stability. For stability in motion, the main difference between a three-legged walker and the other walking devices like the Odetics and Ohio State walking machines is that the Georgia Tech walker depends on dynamic stability to maintain its motion. It must, and does, operate routinely with less than three legs having contact with the ground at any particular moment. As the device pushes off from the ground with its legs, the center of mass undergoes a horizontal and vertical motion. This differs from the previously mentioned walkers dramatically in that they strive to constantly maintain a level motion of the center of mass. While it does take energy to move the center of mass vertically, that energy is recovered when the body and leg are brought back into contact with the ground with no energy expenditure. Since the walker is not statically stable at all times in this motion, the controls complexity increases dramatically. This situation is similar to the unipod and bipods being developed at Carnegie Mellon and Clemson Universities, however with two major distinctions. The first is that while the tripod is not statically stable at all times while in motion, the device can always return to a statically stable position by simply allowing the leg or legs to return to the ground provided that the center of mass of the device is still located inside of the triangle formed by the foot projections on the ground. The second major difference is related to the first one in that only very small movements of the legs are needed to generate motion of the

platform. By taking small steps, the danger of tipping and energy consumption is minimized.

This motion of pushing off from the ground so that the foot actually leaves the ground is also significant from another point of view. That it is possible to establish a rocking motion of the walker such that the inertia and momentum from the restoration of the leg back to the ground, aids in the pushing off from the ground of the other legs. As will be discussed later in this paper, this type of motion is very similar to that of a man on crutches.

### Motion

Mechanical simplicity is a primary design constraint for SKITTER. Although other walkers incorporate many complex linkages and bearings in their design, SKITTER utilizes only six actuators ( two per leg ) and six hinges ( two per leg ) to generate motion (fig. 1 and 2). The femur actuator changes the angular position of the femur relative to the central body, and the tibia actuator changes the angular position of the tibia relative to the femur. By coordinating the position and velocities of the actuators, a variety of platform positions and motions is achieved.

### Lean

SKITTER's basic mode of operation is to reorient its central body or "lean" by reconfiguring the legs while always maintaining three fixed points of contact with the surface. To understand the lean sequence as well as the other modes of operation described later in this section, a fixed axis (X-Y-Z) is established such that the walking surface lies in the X-Z plane and the positive Y axis is normal to the surface following the right hand rule. SKITTER is oriented such that the motions of leg A are confined to the X-Y plane and the feet of leg B and leg C construct a line parallel to the Z axis known as pivot line A. A second axis (x-y-z)



may be established such that the  $x-z$  plane lies in the mid-plane of the central body with the positive  $x$  axis in the direction of leg A and the positive  $z$  axis parallel to and in the same direction as the  $Z$  axis. The positive  $y$  axis is normal to the mid plane and follows the right hand rule. Finally, a third axis ( $x'-y'-z'$ ) may also be established on the femur at the femur A - tibia A hinge line such that the positive  $z'$  axis is parallel and in the same direction as the positive  $Z$  axis (fig 3).

One possible example of the lean mode sequence is shown in fig 4. Starting with SKITTER at a static equilibrium position known as the 90-90 configuration ( the femur-tibia and tibia-surface angles are both 90 degrees ), femur A slowly rotates cw about the  $z$  axis and tibia A slowly rotates ccw about the  $z'$  axis such that foot A never leaves its initial contact point with the surface. The motion of femur A and tibia A is analogous to the motion of a crank and coupler of a slider crank mechanism discussed in detail in the kinematics section of this paper. The movements of leg A cause the central body and other two legs of SKITTER to rotate about pivot line A. As femur A and tibia A reach a new desired position, the central body has rotated and translated from its initial position with respect to the X-Y-Z reference frame.

One interesting motion of the central body is its ability to translate along its local  $y$  axis at an obtainable platform configuration. This can be accomplished by actuating each leg such that all three body-femur joints have a velocity vector parallel to the  $y$  axis and of equal magnitude. Therefore, as a drill rig platform, SKITTER eliminates the need for angular positioning and vertical feed mechanisms by leaning to the correct orientation and then raising and lowering itself along the drill string path by a series of coordinated actuator movements. The operation is completed with SKITTER's feet never losing contact with the ground. Also, the platform is able to achieve a position such that the mid-plane of the central body is parallel to the plane of its feet by simply making the body-femur angle and the femur-tibia angle of all three legs equal respectively. The work volume which encloses all of the possible orientations of the

central body is limited to the range of the actuators and the physical characteristics of the body and leg components.

### Leap

One method for maneuvering around obstacles which might impair the movement of SKITTER such as small rocks or ditches is to "leap" over them. SKITTER simply reorients its central body such that the y axis lies in the intended direction of travel. All three legs move such that the body-femur joints of all three legs have a velocity vector of equal magnitude and parallel to the y axis and supply a sufficient downward force to make SKITTER leap. With increases in control logic and proper frame design, the magnitude of the leap increases giving SKITTER the ability to achieve larger distances, and thus imitating the "skip walk" used by the astronauts on the lunar missions. One advantage of the leap mode is that the magnitude of directions in which SKITTER could translate is limited only by the possible orientations of the central body; therefore, with proper design, true omni-directional motion can be obtained.

### Crutch Walk

SKITTER's crutch walk mode for translational motion differs dramatically from most current walker designs which usually move one or more appendages while keeping at least three points of surface contact at all times. SKITTER, on the other hand, tries to capitalize off of its inertial characteristics and dynamic stability to propel itself forward. One inherent fact of a three legged platform such as SKITTER is that it will always be statically stable as long as all three feet are in contact with the surface and its center of gravity is positioned over the triangle formed by the feet. However, if one of the legs loses contact with the surface, the platform becomes statically unstable and starts rotating due to gravity about the pivot line constructed by the feet of the other two legs. By combining this fact with the lean motion,

SKITTER can be made to translate over a surface similar to a person walking with crutches.

For example, starting at the 90-90 equilibrium position, the femur and tibia of leg A begin the slider crank motion described in the kinematics section of this paper (fig 5). The central body starts to rotate about pivot line A as in the lean mode; however, this time, femur A and tibia A have acquired enough angular acceleration to supply a sufficient force at the foot and consequently a sufficient torque about pivot line A to cause foot A to lose contact with the surface (i.e. foot A pushes off from the surface fig 6). The entire platform continues to rotate about pivot line A until SKITTER's potential energy equals the kinetic energy imparted by leg A as it left the surface. At this point, the entire platform rotates about pivot line A in the opposite direction due to gravity. While leg A is away from the surface, femur A and tibia A rotate into a new configuration causing foot A to swing towards the central body (fig 6). As leg A comes back into contact with the surface, the central body is in a new orientation and foot A has translated to a new location on the surface in relation to the X-Y-Z reference frame (fig 7).

The next stage has leg B and leg C moving identically in their respective planes of motion. legs B & C reconfigure as shown in figure 7 causing a rotation of the central body in the X-Y plane about foot A. During their reconfiguration, legs B & C acquire adequate angular acceleration to supply a sufficient force at foot B and foot C, and consequently adequate torque about foot A, to cause the feet to lose contact with the surface (fig 8). Again, the platform will continue to rotate in the X-Y plane about foot A until its potential energy equals the kinetic energy imparted by legs B & C as they left the surface. At that time, the platform begins to rotate in the opposite direction due to gravity. While away from the surface all three legs reconfigure to their original 90-90 starting configuration (fig 8). As foot B and foot C reestablish contact with the surface, it is seen that the feet are in a new location and that the center of gravity has translated (fig 9). It is important to point out that if the roles of

leg A and legs B & C are interchanged (Reversed Crutch Walk mode) then a translation in the opposite direction occurs giving six radial directions of translation without a required rotation of the platform.

Surprisingly, it has been found by analysis that it takes little energy to have a leg push off from the surface with adequate force to give the leg time to reconfigure into a new position. Similarly, only a small rotation of the platform about the pivot line is needed to give adequate space for reconfiguration of the leg; therefore, the chances of the platform tipping over are small.

One important benefit which arises out of the Crutch Walk motion is a decrease in the energy input to the system as the platform gains momentum while it walks. With an increase in the gait of the crutch walk sequence and the proper control strategy, SKITTER is able to achieve a stable rocking motion. Just as a person who is walking quickly on crutches uses his momentum to swing himself forward, SKITTER uses its momentum to propel itself forward. Therefore, the horsepower to maintain the rocking motion is small since the energy input to the system only has to account for the losses in the system due to SKITTER contacting the surface.

Slopes can be negotiated quite easily using either the Crutch Walk or Reverse Crutch Walk mode with the requirement that the force vector due to gravity acting through the cg of SKITTER always intersect the triangle formed by the three feet. This requirement insures that SKITTER will not over turn and can always revert to a statically stable position. The grade of slope that SKITTER can effectively negotiate is primarily determined by the femur and tibia dimensions which determine the size of the triangle. A larger foot print triangle results in a larger margin of safety from over turning and therefore a larger grade of slope can be negotiated. The platform is able to walk up, down or tact a slope by assuming an optimum

nominal position (i.e. taking the largest step possible without overturning) and proceeding with one of the sequences described above.

### **Squat**

SKITTER has the ability to lower its central body by having each leg repeat the sequence of pushing off of the surface, reconfiguring so that the foot swings away from the central body and landing on the surface to reestablish static equilibrium. If the sequence is carried through enough iterations, the central body of SKITTER would come to rest on the surface with the legs extended outward (fig. 10). This particular position is extremely advantageous if the platform is being used in conjunction with a lifting device such as a crane. In the squat mode, the legs form outriggers to counter the weight of the cargo being lifted and eliminate the need for counter weights or other stability mechanisms.

### **Pivoting**

Although the platform has six radial directions for translation, there will be situations that will require for the platform to rotate about the surface Y axis. SKITTER is capable of two different pivoting modes. The foot pivoting mode allows the platform to pivot around one foot while the complex pivot mode allows SKITTER to swing one foot through an arc in the surface X-Z plane .

In the foot pivoting mode, Skitter pushes leg A off the surface, and while in the air, leg B reconfigures resulting in a torque about foot C . The platform will pivot around foot C, and as leg A contacts the surface, SKITTER will once again be statically stable. Since the hinge lines of the platform dictate a 120 degree interval between the planes of motion of the legs, the foot pivoting mode would also require that foot B either slide in a arc about foot C or be away

from the surface for the rotation to occur.

In the complex pivoting mode, SKITTER pushes leg A off the surface, and while in the air, leg B increases its body-femur angle while leg C decreases its body-femur angle. The reconfiguring of the legs in this manner will cause the platform to undergo rotations around the surface Y & X axis causing leg A to swing in a arc in the surface X-Z plane while both leg A and leg B remain fixed to the ground. Once leg A contacts the ground again, the platform becomes statically stable. If the motion is carried through for all three legs, SKITTER achieves a net rotation about its cg. Unlike the foot pivoting mode, the complex mode does not require foot B to slide or leave the surface for the rotation to occur.

#### **Self Righting Mode**

If, for some reason, the platform tripped or fell during one of the modes of operation, it has the capability of righting itself since the legs have a range of motion extending above and below the mid-plane of the central body (fig 11). As an extreme example, if SKITTER tripped and landed completely upside down on the surface, the platform could tuck two legs in toward the central body while the third leg pushed down on the ground to flip the platform over to the correct orientation (fig. 12). The resulting motion would simulate a person summersaulting and landing on his feet. This unique fault tolerant capability of SKITTER makes the platform a valuable remote field robot.

The movements just discussed were achieved by utilizing the inertia characteristics of SKITTER in conjunction with the coordinated actions of the six linear actuators. A direct relation between movement complexity and control complexity is apparent; however, the movements discussed be realized by current control strategies and devices.

## DYNAMICS and KINEMATICS

Of the many possible combinations of motions of the femur and tibia joints, only two possible combinations of motions exist such that the feet do not slid on the ground. The first motion is a linear movement of the center of mass as in the jump mode of operation. To accomplish this, the femur and tibia joints must combine their motions to produce a linear motion at the each of the hinge lines of the femur and the body. To model this linear motion of the center of mass, each leg is modeled as an offset slider crank mechanism. If it is desired for the walker to actually leave the ground, then the device must supply enough force that the walker has sufficient velocity at the end of the leg movements to leave the ground, or jump. The derivation of this model is developed here and the results of that model presented later.

The second method of motion for the walker is one that produces a rotational displacement of the center of mass relative to a pivot line, or the lean motion. This lean motion is also the fundamental motion for the crutch walk discussed earlier. The only difference between the motions of the femur and tibia joints in the lean configuration and the crutch walk is that at the end of the crutch walk motion, the body has enough angular velocity to allow the foot to leave the ground and this allows the leg to reconfigure while off the ground. To model this rotational motion of the body, a four-bar linkage is constructed where the four links are the ground, the tibia, the femur, and a link that is composed of the rigid structure of the body and the other two legs. The final link is connected to the hinge of the femur and body of the leg that is moving and terminates at the pivot line formed by the two feet that do not move. Derivation of the kinematic and dynamic model is discussed in this section and the results of the computer simulation is discussed in a latter section.

### *JUMP MODE*

The first motion to be discussed is the jump motion. This motion can be divided into two distinct phases. The first phase is the acceleration of the body in the local vertical axis to a prescribed velocity such that during phase two, the jump phase, the body is off the ground and it begins to decelerate under the force of gravity. Since linear motion is required, an offset slider crank mechanism is used to simulate the motion of the legs (Fig. 13a). As a simplification of the mathematical model, the problem was inverted such that the body of SKITTER was considered to be ground and the foot was constrained to move in the linear fashion. Thus, the femur is considered to be the crank and the foot is the slider. A derivation of that model is presented here along with a measure of the forces, torques, and angular velocities needed to produce this motion.

To conduct this analysis, several parameters must be defined by the user of the program developed to model this motion. The first parameter is the vertical distance, H, desired for SKITTER to jump (Fig. 13c). The second parameter is the vertical distance, D, that the legs are allowed to displace to accelerate the body to the desired velocity (Fig. 13b). This velocity,  $V_0$ , is obtained using the conservation of potential and kinetic energy theory where the final height is the jump distance, H, and the initial velocity is V. This results in,

$$V_0 = (2gH)^{1/2}$$

In this equation, g represents the gravitational constant. For simplicity, constant linear acceleration,  $A_0$ , is maintained at the center of mass for the acceleration phase. Thus, the acceleration necessary to accelerate the body to the desired velocity,  $V_0$ , over the linear distance, D, is,

$$A_0 = V_0^2/2D$$



Given the initial joint angles of the leg (the angle between the local x axis and the femur centerline,  $\theta_1$ , and the angle between the femur and tibia centerlines,  $\theta_2$ ), the other critical angle for the dynamic and kinematic analysis is the angle between the tibia and the vertical axis,  $\phi$  (Fig. 13a). This angle is determined from the joint angles to be,

$$\theta_2 - \theta_1 - \pi/2 = \phi$$

The next step is to derive the angular velocities for the femur and tibia links. The angular velocity equations are written using the standard equations for finding the velocity of a link. The equations are written from both ends of the link (i.e. one equation relates the velocity of point B to the ground, A, and the other relates the velocity of point B to the foot, point C, where the velocity at point C is a known parameter). The two equations are,

$$\underline{V}_b = \underline{V}_a + \omega_{ab} \times \underline{R}_{ab}$$

$$\underline{V}_b = \underline{V}_c + \omega_{bc} \times \underline{R}_{cb}$$

When the above equations are evaluated for this particular geometry, the following equation is derived,

$$(-\omega_{ab} L \sin(\theta_1)) \underline{i} - (\omega_{ab} L \cos(\theta_1)) \underline{j} = (-\omega_{bc} L \cos(\phi)) \underline{i} - (V_0 + \omega_{bc} L \sin(\phi)) \underline{j}$$

By comparing the  $\underline{i}$  and  $\underline{j}$  terms of the above equations, then the magnitude of the angular velocities can be derived,

$$\omega_{ab} = V_0 \cos(\phi) / L(\cos(\theta_1 + \phi))$$

$$\omega_{bc} = \omega_{ab} \sin(\theta_1) / \cos(\phi)$$

Like the velocity equations presented earlier, acceleration equations can be written in a similar manner.

$$\underline{\underline{A}}_b = \underline{\underline{A}}_a + \underline{\underline{\alpha}}_{ab} \times \underline{\underline{R}}_{ab} + \underline{\underline{\omega}}_{ab} \times (\underline{\underline{\omega}}_{ab} \times \underline{\underline{R}}_{ab})$$

$$\underline{\underline{A}}_b = \underline{\underline{A}}_c + \underline{\underline{\alpha}}_{bc} \times \underline{\underline{R}}_{cb} + \underline{\underline{\omega}}_{bc} \times (\underline{\underline{\omega}}_{bc} \times \underline{\underline{R}}_{cb})$$

Evaluating the above equations for the geometry of this problem yields,

$$(-\alpha_{ab} L \sin(\theta_1) - \omega_{ab}^2 L^2 \cos(\theta_1)) \underline{\underline{i}} + (-\alpha_{ab} L \cos(\theta_1) + \omega_{ab}^2 L \sin(\theta_1)) \underline{\underline{j}} =$$

$$(-\alpha_{bc} L \cos(\phi) + \omega_{bc}^2 L \sin(\phi)) \underline{\underline{i}} + (-A_0 - \alpha_{bc} L \sin(\phi) - \omega_{bc}^2 L \cos(\phi)) \underline{\underline{j}}$$

Solving the above equations properly for the magnitude of the angular accelerations gives,

$$\alpha_{bc} = -A_0 L \sin(\theta_1) - \omega_{bc}^2 L^2 \cos(\phi) \sin(\theta_1) - \omega_{bc}^2 L^2 \sin(\phi) \cos(\theta_1) - \omega_{ab}^2 L^2 / -L^2 (\cos(\phi + \theta_1))$$

$$\alpha_{ab} = (\alpha_{bc} L \cos(\phi) - \alpha_{bc}^2 L \sin(\phi) - \omega_{ab}^2 L \cos(\theta_1)) / L \sin(\theta_1)$$

For a complete force and torque analysis of the leg motion, it is necessary to obtain the linear acceleration of the center of mass for the tibia,  $\underline{\underline{X}}_{cg}$  and  $\underline{\underline{Y}}_{cg}$ . Using the acceleration equation for the center of mass relative to the foot gives,

$$\underline{\underline{A}}_{cg} = \underline{\underline{A}}_c + \underline{\underline{\alpha}}_{bc} \times \underline{\underline{R}}_{ccg} + \underline{\underline{\omega}}_{bc} \times (\underline{\underline{\omega}}_{bc} \times \underline{\underline{R}}_{ccg})$$

Evaluating this equation for the geometry yields,

$$\ddot{A}_{cg} = (-\alpha_{bc} L \cos(\phi)/2 + \omega_{bc}^2 L \sin(\phi)/2) \underline{i} + (-A_0 - \alpha_{bc} L \sin(\phi)/2 - \omega_{bc}^2 L \cos(\phi)/2) \underline{j}$$

Which leads to,

$$\ddot{X}_{cg} = -\alpha_{bc} L \cos(\phi)/2 + \omega_{bc}^2 L \sin(\phi)/2$$

$$\ddot{Y}_{cg} = -A_0 - \alpha_{bc} L \sin(\phi)/2 - \omega_{bc}^2 L \cos(\phi)/2$$

Summing the forces for the tibia (note that the force P is the reaction force from the ground due to the weight of SKITTER) enables a determination of the reaction forces at point B,

$$\Sigma F_x = m_t \ddot{X}_{cg} = F_{bx}$$

$$\Sigma F_y = m_t \ddot{Y}_{cg} = P - F_{by} - m_t g$$

Solving for the reaction forces,

$$F_{bx} = m_t \ddot{X}_{cg}$$

$$F_{by} = m_t \ddot{Y}_{cg} + m_t g - P$$

The summation of the moments about points A and B are.

$$\Sigma M_b = I_t \alpha_{bc} = PL \sin(\phi) - m_t g L \sin(\phi)/2 - T_t$$

$$\Sigma M_a = I_f \alpha_{ab} = F_{by} L \cos(\theta_1) - F_{bx} L \sin(\theta_1) - m_f g L \cos(\theta_1) / 2 - T_f$$

Thus, the torque needed about the femur and tibia are,

$$T_t = I_t \alpha_{bc} - PL \sin(\phi) + m_t g L \sin(\phi) / 2$$

$$T_f = I_f \alpha_{ab} + F_{bx} L \sin(\theta_1) + m_f g L \cos(\theta_1) / 2 - F_{by} L \cos(\theta_1)$$

The calculations for the mass moments of inertia are approximations based upon a rectangular cross section of the leg. The variable "a" is equal to the length of the leg and the variable "b" is equal to the width of the cross section of the leg.

$$I_x = m_{leg} (a^2 + b^2) / 12$$

From the above equations, all the variables are known except for the torques. Therefore, the torque needed at either the femur or tibia joints is known for any given position of the legs.

These equations were then implemented in a computer program to evaluate the angular velocity, torque, and horse power requirements for each joint of the leg as it attempts to make the leg jump the desired distance.

### **LEAN MOTION**

As stated earlier, the lean motion is the foundation of the crutch walk and provides dexterity to the platform. The kinematic model for this mode of operation is a four-bar

linkage where the four links are (Fig. 14a),

- Tibia
- Femur
- Rigid structure consisting of  
everything but links one and two
- Ground

The lean motion can also be divided into two separate phases of motion. The first is the acceleration phase where the femur and tibia move in such a fashion as to impart to the center of mass a prescribed angular velocity. This angular velocity is sufficient to allow the foot to leave the ground after the forces have ceased to be applied to the joints.

As in the jump mode analysis, it is necessary to input two parameters into the program to allow the remainder of the variables to be set. For the lean mode, it is necessary for the user to define the acceleration angle,  $\Psi$ , which is the angle that the center of mass of SKITTER is to undergo to obtain the necessary angular velocity (Fig. 14b). The other input parameter is the angular displacement,  $\Phi$ , that the user wants the center of mass to undergo after the foot leaves the ground (Fig. 14b).

With these input parameters, an initial angular velocity must be calculated so that the center of mass will undergo the desired amount of rotation. This initial angular velocity,  $\omega_0$ , is calculated from the conservation of potential and kinetic energy theory where the final angular velocity is zero and the final height of the center of mass is  $H_2$ , where  $H_2$  is defined as (Fig. 14b):

$$H_2 = r \sin(\Psi + \Phi)$$

Where  $r$  is defined as the constant perpendicular distance from the center of mass of SKITTER to the pivot line formed by the stationery feet. Thus, the initial angular velocity must equal:

$$\omega_0 = (2mgH_2 / I)^{1/2}$$

For this equation, the  $m$  terms refers to the mass of SKITTER and the  $I$  term is the moment of inertia for SKITTER about the pivot line.

As with the jump motion, a constant angular acceleration,  $\alpha_0$ , is assumed over the acceleration angle for the center of mass. This acceleration results in the desired angular velocity,  $\omega_0$ , at the end of the acceleration angle,  $\Psi$ .

$$\alpha_0 = \omega_0^2 / (2\Psi)$$

For this motion, there are a number of geometric parameters that must be summarized for the following analysis to be clear. First, the joint angles for the different links are designated as follows (Fig. 14a):

$\theta_1$ : angle between the tibia link and the ground

$\theta_2$ : angle between the femur link and  
horizontal at the femur-tibia joint

$\theta_3$ : angle between the rigid body of  
SKITTER and the ground.

As in the jump motion, all of these angles are measured in the right hand sense. The letters designate the various joints in this analysis. Point A is the hip joint, point B is the knee joint, point C represents the contact point between the ground and the tibia, and point D is the contact point for the line that is perpendicular to the pivot line and contains the center of mass.

With this variable definition, the following equations can be written for the velocity of the points on the links:

$$\underline{V}_b = \underline{V}_c + \underline{\omega}_{bc} \times \underline{R}_{cb}$$

$$\underline{V}_a = \underline{V}_b + \underline{\omega}_{ab} \times \underline{R}_{ba}$$

$$\underline{V}_a = \underline{V}_d + \underline{\omega}_{ad} \times \underline{R}_{da}$$

The above equations are evaluated using the following conditions,  $\underline{V}_c = \underline{V}_d = \underline{0}$ . With this above condition and the fact that  $\underline{\omega}_{ad}$  is the input velocity following a prescribed acceleration profile, then the remaining angular velocity terms can be calculated by solving the above equations for the scalar magnitudes of the angular velocities:

$$\omega_{ab} = R\omega_{ad}(\sin(\theta_1 - \theta_3))/(L\sin(\theta_1 - \theta_2))$$

$$\omega_{bc} = (\omega_{ad}R\sin(\theta_3) - \omega_{ab}L\sin(\theta_2))/L\sin(\theta_1)$$

As with the velocity equations, similar equations for the angular acceleration can be written.

$$\underline{A}_b = \underline{A}_c + \underline{\alpha}_{bc} \times \underline{R}_{cb} + \underline{\omega}_{bc} \times (\underline{\omega}_{bc} \times \underline{R}_{cb})$$

$$\underline{\dot{A}}_a = \underline{\dot{A}}_b + \underline{\alpha}_{ab} \times \underline{R}_{ba} + \underline{\omega}_{ab} \times (\underline{\omega}_{ab} \times \underline{R}_{ba})$$

$$\underline{\dot{A}}_a = \underline{\dot{A}}_d + \underline{\alpha}_{ad} \times \underline{R}_{da} + \underline{\omega}_{ad} \times (\underline{\omega}_{ad} \times \underline{R}_{da})$$

Again applying the boundary conditions of  $\underline{\dot{A}}_c = \underline{\dot{A}}_d = \underline{0}$  and the fact that the angular acceleration,  $\alpha_{ad}$ , is the  $\alpha_0$  calculated earlier, the following magnitudes for the angular accelerations may be derived.

$$\alpha_{ab} = (R\alpha_{ad}\sin(\theta_1 - \theta_3) - R\omega_{ad}^2\cos(\theta_1 - \theta_3) + L\omega_{ab}^2\cos(\theta_2 - \theta_1) + L\omega_{bc}^2)/L\sin(\theta_1 - \theta_2)$$

$$\alpha_{bc} = (\alpha_{ad}R\sin(\theta_3) + \omega_{ad}^2R\cos(\theta_3) - \omega_{bc}^2L\cos(\theta_1) - \alpha_{ab}L\sin(\theta_2) - \omega_{ab}^2L\cos(\theta_2))/L\sin(\theta_1)$$

Up to this point, the angular velocity and angular accelerations have been calculated for the various links. To complete the analysis for the forces and torques necessary to achieve the input motion, the linear acceleration of the center of mass for the femur, tibia, and rigid body must be calculated. Since all the velocities and accelerations are already known, this is very simple.

$$\ddot{X}_{T,CG} = A_{b,x}/2$$

$$\ddot{Y}_{T,CG} = A_{b,y}/2$$

$$\ddot{X}_{F,CG} = A_{bx} - (\alpha_{ab}L\sin(\theta_2) - \omega_{ab}^2L\cos(\theta_2))/2$$

$$\ddot{Y}_{F,CG} = A_{by} + (\alpha_{ab}L\cos(\theta_2) - \omega_{ab}^2L\sin(\theta_2))/2$$



$$\bar{X}_{\text{BODY,CG}} = A_{a,x}/2$$

$$\bar{Y}_{\text{BODY,CG}} = A_{a,y}/2$$

Now the free body diagrams can be written for this system. Unfortunately, the system is a coupled one, which means that the input torque, T, about the femur can not be easily solved for without solving for a eight other variables at the same time. These additional variables are the reaction forces at the joints and ground. Thus, the equations can be best solved by putting the equations into matrix form and then solving them using Gaussian elimination with pivoting. The form of the equations to be solved is as follows.

$$\begin{bmatrix}
 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & L\sin(\theta) & -L\cos(\theta) & 0 & 0 & 0 & 0 & 0 \\
 -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 L\sin(\theta_2) & -L\cos(\theta_2) & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 R\sin(\theta_3) & -R\cos(\theta_3) & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{bmatrix}
 \begin{bmatrix}
 F_{a,x} \\
 F_{a,y} \\
 F_{b,x} \\
 F_{b,y} \\
 F_{c,x} \\
 F_{c,y} \\
 F_{d,x} \\
 F_{d,y} \\
 T
 \end{bmatrix}
 =
 \begin{bmatrix}
 m_T X_{T,CG} \\
 m_T Y_{T,CG} \\
 I_c \alpha_{bc} + m_t g L \cos(\theta_1)/2 \\
 m_F X_{F,CG} \\
 m_F Y_{F,CG} \\
 I_b \alpha_{ab} + m_F g \cos(\theta_2)/2 \\
 m_{\text{Body}} X_{\text{Body,CG}} \\
 m_{\text{Body}} Y_{\text{Body,CG}} \\
 I_d \alpha_{ad} - m_{\text{Body}} g \cos(\theta_3)/2
 \end{bmatrix}$$

From the solution of this matrix for the variable T, the input torque required to have SKITTER lean and lift off from the ground can be calculated.

## Dynamic Simulation of the Motion of SKITTER

To simulate the motion of SKITTER, the dynamic and kinematic equations were developed for the basic motions, the jump and the lean motion, and then implemented in a computer program. These motions do not cover the broad scope of motions possible, especially all the possible motions resulting from the non-symmetric configurations of the legs. However, these two motions do represent the basic modes of operation for the machine (the other gaits are a combination of these two motions). Also included in this analysis is an actuator sizing routine which allows the user to determine if a given actuator (rotary or linear) can supply the necessary torque and speed.

The programs developed are written in a general format to allow the user to vary the physical parameters of SKITTER as well as modify its performance parameters. These performance parameters include the distance that the walker will jump in the air, the distance that the legs accelerate through before they leave the ground, and the actuator specifications (torque and velocity limitations). The physical parameters that can be varied on the model include all of the actuator attach points, the length of the femur and tibia, the weight of the femur and tibia (and thus its inertia), the weight of SKITTER, the gravity, and the type of actuator (rotary or linear).

From the dynamic analysis of the motion of SKITTER, a maximum torque and angular velocity about the hip and knee joints of the walker are calculated. If the type of actuator is a rotary one, then these values are compared against the input specifications for the actuator to determine if they will suffice. For a linear actuator, the moment arm about each joint must be calculated for that instant in time. The reason for this is that as the leg undergoes its motion, a linear actuator will not maintain a constant perpendicular

distance from the joint. Thus, whether a specified actuator will provide the prescribed motion is a function of two variables: torque required and perpendicular distance. To determine if the actuator will work, the two worst cases must be compared the actuator specifications. If the actuator is able to provide the necessary linear force and velocity for the two cases of maximum torque and minimum perpendicular distance, then the actuator will work.

### Jump Motion

The jump motion is described in detail in the another section of this paper and will not be redefined here. The dynamic model for this motion is one that provides linear motion of the hip joint (and thus the central body) as compared to the foot. Thus, the kinematic model for this motion is an off-set slider crank mechanism where the input is from the hip joint and the knee joint is a passive joint.

The solution to this kinematic problem was implemented in a computer program in order to solve for the angular velocities and accelerations of each joint. These values were very important in order that the dynamic problem could be solved completely. To accomplish this, the Newtonian Force equations were derived for the linkage so that the torque about each of the joints could be solved for as a function of its position.

### Lean Motion

As with the jump motion, the lean motion is described in detail else where in this paper. For this problem, the motion is modeled as a four-bar linkage where the joints are comprised of:

- 1) Tibia
- 2) Femur

- 3) Rigid structure consisting of everything but links 1 and 2 (the rest of SKITTER)
- 4) Ground

The kinematic solution to this problem is also well known but with one small difference: the input to the system undergoes a constant acceleration. While this is not a drastic change in the problem, it does considerably complicate the solution. With this accomplished, the next task was to derive the dynamic solution to determine the input torque needed to generate the angular velocities calculated by the kinematic equations as necessary to move SKITTER. Again using Newtonian mechanics and the angular accelerations and velocities from the kinematic problem, a system matrix of rank nine was derived that had to be solved to determine the input torque. The system matrix, while not triangular, was solvable using Gaussian elimination with row pivoting to obtain a solution to the input torque needed at the hip joint to provide the desired motion.

## Results

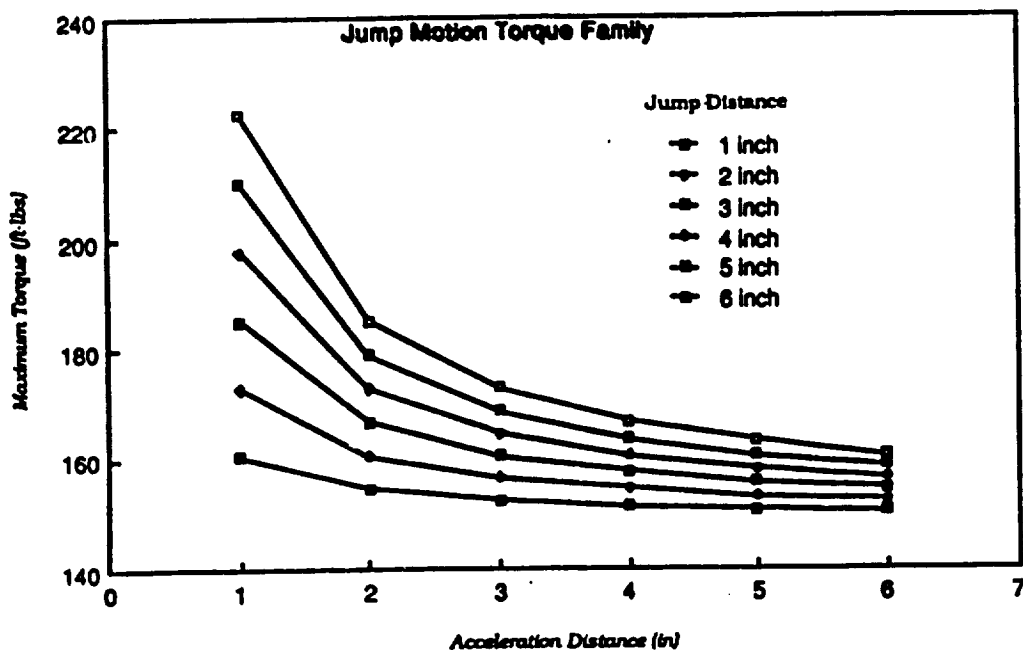
Presented below are the results of the numerical calculations and simulations performed under this contract. All of the number represent the power requirements of SKITTER while operating on earth and with a weight of three hundred pounds.

### Jump Motion

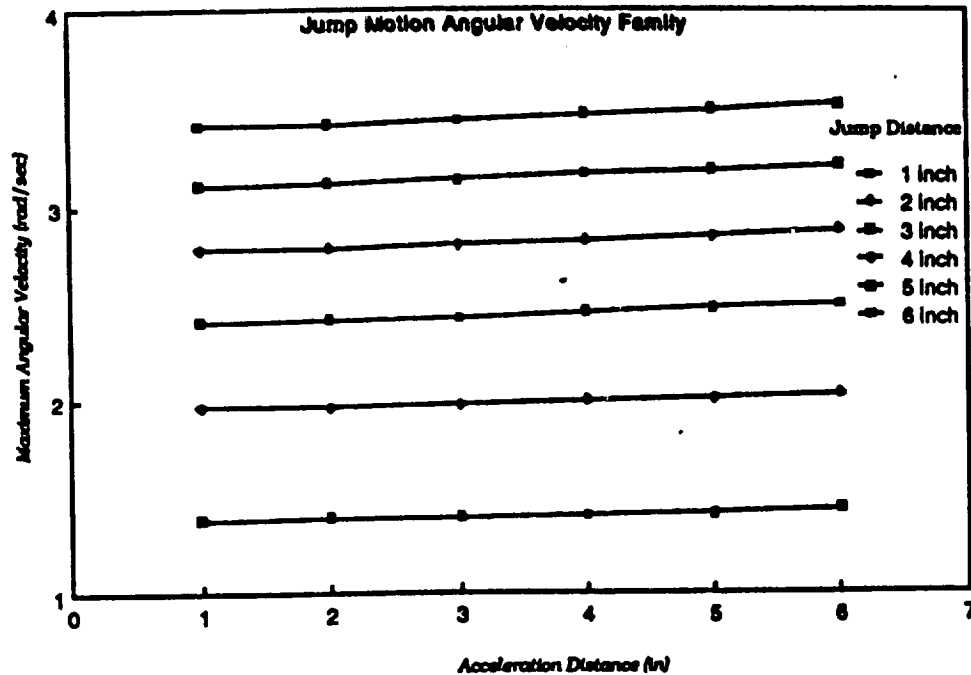
Sample data presented below is from the computer simulation of the dynamics of SKITTER. The family of curves shown below is for a variety of jump heights and acceleration distances. The physical parameters for the SKITTER model were determined from the envisioned SKITTER II model. The values shown are the requirements for each

leg as it attempts to jump. The total system requires three times as much torque and horse power to jump the distance desired.

The first plot shows a family of curves for the input torque at the hip joint at various jump distances and acceleration distances.



The following plot shows required maximum angular velocity at the hip joint to accelerate SKITTER to jump various heights through different acceleration distances.



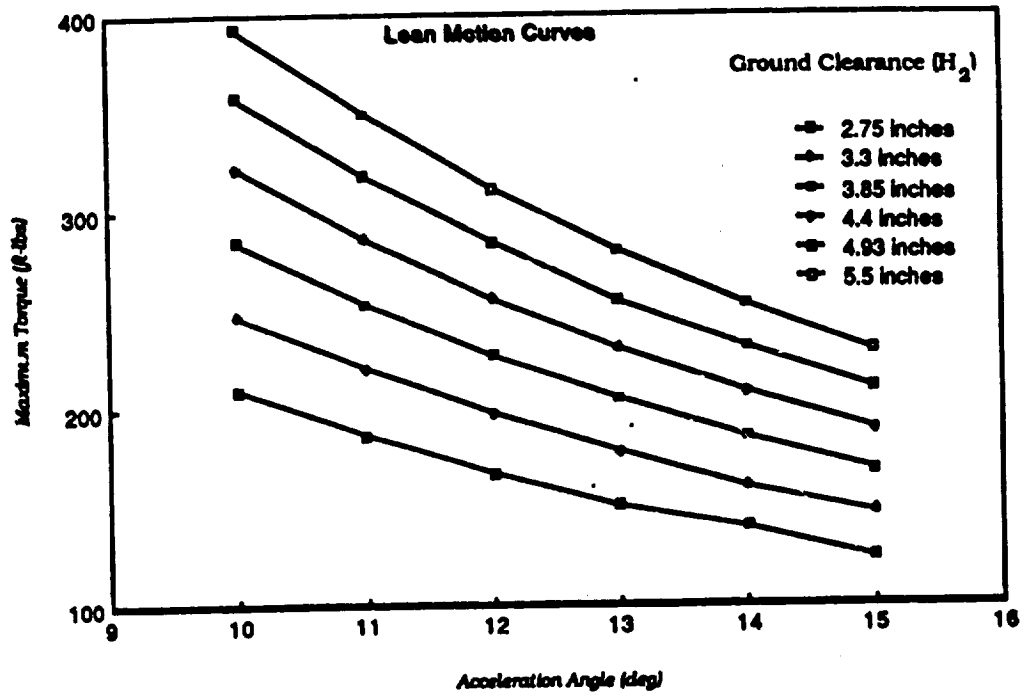
As can be seen from these plots, the requirements to jump are not as strenuous as originally thought. Sizing actuators to meet these specifications is not difficult and a vendor has already been identified that can meet these requirements. The Helac Corporation makes a series of planetary rotary actuators that are capable of supplying 4300 inch-pounds of torque with a full 360 degrees of rotation in the joint, while weighing only 24 pounds. These actuators will be ideal choice for use on SKITTER II.

### Lean Motion

For the lean motion, a family of curves was generated for various angles of acceleration (this is the angle that SKITTER accelerates through till it reaches the desired angular velocity to lift off from the ground) and ground clearances (the distance that the

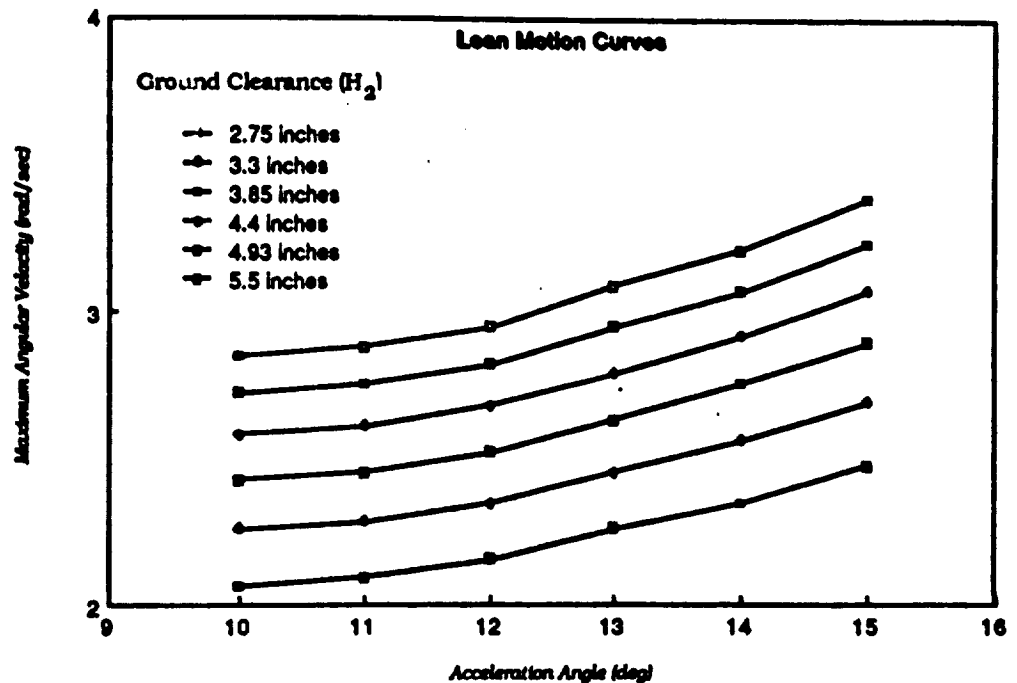
foot is from the ground at its maximum point). It is important to note that this simulation is for the case where one leg is providing the force to rotate SKITTER about the other two legs. In the crutch walk motion described earlier, it is also desired for the other two legs to push off from the ground and rotate about the stationary third leg during part of its motion. For this case, the angular velocity about each of the moving legs hip joint is the same whether one or two legs is pushing, but the torque can be divided between the two legs. Thus, the angular velocity requirements stay the same for this motion, but the torque requirements are divided between the two legs.

The first plot illustrates the family of curves relating the maximum input torque at the hip joint for a variety of acceleration angles and ground clearances (the distance that the foot of SKITTER leaves the ground during the rotation motion).



The next plot relates the maximum angular velocity at the hip joint and a family of acceleration angles and ground clearance.





The data presented here has been checked in various manners. For the lean motion, graphical techniques were used to verify the angular velocities and accelerations generated by the computer program. Since these numbers are then used to determine the input torque, this number is believed to be correct also.

### Future Work

The work completed under this contract to develop kinematic and dynamic equations for the motions of SKITTER has been completed. Actuator sizing programs have been developed so that the designer can vary the size of the structural members and optimize the power consumption for a given size actuator. The joint angles from the jump and lean motions can also be written to a file so that the graphical simulation program can display

the motions of SKITTER in 3-D. The next step in this process is to improve the computer models and incorporate in the program a control algorithm and inefficiencies in the power transmission. This will give the most accurate computer simulation of what the actual SKITTER II will be like when it eventually is built. Again, by using the computer simulations, the designer is able to optimize the design before any hardware is built. This allows the best prototype to be built.

### PROOF OF PRINCIPLE MODEL

To test the theory on the motion of a three-legged walker, a proof of principle working model was constructed. The walker was a one-tenth scale model of the conceptual design of the Lunar model and its purpose was only to obtain some translatory motion. This model, which weighed approximately eighty-five pounds and was completely self-contained, was demonstrated to NASA on several occasions where it fulfilled its intended goal and also demonstrated several other of the modes of operation discussed earlier.

The SKITTER proof of principle model was pneumatically actuated for cost effective reasons. A small scuba tank was attached to the underside of the model to serve as a high pressure reservoir of air. With a pneumatic actuation system, each actuator was only able to move the joint into two discrete positions, either the actuator was all the way in or all the way out (as compared to hydraulic or electromechanical actuators that can reach an infinite number of positions over its stroke length). Since the original design called for two actuators per leg, this results in four positions of the foot. To improve on this, the design for the prototype was modified such that each joint consisted of two actuators, four actuators per leg, and the actuators were connected by a free floating member. This gave each joint four possible positions that it could obtain and thus the foot could obtain sixteen different positions.

To control the actuation of the joints, a small computer was located aboard the model. Since no usable position or velocity feedback can be obtained from a pneumatic actuator, the computer was programmed with a series of commands that controlled the actuators and therefore the walker could demonstrate the modes of operations discussed earlier.

There are a number of fundamental differences between the proof of principle model and the SKITTER proposed for use on the moon. The first difference is that the prototype operated outside of its intended environment on the moon where the gravity is one-sixth's of

the earth's gravity. Thus, the power requirements were much higher on the prototype as compared to a comparable SKITTER on the moon. The second difference is that the prototype used a discrete actuation system that allows the feet to be in only sixteen discrete motions. Of those positions, only certain movements from these discrete positions allowed the prototype to move as designed. The final difference was that no control strategy could be implemented to provide smooth motion of the legs because of the pneumatic system's lack of useful feedback.

The fact that a successful prototype was built and demonstrated with these negative factors inhibiting its performance, shows that the idea is feasible and much easier to implement than originally thought. With the addition of servo-actuators, the motion can only be improved, but the fundamental concepts on the modes of operation of a three-legged walker have already been proved correct.

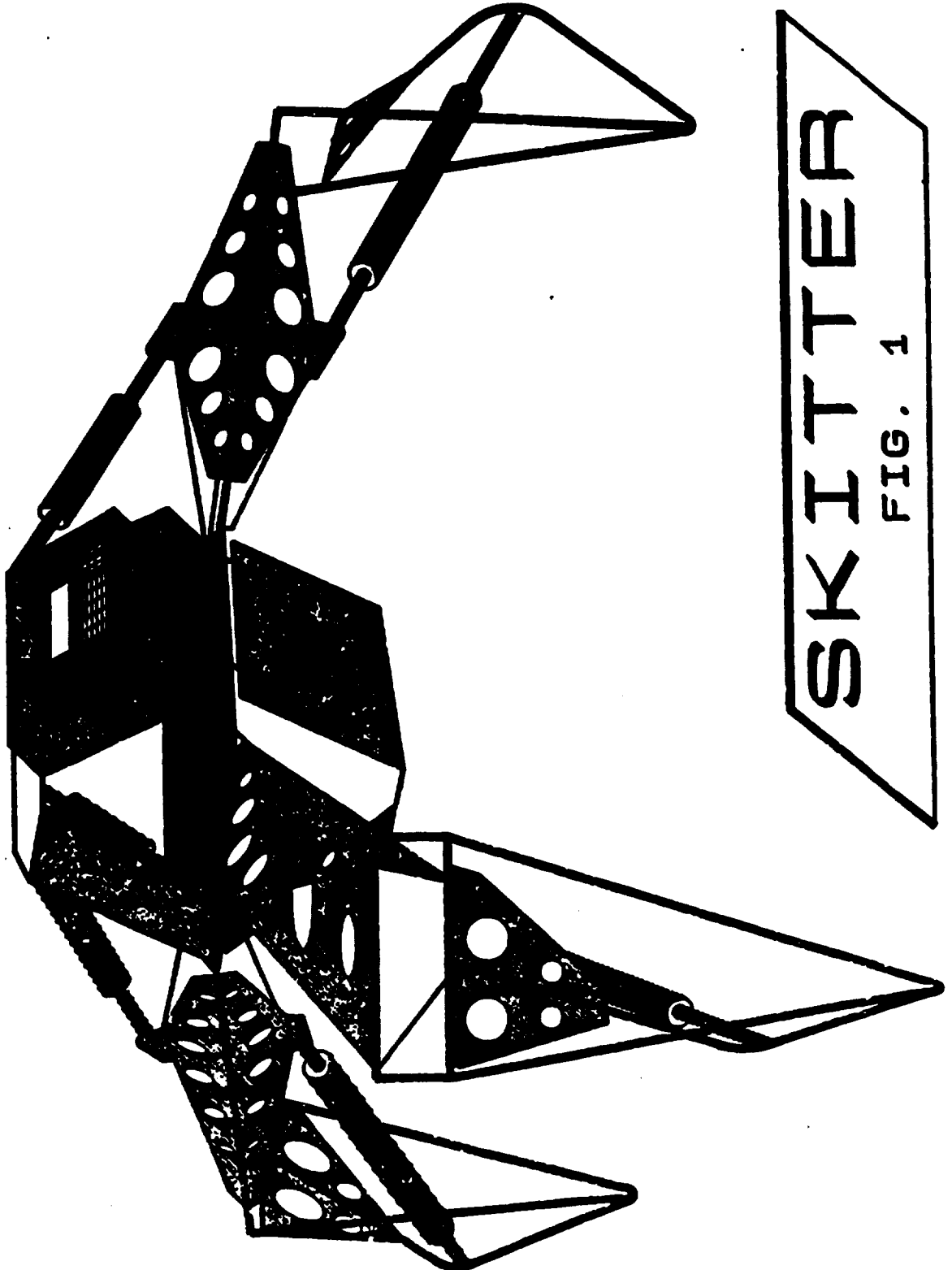
### CONCLUSION

With a successful proof of principle model developed, the Georgia Institute of Technology is continuing the development of the three-legged walker, SKITTER. To further a complete understanding of the dynamics and kinematics of the walker, computer simulation is being written to incorporate the equations of motion and display graphically SKITTER as it moves. Once this work has been completed, a control strategy and hardware (actuators and sensors) will be incorporated into the computer mode! to provide a realistic simulation of the next generation prototype. This model can then be evaluated and modified by the user before any hardware is actually built.

The next version of SKITTER, or SKITTER II, will have servo-actuators at the joints to allow feedback of the position and velocity of the joint so that the motion of the legs can be accurately controlled throughout the range of their motions. This new model will be capable of all the gaits and modes of operation described previously, but it will have much greater range

of motion and a much smoother motion. This next generation SKITTER model will also be compared against other walking and wheeled vehicles for overall efficiency, as well as the accuracy of the computer models.

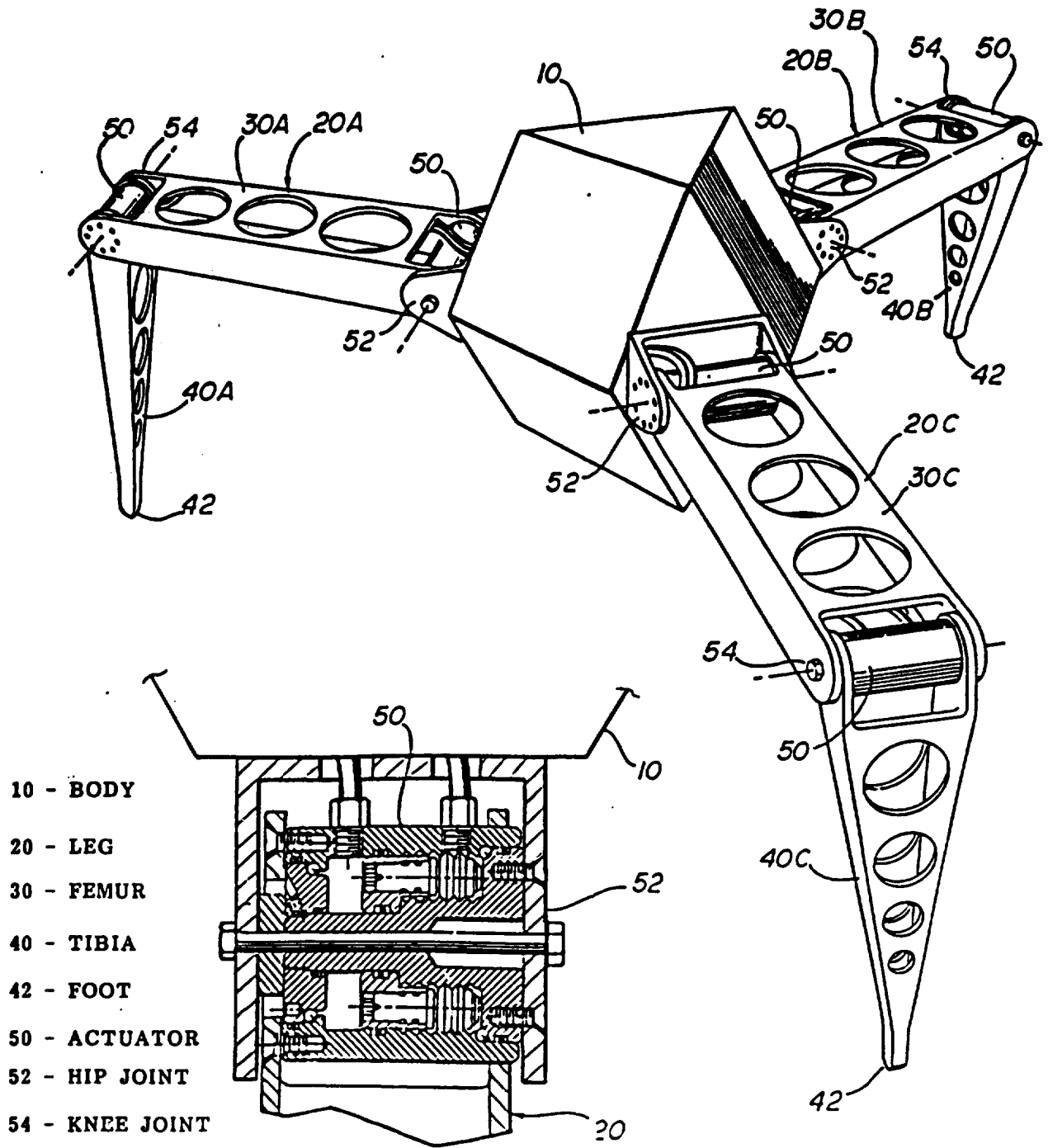
**Appendix A**  
**Figures**



**SKITTER**

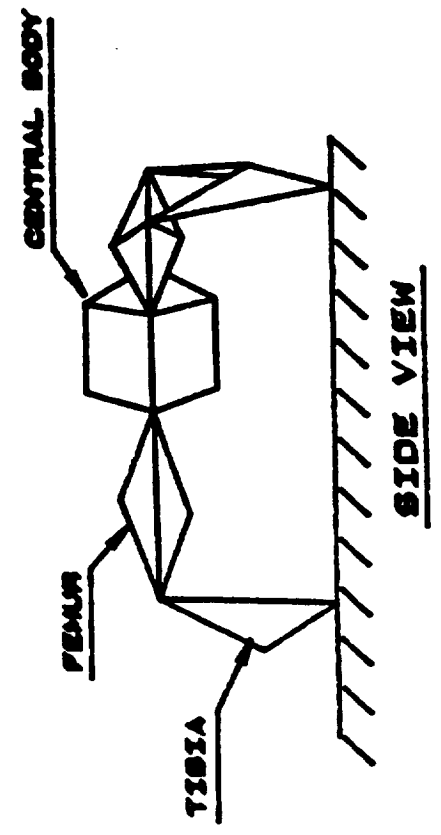
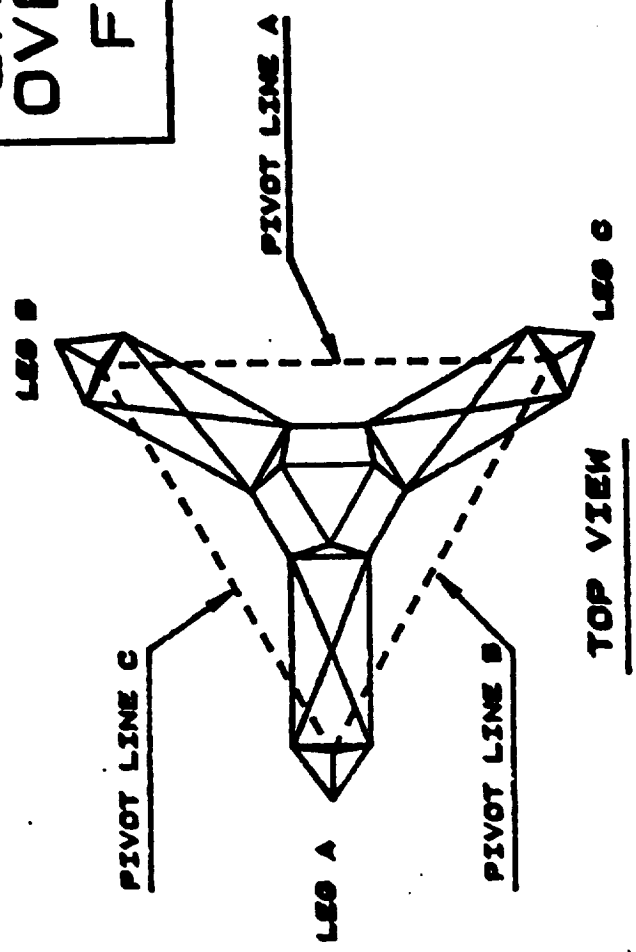
**FIG. 1**

**SKITTER 2**  
**WALKING MOBILE PLATFORM**





**SKITTER  
OVERVIEW  
FIG. 2**



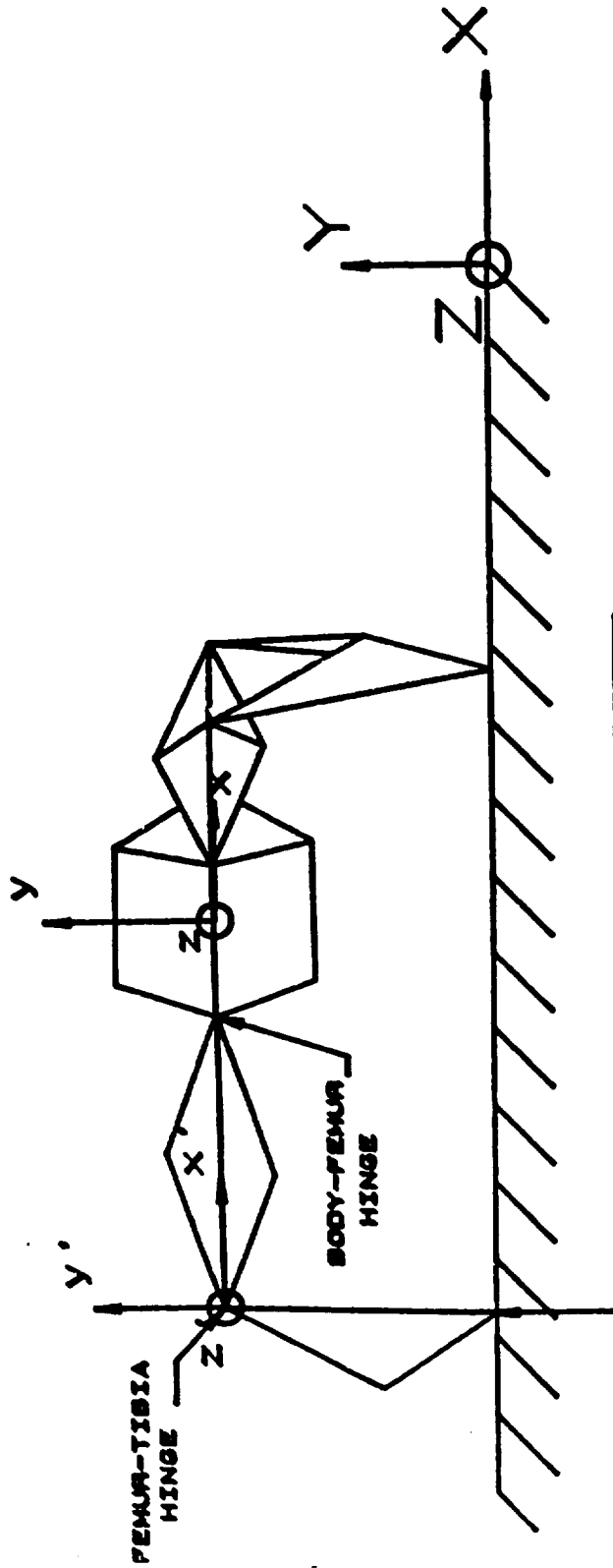
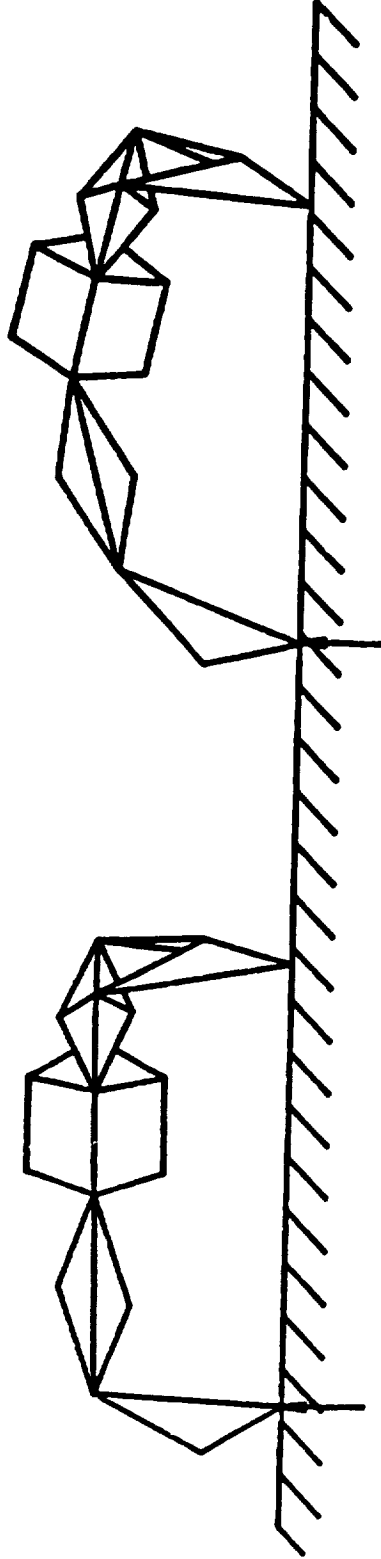


FIG. 3

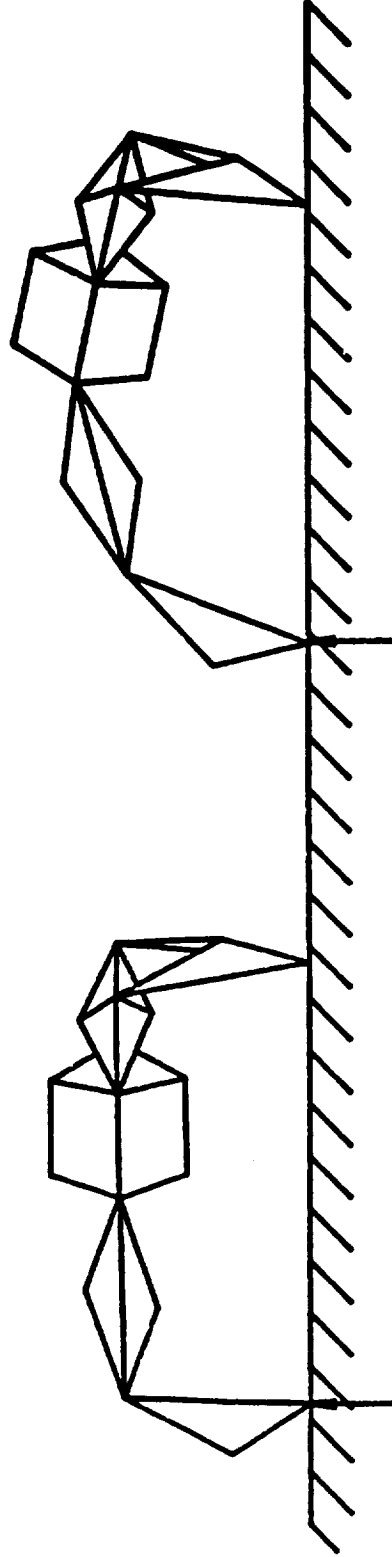
LEAN MODE



NOMINAL  
CONFIGURATION  
FIG. 4A

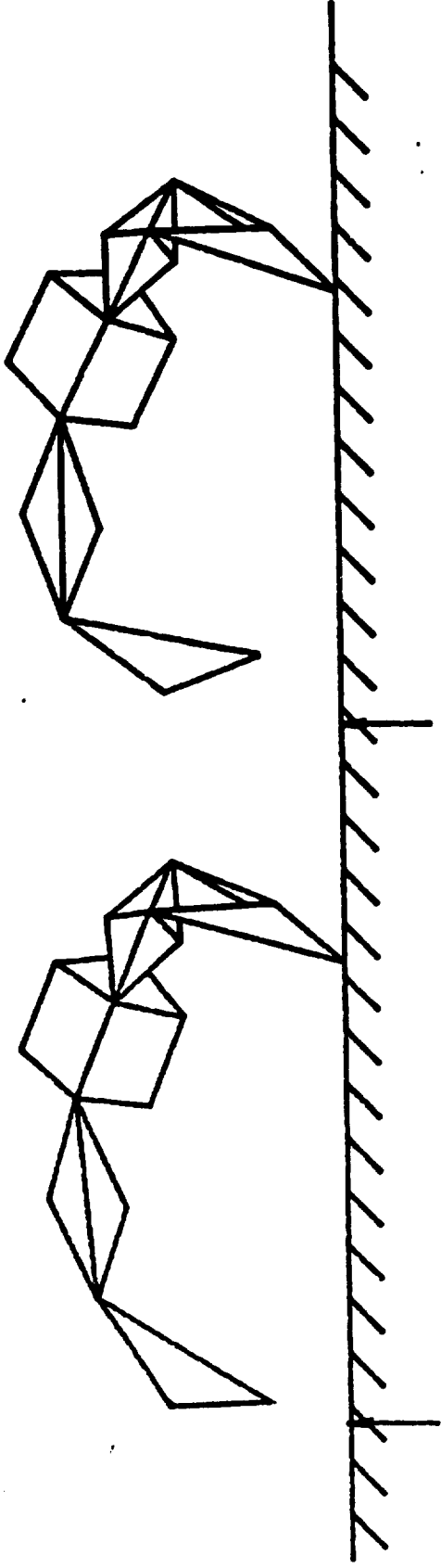
LEAN  
FIG. 4B

# CRUTCH WALK



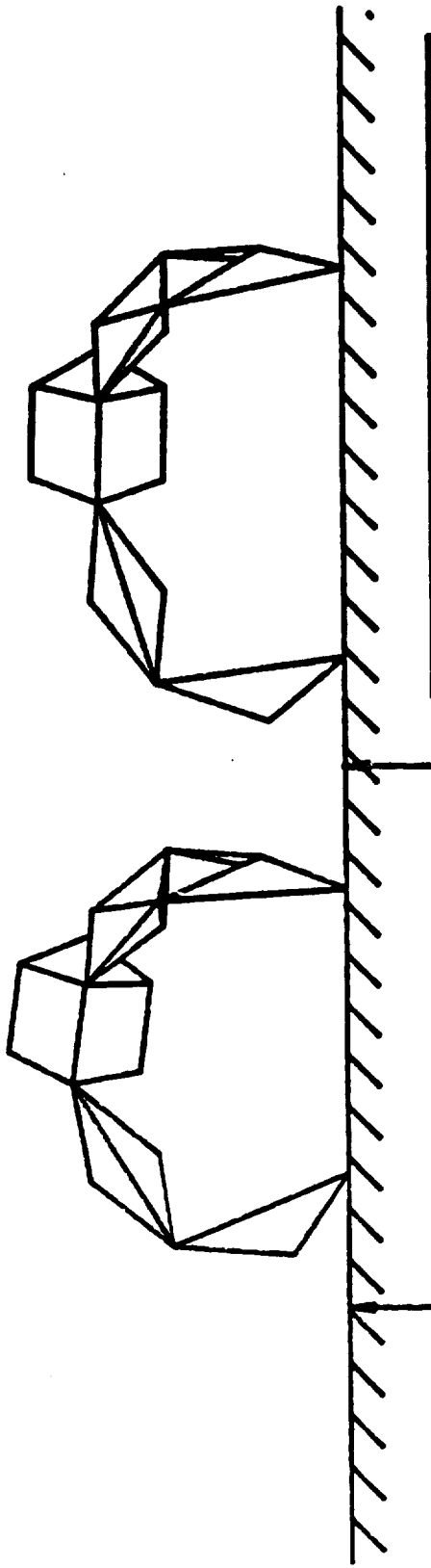
NOMINAL  
CONFIGURATION  
FIG. 5A

LEAN  
FIG. 5B



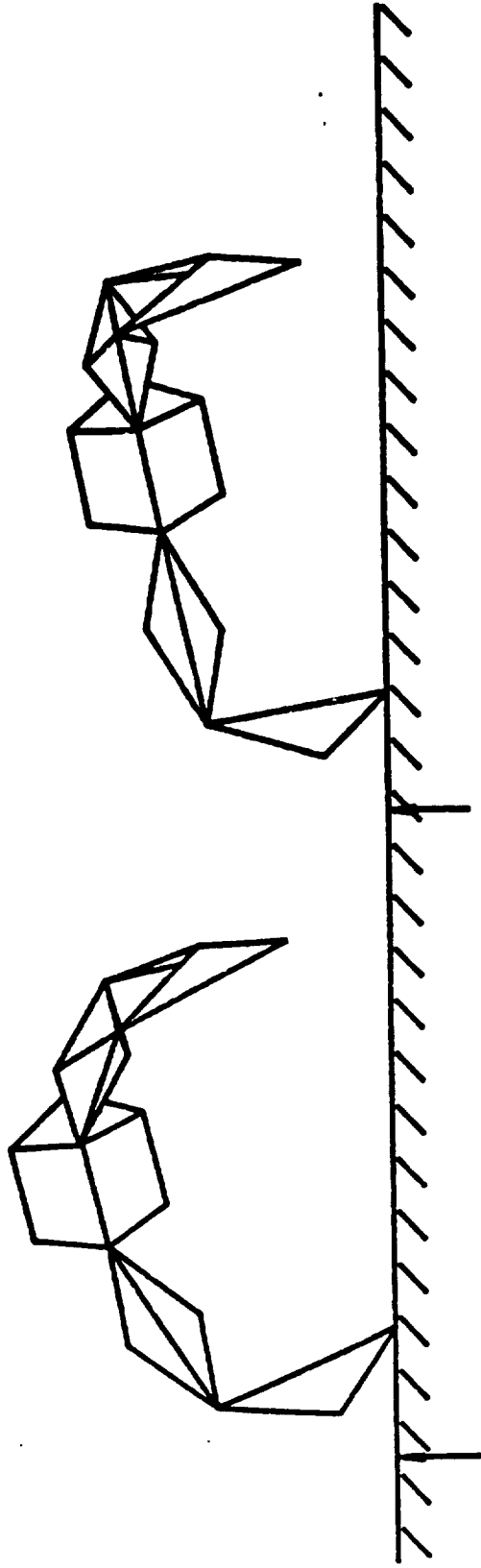
RECONFIGURE  
LEG A  
FIG. 6B

PUSH OFF FROM  
SURFACE WITH  
LEG A  
FIG. 6A



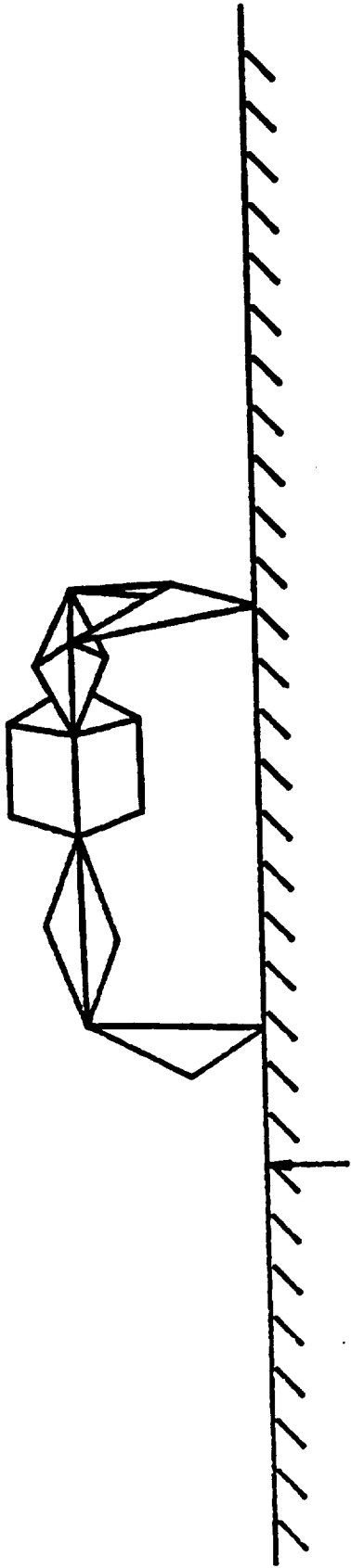
BEGIN LEAN MODE  
WITH LEGS B & C  
FIG. 7B

RETURN TO SURFACE  
FIG. 7A



RECONFIGURE  
ALL LEGS BACK  
TO NOMINAL  
CONFIGURATION  
FIG. 8B

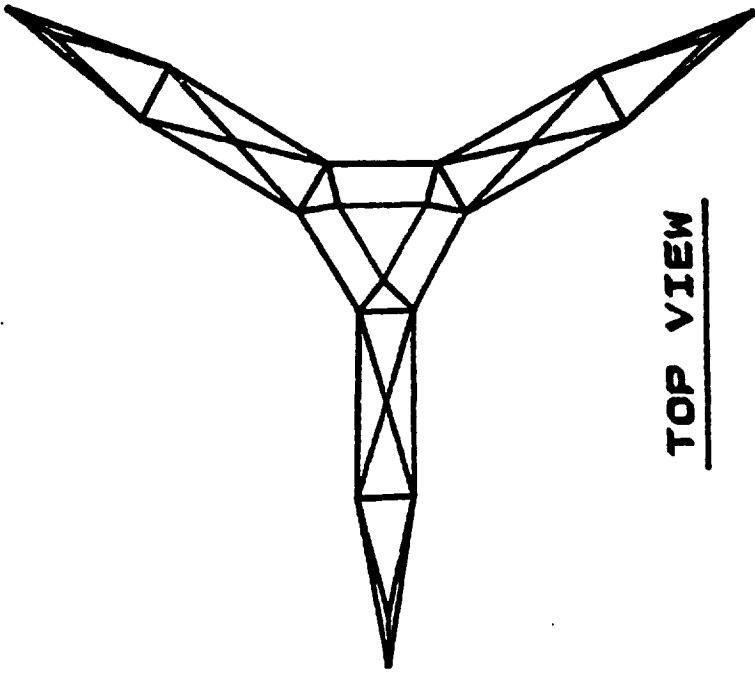
PUSH OFF FROM  
SURFACE WITH  
LEGS B & C  
FIG. 8A



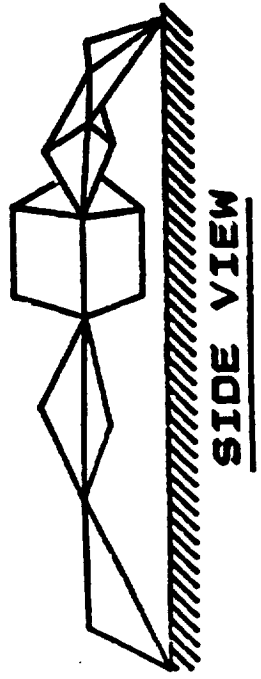
RETURN TO SURFACE  
FIG. 9



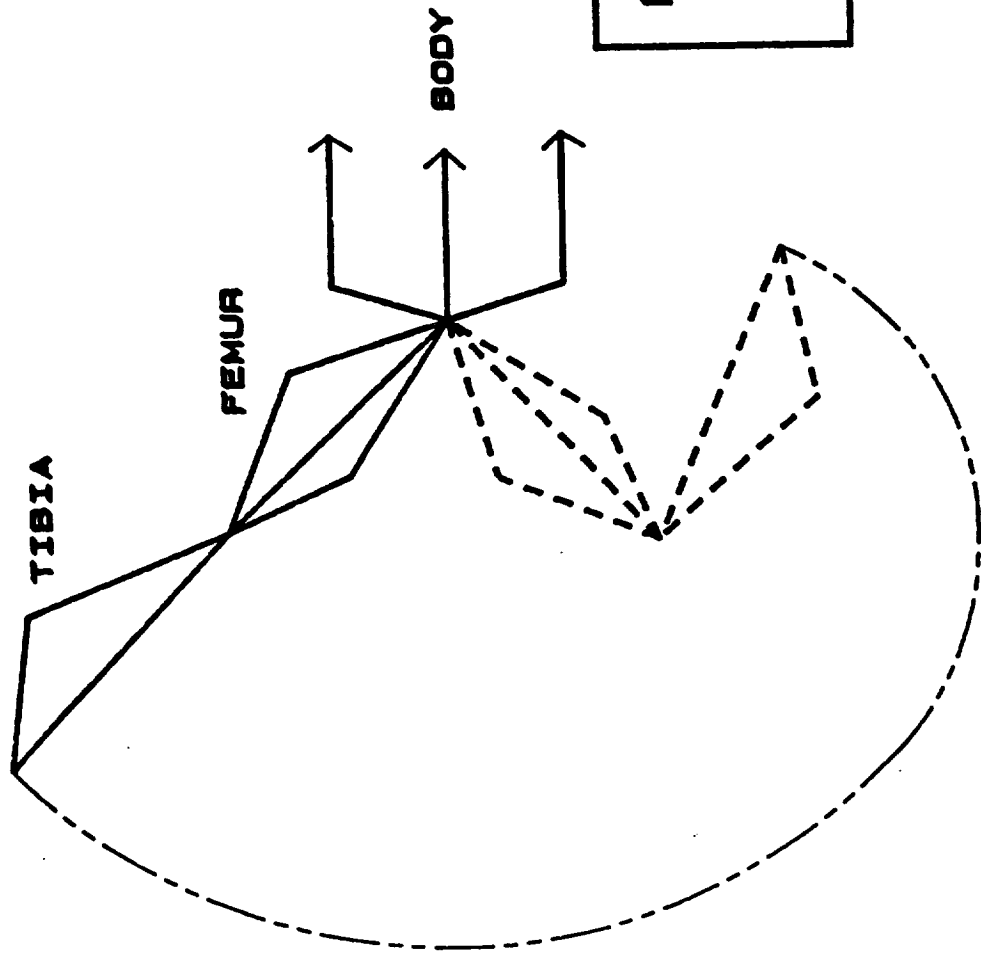
**SQUAT  
POSITION  
FIG. 10**



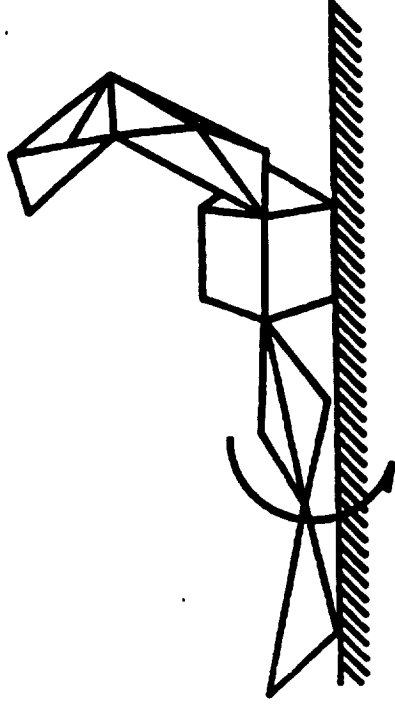
TOP VIEW



SIDE VIEW

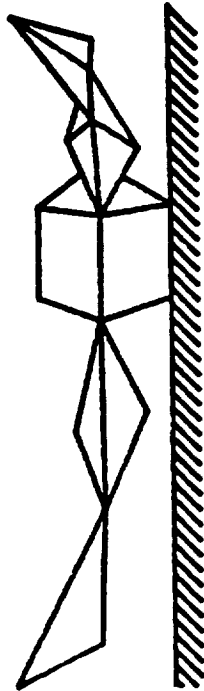


**RANGE OF MOTION  
FOR LEG  
FIG. 11**

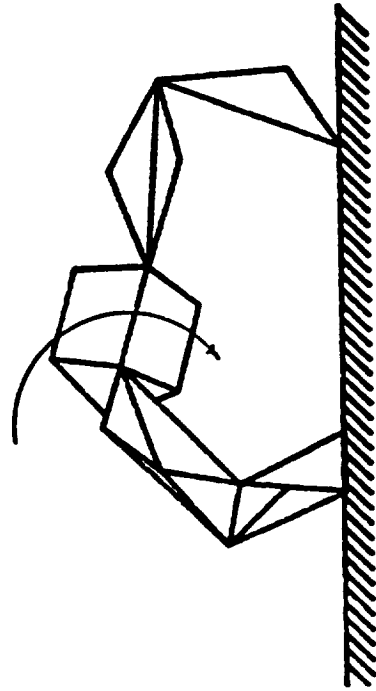


**TWO LEGS TUCK IN  
WHILE THIRD LEG  
PUSHES AWAY FROM  
SURFACE**

**SELF RIGHTING  
MODE  
FIG. 12**

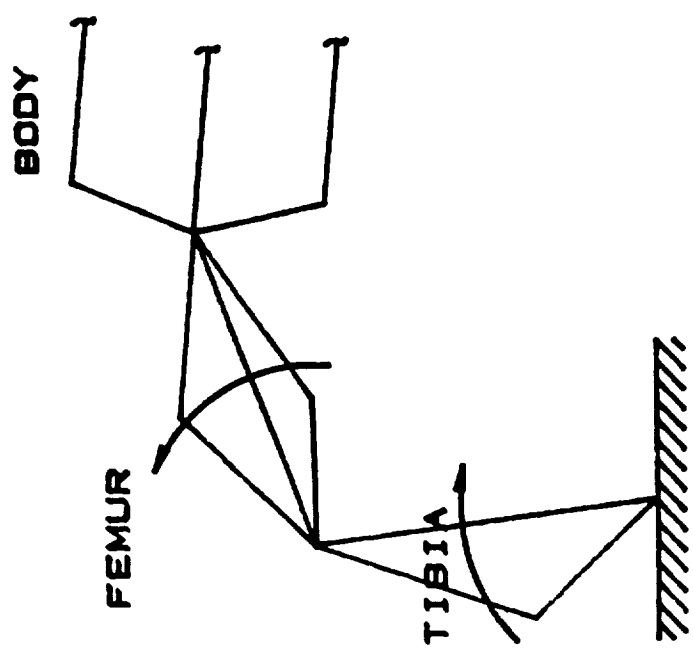
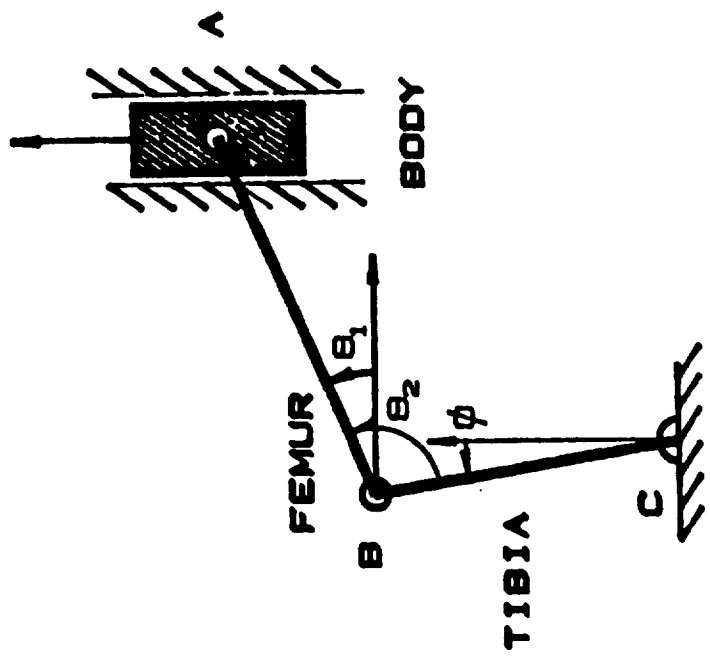


**SKITTER UPSIDE DOWN**

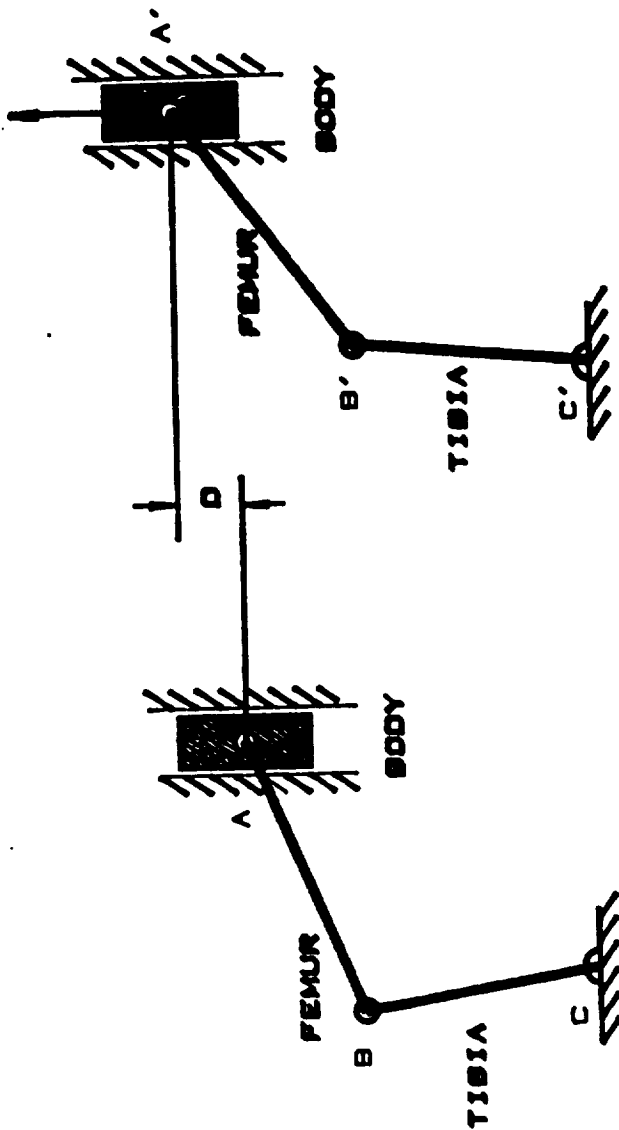


**SKITTER SUMMERSALTS  
TO RIGHT ITSELF**

DESIRED MOTION



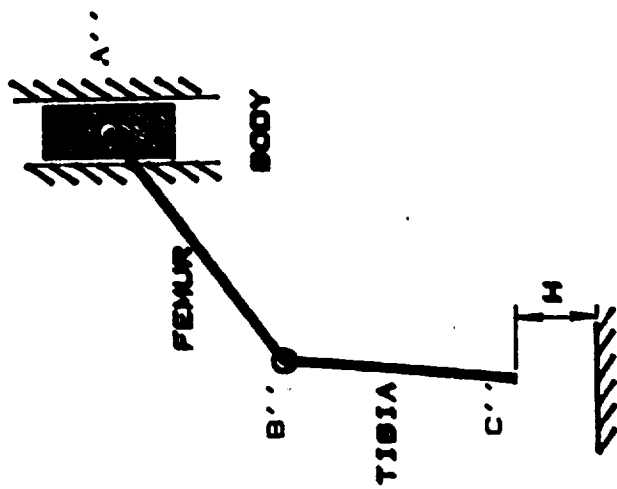
SLIDER-CRANK MECHANISM  
FOR JUMP MOTION  
VARIABLE DEFINITION  
FIG. 130



END OF ACCELERATION

BEGINNING OF JUMP

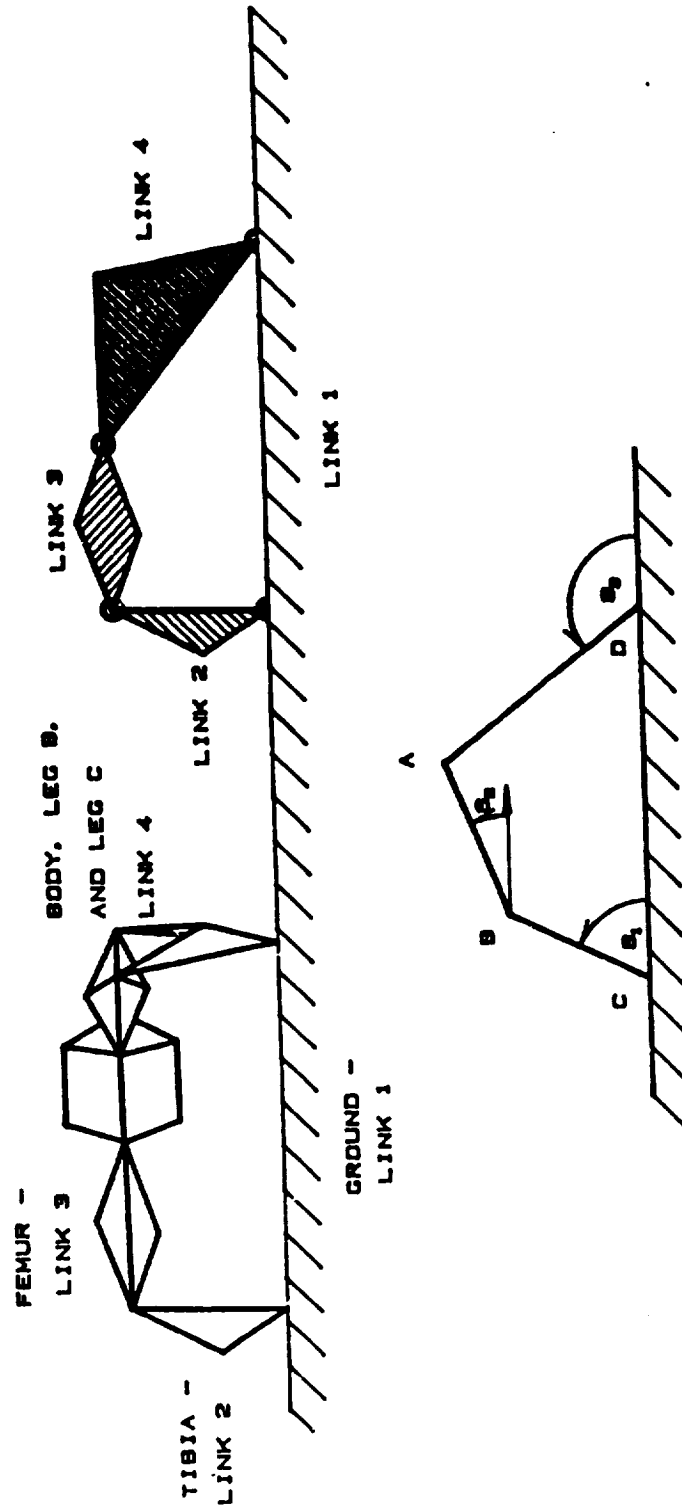
SLIDER-CRANK MECHANISM  
FOR JUMP MOTION  
VARIABLE DEFINITION  
FIG. 13b



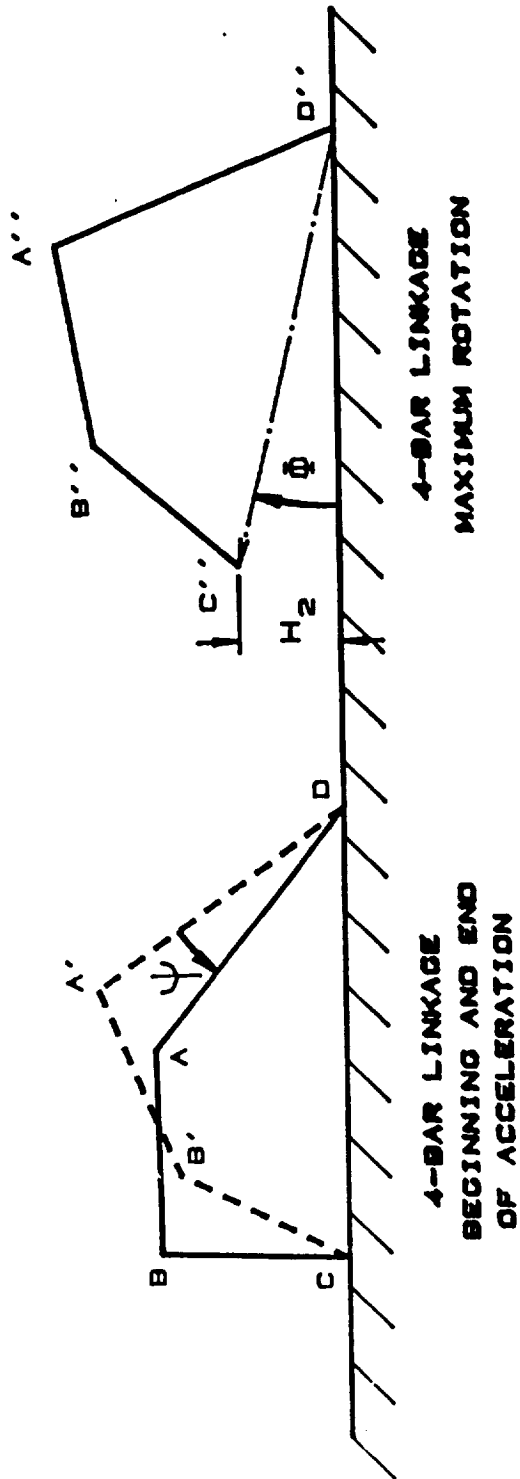
MAXIMUM HEIGHT

SLIDER-CRANK MECHANISM  
 FOR JUMP MOTION  
 VARIABLE DEFINITION

FIG. 13c



**4-BAR LINKAGE  
FOR LEANING MOTION  
VARIABLE DEFINITION  
FIG. 14a**



**4-BAR LINKAGE**  
**FOR LEANING MOTION**  
**VARIABLE DEFINITION**  
**FIG. 14b**



D  
D  
D  
D  
D  
D  
D  
D  
D  
D  
D  
D  
D  
D  
D

---

**Appendix B**  
**SKIT\_3D Graphic Simulation**  
**Figures and Program Listing**

---

## **SKIT\_3D**

### **Three Dimensional Graphic Simulation Program**

Computer graphic simulation is an excellent engineering tool for analyzing and designing spatial mechanisms. SKIT\_3D is a three dimensional graphic simulation program which allows the user to visualize SKITTER's spatial configurations by controlling system, leg segment, or actuator movements of a screen representation of the platform. User input is in the form of incremental positioning, direct positioning, or time based data files which can be used for platform animation. Output is directed to a screen, plotter, or dump device.

#### **SYSTEM REQUIREMENTS:**

**Computer:** Hewlett-Packard 200 or 300 series computer with input knob present on the key board.

**Mass Storage:** One 3.5" 720 kbyte disc drive.

**Software:** Hewlett-Packard Basic 4.0 or greater with the Knob\_20 bin loaded.

**Options:** Hewlett-Packard Graphics Language (HPGL) plotter.  
Hewlett-Packard LaserJet printer (or equivalent).

#### **LOADING THE PROGRAM:**

To load the program:

- 1) Boot the computer into the Basic System environment.
- 2) Make the disc drive the default mass storage device by using the **Mass Storage Is** command.
- 3) Insert the SKIT\_3D disc into the disc drive.
- 4) Type **load "SKIT\_3D" <cr>**.

5) Hit <RUN> key.

6) Program will begin.

### **USING THE PROGRAM:**

The Program can be divided into two different sections depending on the type of data input. The Manual Mode of operation allows the user to input data directly or via the keyboard knob to control SKITTER's movements while the Data File Mode accepts time based data files for animating platform position sequences. All movements by the platform are relative to a local coordinate system on SKITTER. Figure 3 shows the orientation of the coordinant system ( x-y-z) relative to the initial screen display.

### **MANUAL MODE**

#### **Main Menu:**

The initial display shows a Z axis view of SKITTER with the main menu appearing at the bottom of the screen. The menu items can be accessed by pressing the corresponding function keys on the key board. Definition of the keys are as follows

**SYSTEM:** Allows the user to reorient the entire platform.

**PIVOT LINES:** Allows the user to pivot the entire platform about a line constructed by any two feet of SKITTER.

**ACTUATORS:** Allows the user to reorient either a leg segment or the entire platform by engaging a particular actuator.

**EXIT:** Allows the user to exit the program.

**OUTPUT:** Allows the user to output the screen display to either a plotter or dump device.

**MOVIE:** Allows the user to enter DATA FILE Mode.

**ATTRIBUTES:** Allows the user to change views, window parameters, and output devices.

**WHAT:** Allows the user to view screen, output device, and platform parameters.

**SYSTEM :**

The SYSTEM function allows the user to rotate or translate the entire platform about or along all three axis. When the key is pressed, a new menu will appear and is defined as follows:

**Rotate X:** Rotates the platform incrementally about the X axis when the knob is turned.

**Rotate Y:** Rotates the platform incrementally about the Y axis when the knob is turned.

**Rotate Z:** Rotates the platform incrementally about the Z axis when the knob is turned.

**Rotation Angle:** Rotates the platform about the last rotation axis by a user defined angle (<cr> quits).

**Knob Increment:** Allows the user to input a new knob increment.

**Translate X:** Translates the platform incrementally along the X axis when the knob is turned.

**Translate Y:** Translates the platform incrementally along the y axis when the knob is turned.

**Translate Z:** Translates the platform incrementally along the z axis when the knob is turned.

**Trans Vector:** Translates the platform along a user defined vector (<cr> to quit)

**Main Menu:** Returns the user to the main menu.

### **Pivot Lines:**

The PIVOT LINES function allows the user to pivot the entire platform about a line constructed by two of SKITTER's feet. When the key is depressed, a new menu will appear and is defined as follows:

**Leg A:** Pivots the platform about the leg a pivot line constructed by the feet of legs b & c.

**Leg B:** Pivots the platform about the leg b pivot line constructed by the feet of legs a & c.

**Leg C:** Pivots the platform about the leg c pivot line constructed by the feet of legs a & b.

**Main Menu:** Returns the user to the main menu.

### **ACTUATORS:**

The ACTUATORS function key allows the user to reorient a leg segment or the entire platform by engaging a particular actuator. When the key is depressed a new menu appears and is defined as follows:

**Femur A:** Engages the femur a actuator and will incrementally change the femur a-body angle when the knob is turned.

**Femur B:** Engages the femur b actuator and will incrementally change the femur b-body angle when the knob is turned.

**Femur C:** Engages the femur c actuator and will incrementally change the femur c-body angle when the knob is turned.

**Fixed/Free:** A Toggle switch which allows the user to move the platform with either its legs always in contact with the ground (fixed) or unconstrained by the ground (free).

**Knob Increment:** Allows the user to define a new knob increment.

**Tibia A:** Engages the tibia A actuator and will incrementally change the tibia-femur angle of leg a when the knob is turned.

**Tibia B:** Engages the tibia b actuator and will incrementally change the tibia-femur angle of leg b when the knob is turned.

**Tibia C:** Engages the tibia c actuator and will incrementally change the tibia-femur angle of leg c when the knob is turned.

**Main Menu:** Returns the user to the main menu.

#### **OUTPUT:**

The OUTPUT function key allows the user to send the screen display to a plotter or a dump device. When the key is pressed, a new menu will appear and is defined as follows:

**Plotter:** Outputs the screen display to the designated plotter.

**Raster Dump:** Outputs the screen display to the designated dump device.

**Main Menu:** Returns the user to the main menu.

#### **ATTRIBUTES:**

The ATTRIBUTES key allows the user to define views, window parameters, and output devices. When the key is pressed, a new menu will appear and is defined as follows:

**View:** Allows the user to define a new view. When this key is depressed, a new menu will appear and is defined as follows:

**X Axis:** Changes the users view to looking down the X axis.

**Y Axis:** Changes the users view to looking down the Y axis.

**Z Axis:** Changes the users view to looking down the Z axis.

**Quit:** Returns the user to the ATTRIBUTES menu.

**Window:** Allows the user to change window parameters. When this key is pressed, a new menu will appear and is defined as follows:

**Zoom:** Allows the user to zoom in and out from the current window.

**Pan X:** Allows the user to pan horizontally.

**Pan Y:** Allows the user to pan vertically.

**Input Data:** Allows the user to input specific window coordinants.

**Quit:** Returns the user to the ATTRIBUTES menu.

**Disp Quantities:** Displays current positions and incremental changes of the entire system or body segments as they are moved.

**Dump Device:** Allows the user to specify a new dump device.

**Plotter Port:** Allows the user to specify a new plotter port.

**Main Menu:** Returns the user to the main menu.

### **WHAT:**

The WHAT function key allows the user to view the values of all parameters such as joint angles, output devices, window variables, and current view. To exit to the main menu, simply hit <cr>.

### **DATA FILE MODE**

The MOVIE key accessible on the main menu allows the user to enter DATA FILE MODE. This mode of operation accepts time based data files created previously by the user and determines the transformation matrices for each time increment. A new file is built on the disc drive named SKITWORKS which contains the SKITTER position sequence.

After the SKITWORKS file is closed the program begins animating the position sequence on the screen.

The DATA FILE mode is extremely useful for integrating output data files from kinematic or control programs with computer graphics. The user is able to vary any particular parameter, such as mass, inertia, or gravity in his application program, create a input data file, and see graphically the effects on SKITTER as it goes through a position sequence. Once a theoretical SKITTER model is complete design of the system components can begin using the model parameters (i.e. control parameters, leg lengths, weights etc.) as design constraints.

#### INPUT DATA FILE STRUCTURE

The input file structure consists of values for time, free or fixed leg segment movements, and system rotations and translations. The file can be 30 lines long and each line is arranged as follows:

TIME	FREE	FEMUR A	TIBIA A	FEMUR B	TIBIA B	FEMUR C	TIBIA C	ROT X	ROT Y	ROT Z	TRAN X	TRAN Y	TRAN Z
.2	0	3.2	5	0	0	0	0	15	0	0	0	0	0

This particular example shows that the user wishes to rotate the system around the x axis 15 degrees, move femur a 3.2 degrees in fixed mode, and tibia a 5 degrees in the fixed mode at time increment .2.

**TIME:** Time can have any value; however the value of 999 is reserved as a pointer to tell the program that it is at the end of the input data file. On the final line of the data file the user has to make the time value equal to 999.

**FREE:** If the value for free equals 0, then the foot will be fixed; however, if the value for free equals 1 then the foot is free ( see manual mode - actuators menu).

When the MOVIE keys is pressed, the user will be asked :



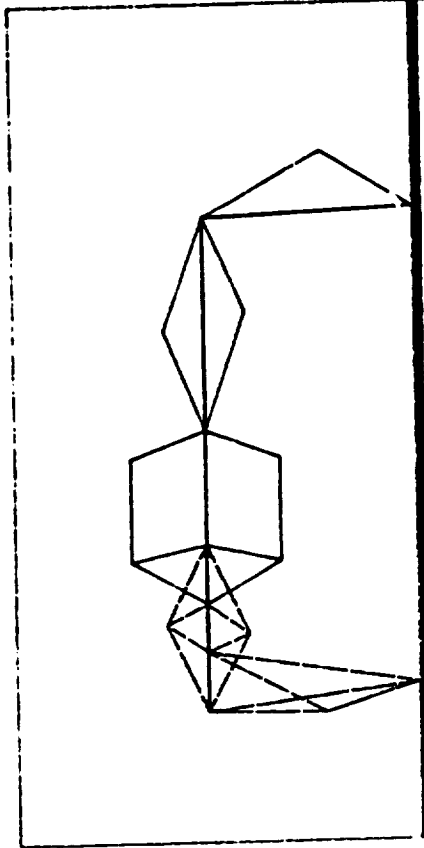
**DO YOU WANT TO RUN AN ALREADY COMPUTED FILE ? Y OR N**

If the user has run the movie function before and as saved a SKITWORKS file, he may enter Y. However, if this is the first time through for the user or a brand new input file, he should enter N.

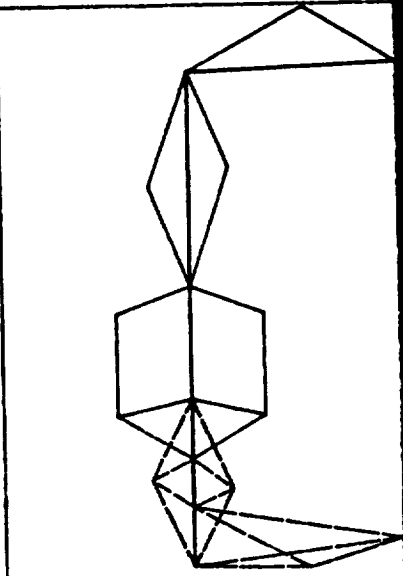
At the prompt, enter in the name of either the computed file or new input file depending on your previous answer. The program will proceed and animate the position sequence. To stop the animation sequence simply hit <cr>.

Once the animation sequence is stopped, the program will ask the user if he would like to save the SKITWORKS file as a computed file if the input data file was new. If so enter Y and enter the name of the file at the next prompt.

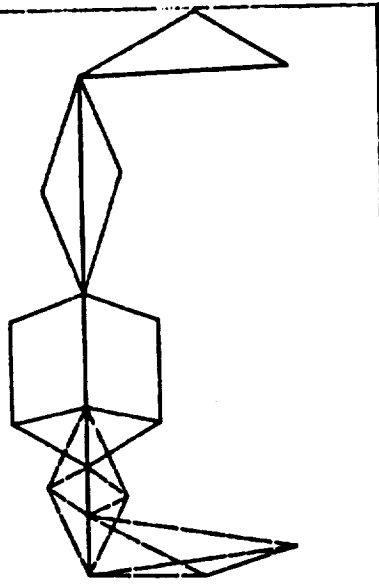
A sample input file , SHOW, and a computed file, WALK, are store on the SKIT\_3D disc an can be used to demonstrate the DATA FILE mode for the user.



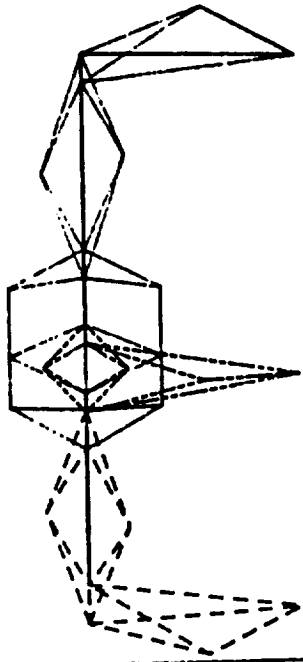
INITIAL DISPLAY



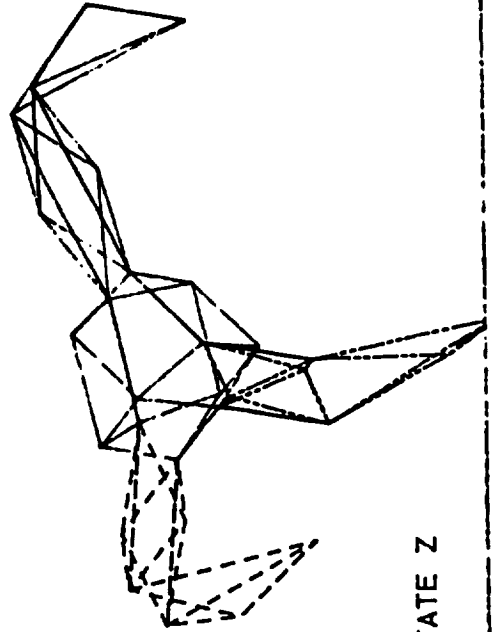
TRANSLATE X



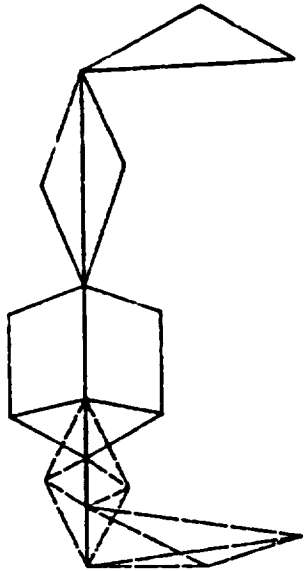
TRANSLATE Y



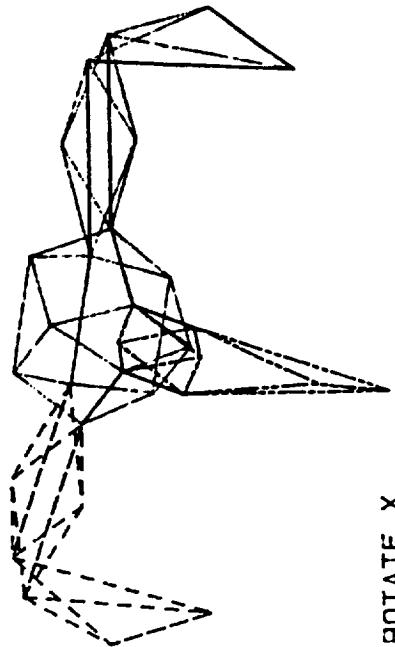
ROTATE Y



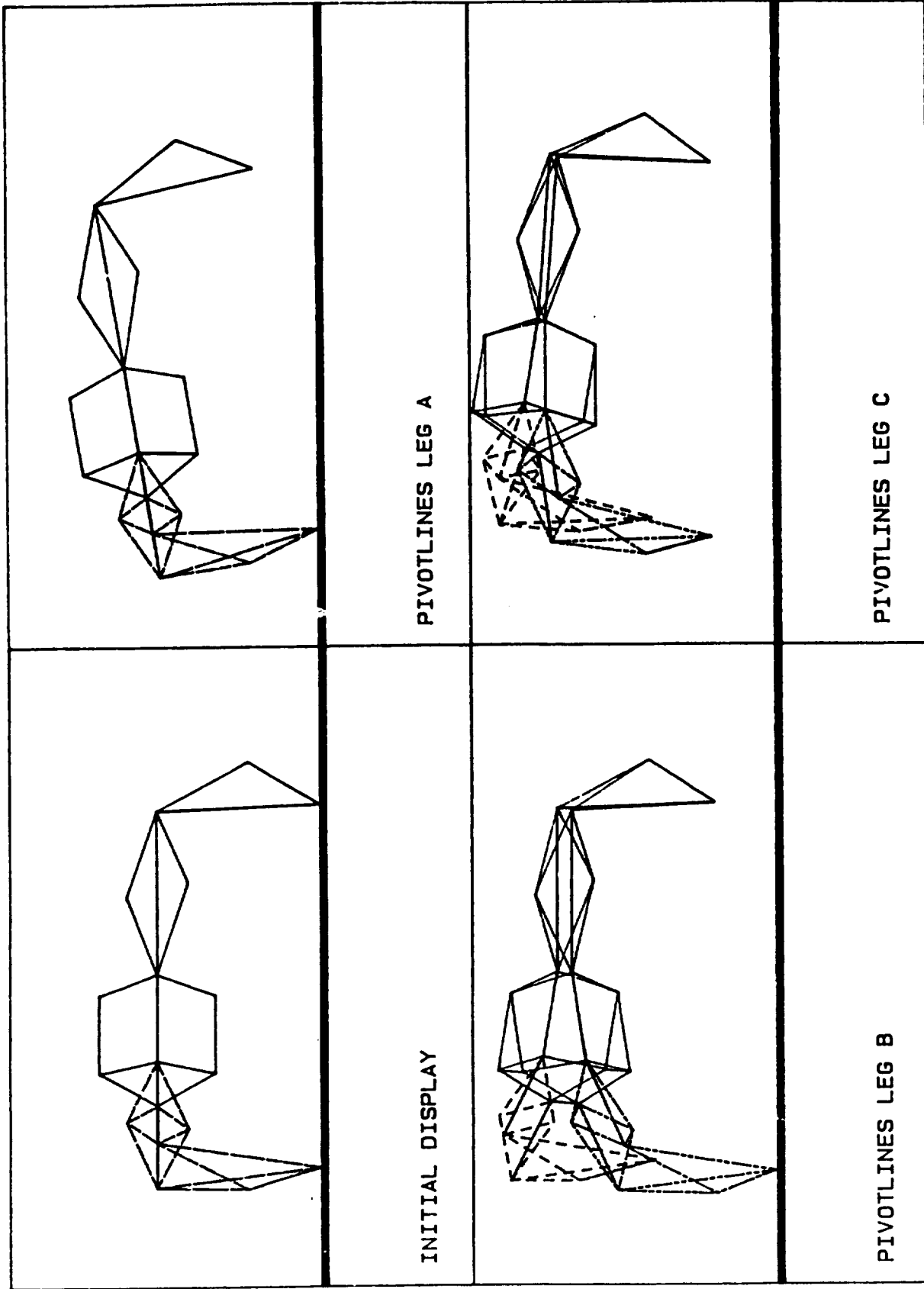
ROTATE Z

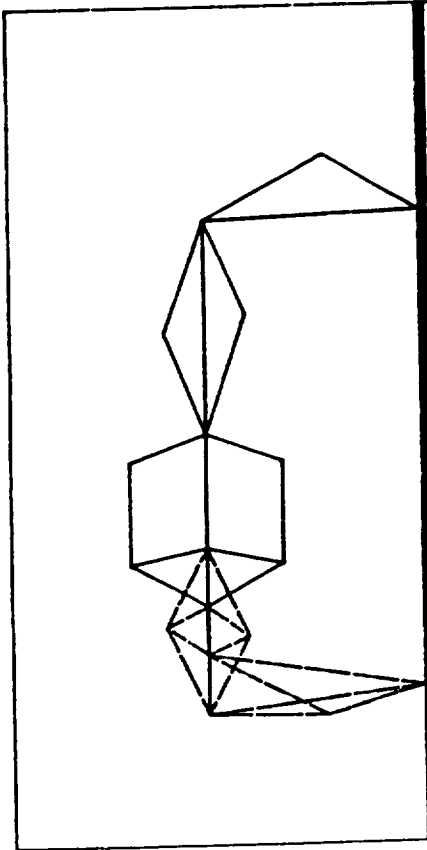


INITIAL DISPLAY

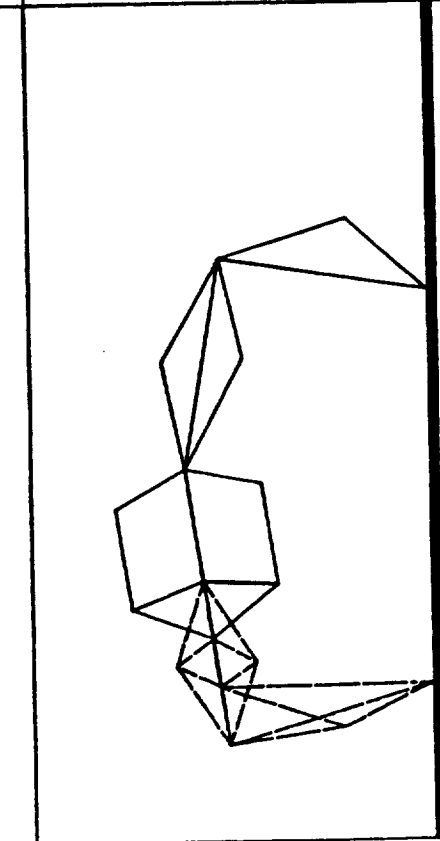


ROTATE X

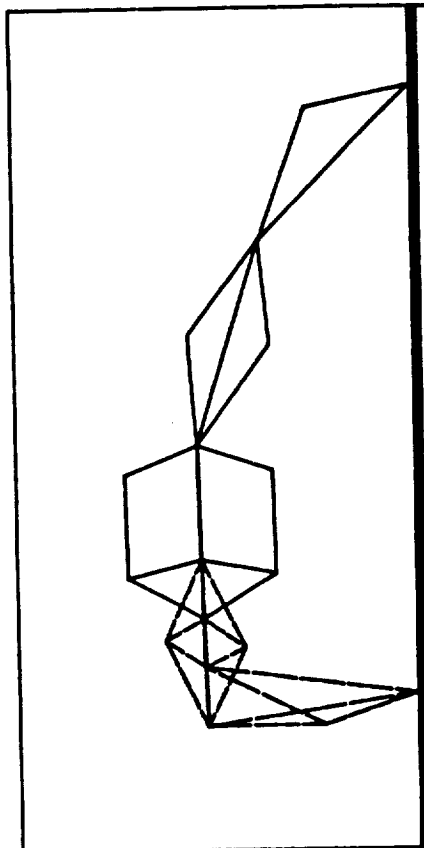




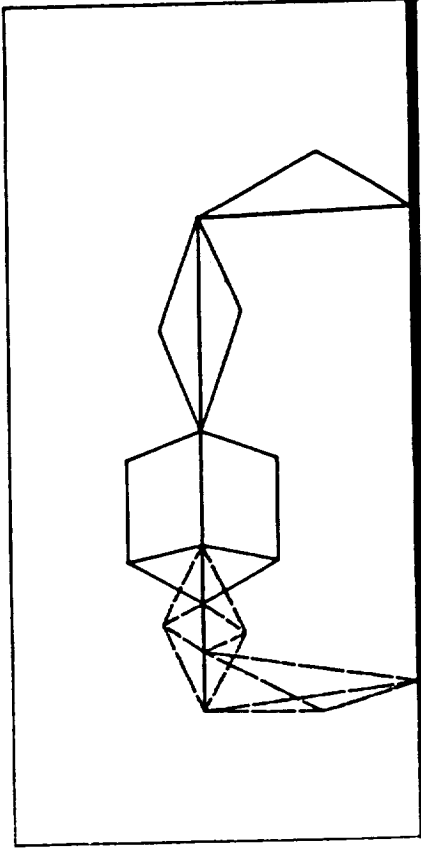
INITIAL DISPLAY



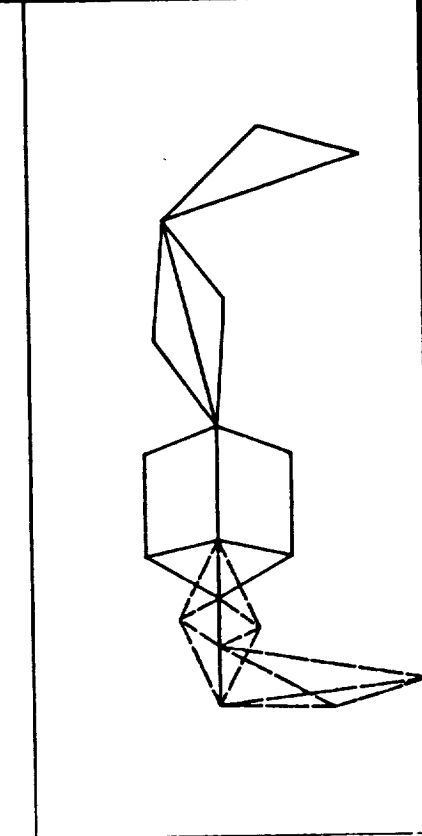
ACTUATORS FEMUR A



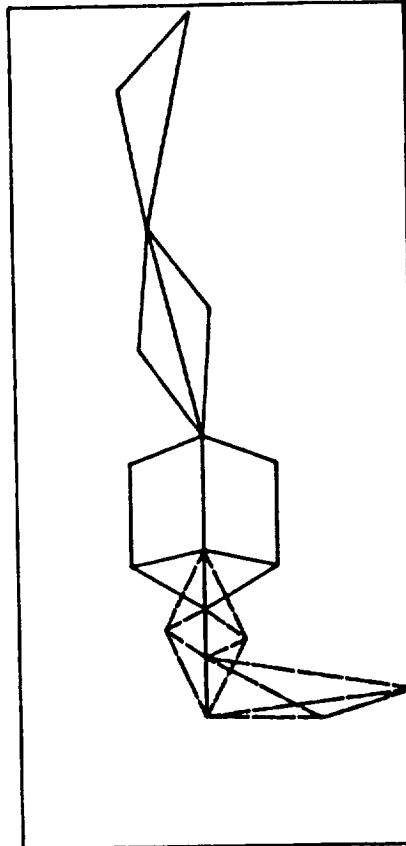
ACTUATORS TIBIA A



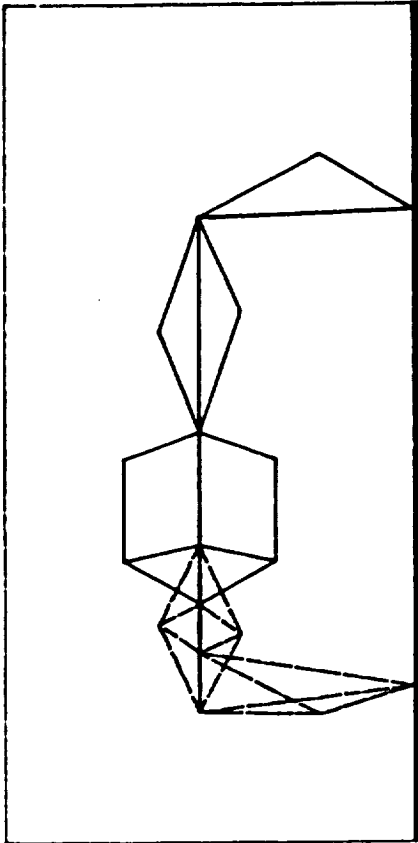
INITIAL DISPLAY



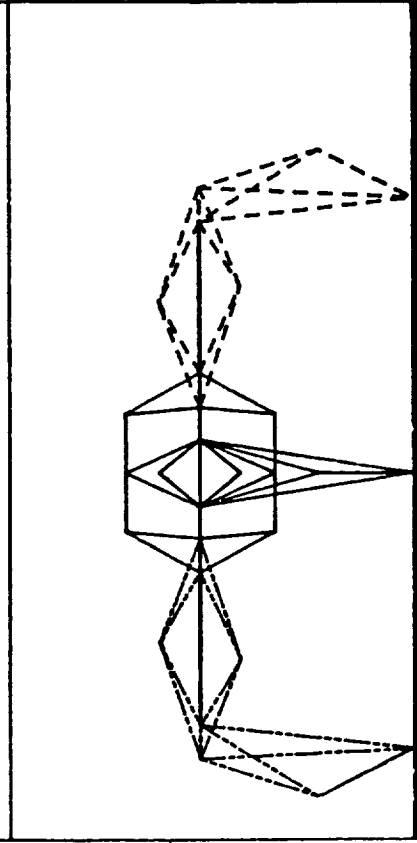
ACTUATORS FEMUR A FREE



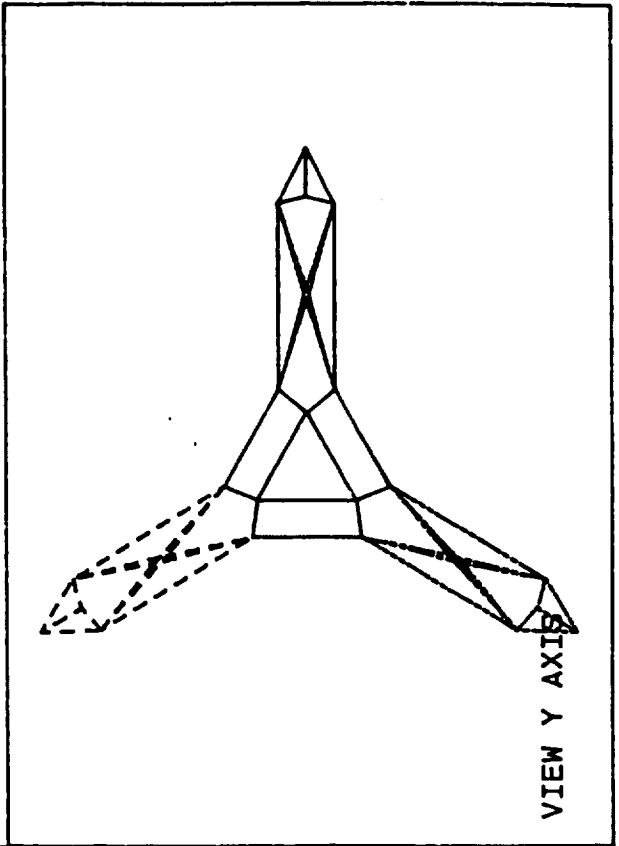
ACTUATORS TIBIA A FREE



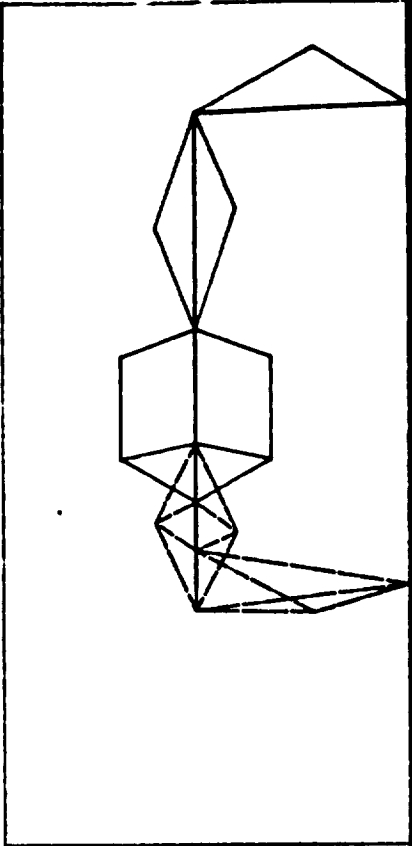
INITIAL DISPLAY Z AXIS



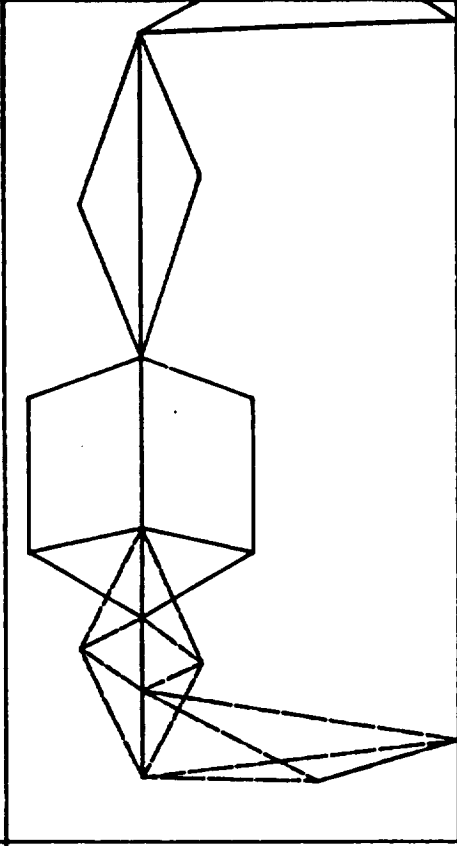
VIEW X AXIS



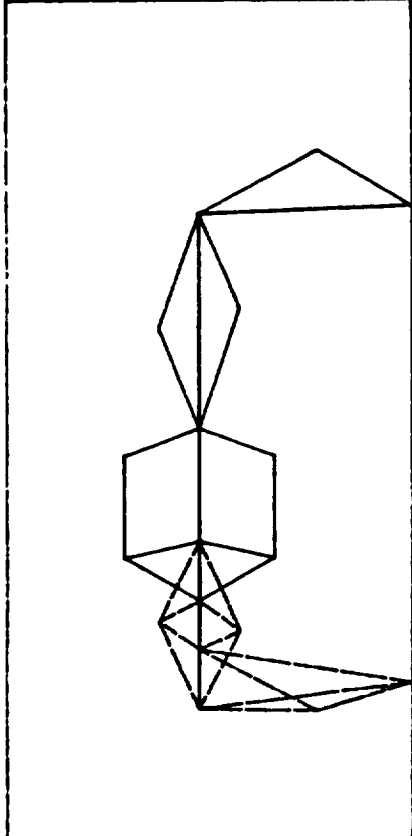
VIEW Y AXIS



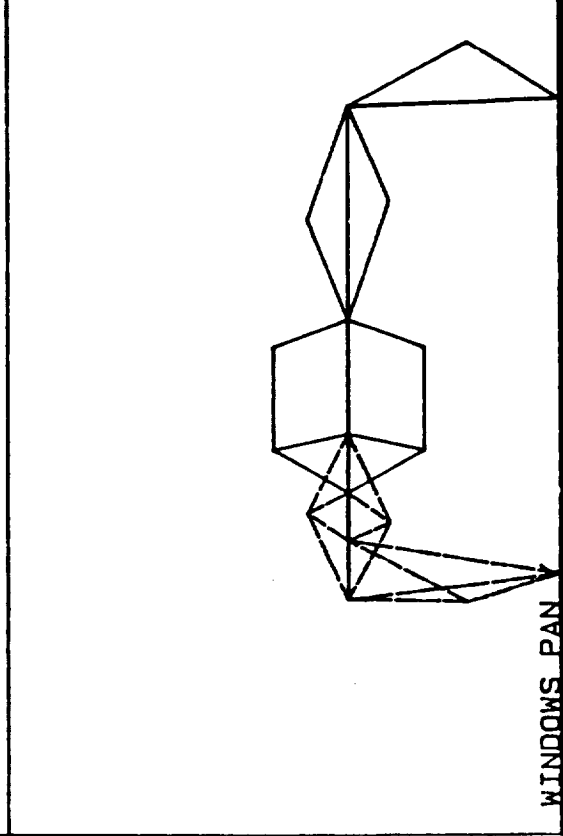
WINDOWS PAN X



WINDOWS ZOOM



INITIAL DISPLAY



WINDOWS PAN



;  
THESE ARE THE CURRENT PARAMETERS:

WINDOW:           X MIN = -20           X MAX = 40  
                  Y MIN = -20           Y MAX = 40

VIEW:   LOOKING DOWN Z AXIS

PLOTTER LOCATION: 705           PRINTER LOCATION: 9

DISPLAY VALUES IS OFF

	TRANS (in.)	ROT (deg)		FEMUR ANGLE	TIBIA ANGLE
X	0.00	0.00	A	0.00	90.00
Y	0.00	0.00	B	0.00	90.00
Z	0.00	0.00	C	0.00	90.00

PIVOT ANG A: 0   PIVOT ANG B: 0   PIVOT ANG C: 0

FORM FEED	LASER PRINTER	SCREEN	HARD DISK	DISK DRIVE
SCRATCH	LOAD ""	CATALOG DRIVE	LIST PROGRAM	RE-STORE ""

TYPICAL OUTPUT FROM 'WHAT' COMMAND

```

10 REM*****
10 REM
10 REM          SKIT_3D  V.10
10 REM          THREE DIMENSIONAL GRAPHIC SIMULATION PROGRAM
10 REM
10 REM          WRITTEN BY:
10 REM          BRICE K. MACLAREN
10 REM          GARY V. MCMURRAY
10 REM          06/23/88
10 REM
10 REM          THIS PROGRAM WILL ACCURATELY DEPICT THE SKITTER MOBILE PLATFORM
10 REM          AND THE MOTIONS THAT IT IS ABLE TO ACHIEVE BY USING EITHER
10 REM          MANUAL OR DATA FILE INPUT. USE FUNCTION KEYS AND KNOB FOR INPUT.
10 REM
10 REM          HARDWARE:  HEWLETT PACKARD 200/300 SERIES COMPUTER
10 REM          3.5' DISC DRIVE
10 REM          KEYBOARD WITH KNOB
10 REM          SOFTWARE:  HEWLETT-PACKARD BASIC 4.0 WITH KNOB_20 BIN LOADED
10 REM          OPTIONS:  HEWLETT-PACKARD (HPGL) PLOTTER
10 REM          PRINTER
10 REM
10 REM          *****
10 REM
10 REM          *****
10 REM          SKITTER DATA FILE: SEE MANUAL FOR DESCRIPTION OF BODY POINTS
10 REM          *****
10 DATA      0,0,0,10          !          UPPER BODY
10 DATA      6.5482,28.4912,0,-2          !          A
10 DATA      9.1602,20.9817,3.2283,-1          !          H
10 DATA      -1.7468,20.9817,9.5688,-1          !          I
10 DATA      -3.2741,28.4912,5.6709,-1          !          B
10 DATA      0,0,0,7
10 !
10 DATA      0,0,0,10
10 DATA      -3.2741,28.4912,5.6709,-2          !          B
10 DATA      -7.3759,20.9817,6.3188,-1          !          J
10 DATA      -7.3759,20.9817,-6.3188,-1          !          K
10 DATA      -3.2741,28.4912,-5.6709,-1          !          C
10 DATA      0,0,0,7
10 !
10 DATA      0,0,0,10
10 DATA      -3.2741,28.4912,-5.6709,-2          !          C
10 DATA      -1.7468,20.9817,-9.5688,-1          !          L
10 DATA      9.1602,20.9817,-3.2283,-1          !          G
10 DATA      6.5482,28.4912,0,-1          !          A
10 DATA      0,0,0,7
10 !
10 DATA      0,0,0,10          !          BOTTOM BODY
10 DATA      6.5482,13.4722,0,-2          !          D
10 DATA      9.1602,20.9817,3.2283,-1          !          H
10 DATA      -1.7468,20.9817,9.5688,-1          !          I
10 DATA      -3.2741,13.4722,5.6709,-1          !          E
10 DATA      0,0,0,7
10 !
10 DATA      0,0,0,10
10 DATA      -3.2741,13.4722,5.6709,-2          !          E
10 DATA      -7.3759,20.9817,6.3188,-1          !          J
10 DATA      -7.3759,20.9817,-6.3188,-1          !          K

```

480	DATA	-3.2741,13.4722,-5.6709,-1	!	F	
490	DATA	0,0,0,7			
500	!				
510	DATA	0,0,0,10			
520	DATA	-3.2741,13.4722,-5.6709,-2	!	F	
530	DATA	-1.7468,20.9817,-9.5688,-1	!	L	
540	DATA	9.1602,20.9817,-3.2283,-1	!	G	
550	DATA	6.5482,13.4722,0,-1	!	D	
560	DATA	0,0,0,7			
570	!				
580	DATA	0,0,0,4			! RESERVED FOR PEN
590	DATA	0,0,0,10	!		FEMUR ONE
600	DATA	9.1602,20.9817,-3.2283,-2	!	G	
610	DATA	29.6602,20.9817,-3.2283,-1	!	M	
620	DATA	18.7362,25.0731,0,-1	!	O	
630	DATA	0,0,0,7			
640	!				
650	!				
660	DATA	0,0,0,10			
670	DATA	29.6602,20.9817,3.2283,-2	!	N	
680	DATA	18.7362,25.0731,0,-1	!	O	
690	DATA	9.1602,20.9817,3.2283,-1	!	H	
700	DATA	0,0,0,7			
710	!				
720	DATA	0,0,0,10			
730	DATA	9.1602,20.9817,-3.2283,-2	!	G	
740	DATA	29.6602,20.9817,-3.2283,-1	!	M	
750	DATA	20.571,16.8903,0,-1	!	P	
760	DATA	0,0,0,7			
770	!				
780	DATA	0,0,0,10			
790	DATA	29.6602,20.9817,3.2283,-2	!	N	
800	DATA	20.571,16.8903,0,-1	!	P	
810	DATA	9.1602,20.9817,3.2283,-1	!	H	
820	DATA	0,0,0,7			
830	!				
840	DATA	0,0,0,10	!		TIBIA ONE
850	DATA	29.6602,20.9817,-3.2283,-2	!	M	
860	DATA	30.4255,0,0,-1	!	A'	
870	DATA	35.7901,9.2132,0,-1	!	B'	
880	DATA	0,0,0,7			
890	!				
900	DATA	0,0,0,10			
910	DATA	29.6602,20.9817,3.2283,-2	!	N	
920	DATA	30.4255,0,0,-1	!	A'	
930	DATA	35.7901,9.2132,0,-1	!	B'	
940	DATA	0,0,0,7			
950	!				
960	DATA	0,0,0,4	!		RESERVED FOR PEN
970	DATA	0,0,0,10	!		FEMUR TWO
980	DATA	-1.7468,20.9817,9.5688,-2	!	I	
990	DATA	-9.3493,25.0731,16.2368,-1	!	S	
1000	DATA	-11.9968,20.9817,27.3223,-1	!	Q	
1010	DATA	0,0,0,7			
1020	!				
1030	DATA	0,0,0,10			
1040	DATA	-17.6259,20.9817,24.0723,-2	!	R	
1050	DATA	-9.3493,25.0731,16.2368,-1	!	S	
1060	DATA	-7.3759,20.9817,6.3188,-1	!	J	
1070	DATA	0,0,0,7			

1080	!				
1090	DATA	0,0,0,10			
1100	DATA	-1.7468,20.9817,9.5688,-2	!	I	
1110	DATA	-10.2667,16.8903,17.8258,-1	!	T	
1120	DATA	-11.9968,20.9817,27.3223,-1	!	Q	
1130	DATA	0,0,0,7			
1140	!				
1150	DATA	0,0,0,10			
1160	DATA	-17.6259,20.9817,24.0723,-2	!	R	
1170	DATA	-10.2667,16.8903,17.8258,-1	!	T	
1180	DATA	-7.3759,20.9817,6.3188,-1	!	J	
1190	DATA	0,0,0,7			
1200	!				
1210	!				
1220	DATA	0,0,0,10	!		
1230	DATA	-11.9968,20.9817,27.3223,-2	!	Q	
1240	DATA	-15.1940,0,26.3601,-1	!	C'	
1250	DATA	-17.8763,9.2132,31.0059,-1	!	D'	
1260	DATA	0,0,0,7			
1270	!				
1280	DATA	0,0,0,10			
1290	DATA	-17.6259,20.9817,24.0723,-2	!	N	
1300	DATA	-15.1940,0,26.3601,-1	!	C'	
1310	DATA	-17.8763,9.2132,31.0059,-1	!	D'	
1320	DATA	0,0,0,7			
1330	!				
1340	DATA	0,0,0,4	!	RESERVED FOR PEN	
1350	DATA	0,0,0,10	!		
1360	DATA	-7.3759,20.9817,-6.3188,-2	!	K	
1370	DATA	-9.3493,25.0731,-16.2368,-1	!	W	
1380	DATA	-17.6259,20.9817,-24.0723,-1	!	U	
1390	DATA	0,0,0,7			
1400	!				
1410	DATA	0,0,0,10			
1420	DATA	-11.9968,20.9817,-27.3223,-2	!	V	
1430	DATA	-9.3493,25.0731,-16.2368,-1	!	W	
1440	DATA	-1.7468,20.9817,-9.5688,-1	!	L	
1450	DATA	0,0,0,7			
1460	!				
1470	DATA	0,0,0,10			
1480	DATA	-7.3759,20.9817,-6.3188,-2	!	K	
1490	DATA	-10.2667,16.8903,-17.8258,-1	!	X	
1500	DATA	-17.6259,20.9817,-24.0723,-1	!	U	
1510	DATA	0,0,0,7			
1520	!				
1530	!				
1540	DATA	0,0,0,10			
1550	DATA	-11.9968,20.9817,-27.3223,-2	!	V	
1560	DATA	-10.2667,16.8903,-17.8258,-1	!	X	
1570	DATA	-1.7468,20.9817,-9.5688,-1	!	L	
1580	DATA	0,0,0,7			
1590	!				
1600	DATA	0,0,0,10	!		
1610	DATA	-17.6259,20.9817,-24.0723,-2	!	U	
1620	DATA	-15.1940,0,-26.3601,-1	!	E'	
1630	DATA	-17.8763,9.2132,-31.0059,-1	!	F'	
1640	DATA	0,0,0,7			
1650	!				
1660	DATA	0,0,0,10			
1670	DATA	-11.9968,20.9817,-27.3223,-2	!	V	

TIBIA TWO

FEMUR THREE

TIBIA THREE

```

1680 DATA      -15.1940,0,-26.3601,-1      ! E'
1690 DATA      -17.8763,9.2132,-31.0059,-1 ! F'
1700 DATA      0,0,0,7
1710 REM
1720 !
1730 !
1731 REM*****
1732 REM
1733 REM          MAIN PROGRAM
1734 REM
1735 REM*****
1740 OPTION BASE 1
1750 REAL Skitter(129,4),Newskit(129,3)      ! DEFINE VAR
1760 REAL Trans(4,4),Temp(129,4),Tempa(129,4)!TRANSFORMATION MATRIX
1770 REAL Total(4,4),Skitmod(129,4)        !TOTAL TRANFORM MATRIX
1780 REAL Femur(31,4),Femurmod(31,4),Femurtemp(31,4)
1790 REAL Tibia(10,4),Tibiatemp(10,4),Tibiamod(10,4)
1800 GOSUB Init                               ! INITIALIZATION ROUTINE
1810 CALL Display_skit(Skitter(*),Newskit(*),Screen_x,Screen_y)
1820                                           ! DRAW SKITTER
1821 REM*****
1822 REM
1823 REM          MENU SELECTION AND KEY DEFINITION
1824 REM
1825 REM*****
1830 Menu:                                     ! MENU SELECT
1840 SELECT Menu$
1850 !
1860 CASE "MAIN"
1870 ON KEY 0 LABEL "SYSTEM" GOTO System
1880 ON KEY 2 LABEL "ACTATORS" GOTO Actuator
1890 ON KEY 5 LABEL "OUTPUT" GOTO Output
1900 ON KEY 7 LABEL "ATTRIBUTES" GOTO Attributes
1910 ON KEY 9 LABEL "WHAT ??" GOTO What
1920 ON KEY 4 LABEL "EXIT" GOTO Finished
1930 ON KEY 1 LABEL "PIVOT LINES" GOTO Pivotlines
1940 ON KEY 6 LABEL "MOVIE" GOTO Movie
1950 ON KEY 3 LABEL "" GOTO Main
1960 ON KEY 8 LABEL "" GOTO Main
1970 GOTO 1970
1980 CASE "PIVOTLINES"
1990 ON KNOB .55 GOTO Knob_isr
2000 ON KEY 0 LABEL "LEG A" GOSUB Pivotlega
2010 ON KEY 1 LABEL "" GOTO Pivotlines
2020 ON KEY 2 LABEL "LEG B" GOSUB Pivotlegb
2030 ON KEY 3 LABEL "" GOTO Pivotlines
2040 ON KEY 4 LABEL "LEG C" GOSUB Pivotlegc
2050 ON KEY 5 LABEL "" GOTO Pivotlines
2060 ON KEY 6 LABEL "" GOTO Pivotlines
2070 ON KEY 7 LABEL "" GOTO Pivotlines
2080 ON KEY 8 LABEL "" GOTO Pivotlines
2090 ON KEY 9 LABEL "MAIN MENU" GOTO Main
2100 GOTO 2100
2110 CASE "SYSTEM"
2120 ON KNOB .2 GOTO Knob_isr
2130 ON KEY 1 LABEL "ROTATE Y" GOSUB Rot_y
2140 ON KEY 0 LABEL "ROTATE X" GOSUB Rot_x
2150 ON KEY 2 LABEL "ROTATE Z" GOSUB Rot_z
2160 ON KEY 5 LABEL "TRANSLATE X" GOSUB Trans_x
2170 ON KEY 6 LABEL "TRANSLATE Y" GOSUB Trans_y

```

```

2180      ON KEY 7 LABEL "TRANSLATE Z" GOSUB Trans_z
2190      ON KEY 9 LABEL "MAIN MENU" GOTO Main
2200      ON KEY 8 LABEL "TRANS VECTOR" GOSUB Vector
2210      ON KEY 3 LABEL "ROTATION ANGLE" GOSUB Angle
2220      ON KEY 4 LABEL "KNB INCREMENT" GOTO Increment
2230      GOTO 2230
2240 !
2250      CASE "ATTRIBUTES"
2260          ON KEY 0 LABEL "VIEWS" GOTO Windowpane
2270          ON KEY 9 LABEL "MAIN" GOTO Main
2280          ON KEY 1 LABEL "" GOTO Attributes
2290          ON KEY 2 LABEL "WINDOW" GOTO Windows
2300          ON KEY 3 LABEL "" GOTO Attributes
2310          ON KEY 4 LABEL "DISP QUANTITY" GOTO Print_flag
2320          ON KEY 5 LABEL "DUMP DEVICE" GOTO Printer
2330          ON KEY 6 LABEL "PLOTTER PORT" GOTO Plotter
2340          ON KEY 7 LABEL "" GOTO Attributes
2350          ON KEY 8 LABEL "" GOTO Attributes
2360          GOTO 2360
2370 !
2380 !
2390      CASE "OUTPUT"
2400          ON KEY 0 LABEL "PLOT" GOTO Plot
2410          ON KEY 1 LABEL "" GOTO Output
2420          ON KEY 2 LABEL "RASTER DUMP" GOTO Dump
2430          ON KEY 3 LABEL "" GOTO Output
2440          ON KEY 4 LABEL "" GOTO Output
2450          ON KEY 5 LABEL "" GOTO Output
2460          ON KEY 6 LABEL "" GOTO Output
2470          ON KEY 7 LABEL "" GOTO Output
2480          ON KEY 8 LABEL "" GOTO Output
2490          ON KEY 9 LABEL "MAIN MENU" GOTO Main
2500          GOTO 2500
2510 !
2520      CASE "ACTUATOR"
2530          ON KNOB .21 GOTO Knob_leg_isr
2540          ON KEY 0 LABEL "FEMUR A" GOTO Femur_a
2550          ON KEY 1 LABEL "FEMUR B" GOTO Femur_b
2560          ON KEY 2 LABEL "FEMUR C" GOTO Femur_c
2570          ON KEY 5 LABEL "TIBIA A" GOTO Tibia_a
2580          ON KEY 6 LABEL "TIBIA B" GOTO Tibia_b
2590          ON KEY 7 LABEL "TIBIA C" GOTO Tibia_c
2600          ON KEY 8 LABEL "" GOTO Menu
2610          ON KEY 3 LABEL Freeleg$ GOTO Free_leg
2620          ON KEY 4 LABEL "KNB INCREMENT" GOTO Increment
2630          ON KEY 9 LABEL "MAIN MENU" GOTO Main
2640          GOTO 2640
2650      END SELECT
2660 !
2670 !
2671 REM*****
2672 REM
2673 REM      KNOB_ISR: KNOB INTERUPT SERVICE ROUTINE.  ON KNOB ROTATION, THE
2674 REM      APPROPRIATE FUNTION WILL BE CARRIED OUT BY THE KNOB INCREMENT
2675 REM      AMOUNT
2676 REM
2677 REM*****
2680 Knob_isr:      !
2690                  Theta=Increment*SGN(KNOBX)
2700                  SELECT Twirl$

```

```

2701 REM***** SYSTEM ROTATION *****
2702 !
2710 CASE "ROTATE Y"
2720 Theta=-Theta
2730 CALL Rotate_y(Skitter(*),Skitmod(*),Total(*),Trans(*),Temp(*),Tempa(*),Sys_rot_y,Theta,Printflag$)
2731 !
2740 CASE "ROTATE X"
2750 CALL Rotate_x(Skitter(*),Skitmod(*),Total(*),Trans(*),Temp(*),Tempa(*),Sys_rot_x,Theta,Printflag$)
2760 !
2770 CASE "ROTATE Z"
2780 CALL Rotate_z(Skitter(*),Skitmod(*),Total(*),Trans(*),Temp(*),Tempa(*),Sys_rot_z,Theta,Printflag$)
2790 !
2791 REM***** SYSTEM TRANSLATION *****
2792 !
2800 CASE "TRANSLATE"
2810 CALL Translate_3d(Skitter(*),Skitmod(*),Total(*),Temp(*),Tempa(*),Trans(*),Sys_trans_x,Sys_trans_y,Sys_trans_z,Way,Theta,Printflag$)
2820 !
2821 REM***** PIVOT LINES *****
2822 !
2830 CASE "PIVOTLINES"
2840 Theta=-Theta
2841 !
2842 !
2850 IF Leg_flag$="FEMUR A" THEN
2860 Pivot_ang_a=Pivot_ang_a+Theta
2870 DISP " MODE: ROTATE ABOUT PIVOT LINE A";Theta;" TOTAL ANGLE:";Pivot_ang_a
2880 Pivotx=(Skitter(96,1)+Skitter(127,1))/2
2890 Pivoty=(Skitter(96,2)+Skitter(127,2))/2
2900 Pivotz=(Skitter(96,3)+Skitter(127,3))/2
2910 Dist_piv_foot=SQR((Pivotx-Skitter(65,1))^2+(Pivoty-Skitter(65,2))^2+(Pivotz-Skitter(65,3))^2)
2920 CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pivoty,-Pivotz)
2930 Flag=0
2940 CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,-Theta,Printflag$,Fem_a_ang,Flag)
2950 CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivoty,Pivotz)
2960 MAT Skitter= Skitmod
2970 END IF
2980 !
2981 !
2982 !
2990 IF Leg_flag$="FEMUR B" THEN
3000 Pivot_ang_b=Pivot_ang_b+Theta
3010 DISP " MODE: ROTATE ABOUT PIVOT LINE B BY:";Theta;" TOTAL ANG:";Pivot_ang_b
3020 Pivotx=(Skitter(65,1)+Skitter(127,1))/2
3030 Pivoty=(Skitter(65,2)+Skitter(127,2))/2
3040 Pivotz=(Skitter(65,3)+Skitter(127,3))/2
3050 Dist_piv_foot=SQR((Pivotx-Skitter(96,1))^2+(Pivoty-Skitter(96,2))^2+(Pivotz-Skitter(96,3))^2)
3060 CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pivoty,-Pivotz)
3070 CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)

```

```

80                                     Flag=0
90                                     CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Theta,Pr
tflag$,Fem_b_ang,Flag)
00                                     CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
10                                     CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
,Pivotz)
20                                     MAT Skitter= Skitmod
30                                     END IF
40 !
41 !
42 !
50                                     IF Leg_flag$="FEMUR C" THEN
60                                     Pivot_ang_c=Pivot_ang_c+Theta
70                                     DISP " MODE: ROTATE ABOUT PIVOT LINE C BY:";Theta;"
OTAL ANG:";Pivot_ang_c
80                                     Pivotx=(Skitter(65,1)+Skitter(96,1))/2
90                                     Pivoty=(Skitter(65,2)+Skitter(96,2))/2
00                                     Pivotz=(Skitter(65,3)+Skitter(96,3))/2
10                                     Dist_piv_foot=SQR((Pivotx-Skitter(127,1))^2+(Pivoty-S
tter(127,2))^2+(Pivotz-Skitter(127,3))^2)
20                                     CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
ty,-Pivotz)
30                                     CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
40                                     Flag=0
50                                     CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Theta,Pr
tflag$,Fem_c_ang,Flag)
60                                     CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
70                                     CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
,Pivotz)
80                                     MAT Skitter= Skitmod
90                                     END IF
00 !
10                                     MAT Tempa(*,1:3)= Skitter(*,1:3)
20                                     MAT Temp= Tempa*Total
30                                     MAT Skitmod(*,1:3)= Temp(*,1:3)
40                                     END SELECT
50                                     CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
60                                     GOTO Menu
70 !
71 REM*****
72 REM
73 REM      KNOB_LEG_ISR:  KNOB LEG INTERRUPT SERVICE ROUTINE FOR ACTUATOR MENU
74 REM
75 REM*****
76 !
80 Knob_leg_isr:
90      Theta=Increment*SGN(KNOBX)
00      SELECT Leg$
01!
02 !
10      CASE "FEMUR A"
20      Theta=-Theta
30      MAT Femur= Skitter(37:67,*)
40      CALL Trans_fem_orig(Femur(*),Femurmod(*),-Femur(3,1),
Femur(3,2),-Femur(3,3))
50      Flag=1
60      CALL Rotate_leg_y(Femurmod(*),Femurmod(*),180)
70      CALL Rotate_leg_z(Femurmod(*),Femurmod(*),Leg$,Theta,
intflag$,Fem_a_ang,Flag)
80      CALL Rotate_leg_y(Femurmod(*),Femurmod(*),180)

```



```

490      CALL Trans_fem_orig(Femurmod(*),Femurmod(*),Femur(3,1
,Femur(3,2),Femur(3,3))
500      MAT Skitter(37:67,*)= Femurmod
510!
520      IF Freeleg$="FREE" THEN
530      MAT Femurtemp(*,1:3)= Femurmod(*,1:3)
540      MAT Femur= Femurtemp*Total
550      MAT Skitmod(37:67,1:3)= Femur(*,1:3)
560      GOTO End_leg
570      END IF
580!
590      Pivotx=(Skitter(96,1)+Skitter(127,1))/2
600      Pivoty=(Skitter(96,2)+Skitter(127,2))/2
610      Pivotz=(Skitter(96,3)+Skitter(127,3))/2
620      Dist_piv_foot=SQR((Pivotx-Skitter(65,1))^2+(Pivoty-Sk
tter(65,2))^2+(Pivotz-Skitter(65,3))^2)
630      Dist_y=Pivoty-Skitter(65,2)
640      Rot_ang=ASN(Dist_y/Dist_piv_foot)
650      CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
oty,-Pivotz)
660      Flag=0
670      CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,-Rot_ang
Printflag$,Fem_a_ang,Flag)
680      CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
y,Pivotz)
690      MAT Skitter= Skitmod
700      MAT Tempa(*,1:3)= Skitter(*,1:3)
710      MAT Temp= Tempa*Total
720      MAT Skitmod(*,1:3)= Temp(*,1:3)
730 !
740 !
750 !
760      CASE "FEMUR B"
770      MAT Femur= Skitter(68:98,*)
780      CALL Trans_fem_orig(Femur(*),Femurmod(*),-Femur(3,1),
Femur(3,2),-Femur(3,3))
790      CALL Rotate_leg_y(Femurmod(*),Femurmod(*),60)
800      Flag=1
810      CALL Rotate_leg_z(Femurmod(*),Femurmod(*),Leg$,Theta,
rintflag$,Fem_b_ang,Flag)
820      CALL Rotate_leg_y(Femurmod(*),Femurmod(*),-60)
830      CALL Trans_fem_orig(Femurmod(*),Femurmod(*),Femur(3,1
,Femur(3,2),Femur(3,3))
840      MAT Skitter(68:98,*)= Femurmod
850 !
860      IF Freeleg$="FREE" THEN
870      MAT Femurtemp(*,1:3)= Femurmod(*,1:3)
880      MAT Femur= Femurtemp*Total
890      MAT Skitmod(68:98,1:3)= Femur(*,1:3)
900      GOTO End_leg
910      END IF
920 !
930      Pivotx=(Skitter(65,1)+Skitter(127,1))/2
940      Pivoty=(Skitter(65,2)+Skitter(127,2))/2
950      Pivotz=(Skitter(65,3)+Skitter(127,3))/2
960      Dist_piv_foot=SQR((Pivotx-Skitter(96,1))^2+(Pivoty-Sk
tter(96,2))^2+(Pivotz-Skitter(96,3))^2)
970      Dist_y=Pivoty-Skitter(96,2)
980      Rot_ang=ASN(Dist_y/Dist_piv_foot)
990      CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi

```

```

oty,-Pivotz)
000 CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
010 Flag=0
020 CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Rot_ang,
rintflag$,Fem_b_ang,Flag)
030 CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
040 CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
y,Pivotz)
050 MAT Skitter= Skitmod
060 MAT Tempa(*,1:3)= Skitter(*,1:3)
070 MAT Temp= Tempa*Total
080 MAT Skitmod(*,1:3)= Temp(*,1:3)
090 !
100 !
110 CASE "FEMUR C"
120 Theta=-Theta
130 MAT Femur= Skitter(99:129,*)
140 CALL Trans_fem_orig(Femur(*),Femurmod(*),-Femur(3,1),
Femur(3,2),-Femur(3,3))
150 CALL Rotate_leg_y(Femurmod(*),Femurmod(*),-60)
160 Flag=1
170 CALL Rotate_leg_z(Femurmod(*),Femurmod(*),Leg$,Theta,
rintflag$,Fem_c_ang,Flag)
180 CALL Rotate_leg_y(Femurmod(*),Femurmod(*),60)
190 CALL Trans_fem_orig(Femurmod(*),Femurmod(*),Femur(3,1)
,Femur(3,2),Femur(3,3))
200 MAT Skitter(99:129,*)= Femurmod
210 !
220 IF Freeleg$="FREE" THEN
230 MAT Femurtemp(*,1:3)= Femurmod(*,1:3)
240 MAT Femur= Femurtemp*Total
250 MAT Skitmod(99:129,1:3)= Femur(*,1:3)
260 GOTO End_leg
270 END IF
280 !
290 Pivotx=(Skitter(65,1)+Skitter(96,1))/2
300 Pivoty=(Skitter(65,2)+Skitter(96,2))/2
310 Pivotz=(Skitter(65,3)+Skitter(96,3))/2
320 Dist_piv_foot=SQR((Pivotx-Skitter(127,1))^2+(Pivoty-S
itter(127,2))^2+(Pivotz-Skitter(127,3))^2)
330 Dist_y=Pivoty-Skitter(127,2)
340 Rot_ang=ASN(Dist_y/Dist_piv_foot)
350 CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
oty,-Pivotz)
360 CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
370 Flag=0
380 CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Rot_ang,
rintflag$,Fem_c_ang,Flag)
390 CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
400 CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
y,Pivotz)
410 MAT Skitter= Skitmod
420 MAT Tempa(*,1:3)= Skitter(*,1:3)
430 MAT Temp= Tempa*Total
440 MAT Skitmod(*,1:3)= Temp(*,1:3)
450 !
460 !
470 !
480 CASE "TIBIA A"
490 Theta=-Theta

```

```

500      MAT Tibia= Skitter(58:67,*)
510      CALL Trans_fem_orig(Tibia(*),Tibiamod(*),-Tibia(2,1),
Tibia(2,2),-Tibia(2,3))
520      Flag=1
530      CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),180)
540      CALL Rotate_leg_z(Tibiamod(*),Tibiamod(*),Leg$,Theta,
rintflag$,Tib_a_ang,Flag)
550      CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),180)
560      CALL Trans_fem_orig(Tibiamod(*),Tibiamod(*),Tibia(2,1
,Tibia(2,2),Tibia(2,3))
570      MAT Skitter(58:67,*)= Tibiamod
580 !
590      IF Freeleg$="FREE" THEN
600      MAT Tibiatemp(*,1:3)= Tibiamod(*,1:3)
610      MAT Tibia= Tibiatemp*Total
620      MAT Skitmod(58:67,1:3)= Tibia(*,1:3)
630      GOTO End_leg
640      END IF
650 !
660      Pivotx=(Skitter(96,1)+Skitter(127,1))/2
670      Pivoty=(Skitter(96,2)+Skitter(127,2))/2
680      Pivotz=(Skitter(96,3)+Skitter(127,3))/2
690      Dist_piv_foot=SQR((Pivotx-Skitter(65,1))^2+(Pivoty-Sk
tter(65,2))^2+(Pivotz-Skitter(65,3))^2)
700      Dist_y=Pivoty-Skitter(65,2)
710      Rot_ang=ASN(Dist_y/Dist_piv_foot)
720      CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
oty,-Pivotz)
730      Flag=0
740      CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,-Rot_ang
Printflag$,Tib_a_ang,Flag)
750      CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
y,Pivotz)
760      MAT Skitter= Skitmod
770      MAT Tempa(*,1:3)= Skitter(*,1:3)
780      MAT Temp= Tempa*Total
790      MAT Skitmod(*,1:3)= Temp(*,1:3)
300 !
310 !
320      CASE "TIBIA B"
330      MAT Tibia= Skitter(89:98,*)
340      CALL Trans_fem_orig(Tibia(*),Tibiamod(*),-Tibia(2,1),
Tibia(2,2),-Tibia(2,3))
350      CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),60)
360      Flag=1
370      CALL Rotate_leg_z(Tibiamod(*),Tibiamod(*),Leg$,Theta,
rintflag$,Tib_b_ang,Flag)
380      CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),-60)
390      CALL Trans_fem_orig(Tibiamod(*),Tibiamod(*),Tibia(2,1
,Tibia(2,2),Tibia(2,3))
400      MAT Skitter(89:98,*)= Tibiamod
410 !
420      IF Freeleg$="FREE" THEN
430      MAT Tibiatemp(*,1:3)= Tibiamod(*,1:3)
440      MAT Tibia= Tibiatemp*Total
450      MAT Skitmod(89:98,1:3)= Tibia(*,1:3)
460      GOTO End_leg
470      END IF
480 !
490      Pivotx=(Skitter(65,1)+Skitter(127,1))/2

```

```

0000          Pivoty=(Skitter(65,2)+Skitter(127,2))/2
0010          Pivotz=(Skitter(65,3)+Skitter(127,3))/2
0020          Dist_piv_foot=SQR((Pivotx-Skitter(96,1))^2+(Pivoty-Sk
0030          itter(96,2))^2+(Pivotz-Skitter(96,3))^2)
0040          Dist_y=Pivoty-Skitter(96,2)
0050          Rot_ang=ASN(Dist_y/Dist_piv_foot)
0060          CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
0070          voty,-Pivotz)
0080          CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
0090          Flag=0
0100          CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Rot_ang,
0110          rintflag$,Tib_b_ang,Flag)
0120          CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
0130          CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
0140          y,Pivotz)
0150          MAT Skitter= Skitmod
0160          MAT Tempa(*,1:3)= Skitter(*,1:3)
0170          MAT Temp= Tempa*Total
0180          MAT Skitmod(*,1:3)= Temp(*,1:3)
0190          !
0200          !
0210          CASE "TIBIA C"
0220          MAT Tibia= Skitter(120:129,*)
0230          CALL Trans_fem_orig(Tibia(*),Tibiamod(*),-Tibia(2,1),
0240          Tibia(2,2),-Tibia(2,3))
0250          CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),-60)
0260          Flag=1
0270          CALL Rotate_leg_z(Tibiamod(*),Tibiamod(*),Leg$,Theta,
0280          rintflag$,Tib_c_ang,Flag)
0290          CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),60)
0300          CALL Trans_fem_orig(Tibiamod(*),Tibiamod(*),Tibia(2,1
0310          ,Tibia(2,2),Tibia(2,3))
0320          MAT Skitter(120:129,*)= Tibiamod
0330          !
0340          IF Freeleg$="FREE" THEN
0350          MAT Tibiatemp(*,1:3)= Tibiamod(*,1:3)
0360          MAT Tibia= Tibiatemp*Total
0370          MAT Skitmod(120:129,1:3)= Tibia(*,1:3)
0380          GOTO End_leg
0390          END IF
0400          !
0410          Pivotx=(Skitter(65,1)+Skitter(96,1))/2
0420          Pivoty=(Skitter(65,2)+Skitter(96,2))/2
0430          Pivotz=(Skitter(65,3)+Skitter(96,3))/2
0440          Dist_piv_foot=SQR((Pivotx-Skitter(127,1))^2+(Pivoty-S
0450          kitter(127,2))^2+(Pivotz-Skitter(127,3))^2)
0460          Dist_y=Pivoty-Skitter(127,2)
0470          Rot_ang=ASN(Dist_y/Dist_piv_foot)
0480          CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
0490          voty,-Pivotz)
0500          CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
0510          Flag=0
0520          CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Rot_ang,
0530          rintflag$,Tib_c_ang,Flag)
0540          CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
0550          CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
0560          y,Pivotz)
0570          MAT Skitter= Skitmod
0580          MAT Tempa(*,1:3)= Skitter(*,1:3)
0590          MAT Temp= Tempa*Total

```

```

5490                                     MAT Skitmod(*,1:3)= Temp(*,1:3)
5500 !
5510 End_leg: !
5520     END SELECT
5530 !
5540     CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
5550     GOTO Menu
5551 REM *****
5552 REM
5553 REM     GOSUB ROUTINES FOR MENU AND KNOB ISR CASE SELECTION
5554 REM
5555 REM*****
5560 Main:!
5570     Menu$="MAIN"
5580     GOTO Menu
5590 Movie:!
5600     CALL Movie
5610     RESTORE 160
5620     GOSUB Init
5630     CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
5640     GOTO Menu
5650 Pivotlines:!
5660     Menu$="PIVOTLINES"
5670     GOTO Menu
5680 !
5690 Pivotlega:!
5700     Twirl$="PIVOTLINES"
5710     Leg_flag$="FEMUR A"
5720     RETURN
5730 !
5740 Pivotlegb:!
5750     Twirl$="PIVOTLINES"
5760     Leg_flag$="FEMUR B"
5770     RETURN
5780 !
5790 Pivotlegc:!
5800     Twirl$="PIVOTLINES"
5810     Leg_flag$="FEMUR C"
5820     RETURN
5830 !
5840 System: !
5850     Menu$="SYSTEM"
5860     GOTO Menu
5870 !
5880 Attributes:!
5890     Menu$="ATTRIBUTES"
5900     GOTO Menu
5910 !
5920 Output:!
5930     Menu$="OUTPUT"
5940     GOTO Menu
5950 !
5960 !
5970 Actuator:!
5980     Menu$="ACTUATOR"
5990     GOTO Menu
6000 !
6010 Increment:!
6020     DISP " INPUT NEW INCREMENT CURRENT VALUE:",Increment;
6030     LINPUT Increment$

```

```

5040     IF Increment$="" THEN
5050         GOTO 6090
5060     ELSE
5070         Increment=VAL(Increment$)
5080     END IF
5090     GOTO Menu
5100 Free_leg:!
5110     IF Freeleg$="FIXED" THEN
5120         DISP " LEG IS NOW FREE TO ROTATE"
5130         Freeleg$="FREE"
5140     ELSE
5150         DISP " LEG IS NOW FIXED "
5160         Freeleg$="FIXED"
5170     END IF
5180     GOTO Menu
5190 Femur_a:!
5200     Leg$="FEMUR A"
5210     DISP " MODE: FEMUR A"
5220     GOTO Menu
5230 !
5240 Femur_b:!
5250     Leg$="FEMUR B"
5260     DISP " MODE: FEMUR B"
5270     GOTO Menu
5280 !
5290 Femur_c:!
5300     Leg$="FEMUR C"
5310     DISP " MODE: FEMUR C"
5320     GOTO Menu
5330 !
5340 Tibia_a:!
5350     Leg$="TIBIA A"
5360     DISP " MODE: TIBIA A"
5370     GOTO Menu
5380 !
5390 Tibia_b:!
5400     Leg$="TIBIA B"
5410     DISP " MODE: TIBIA B"
5420     GOTO Menu
5430 !
5440 Tibia_c:!
5450     Leg$="TIBIA C"
5460     DISP " MODE: TIBIA C"
5470     GOTO Menu
5480 !
5490 Windows:!
5500     CALL Zoom_pan(Window$,Screenx_win_min,Screenx_win_max,Screeny_wi
_min,Screeny_win_max,Skitmod(*),Newskit(*),Screen_x,Screen_y)
5510     GOTO Menu
5520 What:!
5530 !
5540     Ap=Screenx_win_min
5550     Bp=Screenx_win_max
5560     Cp=Screeny_win_min
5570     Dp=Screeny_win_max
5580     Ep=Sys_trans_x
5590     Fp=Sys_trans_y
5600     Gp=Sys_trans_z
5610     Hp=Sys_rot_x
5620     Ip=Sys_rot_y

```

```

6630      Kp=Sys_rot_z
6640      Lp=Fem_a_ang
6650      Mp=Fem_b_ang
6660      Np=Fem_c_ang
6670      Op=Tib_a_ang
6680      Qp=Tib_b_ang
6690      Pp=Tib_c_ang
6700      Rp=Increment
6710      Sp=Pivot_ang_a
6720      Tp=Pivot_ang_b
6730      Up=Pivot_ang_c
6740 CALL What(Ep,Fp,Gp,Hp,Ip,Kp,Ap,Bp,Cp,Dp,Printflag$,Screen_x,Screen_y,Plot_d
vice,Dump_device,Lp,Mp,Np,Op,Qp,Pp,Rp,Sp,Tp,Up)
6750      GOTO Menu
6760 !
6770 Windowpane:!
6780      CALL Windows(Skitmod(*),Newskit(*),Screen_x,Screen_y)
6790      GOTO Menu
6800 !
6810 Window_limits:!
6820      CALL Window_limits(Skitmod(*),Newskit(*),Screen_x,Screen_y,Screenx_win_min,Screenx_win_max,Screeny_win_min,Screeny_win_max)
6830      GOTO Menu
6840 !
6850 Print_flag:!
6860      IF Printflag$="OFF" THEN
6870          DISP " DISPLAY QUANTITIES IS ";CHR$(129);" ON ";CHR$(128)
6880          Printflag$="ON"
6890      ELSE
6900          DISP " DISPLAY QUANTITIES IS OFF"
6910          Printflag$="OFF"
6920      END IF
6930      GOTO Menu
6940 Rot_y:!
6950      Twirl$="ROTATE Y"
6960      DISP " MODE:";Twirl$
6970      RETURN
6980 !
6990 Rot_x:!
7000      Twirl$="ROTATE X"
7010      DISP " MODE:";Twirl$
7020      RETURN
7030 !
7040 Rot_z:!
7050      Twirl$="ROTATE Z"
7060      DISP " MODE:";Twirl$
7070      RETURN
7080 !
7090 Trans_x:!
7100      Twirl$="TRANSLATE"
7110      DISP " MODE: TRANSLATE X"
7120      Way=1
7130      RETURN
7140 !
7150 Trans_y:!
7160      Twirl$="TRANSLATE"
7170      DISP " MODE: TRANSLATE Y"
7180      Way=2
7190      RETURN
7200 !

```

```

7210 Trans_z:!
7220 Twirl$="TRANSLATE"
7230 DISP " MODE: TRANSLATE Z"
7240 Way=3
7250 RETURN
7260 !
7270 Vector:!
7280 DISP " MODE: VECTOR TRANSLATION"
7290 CALL Vector(Skitter(*),Skitmod(*),Temp(*),Tempa(*),Total(*),Tran
s(*),Sys_trans_x,Sys_trans_y,Sys_tran_z,Newskit(*),Screen_x,Screen_y,Printflag$)
7300 RETURN
7310 !
7320 Angle:!
7330 DISP " MODE:INPUT ANGLE ROTATION"
7340 CALL Angle(Skitter(*),Skitmod(*),Temp(*),Tempa(*),Total(*),Trans(*),S
ys_rot_x,Sys_rot_y,Sys_rot_z,Newskit(*),Screen_x,Screen_y,Printflag$,Twirl$)
7350 RETURN
7360 !
7370 !
7380 Plot:!
7381 CALL Plot_it(Plot_device)
7382 PLOTTER IS Plot_device,"HPGL"
7383 DISP "PLOT BEING GENERATED"
7384 CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
7385 PLOTTER IS CRT,"INTERNAL"
7386 PRINT "IN;ROO;IP;UP;SPO;"
7387 PRINTER IS CRT
7390 BEEP 1464.84,.5
7400 DISP " PLOT FINISHED"
7410 GOTO Menu
7420 !
7430 Dump:!
7431 DISP "GRAPHICS DUMP BEING GENERATED"
7433 DUMP DEVICE IS Dump_device
7434 DUMP GRAPHICS
7435 PRINTER IS Dump_device
7436 PRINT CHR$(12)
7437 PRINTER IS CRT
7440 BEEP 1464.84,.5
7450 DISP " GRAPHICS DUMP FINISHED"
7460 GOTO Menu
7470 !
7480 Printer:!
7490 DISP "WHERE IS THE LOCATION OF THE EXTERNAL PRINTER";
7500 LINPUT Temp$
7510 IF Temp$="" THEN GOTO 7540
7520 Dump_device=VAL(Temp$)
7530 DISP " PRINTER IS AT ";CHR$(129);Dump_device;CHR$(128)
7540 GOTO Menu
7550 !
7560 Plotter:!
7570 DISP " WHERE IS THE LOCATION OF THE PLOTTER";
7580 LINPUT Temp$
7590 IF Temp$="" THEN GOTO 7620
7600 Plot_device=VAL(Temp$)
7610 DISP " PLOTTER IS AT ";CHR$(129);Plot_device;CHR$(128)
7620 GOTO Menu
7630 !*****
7640 !
7641 ! INITIALIZATION OF PARAMETERS

```



```

7642 !
7643 !*****
7650 Init: ! INITIALIZE SCREEN,SKITTER
7660 !
7670 DEG ! SET TO DEGREES
7680 GINIT ! INITIATE GRAPHICS
7690 GRAPHICS ON ! TURN G-PLANE ON
7700 PLOTTER IS CRT,"INTERNAL" ! INIT PLOTTER
7710 !
7720 Dump_device=9
7730 !
7740 Plot_device=705
7750 !
7760 READ Skitter(*) ! READ SKITTER DATA
7770 !
7780 MAT Skitmod= Skitter
7790 !
7800 MAT Femurtemp= (1)
7810 !
7820 MAT Tibiatemp= (1)
7830 MAT Trans= IDN ! INIT TRANS MATRIX TO IDN
7840 !
7850 MAT Tempa= (1)
7860 !
7870 MAT Total= IDN
7880 !
7890 Menu$="MAIN" ! INIT MAIN MENU
7900 !
7910 Twirl$="ROTATE Y"
7920 !
7930 Freeleg$="FIXED"
7940 !
7950 Way=1 ! AXIS OF TRANS X=1,Y=2,Z=3
7960 !
7970 Printflag$="OFF"
7980 !
7990 Increment=5 ! TRANS INC.
8000 !
8010 Rot_increment=3
8020 !
8030 Screen_x=1 ! INIT VIEW PLANE
8040 Screen_y=2 ! X=1,Y=2,Z=3
8050 !
8060 Sys_trans_x=0 ! INIT POSITONS OF SYSTEM
8070 Sys_trans_y=0
8080 Sys_trans_z=0
8090 Sys_rot_x=0
8100 Sys_rot_y=0
8110 Sys_rot_z=0
8120 !
8130 Fem_a_ang=0 ! INIT LEG ANGLES
8140 Fem_b_ang=0
8150 Fem_c_ang=0
8160 Tib_a_ang=90
8170 Tib_b_ang=90
8180 Tib_c_ang=90
8190 !
8200 Pivot_ang_a=0
8210 Pivot_ang_b=0
8220 Pivot_ang_c=0

```

```

8230 !
8240     Screenx_win_max=40                                ! SET WINDOW
8250     Screenx_win_min=-20
8260     Screeny_win_max=40
8270     Screeny_win_min=-20
8280     SHOW Screenx_win_min,Screenx_win_max,Screeny_win_min,Screeny_win_max
8290 !
8300     RETURN
8310 !*****
8320 !
8330 !
8340 !*****
8350 !
8351 !     EXIT ROUTINE TO CLEAR SCREEN AND ENTER BASIC ENVIROMENT
8352 !
8353 !*****
8360 Finished:                                           ! DONE WITH PROGRAM
8370     GCLEAR
8380     GRAPHICS OFF
8390     CLEAR SCREEN
8400     END
8410 !
8411 !
8412 !
8413 !
8414 !
8415 !
8416 !
8417 !
8420 !
8430 !*****
8440 !
8441 !     SUBROUTINE ROTATE:  ROTATES SYTEM ABOUT LOCAL Y AXIS
8442 !
8443 !*****
8444 !
8450 SUB Rotate_y(Skitter(*),Skitmod(*),Total(*),Trans(*),Tempa(*),Sys_rot_y,Theta,Printflag$)
8460 !
8470 DIM Bogus(4,4)
8480 !
8490     Sys_rot_y=Sys_rot_y+Theta
8500     IF Printflag$="ON" THEN
8510         DISP " MODE: ROTATE Y BY ANGLE OF ",Theta," TOTAL ANGLE=",Sys_rot_y
8520     END IF
8530 MAT Tempa(*,1:3)= Skitter(*,1:3)
8540 !
8550 ! SET UP ROTATION MATRIX
8560 !
8570     MAT Trans= IDN
8580     Sine=SIN(Theta)
8590     Cosine=COS(Theta)
8600     Trans(1,1)=Cosine
8610     Trans(1,3)=Sine
8620     Trans(3,1)=-Sine
8630     Trans(3,3)=Cosine
8640 !
8650 ! INCREMENT Y ROTATION VARIABLE
8660 !

```

```

8670 !
8680     MAT Bogus= Trans*Total
8690     MAT Total= Bogus
8700     MAT Temp= Tempa*Total
8710     MAT Skitmod(*,1:3)= Temp(*,1:3)
8720 !
8730     SUBEND
8740 !
8750 !*****
8760 !
8761 !     SUBROUTINE ROTATE X:  ROTATES SYSTEM ABOUT LOCAL X AXIS
8762 !
8763 !*****
8764 !
8770 SUB Rotate_x(Skitter(*),Skitmod(*),Total(*),Trans(*),Temp(*),Tempa(*),Sys_rot_x,Theta,Printflag$)
8780 OPTION BASE 1
8790 DIM Bogus(4,4)
8800 !
8810     Sys_rot_x=Sys_rot_x+Theta
8820     IF Printflag$="ON" THEN
8830         DISP " MODE: ROTATE X BY ANGLE OF ",Theta," TOTAL ANGLE=",Sys_rot_x
8840     END IF
8850 MAT Tempa(*,1:3)= Skitter(*,1:3)
8860 ! SET UP ROTATION MATRIX
8870 !
8880     MAT Trans= IDN
8890     Sine=SIN(Theta)
8900     Cosine=COS(Theta)
8910     Trans(2,2)=Cosine
8920     Trans(3,3)=Cosine
8930     Trans(2,3)=-Sine
8940     Trans(3,2)=Sine
8950 !
8960 ! INCREMENT ROTATION VARIABLE
8970 !
8980     MAT Bogus= Trans*Total
8990     MAT Total= Bogus
9000     MAT Temp= Tempa*Total
9010     MAT Skitmod(*,1:3)= Temp(*,1:3)
9020 !
9030     SUBEND
9040 !
9050 !*****
9060 !
9061 !     SUBROUTINE ROTATE Z:  ROTATES SYSTEM ABOUT LOCAL Z AXIS
9062 !
9063 !*****
9064 !
9070 SUB Rotate_z(Skitter(*),Skitmod(*),Total(*),Trans(*),Temp(*),Tempa(*),Sys_rot_z,Theta,Printflag$)
9080 OPTION BASE 1
9090 DIM Bogus(4,4)
9100     Sys_rot_z=Sys_rot_z+Theta
9110     IF Printflag$="ON" THEN
9120         DISP " MODE: ROTATE Z BY ANGLE OF ",Theta," TOTAL ANGLE=",Sys_rot_z
9130     END IF
9140 MAT Tempa(*,1:3)= Skitter(*,1:3)

```

D

```

)150 !
)160 ! SET UP ROTATION MATRIX
)170 !
)180     MAT Trans= IDN
)190     Sine=SIN(Theta)
)200     Cosine=COS(Theta)
)210     Trans(1,1)=Cosine
)220     Trans(1,2)=-Sine
)230     Trans(2,1)=Sine
)240     Trans(2,2)=Cosine
)250 !
)260 ! INCREMENT COUNTER; FIND NEW SKITTER MATRIX
)270     MAT Bogus= Trans*Total
)280     MAT Total= Bogus
)290     MAT Temp= Tempa*Total
)300     MAT Skitmod(*,1:3)= Temp(*,1:3)
)310     SUBEND
)320 !
)330 !*****
)340 !
)341 ! SUBROUTINE TRANSLATE 3D: TRANSLATES SYSTEM ALONG A LOCAL X-Y-Z AXIS
)342 !
)343 !*****
)344 !
)350 SUB Translate_3d(Skitter(*),Skitmod(*),Total(*),Temp(*),Tempa(*),Trans(*),S
)360 /s_trans_x,Sys_trans_y,Sys_trans_z,Way,Theta,Printflag$)
)370 DIM Laurie(4,4)
)380 !
)390 ! SET UP TEMP ARRAY SO AS NOT TO LOOSE PENS --SKITTER(*,4)
)400 !
)410     MAT Tempa(*,1:3)= Skitter(*,1:3)
)420 !
)430 ! DETERMINE DIRECTION OF TRANSLATION AND SET UP TRANS MATRIX
)440 !
)450     IF Way=1 THEN
)460         Sys_trans_x=Sys_trans_x+Theta
)470         Tx=Theta
)480         Ty=0
)490         Tz=0
)500     IF Printflag$="ON" THEN
)510         DISP " MODE: TRANSLATE X BY ",Theta,"IN. TOTAL TRANSLATION=",Sys_t
)520     END IF
)530     END IF
)540     IF Way=2 THEN
)550         Sys_trans_y=Sys_trans_y+Theta
)560         Tx=0
)570         Ty=Theta
)580         Tz=0
)590     IF Printflag$="ON" THEN
)600         DISP " MODE: TRANSLATE Y BY ",Theta,"IN. TOTAL TRANSLATION=",Sys_t
)610     END IF
)620     END IF
)630     IF Way=3 THEN
)640         Sys_trans_z=Sys_trans_z+Theta
)650         Tx=0
)660         Ty=0
)670         Tz=Theta

```

```

0680     IF Printflag$="ON" THEN
0690         DISP " MODE: TRANSLATE Z BY ";Theta,"IN.  TOTAL TRNASLATION=";Sys_t
ans_z
0700     END IF
0710         END IF
0720     MAT Trans= IDN
0730     Trans(4,1)=Tx
0740     Trans(4,2)=Ty
0750     Trans(4,3)=Tz
0760 !
0770 ! FIND NEW SKITTER MATRIX WITH CORRECT PENS
0780 !
0790     MAT Laurie= Trans*Total
0800     MAT Total= Laurie
0810     MAT Temp= Tempa*Total
0820     MAT Skitmod(*,1:3)= Temp(*,1:3)
0830     SUBEND
0840 !
0850 !*****
0860 !
0870 SUB Scaling_3d(Sx,Sy,Sz,Array(*))
0880     MAT Array= IDN
0890     Array(1,1)=Sx
0900     Array(2,2)=Sy
0910     Array(3,3)=Sz
0920     SUBEND
0930 !
0940 !*****
0950 !
0951 !     SUBROUTINE DISPLAY_SKIT:  PLOTS SKITTER TO LOCAL PLOTTING DEVICE
0952 !                               OR SCREEN
0953 !
0954 !*****
0955 !
0960 SUB Display_skit(Skitter(*),Newskit(*),Screen_x,Screen_y)
0970     OPTION BASE 1
0980     DATA 1,1,4           ! PEN1
0990     DATA 4,4,4           ! PEN 2
0000     DATA 8,8,4           ! PEN 3
0010     DIM Pen1(1,3),Pen2(1,3),Pen3(1,3),Temp(1,3)
0020     READ Pen1(*),Pen2(*),Pen3(*)
0030     IF Screen_x<>1 AND Screen_y<>2 THEN
0040     MAT Temp= Pen2
0050     MAT Pen2= Pen3
0060     MAT Pen3= Temp
0070     END IF
0080     MAT Newskit(*,1)= Skitter(*,Screen_x)
0090     MAT Newskit(*,2)= Skitter(*,Screen_y)
0100     MAT Newskit(*,3)= Skitter(*,4)
0110     MAT Newskit(37:37,*)= Pen1
0120     MAT Newskit(68:68,*)= Pen2
0130     MAT Newskit(99:99,*)= Pen3
0140     CLEAR SCREEN
0150     GCLEAR
0160     FRAME
0170     IF Screen_x<>3 AND Screen_y<>2 THEN GOTO 10300
0180     MOVE -100,-.9
0190     RECTANGLE 200,.15
0200     MOVE -100,-.75
0210     RECTANGLE 200,.15

```

```

10220 MOVE -100,-.6
10230 RECTANGLE 200,.15
10240 MOVE -100,-.45
10250 RECTANGLE 200,.15
10260 MOVE -100,-.3
10270 RECTANGLE 200,.15
10280 MOVE -100,-.15
10290 RECTANGLE 200,.15
10300 PLOT Newskit(*)
10310 LINE TYPE 1
10320 SUBEND
10330!
10340!
10350! *****
10360!
10370 SUB Printmat(Array(*))
10380 OPTION BASE 1
10390 FOR Row=BASE(Array,1) TO SIZE(Array,1)+BASE(Array,1)-1
10400 FOR Column=BASE(Array,2) TO SIZE(Array,2)+BASE(Array,2)-1
10410 PRINT USING "DDDD.DD,XX,#";Array(Row,Column)
10420 NEXT Column
10430 PRINT
10440 NEXT Row
10450 SUBEND
10460 !
10470 ! *****
10480 !
10481 ! SUBROUTINE TRANS_TO_VECTOR: TRANSLATES SYSTEM TO A GIVEN POINT
10482 !
10483 ! *****
10490 SUB Trans_to_vector(Skitter(*),Skitmod(*),Temp(*),Tempa(*),Total(*),Trans(
*) ,Sys_trans_x,Sys_trans_y,Sys_trans_z)
10500 !
10510 OPTION BASE 1
10520 DIM Bogus(4,4)
10530 ! SET UP STORAGE ARRAY TO KEEP SKITTER PENS CORRECT SKITTER(*,4)
10540 !
10550 MAT Tempa(*,1:3)= Skitter(*,1:3)
10560 MAT Trans= IDN
10570 !
10580 ! SET UP TRANS MATRIX
10590 !
10600 Trans(4,1)=Sys_trans_x
10610 Trans(4,2)=Sys_trans_y
10620 Trans(4,3)=Sys_trans_z
10630 !
10640 ! FIND NEW MATRIX
10650 !
10660 MAT Bogus= Trans*Total
10670 MAT Total= Bogus
10680 MAT Temp= Tempa*Total
10690 MAT Skitmod(*,1:3)= Temp(*,1:3)
10700 SUBEND
10710 !
10720 !
10730 ! *****
10740 !
10741 ! SUBROUTINE VECTOR: TRANSLATES SYSTEM TO INPUT POINT
10742 !
10743 ! *****

```

```

10744 !
10750 SUB Vector(Skitter(*),Skitmod(*),Temp(*),Tempa(*),Total(*),Trans(*),Sys_tr
ans_x, Sys_trans_y, Sys_trans_z, Newskit(*), Screen_x, Screen_y, Prinflag$)
10760 !
10770 ! ASK FOR INPUT <CR> MEANS LEAVE
10780 !
10790 Ask: !
10800 DISP " X COORDINANT RELATIVE TO ", Sys_trans_x;
10810 LINPUT Temp$
10820 IF Temp$="" THEN
10830     GOTO Leave
10840 ELSE
10850     X=VAL(Temp$)
10860 END IF
10870 DISP " Y COORDINANT RELATIVE TO ", Sys_trans_y;
10880 LINPUT Temp$
10890 IF Temp$="" THEN
10900     GOTO Leave
10910 ELSE
10920     Y=VAL(Temp$)
10930 END IF
10940 DISP " Z COORDINANT RELATIVE TO ", Sys_trans_z;
10950 LINPUT Temp$
10960 IF Temp$="" THEN
10970     GOTO Leave
10980 ELSE
10990     Z=VAL(Temp$)
11000 END IF
11010 !
11020 ! UPDATE TRANSLATION COUNTERS
11030 !
11040 Sys_trans_x=Sys_trans_x+X
11050 Sys_trans_y=Sys_trans_y+Y
11060 Sys_trans_z=Sys_trans_z+Z
11070 !
11080 ! FIND NEW SKITTER MATRIX
11090 !
11100 CALL Trans_to_vector(Skitter(*),Skitmod(*),Temp(*),Tempa(*),Total(*),Trans
(*),X,Y,Z)
11110 !
11120 ! DISPLAY SKITTER
11130 CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
11140 GOTO Ask
11150 Leave: !
11160 SUBEND
11170 !
11180 !
11190 !*****
11200 !
11201 ! SUBROUTINE WINDOWS: ALLOWS USER TO CHANGE VIEWING AXIS
11202 !
11203 !*****
11204 !
11210 SUB Windows(Skitmod(*),Newskit(*),Screen_x,Screen_y)
11220     IF Screen_x=1 AND Screen_y=2 THEN
11230         DISP " CURRENT WINDOW = X-Y PLANE"
11240     END IF
11250     IF Screen_x=1 AND Screen_y=3 THEN
11260         DISP " CURRENT WINDOW = X-Z PLANE"
11270     END IF

```

D

```

.1280      IF Screen_x=3 AND Screen_y=2 THEN
.1290          DISP "  CURRENT WINDOW = Z-Y PLANE"
.1300      END IF
.1310 Menu:      !
.1320          ON KEY 0 LABEL "X AXIS" GOTO Zy_plane
.1330          ON KEY 2 LABEL "Y AXIS" GOTO Xz_plane
.1340          ON KEY 4 LABEL "Z AXIS" GOTO Xy_plane
.1350          ON KEY 9 LABEL "QUIT" GOTO Leave
.1360          GOTO 11360
.1370          !
.1380 Zy_plane:      !
.1390          Screen_x=3
.1400          Screen_y=2
.1410          DISP "  NEW WINDOW = LOOKING DOWN X AXIS"
.1420          CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y

.1430          GOTO Menu
.1440          !
.1450 Xz_plane:      !
.1460          Screen_x=1
.1470          Screen_y=3
.1480          DISP "  NEW WINDOW = LOOKING DOWN Y AXIS"
.1490          CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y

.1500          GOTO Menu
.1510          !
.1520 Xy_plane:      !
.1530          Screen_x=1
.1540          Screen_y=2
.1550          DISP "  NEW WINDOW = LOOKING DOWN Z AXIS"
.1560          CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y

.1570          GOTO Menu
.1580 Leave:      !
.1590          SUBEND
.1600!
.1610!
.1620!
.1630!*****
.1640!
.1641 ! SUBROUTINE WINDOW_LIMITS: ALLOWS THE USER TO INPUT NEW VIEWING WINDOW
.1642 !
.1643 !*****
.1644 !
.1650 SUB Window_limits(Skitmod(*),Newskit(*),Screen_x,Screen_y,Screenx_win_min,
creenx_win_max,Screeny_win_min,Screeny_win_max)
.1660!
.1670!
.1680!
.1690 DISP " INPUT XMIN --- CURRENT VALUE IS",Screenx_win_min," <CR> TO EXIT";
.1700 LINPUT Temp$
.1710 IF Temp$="" THEN GOTO Leave
.1720 DISP " INPUT XMAX --- CURRENT VALUE IS",Screenx_win_max," <CR> TO EXIT";
.1730 LINPUT Temp1$
.1740 IF Temp1$="" THEN GOTO Leave
.1750 IF VAL(Temp$)>VAL(Temp1$) THEN
.1760     BEEP 1464.84,.5
.1770     DISP " XMIN HAS TO BE LESS THAN XMAX"
.1780     WAIT 3
.1790     GOTO 11690

```



```

1800 ELSE
1810   Screenx_win_min=VAL(Temp$)
1820   Screenx_win_max=VAL(Temp1$)
1830 END IF
1840!
1850!
1860!
1870 DISP " INPUT YMIN ---- CURRENT VALUE IS",Screeny_win_min," <CR> TO EXIT";
1880 LINPUT Temp$
1890 IF Temp$="" THEN GOTO Leave
1900 DISP " INPUT YMAX ---- CURRENT VALUE IS",Screeny_win_max," <CR> TO EXIT";
1910 LINPUT Temp1$
1920 IF Temp1$="" THEN GOTO Leave
1930 IF VAL(Temp$)>VAL(Temp1$) THEN
1940   BEEP 1464.84,.5
1950   DISP " Y MIN MUST BE LESS THAN Y MAX"
1960   WAIT 3
1970   GOTO 11870
1980 ELSE
1990   Screeny_win_min=VAL(Temp$)
2000   Screeny_win_max=VAL(Temp1$)
2010 END IF
2020!
2030!
2040 SHOW Screenx_win_min,Screenx_win_max,Screeny_win_min,Screeny_win_max
2050 CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
2060!
2070 Leave:
2080   SUBEND
2090!
2100!*****
2110!
2111 ! SUBROUTINE ANGLE:  ALLOWS TE USER TO ROTATE SYSTEM BY INPUT ANGLE
2112 !                   ABOUT  LAST ROTATION AXIS
2113 !
2114 !*****
2115 !
2120 SUB Angle(Skitter(*),Skitmod(*),Temp(*),Tempa(*),Total(*),Trans(*),Sys_rot
c,Sys_rot_y,Sys_rot_z,Newskit(*),Screen_x,Screen_y,Printflag$,Twirl$)
2130!
2140! FIND MODE AND ROTATE ABOUT CORRECT AXIS
2150!
2160 IF Twirl$="ROTATE X" THEN
2170   DISP " INPUT ANGLE TO ROTATE ABOUT X AXIS -- CURRENT ANG=",Sys_rot_x;
2180   LINPUT Temp$
2190   IF Temp$="" THEN GOTO Leave
2200   Theta=VAL(Temp$)
2210   CALL Rotate_x(Skitter(*),Skitmod(*),Total(*),Trans(*),Temp(*),Tempa(*),Sy
_rot_x,Theta,Printflag$)
2220 END IF
2230!
2240!
2250 IF Twirl$="ROTATE Y" THEN
2260   DISP " INPUT ANGLE TO ROTATE ABOUT Y AXIS -- CURRENT ANG=",Sys_rot_y;
2270   LINPUT Temp$
2280   IF Temp$="" THEN GOTO Leave
2290   Theta=VAL(Temp$)
2300   CALL Rotate_y(Skitter(*),Skitmod(*),Total(*),Trans(*),Temp(*),Tempa(*),Sy
_rot_y,Theta,Printflag$)
2310 END IF

```

**CONTINUED**

```

12320!
12330!
12340 IF Twirl$="ROTATE Z" THEN
12350 DISP " INPUT ANGLE TO ROTATE ABOUT Z AXIS -- CURRENT ANG=",Sys_rot_z;
12360 LINPUT Temp$
12370 IF Temp$="" THEN GOTO Leave
12380 Thcta=VAL(Temp$)
12390 CALL Rotate_z(Skitter(*),Skitmod(*),Total(*),Trans(*),Temp(*),Tempa(*),Sys
_rot_z,Theta,Printflag$)
12400 END IF
12410 !
12420 ! OUTPUT NEW PICTURE
12430 !
12440 CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
12450 GOTO 12150
12460 !
12470 !
12480 Leave: !
12490 SUBEND
12500 !
12510 !*****
12511 !
12512 ! SUBROUTINE WHAT: OUTPUTS PROGRAM VARIABLES TO SCREEN
12513 !
12514 !*****
12520 !
12530 SUB What(Sys_trans_x,Sys_trans_y,Sys_trans_z,Sys_rot_x,Sys_rot_y,Sys_rot_z
,A,B,C,D,Printflag$,Screenx,Screeny,Plotd,Dumpd,Fa,Fb,Fc,Ta,Tb,Tc,Inc,Sp,Tp,Up)
12540 !
12550 GRAPHICS OFF
12560 CLEAR SCREEN
12570 PRINT CHR$(132);"THESE ARE THE CURRENT PARAMETERS:";CHR$(128)
12580 PRINT
12590 PRINT CHR$(129);" WINDOW: ";CHR$(128);" X MIN = ";A;" X MAX =
";B
12600 PRINT " Y MIN = ";C;" Y MAX = ";D
12610 PRINT
12620 IF Screenx=1 AND Screeny=2 THEN PRINT CHR$(129);" VIEW: ";CHR$(128);" LOOK
ING DOWN Z AXIS"
12630 IF Screenx=3 AND Screeny=2 THEN PRINT CHR$(129);" VIEW: ";CHR$(128);" LOOK
ING DOWN X AXIS"
12640 IF Screenx=1 AND Screeny=3 THEN PRINT CHR$(129);" VIEW: ";CHR$(128);" LOOK
ING DOWN Y AXIS"
12650 PRINT
12660 PRINT "PLOTTER LOACTION:";CHR$(129);Plotd;CHR$(128);" PRINTER LOCATIO
N:";CHR$(129);Dumpd;CHR$(128)
12670 PRINT
12680 IF Printflag$="ON" THEN
12690 PRINT " DISPLAY VALUES IS ";CHR$(129);" ON ";CHR$(128)
12700 ELSE
12710 PRINT " DISPLAY VALUES IS OFF"
12720 END IF
12730 PRINT
12740 PRINT TAB(5);CHR$(132);"TRANS (in.) ROT (deg)";CHR$(128);"
";CHR$(132);" FEMUR ANGLE TIBIA ANGLE";CHR$(128)
12750 PRINT
12760 Brice: IMAGE AA,2X,4D.2D,6X,4D.2D,18X,A,6X,4D.2D,8X,4D.2D
12770 PRINT USING Brice;"X ";Sys_trans_x;Sys_rot_x;"A";Fa;Ta
12780 PRINT USING Brice;"Y ";Sys_trans_y;Sys_rot_y;"B";Fb;Tb
12790 PRINT USING Brice;"Z ";Sys_trans_z;Sys_rot_z;"C";Fc;Tc

```

*CS*

```

12800 PRINT
12810 PRINT "PIVOT ANG A:";Sp;" PIVOT ANG B:";Tp;" PIVOT ANG C:";Up
12820 INPUT "HIT <CR> TO CONTINUE",Temp$
12830 CLEAR SCREEN
12840 ALPHA OFF
12850 GRAPHICS ON
12860 SUBEND
12870 !
12880 !*****
12890 !
12891 ! SUBROUTINE ROTATE LEG Z: ROTATES ENTIRE LEG ABOUT Z AXIS
12892 !
12893 !*****
12894 !
12900 SUB Rotate_leg_z(Femur(*),Femurmod(*),Leg$,Theta,Printflag$,OPTIONAL Femur
a_rot,Way)
12910 OPTION BASE 1
12920 DIM Bogus(4,4),Temp(196,4),Tempa(196,4),Trans(4,4)
12930 N=SIZE(Femur,1)
12940 REDIM Tempa(N,4),Temp(N,4)
12950!
12960 IF Way=1 THEN
12970 Femura_rot=Femura_rot+Theta
12980!
12990 IF Printflag$="ON" THEN
13000 DISP " MODE: ";Leg$;" THROUGH ANGLE OF ";Theta;" TOTAL ANGLE=";F
emura_rot
13010 END IF
13020 END IF
13030!
13040 MAT Tempa= (1)
13050 MAT Tempa(*,1:3)= Femur(*,1:3)
13060!
13070! SET UP ROTATION MATRIX
13080!
13090 MAT Trans= IDN
13100 Sine=SIN(Theta)
13110 Cosine=COS(Theta)
13120 Trans(1,1)=Cosine
13130 Trans(1,2)=-Sine
13140 Trans(?,1)=Sine
13150 Trans(2,2)=Cosine
13160!
13170! FIND NEW SKITTER MATRIX
13180!
13190 MAT Temp= Tempa*Trans
13200 MAT Femurmod(*,1:3)= Temp(*,1:3)
13210 SUBEND
13220!
13230!*****
13240!
13241 ! SUBROUTINE TRANS FEM ORIG: TRANSLATES LEG TO ORIGIN
13242 !
13243 !*****
13244 !
13250 SUB Trans_fem_orig(Femur(*),Femurmod(*),X,Y,Z)
13260 !
13270 OPTION BASE 1
13280 DIM Bogus(4,4),Tempa(196,4),Temp(196,4),Trans(4,4)
13290 N=SIZE(Femur,1)

```

```

13300 REDIM Tempa(N,4),Temp(N,4)
13310 !
13320 ! SET UP STORAGE ARRAY TO KEEP SKITTER PENS CORRECT SKITTER(*,4)
13330 !
13340     MAT Tempa= (1)
13350     MAT Tempa(*,1:3)= Femur(*,1:3)
13360     MAT Trans= IDN
13370 !
13380 ! SET UP TRANS MATRIX
13390 !
13400     Trans(4,1)=X
13410     Trans(4,2)=Y
13420     Trans(4,3)=Z
13430 !
13440 ! FIND NEW MATRIX
13450 !
13460     MAT Temp= Tempa*Trans
13470     MAT Femurmod(*,1:3)= Temp(*,1:3)
13480 SUBEND
13490 !
13500 !*****
13510 !
13511 ! SUBROUTINE ROTATE LEG Y: ROTATES LEG ABOUT Y AXIS
13512 !
13513 !*****
13514 !
13520 SUB Rotate_leg_y(Femur(*),Femurmod(*),Theta)
13530!
13540     OPTION BASE 1
13550     DIM Bogus(4,4),Temp(196,4),Tempa(196,4),Trans(4,4)
13560     N=SIZE(Femur,1)
13570     REDIM Tempa(N,4),Temp(N,4)
13580!
13590     MAT Tempa= (1)
13600     MAT Tempa(*,1:3)= Femur(*,1:3)
13610!
13620! SET UP ROTATION MATRIX
13630!
13640     MAT Trans= IDN
13650     Sine=SIN(Theta)
13660     Cosine=COS(Theta)
13670     Trans(1,1)=Cosine
13680     Trans(1,3)=Sine
13690     Trans(3,1)=-Sine
13700     Trans(3,3)=Cosine
13710!
13720! FIND NEW SKITTER MATRIX
13730!
13740     MAT Temp= Tempa*Trans
13750     MAT Femurmod(*,1:3)= Temp(*,1:3)
13760     SUBEND
13770!
13780!
13790!*****
13800!
13801 ! SUBROUTINE ZOOM PAN: ALOOWS USER TO PAN OR ZOOM WINDOW
13802 !
13803 !*****
13804 !
13810 SUB Zoom_pan(Window$,Xmin,Xmax,Ymin,Ymax,Skitmod(*),Newskit(*),Screen_x,Sc

```

```

seen_y)
13820      ON ERROR GOTO Brice
13830!
13840      DISP " YOUR CURRENT WINDOW VALUES ARE XMIN:";Xmin;" XMAX:";Xmax;" YM
IN:";Ymin;" YMAX:";Ymax
13850 Menu:
!
13860      ON KNOB .15 GOTO Knob_isr
13870      ON KEY 9 LABEL "QUIT" GOTO Leave
13880      ON KEY 0 LABEL "ZOOM" GOTO Zoom
13890      ON KEY 1 LABEL "" GOTO 13970
13900      ON KEY 2 LABEL "PAN X" GOTO Pan_x
13910      ON KEY 3 LABEL "" GOTO 13970
13920      ON KEY 4 LABEL "PAN Y" GOTO Pan_y
13930      ON KEY 5 LABEL "" GOTO 13970
13940      ON KEY 6 LABEL "INPUT DATA " GOTO Input_data
13950      ON KEY 7 LABEL "" GOTO 13970
13960      ON KEY 8 LABEL "" GOTO 13970
13970      GOTO 13970
13980      !
13990 Pan_x:!
4000      Window$="PAN X"
4010      GOTO Menu
4020 Pan_y:!
4030      Window$="PAN Y"
4040      GOTO Menu
4050 Zoom:!
4060      Window$="ZOOM"
4070      GOTO Menu
4080 Knob_isr:!
4090!
4100      Theta=KNOBX
4110      IF Window$="ZOOM" AND Theta>0 THEN
4120          Xmin=Xmin-5
4130          Xmax=Xmax+5
4140          Ymin=Ymin-5
4150          Ymax=Ymax+5
4160          SHOW Xmin,Xmax,Ymin,Ymax
4170      END IF
4180!
4190!
4200      IF Window$="ZOOM" AND Theta<0 THEN
4210          Xmin=Xmin+5
4220          Xmax=Xmax-5
4230          Ymin=Ymin+5
4240          Ymax=Ymax-5
4250          SHOW Xmin,Xmax,Ymin,Ymax
4260      END IF
4270!
4280!
4290      IF Window$="PAN X" AND Theta<0 THEN
4300          Xmin=Xmin+5
4310          Xmax=Xmax+5
4320          SHOW Xmin,Xmax,Ymin,Ymax
4330      END IF
4340!
4350!
4360      IF Window$="PAN X" AND Theta>0 THEN
4370          Xmin=Xmin-5
4380          Xmax=Xmax-5
4390          SHOW Xmin,Xmax,Ymin,Ymax

```

```

14400      END IF
14410!
14420!
14430      IF Window$="PAN Y" AND Theta>0 THEN
14440          Ymin=Ymin+5
14450          Ymax=Ymax+5
14460          SHOW Xmin,Xmax,Ymin,Ymax
14470      END IF
14480!
14490!
14500      IF Window$="PAN Y" AND Theta<0 THEN
14510          Ymin=Ymin-5
14520          Ymax=Ymax-5
14530          SHOW Xmin,Xmax,Ymin,Ymax
14540      END IF
14550!
14560      CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
14570      GOTO 13840
14580 !
14590 !
14600 Input_data:!
14610      DISP " INPUT XMIN --- CURRENT VALUE IS",Xmin," <CR> TO EXIT";
14620      LINPUT Temp$
14630      IF Temp$="" THEN GOTO Leave
14640      DISP " INPUT XMAX --- CURRENT VALUE IS",Xmax," <CR> TO EXIT";
14650      LINPUT Temp1$
14660      IF Temp1$="" THEN GOTO Leave
14670      IF VAL(Temp$)>VAL(Temp1$) THEN
14680          BEEP 1464.84,.5
14690          DISP " XMIN HAS TO BE LESS THAN XMAX"
14700          WAIT 3
14710          GOTO 14610
14720      ELSE
14730          Xmin=VAL(Temp$)
14740          Xmax=VAL(Temp1$)
14750      END IF
14760!
14770!
14780!
14790      DISP " INPUT YMIN ---- CURRENT VALUE IS",Ymin," <CR> TO EXIT";
14800      LINPUT Temp$
14810      IF Temp$="" THEN GOTO Leave
14820      DISP " INPUT YMAX ---- CURRENT VALUE IS",Ymax," <CR> TO EXIT";
14830      LINPUT Temp1$
14840      IF Temp1$="" THEN GOTO Leave
14850      IF VAL(Temp$)>VAL(Temp1$) THEN
14860          BEEP 1464.84,.5
14870          DISP " Y MIN MUST BE LESS THAN Y MAX"
14880          WAIT 3
14890          GOTO 14790
14900      ELSE
14910          Ymin=VAL(Temp$)
14920          Ymax=VAL(Temp1$)
14930      END IF
14940!
14950!
14960      SHOW Xmin,Xmax,Ymin,Ymax
14970      CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
14980      GOTO Menu
14990      !

```

```

15000 Brice: !
15010     IF ERRN=31 THEN
15020         BEEP 1464,.5
15030         DISP " THIS IS AS FAR AS YOU CAN GO !!!!!!!!!!!!!!!!!!!!!!"
15040         WAIT 2
15050         Xmin=Xmin-5
15060         Xmax=Xam+5
15070         Ymin=Ymin-5
15080         Ymax=Ymax+5
15090         SHOW Xmin,Xmax,Ymin,Ymax
15100         CALL Display_skit(Skitmod(*),Newskit(*),Screen_x,Screen_y)
15110         GOTO 13840
15120     END IF
15130 Leave: !
15140     SUBEND
15150 !*****
15160 !
15161 ! SUBROUTINE PLOT IT:  SETS UP PLOTTER PARAMETERS FOR HARD COPY
15162 !
15163 !*****
15164 !
15170 SUB Plot_it(Plot_device)
15180 DIM L$(32)
15190 DISP " HOW MANY QUADRANTS 1,2 OR 4  DEFAULT=1";
15200 LINPUT Quad$
15210 IF Quad$="" THEN
15220     Quad=1
15230 ELSE
15240     Quad=VAL(Quad$)
15250 END IF
15260 IF Quad=1 THEN GOTO Label
15270 IF Quad<>1 AND Quad<>2 AND Quad<>4 THEN
15280     BEEP 1464,.5
15290     GOTO 15190
15300 END IF
15310 DISP " WHICH SQUARE  DEFAULT=1";
15320 LINPUT What$
15330 IF What$="" THEN
15340     What=1
15350 ELSE
15360     What=VAL(What$)
15370 END IF
15380 IF What<>1 AND What<>2 AND What<>3 AND What<>4 THEN
15390     BEEP 1464,.5
15400     GOTO 15310
15410 END IF
15420 !
15430 !
15440 Label: !
15450 DISP " INPUT ANY LABELS  DEFAULT= NONE";
15460 LINPUT L$
15470 IF L$="" THEN GOTO P1
15480 P1: !
15490 PRINTER IS Plot_device
15500 IF Quad=1 THEN
15510     PRINT "IN;IP;SP1;SI .5,.5;PA 425,1000,LB";L$;CHR$(3);";SP0;"
15520 END IF
15530 IF Quad=4 THEN
15540     IF What=1 THEN
15550         PRINT "IN;IP250,596,5250,4196;SP1;SI;PA 425,900,LB";L$;CHR$(3);";S

```



```

P0;"
15560      END IF
15570      IF What=2 THEN
15580          PRINT "IN;IP5250,596,10250,4196;SP1;SI;PA 5425,900,LB";L$;CHR$(3);
";SP0;"
15590      END IF
15600      IF What=3 THEN
15610          PRINT "IN;IP250,4196,5250,7796;SP1;SI;PA 425 4500,LB";L$;CHR$(3);"
";SP0;"
15620      END IF
15630      IF What=4 THEN
15640          PRINT "IN;IP5250,4196,10250,7796;SP1,SI,PA 5425, 4500,LB";L$;CHR$(3
);";SP0;"
15650      END IF
15660      END IF
15670      IF Quad=2 THEN
15680          IF What=2 THEN
15690              PRINT "IN;RO90;IP;IW;IP154,244,7354,5122;SP1;SI .35,.35;PA 600,6
00;LB";L$;CHR$(3);";SP0;"
15700          ELSE
15710              PRINT "IN;RO90;IP;IW;IP154,5122,7354,10244;SP1;SI .35,.35;PA 600
,5478;LB";L$;CHR$(3);";SP0;"
15720          END IF
15730      END IF
15740      Leave: !
15750      SUBEND
15751 !
15752 !
15753 !
15754 !
15755 !*****
15756 !
15757 !   SUBROUTINE MOVIE:   ALLOWS USER TO INPUT DATA FILES FOR ANIMATION
15758 !                   OF POSITION SEQUENCES
15759 !
15760 !*****
15761 !
15763 SUB Movie
15770 REM *****
15780 DATA 0,0,0,10 ! UPPER BODY
15790 DATA 6.5482,28.4912,0,-2 ! A
15800 DATA 9.1602,20.9817,3.2283,-1 ! H
15810 DATA -1.7468,20.9817,9.5688,-1 ! I
15820 DATA -3.2741,28.4912,5.6709,-1 ! B
15830 DATA 0,0,0,7
15840 !
15850 DATA 0,0,0,10
15860 DATA -3.2741,28.4912,5.6709,-2 ! B
15870 DATA -7.3759,20.9817,6.3188,-1 ! J
15880 DATA -7.3759,20.9817,-6.3188,-1 ! K
15890 DATA -3.2741,28.4912,-5.6709,-1 ! C
15900 DATA 0,0,0,7
15910 !
15920 DATA 0,0,0,10
15930 DATA -3.2741,28.4912,-5.6709,-2 ! C
15940 DATA -1.7468,20.9817,-9.5688,-1 ! L
15950 DATA 9.1602,20.9817,-3.2283,-1 ! G
15960 DATA 6.5482,28.4912,0,-1 ! A
15970 DATA 0,0,0,7
15980 !

```

)

BOTTOM BODY

15990 DATA	0,0,0,10	!	
16000 DATA	6.5482,13.4722,0,-2	!	D
16010 DATA	9.1602,20.9817,3.2283,-1	!	H
16020 DATA	-1.7468,20.9817,9.5688,-1	!	I
16030 DATA	-3.2741,13.4722,5.6709,-1	!	E
16040 DATA	0,0,0,7		
16050 !			
16060 DATA	0,0,0,10		
16070 DATA	-3.2741,13.4722,5.6709,-2	!	E
16080 DATA	-7.3759,20.9817,6.3188,-1	!	J
16090 DATA	-7.3759,20.9817,-6.3188,-1	!	K
16100 DATA	-3.2741,13.4722,-5.6709,-1	!	F
16110 DATA	0,0,0,7		
16120 !			
16130 DATA	0,0,0,10		
16140 DATA	-3.2741,13.4722,-5.6709,-2	!	F
16150 DATA	-1.7468,20.9817,-9.5688,-1	!	L
16160 DATA	9.1602,20.9817,-3.2283,-1	!	G
16170 DATA	6.5482,13.4722,0,-1	!	D
16180 DATA	0,0,0,7		
16190 !			
16200 DATA	0,0,0,4		! RESERVED FOR PEN
16210 DATA	0,0,0,10	!	
16220 DATA	9.1602,20.9817,-3.2263,-2	!	G
16230 DATA	29.6602,20.9817,-3.2283,-1	!	M
16240 DATA	18.7362,25.0731,0,-1	!	O
16250 DATA	0,0,0,7		
16260 !			
16270 !			
16280 DATA	0,0,0,10		
16290 DATA	29.6602,20.9817,3.2283,-2	!	N
16300 DATA	18.7362,25.0731,0,-1	!	O
16310 DATA	9.1602,20.9817,3.2283,-1	!	H
16320 DATA	0,0,0,7		
16330 !			
16340 DATA	0,0,0,10		
16350 DATA	9.1602,20.9817,-3.2283,-2	!	G
16360 DATA	29.6602,20.9817,-3.2283,-1	!	M
16370 DATA	20.571,16.8903,0,-1	!	P
16380 DATA	0,0,0,7		
16390 !			
16400 DATA	0,0,0,10		
16410 DATA	29.6602,20.9817,3.2283,-2	!	N
16420 DATA	20.571,16.8903,0,-1	!	P
16430 DATA	9.1602,20.9817,3.2283,-1	!	H
16440 DATA	0,0,0,7		
16450 !			
16460 DATA	0,0,0,10	!	
16470 DATA	29.6602,20.9817,-3.2283,-2	!	M
16480 DATA	30.4255,0,0,-1	!	A'
16490 DATA	35.7901,9.2132,0,-1	!	B'
16500 DATA	0,0,0,7		
16510 !			
16520 DATA	0,0,0,10		
16530 DATA	29.6602,20.9817,3.2283,-2	!	N
16540 DATA	30.4255,0,0,-1	!	A'
16550 DATA	35.7901,9.2132,0,-1	!	B'
16560 DATA	0,0,0,7		
16570 !			
16580 DATA	0,0,0,4		! RESERVED FOR PEN

FEMUR ONE

TIBIA ONE

16590 DATA	0,0,0,10	!		FEMUR TWO
16600 DATA	-1.7468,20.9817,9.5688,-2	!	I	
16610 DATA	-9.3493,25.0731,16.2368,-1	!	S	
16620 DATA	-11.9968,20.9817,27.3223,-1!		Q	
16630 DATA	0,0,0,7			
16640 !				
16650 DATA	0,0,0,10			
16660 DATA	-17.6259,20.9817,24.0723,-2!		R	
16670 DATA	-9.3493,25.0731,16.2368,-1	!	S	
16680 DATA	-7.3759,20.9817,6.3188,-1	!	J	
16690 DATA	0,0,0,7			
16700 !				
16710 DATA	0,0,0,10			
16720 DATA	-1.7468,20.9817,9.5688,-2	!	I	
16730 DATA	-10.2667,16.8903,17.8258,-1!		T	
16740 DATA	-11.9968,20.9817,27.3223,-1!		Q	
16750 DATA	0,0,0,7			
16760 !				
16770 DATA	0,0,0,10			
16780 DATA	-17.6259,20.9817,24.0723,-2!		R	
16790 DATA	-10.2667,16.8903,17.8258,-1!		T	
16800 DATA	-7.3759,20.9817,6.3188,-1	!	J	
16810 DATA	0,0,0,7			
16820 !				
16830 !				
16840 DATA	0,0,0,10	!		TIBIA TWO
16850 DATA	-11.9968,20.9817,27.3223,-2	!	Q	
16860 DATA	-15.1940,0,26.3601,-1	!	C'	
16870 DATA	-17.8763,9.2132,31.0059,-1	!	D'	
16880 DATA	0,0,0,7			
16890 !				
16900 DATA	0,0,0,10			
16910 DATA	-17.6259,20.9817,24.0723,-2	!	N	
16920 DATA	-15.1940,0,26.3601,-1	!	C'	
16930 DATA	-17.8763,9.2132,31.0059,-1	!	D'	
16940 DATA	0,0,0,7			
16950 !				
16960 DATA	0,0,0,4	!	RESERVED FOR PEN	
16970 DATA	0,0,0,10	!		FEMUR THREE
16980 DATA	-7.3759,20.9817,-6.3188,-2	!	K	
16990 DATA	-9.3493,25.0731,-16.2368,-1!		W	
17000 DATA	-17.6259,20.9817,-24.0723,-1!		U	
17010 DATA	0,0,0,7			
17020 !				
17030 DATA	0,0,0,10			
17040 DATA	-11.9968,20.9817,-27.3223,-2	!	V	
17050 DATA	-9.3493,25.0731,-16.2368,-1	!	W	
17060 DATA	-1.7468,20.9817,-9.5688,-1	!	L	
17070 DATA	0,0,0,7			
17080 !				
17090 DATA	0,0,0,10			
17100 DATA	-7.3759,20.9817,-6.3188,-2	!	K	
17110 DATA	-10.2667,16.8903,-17.8258,-1	!	X	
17120 DATA	-17.6259,20.9817,-24.0723,-1	!	U	
17130 DATA	0,0,0,7			
17140 !				
17150 !				
17160 DATA	0,0,0,10			
17170 DATA	-11.9968,20.9817,-27.3223,-2	!	V	
17180 DATA	-10.2667,16.8903,-17.8258,-1!		X	

```

17190 DATA      -1.7468,20.9817,-9.5588,-1  !  L
17200 DATA      0,0,0,7
17210 !
17220 DATA      0,0,0,10  !
17230 DATA      -17.6259,20.9817,-24.0723,-2 !  U
17240 DATA      -15.1940,0,-26.3601,-1  !  E'
17250 DATA      -17.8763,9.2132,-31.0059,-1 !  F'
17260 DATA      0,0,0,7
17270 !
17280 DATA      0,0,0,10
17290 DATA      -11.9968,20.9817,-27.3223,-2 !  V
17300 DATA      -15.1940,0,-26.3601,-1  !  E'
17310 DATA      -17.8763,9.2132,-31.0059,-1 !  F'
17320 DATA      0,0,0,7
17330 REM      *****
17340 !
17350 !
17360      OPTION BASE 1
17370      REAL Skitter(129,4),Newskit(129,3)      ! DEFINE VAR
17380      REAL Trans(4,4),Temp(129,4),Tempa(129,4)!TRANSFORMATION MATRIX
17390      REAL Total(4,4),Skitmod(129,4)          !TOTAL TRANFORM MATRIX
17400      REAL Femur(31,4),Femurmod(31,4),Femurtemp(31,4)
17410      REAL Tibia(10,4),Tibiatemp(10,4),Tibiamod(10,4)
17420      INTEGER Pen1(1,3),Pen2(1,3),Pen3(1,3)
17430      GOSUB Init                                ! INITILIZATION ROUTINE
17440      CALL Display_skit(Skitter(*),Newskit(*),Screen_x,Screen_y)
17450      ! DRAW SKITTER
17460 !
17470 DATA 1,1,4      !PEN1
17480 DATA 4,4,4      !PEN2
17490 DATA 8,8,4      !PEN3
17500 !
17510 Start:!
17520 DISP "DO YOU WANT TO RUN AN ALREADY COMPUTED FILE  Y OR N";
17530 LINPUT Ans$
17550 !
17560 !
17561 ! INPUT DATA FILE
17562 !
17570      IF Ans$="Y" THEN
17580          DISP "NAME OF COMPUTED FILE";
17590          LINPUT Name$
17600      IF Name$="" THEN
17610          GOTO 17520
17620      ELSE
17630          Skitwork$=Name$
17640          File_flag=1
17650          GOTO Movie
17660      END IF
17670      END IF
17680 !
17690 IF Ans$="N" THEN
17700 DISP "INPUT MOVIE FILE";
17710 LINPUT File$
17720 IF File$="" THEN
17730      GOTO 17700
17740 ELSE
17750 Skitwork$="SKITWORK"
17760 END IF
17770      CREATE BDAT Skitwork$,400

```

TIBIA THREE

```

17780      ASSIGN @Path1 TO Skitwork$
17790      ASSIGN @Path2 TO File$
17800      END IF
17810!
17820 READ Pen1(*), Pen2(*), Pen3(*)
17830!
17840 Top: !
17850      ENTER @Path2;Tim,Free,Fadat,Tadat,Fbdat,Tbdat,Fcdat,Tcdat,Xr,Yr,Zr,Tx
,Ty,Tz
17851 !*****
17852 !
17853 !   DETERMINE TRANSFORMATION MATRICES FOR EACH TIME UNIT
17854 !
17855 !*****
17856 !
17860      IF Tim=999 THEN GOTO Movie
17870      N=N+1
17880      IF Free=0 THEN Freeleg$="FIXED"
17890      IF Free=1 OR Free=2 THEN Freeleg$="FREE"
17900 !
17910      IF Fadat<>0 AND Fadat<>999 THEN
17920          Leg$="FEMUR A"
17930          IF Free=2 THEN
17940              Leg_flag$="FEMUR A"
17960          END IF
17970          Theta=Fadat
17980          GOSUB Leg_isr
17990      END IF
18000 !
18010      IF Fadat=999 THEN
18020          Leg_flag$="FEMUR A"
18030      END IF
18040 !
18050      IF Tadat<>0 THEN
18060          Leg$="TIBIA A"
18070          Theta=Tadat
18080          GOSUB Leg_isr
18090      END IF
18100 !
18110      IF Fbdat<>0 AND Fbdat<>999 THEN
18120          Leg$="FEMUR B"
18130          IF Free=2 THEN
18140              Leg_flag$="FEMUR B"
18150          END IF
18160          Theta=Fbdat
18170          GOSUB Leg_isr
18180      END IF
18190 !
18200      IF Fbdat=999 THEN
18210          Leg_flag$="FEMUR B"
18220      END IF
18230 !
18240      IF Tbdat<>0 THEN
18250          Leg$="TIBIA B"
18260          Theta=Tbdat
18270          GOSUB Leg_isr
18280      END IF
18290 !
18300      IF Fcdat<>0 AND Fcdat<>999 THEN
18310          Leg$="FEMUR C"

```

```

18320         IF Free=2 THEN
18330             Leg_flag$="FEMUR C"
18340         END IF
18350             Theta=Fcdat
18360             GOSUB Leg_isr
18370         END IF
18380 !
18390         IF Fcdat=999 THEN
18400             Leg_flag$="FEMUR C"
18410         END IF
18420 !
18430         IF Tcdat<>0 THEN
18440             Leg$="TIBIA C"
18450             Theta=Tcdat
18460             GOSUB Leg_isr
18470         END IF
18480 !
18490 !
18500         IF Yr<>0 THEN
18510             Twirl$="ROTATE Y"
18520             Theta=Yr
18530             GOSUB System_isr
18540         END IF
18550 !
18560         IF Xr<>0 THEN
18570             Twirl$="ROTATE X"
18580             Theta=Xr
18590             GOSUB System_isr
18600         END IF
18610 !
18620         IF Zr<>0 AND Free<>2 THEN
18630             Twirl$="ROTATE Z"
18640             Theta=Zr
18650             GOSUB System_isr
18660         END IF
18670 !
18680         IF Zr<>0 AND Free=2 THEN
18690             Twirl$="PIVOT"
18710             Theta=Zr
18720             GOSUB System_isr
18730         END IF
18740 !
18750         IF Tx<>0 THEN
18760             Way=1
18770             Theta=Tx
18780             Twirl$="TRANSLATE"
18790             CALL Printmat(Total(*))
18800             GOSUB System_isr
18810         END IF
18820         IF Ty<>0 THEN
18830             Way=2
18840             Theta=Ty
18850             Twirl$="TRANSLATE"
18860             GOSUB System_isr
18870         END IF
18880         IF Tz<>0 THEN
18890             Way=3
18900             Theta=Tz
18910             Twirl$="TRANSLATE"
18920             GOSUB System_isr

```

```

18930      END IF
18940 !
18950 !
18960 !
18970      MAT Tempa(*,1:3)= Skitter(*,1:3)
18980      MAT Temp= Tempa*Total
18990      MAT Skitmod(*,1:3)= Temp(*,1:3)
19000      MAT Newskit(*,1)= Skitmod(*,Screen_x)
19010      MAT Newskit(*,2)= Skitmod(*,Screen_y)
19020      MAT Newskit(*,3)= Skitmod(*,4)
19030      MAT Newskit(37:37,*)= Pen1
19040      MAT Newskit(68:68,*)= Pen2
19050      MAT Newskit(99:99,*)= Pen3
19060      OUTPUT @Path1;Newskit(*)
19070      GOTO Top
19080!
19090!
19100 Movie:!
19110      ASSIGN @Path1 TO *
19120      ASSIGN @Path2 TO *
19130      CALL Display_movie(N,Skitwork$)
19140      IF File_flag=1 THEN GOTO Finished
19150 !
19160 Save_file:!
19170      DISP "DO YOU WANT TO SAVE THE WORK FILE Y OR N";
19180      LINPUT Ans$
19190      IF Ans$<>"Y" AND Ans$<>"N" THEN GOTO 19170
19200      IF Ans$="N" THEN GOTO Delete
19210      IF Ans$="Y" THEN
19220          DISP "NAME OF FILE TO SAVE UNDER";
19230          LINPUT Name$
19250          Where$=Name$
19260          RENAME Skitwork$ TO Where$
19270          DISP "FILE SAVED UNDER";Where$
19280          GOTO Finished
19290      END IF
19300 Delete:!
19310      PURGE Skitwork$
19320      GOTO Finished
19330 System_isr:      !
19340                  SELECT Twirl$
19350                  CASE "ROTATE Y"
19360                  CALL Rotate_y(Skitter(*),Skitmod(*),Total(*),Trans(*),Te
mp(*),Tempa(*),Sys_rot_y,Theta,Printflag$)
19370                  CASE "ROTATE X"
19380                  CALL Rotate_x(Skitter(*),Skitmod(*),Total(*),Trans(*),Te
mp(*),Tempa(*),Sys_rot_x,Theta,Printflag$)
19390!
19400                  CASE "ROTATE Z"
19410                  Theta=-Theta
19420                  CALL Rotate_z(Skitter(*),Skitmod(*),Total(*),Trans(*),Te
mp(*),Tempa(*),Sys_rot_z,Theta,Printflag$)
19430!
19440                  CASE "TRANSLATE"
19450                  CALL Translate_3d(Skitter(*),Skitmod(*),Total(*),Temp(*),
Tempa(*),Trans(*),Sys_trans_x,Sys_trans_y,Sys_trans_z,Way,Theta,Printflag$)
19460!
19470                  CASE "PIVOT"
19480                  IF Leg_flag$="FEMUR A" THEN
19490                      Theta=-Theta

```

```

19500 Pivotx=(Skitter(96,1)+Skitter(127,1))/2
19510 Pivoty=(Skitter(96,2)+Skitter(127,2))/2
19520 Pivotz=(Skitter(96,3)+Skitter(127,3))/2
19530 Dist_piv_foot=SQR((Pivotx-Skitter(65,1))^2+(Pivoty-Sk
itter(65,2))^2+(Pivotz-Skitter(65,3))^2)
19540 CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
voty,-Pivotz)
19550 Flag=0
19560 CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Theta,Pr
intflag$,Fem_a_ang,Flag)
19570 CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
ty,Pivotz)
19580 MAT Skitter= Skitmod
19590 END IF
19600!
19610 IF Leg_flag$="FEMUR B" THEN
19620 Pivotx=(Skitter(65,1)+Skitter(127,1))/2
19630 Pivoty=(Skitter(65,2)+Skitter(127,2))/2
19640 Pivotz=(Skitter(65,3)+Skitter(127,3))/2
19650 Dist_piv_foot=SQR((Pivotx-Skitter(96,1))^2+(Pivoty-Sk
itter(96,2))^2+(Pivotz-Skitter(96,3))^2)
19660 CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
voty,-Pivotz)
19670 CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
19680 Flag=0
19690 CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Theta,Pr
intflag$,Fem_b_ang,Flag)
19700 CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
19710 CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
ty,Pivotz)
19720 MAT Skitter= Skitmod
19730 END IF
19740!
19750 IF Leg_flag$="FEMUR C" THEN
19760 Pivotx=(Skitter(65,1)+Skitter(96,1))/2
19770 Pivoty=(Skitter(65,2)+Skitter(96,2))/2
19780 Pivotz=(Skitter(65,3)+Skitter(96,3))/2
19790 Dist_piv_foot=SQR((Pivotx-Skitter(129,1))^2+(Pivoty-S
kitter(129,2))^2+(Pivotz-Skitter(129,3))^2)
19800 CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
voty,-Pivotz)
19810 CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
19820 Flag=0
19830 CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Theta,Pr
intflag$,Fem_c_ang,Flag)
19840 CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
19850 CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
ty,Pivotz)
19860 MAT Skitter= Skitmod
19870 END IF
19880!
19890 END SELECT
19900 RETURN
19910!
19920 Leg_isr:
19930 SELECT Leg$
19940 CASE "FEMUR A"
19950 MAT Femur= Skitter(37:67,*)
19960 CALL Trans_fem_orig(Femur(*),Femurmod(*),-Femur(3,1),
-Femur(3,2),-Femur(3,3))

```



```

20480          Dist_piv_foot=SQR((Pivotx-Skitter(96,1))^2+(Pivoty-Sk
Skitter(96,2))^2+(Pivotz-Skitter(96,3))^2)
20490          Dist_y=Pivoty-Skitter(96,2)
20500          Rot_ang=ASN(Dist_y/Dist_piv_foot)
20510          CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
voty,-Pivotz)
20520          CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
20530          Flag=0
20540          CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Rot_ang,
Printflag$,Fem_b_ang,Flag)
20550          CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
20560          CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
ty,Pivotz)
20570          MAT Skitter= Skitmod
20580          MAT Tempa(*,1:3)= Skitter(*,1:3)
20590          MAT Temp= Tempa*Total
20600          MAT Skitmod(*,1:3)= Temp(*,1:3)
20610 !
20620 !
20630          CASE "FEMUR C"
20640          MAT Femur= Skitter(99:129,*)
20650          CALL Trans_fem_orig(Femur(*),Femurmod(*),-Femur(3,1),
-Femur(3,2),-Femur(3,3))
20660          CALL Rotate_leg_y(Femurmod(*),Femurmod(*),-60)
20670          Flag=1
20680          CALL Rotate_leg_z(Femurmod(*),Femurmod(*),Leg$,Theta,
Printflag$,Fem_c_ang,Flag)
20690          CALL Rotate_leg_y(Femurmod(*),Femurmod(*),60)
20700          CALL Trans_fem_orig(Femurmod(*),Femurmod(*),Femur(3,1
),Femur(3,2),Femur(3,3))
20710          MAT Skitter(99:129,*)= Femurmod
20720!
20730          IF Freeleg$="FREE" THEN
20740          MAT Femurtemp(*,1:3)= Femurmod(*,1:3)
20750          MAT Femur= Femurtemp*Total
20760          MAT Skitmod(99:129,1:3)= Femur(*,1:3)
20770          GOTO End_leg
20780          END IF
20790!
20800          Pivotx=(Skitter(65,1)+Skitter(96,1))/2
20810          Pivoty=(Skitter(65,2)+Skitter(96,2))/2
20820          Pivotz=(Skitter(65,3)+Skitter(96,3))/2
20830          Dist_piv_foot=SQR((Pivotx-Skitter(127,1))^2+(Pivoty-S
Skitter(127,2))^2+(Pivotz-Skitter(127,3))^2)
20840          Dist_y=Pivoty-Skitter(127,2)
20850          Rot_ang=ASN(Dist_y/Dist_piv_foot)
20860          CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
voty,-Pivotz)
20870          CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
20880          Flag=0
20890          CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Rot_ang,
Printflag$,Fem_c_ang,Flag)
20900          CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
20910          CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
ty,Pivotz)
20920          MAT Skitter= Skitmod
20930          MAT Tempa(*,1:3)= Skitter(*,1:3)
20940          MAT Temp= Tempa*Total
20950          MAT Skitmod(*,1:3)= Temp(*,1:3)
20960!

```

```

20970!
20980!
20990      CASE "TIBIA A"
21000      MAT Tibia= Skitter(58:67,*)
21010      CALL Trans_fem_orig(Tibia(*),Tibiamod(*),-Tibia(2,1),
-Tibia(2,2),-Tibia(2,3))
21020      Flag=1
21030      CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),180)
21040      CALL Rotate_leg_z(Tibiamod(*),Tibiamod(*),Leg$,Theta,
Printflag$,Tib_a_ang,Flag)
21050      CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),180)
21060      CALL Trans_fem_orig(Tibiamod(*),Tibiamod(*),Tibia(2,1
),Tibia(2,2),Tibia(2,3))
21070      MAT Skitter(58:67,*)= Tibiamod
21080!
21090      IF Freeleg$="FREE" THEN
21100      MAT Tibiatemp(*,1:3)= Tibiamod(*,1:3)
21110      MAT Tibia= Tibiatemp*Total
21120      MAT Skitmod(58:67,1:3)= Tibia(*,1:3)
21130      GOTO End_leg
21140      END IF
21150!
21160      Pivotx=(Skitter(96,1)+Skitter(127,1))/2
21170      Pivoty=(Skitter(96,2)+Skitter(127,2))/2
21180      Pivotz=(Skitter(96,3)+Skitter(127,3))/2
21190      Dist_piv_foot=SQR((Pivotx-Skitter(65,1))^2+(Pivoty-Sk
itter(65,2))^2+(Pivotz-Skitter(65,3))^2)
21200      Dist_y=Pivoty-Skitter(65,2)
21210      Rot_ang=ASN(Dist_y/Dist_piv_foot)
21220      CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
voty,-Pivotz)
21230      Flag=0
21240      CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,-Rot_ang
,Printflag$,Tib_a_ang,Flag)
21250      CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
ty,Pivotz)
21260      MAT Skitter= Skitmod
21270      MAT Tempa(*,1:3)= Skitter(*,1:3)
21280      MAT Temp= Tempa*Total
21290      MAT Skitmod(*,1:3)= Temp(*,1:3)
21300!
21310!
21320      CASE "TIBIA B"
21330      MAT Tibia= Skitter(89:98,*)
21340      CALL Trans_fem_orig(Tibia(*),Tibiamod(*),-Tibia(2,1),
-Tibia(2,2),-Tibia(2,3))
21350      CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),60)
21360      Flag=1
21370      CALL Rotate_leg_z(Tibiamod(*),Tibiamod(*),Leg$,Theta,
Printflag$,Tib_b_ang,Flag)
21380      CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),-60)
21390      CALL Trans_fem_orig(Tibiamod(*),Tibiamod(*),Tibia(2,1
),Tibia(2,2),Tibia(2,3))
21400      MAT Skitter(89:98,*)= Tibiamod
21410!
21420      IF Freeleg$="FREE" THEN
21430      MAT Tibiatemp(*,1:3)= Tibiamod(*,1:3)
21440      MAT Tibia= Tibiatemp*Total
21450      MAT Skitmod(89:98,1:3)= Tibia(*,1:3)
21460      GOTO End_leg

```

```

21470
21480!
21490
21500
21510
21520
21530
21540
21550
21560
21570
21580
21590
21600
21610
21620
21630
21640
21650!
21660!
21670
21680
21690
21700
21710
21720
21730
21740
21750
21760!
21770
21780
21790
21800
21810
21820
21830!
21840
21850
21860
21870
21880
21890
21900
21910
21920
21930
21940
21950

```

```

END IF

Pivotx=(Skitter(65,1)+Skitter(127,1))/2
Pivoty=(Skitter(65,2)+Skitter(127,2))/2
Pivotz=(Skitter(65,3)+Skitter(127,3))/2
Dist_piv_foot=SQR((Pivotx-Skitter(96,1))^2+(Pivoty-Sk
itter(96,2))^2+(Pivotz-Skitter(96,3))^2)
Dist_y=Pivoty-Skitter(96,2)
Rot_ang=ASN(Dist_y/Dist_piv_foot)
CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
voty,-Pivotz)

CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
Flag=0
CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Rot_ang,
Printflag$,Tib_b_ang,Flag)

CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
ty,Pivotz)

MAT Skitter= Skitmod
MAT Tempa(*,1:3)= Skitter(*,1:3)
MAT Temp= Tempa*Total
MAT Skitmod(*,1:3)= Temp(*,1:3)

CASE "TIBIA C"
MAT Tibia= Skitter(120:129,*)
CALL Trans_fem_orig(Tibia(*),Tibiamod(*),-Tibia(2,1),
-Tibia(2,2),-Tibia(2,3))

CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),-60)
Flag=1
CALL Rotate_leg_z(Tibiamod(*),Tibiamod(*),Leg$,Theta,
Printflag$,Tib_c_ang,Flag)

CALL Rotate_leg_y(Tibiamod(*),Tibiamod(*),60)
CALL Trans_fem_orig(Tibiamod(*),Tibiamod(*),Tibia(2,1
),Tibia(2,2),Tibia(2,3))

MAT Skitter(120:129,*)= Tibiamod

IF Freeleg$="FREE" THEN
MAT Tibiatemp(*,1:3)= Tibiamod(*,1:3)
MAT Tibia= Tibiatemp*Total
MAT Skitmod(120:129,1:3)= Tibia(*,1:3)
GOTO End_leg
END IF

Pivotx=(Skitter(65,1)+Skitter(96,1))/2
Pivoty=(Skitter(65,2)+Skitter(96,2))/2
Pivotz=(Skitter(65,3)+Skitter(96,3))/2
Dist_piv_foot=SQR((Pivotx-Skitter(127,1))^2+(Pivoty-S
kitter(127,2))^2+(Pivotz-Skitter(127,3))^2)
Dist_y=Pivoty-Skitter(127,2)
Rot_ang=ASN(Dist_y/Dist_piv_foot)
CALL Trans_fem_orig(Skitter(*),Skitmod(*),-Pivotx,-Pi
voty,-Pivotz)

CALL Rotate_leg_y(Skitmod(*),Skitmod(*),-60)
Flag=0
CALL Rotate_leg_z(Skitmod(*),Skitmod(*),Leg$,Rot_ang,
Printflag$,Tib_c_ang,Flag)

CALL Rotate_leg_y(Skitmod(*),Skitmod(*),60)
CALL Trans_fem_orig(Skitmod(*),Skitmod(*),Pivotx,Pivo
ty,Pivotz)

```

```

21960          MAT Skitter= Skitmod
21970          MAT Tempa(*,1:3)= Skitter(*,1:3)
21980          MAT Temp= Tempa*Total
21990          MAT Skitmod(*,1:3)= Temp(*,1:3)
22000 !
22010 End_leg:!
22020          END SELECT
22030          RETURN
22040!
22050!
22060 Windows:!
22070!          CALL Zoom_pan(Window$,Screenx_win_min,Screenx_win_max,Screeny_wi
n_min,Screeny_win_max,Skitmod(*),Newskit(*),Screen_x,Screen_y)
22080          GOTO Menu
22090!*****
22100!
22110 Init:          ! INITIALIZE SCREEN,SKITTER
22120!
22130          DEG          ! SET TO DEGREES
22140          GINIT          ! INITIATE GRAPHICS
22150          GRAPHICS ON          ! TURN G-PLANE ON
22160          PLOTTER IS CRT,"INTERNAL"          ! INIT PLOTTER
22170!
22180          Dump_device=9
22190!
22200          Plot_device=705
22210!
22220          READ Skitter(*)          ! READ SKITTER DATA
22230!
22240          MAT Skitmod= Skitter
22250!
22260          MAT Femurtemp= (1)
22270!
22280          MAT Tibiatemp= (1)
22290          MAT Trans= IDN          ! INIT TRANS MATRIX TO IDN
22300!
22310          MAT Tempa= (1)
22320!
22330          MAT Total= IDN
22340!
22350          Menu$="MAIN"          ! INIT MAIN MENU
22360!
22370          Twirl$="ROTATE Y"
22380!
22390          Freeleg$="FIXED"
22400!
22410          Way=1          ! AXIS OF TRANS X=1,Y=2,Z=3
22420!
22430          Printflag$="OFF"
22440!
22450          Increment=5          ! TRANS INC.
22460!
22470          Rot_increment=3
22480!
22490          Screen_x=1          ! INIT VIEW PLANE
22500          Screen_y=2          ! X=1,Y=2,Z=3
22510!
22520          Sys_trans_x=0          ! INIT POSITONS OF SYSTEM
22530          Sys_trans_y=0
22540          Sys_trans_z=0

```

```

22550 Sys_rot_x=0
22560 Sys_rot_y=0
22570 Sys_rot_z=0
22580!
22590 Fem_a_ang=0 ! INIT LEG ANGLES
22600 Fem_b_ang=0
22610 Fem_c_ang=0
22620 Tib_a_ang=90
22630 Tib_b_ang=90
22640 Tib_c_ang=90
22650!
22660 Screenx_win_max=40 ! SET WINDOW
22670 Screenx_win_min=-20
22680 Screeny_win_max=40
22690 Screeny_win_min=-20
22700 SHOW Screenx_win_min,Screenx_win_max,Screeny_win_min,Screeny_win_max
22710!
22720 RETURN
22730!*****
22740!
22750!
22760!*****
22770!
22780 Finished: ! DONE WITH PROGRAM
22790 GCLEAR
22800 CLEAR SCREEN
22810 SUBEND
22820!
22830 SUB Display_movie(N,Skitwork$)
22840 !
22850 OPTION BASE 1
22860 DIM Newskit(129,3)
22870 ON KBD GOTO Leave
22880 DISP "HIT ANY KEY TO QUIT"
22890 ASSIGN @Path1 TO Skitwork$
22900 ON END @Path1 GOTO Start_again
22910 ENTER @Path1;Newskit(*)
22920 CLEAR SCREEN
22930 GCLEAR
22940 FRAME
22950 AREA INTENSITY 0,0,.2
22960 MOVE -100,-.5
22970 RECTANGLE 200,.5,FILL,EDGE
22980 PLOT Newskit(*)
22990 LINE TYPE 1
23000 GOTO 22910
23010 Start_again:!
23020 ASSIGN @Path1 TO *
23030 GOTO 22890
23040 Leave:!
23050 ASSIGN @Path1 TO *
23060 SUBEND

```

---

**Appendix C**  
**SKITTER Dynamic Simulation**  
**Program Listing**

```

10  REM*****
20  REM
30  REM  THIS PROGRAM WILL ALLOW THE USER
40  REM  TO DETERMINE WHETHER OR NOT A
50  REM  PARTICULAR ACTUATOR WILL PROVIDE
60  REM  SUFFICIENT TORQUE AND ANGULAR
70  REM  VELOCITY TO HAVE SKITTER JUMP A
80  REM
90  REM  TO DO THIS, A INVERSE SLIDER CRANK
100 REM  MECHANISM WILL BE SIMULATED. JOINT
110 REM  ANGLES, VELOCITIES, AND ACCELERATIONS
120 REM  WILL BE CALCULATED ALONG WITH THE TORQUES.
130 REM
140 REM          PROGRAM WRITTEN BY:
150 REM          BRICE MACLAREN
160 REM          GARY MCMURRAY
170 REM
180 REM*****
190 RAD
200 OPTION BASE 1
210 REM*****
220 REM
230 DIM A(4),B(4),C(4),D(4),E(4),F(4),Rot1(4,4),Rot2(4,4),Rot3(4,4)
240 DIM Femur(4),Foot(4),Tibcon(4),Oflag(6),Trans1(4,4),Matrix(4,4)
250 DIM Trans2(4,4),Trans3(4,4),Newfem(4),Newfoot(4),Temp1(4),Dist(4)
260 DIM Newb(4),Newd(4),Newe(4),Newfem2(4),Temp2(4),Temp3(4)
270 DIM Ftorque(800),Ttorque(800),Fomega(800),Tomega(800),Fhp(800),Thp(200)
280 DIM Origb(4),Origd(4),Orige(4),Newfoot2(4)
290 DIM Temp$(8)
300 REM
310 REM*****
320 REM
330 REM          INITIALIZE VARIABLES
340 REM
350 Delstep=1
360 G=32.2
370 Jdist=6/12
380 Adist=3/12
390 Wgt=300
400 Mfemur=10/G
410 Mtibia=6/G
420 Flen=20/12
430 Tlen=20/12
440 Beta=0
450 Iota=PI/2
460 CALL Invar(Mfemur,Ifemur,Flen,Mtibia,Itibia,Tlen)
470 MAT Trans3= IDN
480 MAT Trans2= IDN
490 MAT Trans1= IDN
500 Angf=23.62
510 Angt=27.5
520 Aforce=100
530 Avel=5
540 Avel=Avel/12
550 Atorque=100
560 Aomega=2
570 REM*****
580 REM
590 REM  WHAT FOLLOWS ARE THE ORIGINAL POINT LOCATIONS FOR THE VARIOUS

```

```

600 REM CONNECT POINTS -- THEY ARE, IN ORDER, PNT.A,B,C,D,E,& F. PLEASE
610 REM THE DOCUMENTATION FOR DEFINITIONS OF THESE POINTS.
620 REM
630 REM*****
640 DATA -1.95,7.5095,0,1
650 DATA 9.576,4.09,0,1
660 DATA 0,0,0,1
670 DATA 11.411,-4.09,0,1
680 DATA 23.714,-11.7685,0,1
690 REM
700 READ A(*)
710 READ B(*)
720 READ C(*)
730 READ D(*)
740 READ E(*)
750 REM*****
760 REM
770 REM THE NEXT 3 DATA STATEMENTS ARE FOR THE ORIGINAL VECTORS OF THE
780 REM FEMUR, FOOT, AND THE CONNECTION POINT OF THE ACTUATOR TO THE
790 REM TIBIA (THE LAST TWO ARE RELATIVE TO THE FEMUR END POINT)
800 REM
810 REM*****
820 DATA 20,0,0,1
830 DATA -20,0,0,1
840 DATA 6.95,-11.71,0,1
850 REM
860 READ Femur(*)
870 READ Foot(*)
880 READ Tibcon(*)
890 ! CONVERT INCHES TO FEET
900 FOR I=1 TO 3
910 A(I)=A(I)/12
920 B(I)=B(I)/12
930 C(I)=C(I)/12
940 D(I)=D(I)/12
950 E(I)=E(I)/12
960 Femur(I)=Femur(I)/12
970 Foot(I)=Foot(I)/12
980 Tibcon(I)=Tibcon(I)/12
990 NEXT I
1000 Flag=0 ! FLAG CONTROLS WHICH MENU YOU GO TO
1010 Simflag=0 ! FLAG CONTROLS IF SIMULATION FILE IS DESIRED
1020 Rflag=1 ! FLAG CONTROLS TYPE OF RUN USER DESIRES
1030 FOR I=1 TO 6
1040 Oflag(I)=0 ! OFLAG CONROLS WHETHER DATA GOES TO CRT OR FILE
1050 NEXT I
1060 Menu: !
1070 CLEAR SCREEN
1080 CALL Invar(Mfemur,Ifemur,Flen,Mtibia,Itibia,Tlen)
1090 IF Flag<>0 THEN
1100 CALL Printvar(Jdist,Adist,Wgt,G,Mfemur,Flen,Mtibia,Tlen,Angf,
Angt,A(*),B(*),C(*),D(*),E(*),Beta,Iota,Aforce,Avel,Actflag,Atorque,Aomega)
1110 ELSE
1120 DISP "CHOOSE TYPE OF ACTUATOR"
1130 END IF
1140 REM
1150 REM NOW THE MENUS ARE DEFINED
1160 REM
1170 SELECT Flag
1180 CASE 0

```



```

1190     ON KEY 0 LABEL "ROTARY ACT." GOTO Ract
1200     ON KEY 1 LABEL "" GOTO Try
1210     ON KEY 2 LABEL "" GOTO Try
1220     ON KEY 3 LABEL "" GOTO Try
1230     ON KEY 4 LABEL "LINEAR ACT." GOTO Lact
1240     ON KEY 5 LABEL "" GOTO Try
1250     ON KEY 6 LABEL "" GOTO Try
1260     ON KEY 7 LABEL "STOP" GOTO St
1270     ON KEY 8 LABEL "" GOTO Try
1280     ON KEY 9 LABEL "" GOTO Try
1290     CASE 1
1300         ON KEY 0 LABEL "JUMP DISTANCE" GOTO Jd
1310         ON KEY 1 LABEL "ACCELERATION DIST" GOTO Ad
1320         ON KEY 2 LABEL "LEG PROPERTIES" GOTO Lp
1330         ON KEY 3 LABEL "JOINT LOCATIONS" GOTO Jnt
1340         ON KEY 4 LABEL "ACTUATORS" GOTO Act
1350         ON KEY 5 LABEL "CHANGE ACT" GOTO Cha
1360         ON KEY 6 LABEL "DATA FILES" GOTO Fn
1370         ON KEY 7 LABEL "STOP" GOTO St
1380         ON KEY 8 LABEL "PRINTER IS ?" GOTO Prnt
1390         ON KEY 9 LABEL "RUN DATA" GOTO Rn
1400     CASE 2
1410         ON KEY 0 LABEL "FEMUR MASS" GOTO Fm
1420         ON KEY 1 LABEL "" GOTO Try
1430         ON KEY 2 LABEL "INIT FEMUR ANGLE" GOTO Fang
1440         ON KEY 3 LABEL "" GOTO Try
1450         ON KEY 4 LABEL "LEG LENGTH" GOTO Llen
1460         ON KEY 5 LABEL "TIBIA MASS" GOTO Tm
1470         ON KEY 6 LABEL "" GOTO Try
1480         ON KEY 7 LABEL "INIT TIBIA ANGLE" GOTO Tang
1490         ON KEY 8 LABEL "MAIN MENU" GOTO Mm
1500         ON KEY 9 LABEL "RUN DATA" GOTO Rn
1510     CASE 3
1520         ON KEY 0 LABEL "POINT A" GOTO Ba1
1530         ON KEY 1 LABEL "" GOTO Try
1540         ON KEY 2 LABEL "POINT B" GOTO Ba2
1550         ON KEY 3 LABEL "" GOTO Try
1560         ON KEY 4 LABEL "POINT D" GOTO Ta1
1570         ON KEY 5 LABEL "" GOTO Try
1580         ON KEY 6 LABEL "POINT E" GOTO Ta2
1590         ON KEY 7 LABEL "" GOTO Try
1600         ON KEY 8 LABEL "MAIN MENU" GOTO Mm
1610         ON KEY 9 LABEL "RUN DATA" GOTO Rn
1620     CASE 4
1630         ON KEY 0 LABEL "FEMUR FILES" GOTO Cf
1640         ON KEY 1 LABEL "" GOTO Try
1650         ON KEY 2 LABEL "TIBIA FILES" GOTO Tibfiles
1660         ON KEY 3 LABEL "" GOTO Try
1670         ON KEY 4 LABEL "SIMULATION " GOTO Simfil
1680         ON KEY 5 LABEL "" GOTO Try
1690         ON KEY 6 LABEL "CLOSE FILES" GOTO Clf
1700         ON KEY 7 LABEL "" GOTO Try
1710         ON KEY 8 LABEL "MAIN MENU" GOTO Mm
1720         ON KEY 9 LABEL "RUN DATA" GOTO Rn
1730     CASE 5
1740         ON KEY 0 LABEL "TORQ VS JDIST" GOTO Ftj
1750         ON KEY 1 LABEL "" GOTO Try
1760         ON KEY 2 LABEL "OMEGA VS JDIST" GOTO Foj
1770         ON KEY 3 LABEL "" GOTO Try
1780         ON KEY 4 LABEL "HP VS JDIST" GOTO Fhj

```

```

1790     ON KEY 5 LABEL "" GOTO Try
1800     ON KEY 6 LABEL "" GOTO Try
1810     ON KEY 7 LABEL "" GOTO Try
1820     ON KEY 8 LABEL "MAIN MENU" GOTO Mm
1830     ON KEY 9 LABEL "RUN DATA" GOTO Rn
1840     CASE 6
1850     ON KEY 0 LABEL "TORQ VS JDIST" GOTO Ttj
1860     ON KEY 1 LABEL "" GOTO Try
1870     ON KEY 2 LABEL "OMEGA VS JDIST" GOTO Toj
1880     ON KEY 3 LABEL "" GOTO Try
1890     ON KEY 4 LABEL "HP VS JDIST" GOTO Thj
1900     ON KEY 5 LABEL "" GOTO Try
1910     ON KEY 6 LABEL "" GOTO Try
1920     ON KEY 7 LABEL "" GOTO Try
1930     ON KEY 8 LABEL "MAIN MENU" GOTO Mm
1940     ON KEY 9 LABEL "RUN DATA" GOTO Rn
1950     CASE 7
1960     ON KEY 0 LABEL "MAX FORCE" GOTO Actf
1970     ON KEY 1 LABEL "" GOTO Try
1980     ON KEY 2 LABEL "MAX VELOCITY" GOTO Actv
1990     ON KEY 3 LABEL "" GOTO Try
2000     ON KEY 4 LABEL "" GOTO Try
2010     ON KEY 5 LABEL "" GOTO Try
2020     ON KEY 6 LABEL "" GOTO Try
2030     ON KEY 7 LABEL "" GOTO Try
2040     ON KEY 8 LABEL "MAIN MENU" GOTO Mm
2050     ON KEY 9 LABEL "RUN DATA" GOTO Rn
2060     CASE 8
2070     ON KEY 0 LABEL "MAX TORQUE" GOTO Actt
2080     ON KEY 1 LABEL "" GOTO Try
2090     ON KEY 2 LABEL "MAX OMEGA" GOTO Acto
2100     ON KEY 3 LABEL "" GOTO Try
2110     ON KEY 4 LABEL "" GOTO Try
2120     ON KEY 5 LABEL "" GOTO Try
2130     ON KEY 6 LABEL "" GOTO Try
2140     ON KEY 7 LABEL "" GOTO Try
2150     ON KEY 8 LABEL "MAIN MENU" GOTO Mm
2160     ON KEY 9 LABEL "RUN DATA" GOTO Rn
2170     CASE 9
2180     ON KEY 0 LABEL "CRT" GOTO Pprt
2190     ON KEY 1 LABEL "" GOTO Try
2200     ON KEY 2 LABEL "" GOTO Try
2210     ON KEY 3 LABEL "" GOTO Try
2220     ON KEY 4 LABEL "LASER" GOTO Las
2230     ON KEY 5 LABEL "" GOTO Try
2240     ON KEY 6 LABEL "" GOTO Try
2250     ON KEY 7 LABEL "" GOTO Try
2260     ON KEY 8 LABEL "MAIN MENU" GOTO Mm
2270     ON KEY 9 LABEL "RUN DATA" GOTO Rn
2280     CASE 10
2290     ON KEY 0 LABEL "JUMP DISTANCE" GOTO Jd
2300     ON KEY 1 LABEL "" GOTO Try
2310     ON KEY 2 LABEL "" GOTO Try
2320     ON KEY 3 LABEL "" GOTO Try
2330     ON KEY 4 LABEL "ACCELERATION" GOTO Ad
2340     ON KEY 5 LABEL "" GOTO Try
2350     ON KEY 6 LABEL "" GOTO Try
2360     ON KEY 7 LABEL "" GOTO Try
2370     ON KEY 8 LABEL "MAIN MENU" GOTO Mm
2380     ON KEY 9 LABEL "RUN DATA" GOTO Rn

```

```

2390 CASE 11
2400 ON KEY 0 LABEL "FIND MAX VALS" GOTO Fmv
2410 ON KEY 1 LABEL "" GOTO Try
2420 ON KEY 2 LABEL "" GOTO Try
2430 ON KEY 3 LABEL "" GOTO Try
2440 ON KEY 4 LABEL "INCREMENTAL" GOTO Inc
2450 ON KEY 5 LABEL "" GOTO Try
2460 ON KEY 6 LABEL "" GOTO Try
2470 ON KEY 7 LABEL "" GOTO Try
2480 ON KEY 8 LABEL "MAIN MENU" GOTO Mm
2490 ON KEY 9 LABEL "RUN DATA" GOTO Rn
2500 END SELECT
2510 GOTO 2510
2520 !
2530 ! BEGIN SOFTKEY DEFINATIONS
2540 !
2550 Ract: ! ACTUATORS WILL BE ROTARY
2560 Actflag=1
2570 Flag=1
2580 GOTO Menu
2590 Lact: ! ACTUATORS WILL BE LINEAR
2600 Actflag=2
2610 Flag=1
2620 GOTO Menu
2630 Rcond: ! GO TO RUN CONDITIONS MENU
2640 Flag=11
2650 GOTO Menu
2660 Jd: ! INPUT A NEW JUMP DISTANCE
2670 DISP "DISTANCE YOU DESIRE SKITTER TO JUMP (IN INCHES)";
2680 INPUT Jdist
2690 Jdist=Jdist/12
2700 GOTO Menu
2710 Ad: ! INPUT A NEW ACCELERATION DISTANCE
2720 DISP "ACCELERATION DISTANCE FOR FOOT (IN INCHES)";
2730 INPUT Adist
2740 Adist=Adist/12
2750 GOTO Menu
2760 Lp: ! GO TO LEG MENU
2770 Flag=2
2780 GOTO Menu
2790 Jnt: ! GO TO JOINT MENU
2800 Flag=3
2810 GOTO Menu
2820 Act: ! GO TO ACTUATOR MENU
2830 IF Actflag=1 THEN
2840 Flag=8
2850 ELSE
2860 Flag=7
2870 END IF
2880 GOTO Menu
2890 Cha: ! CHANGE TYPE OF ACTUATORS
2900 Flag=0
2910 GOTO Menu
2920 Prnt: ! CHANGE PRINTER
2930 Flag=9
2940 GOTO Menu
2950 Try: ! UNDEFINED SOFTKEY CHOSEN, TRY AGAIN
2960 DISP "BAD CHOICE -- TRY AGAIN"
2970 GOTO Menu
2980 Fn: ! GO TO DATA FILE MENU

```

```

090      Flag=4
000      GOTO Menu
010 St:  !                               EXIT THE PROGRAM
020      PRINTER IS CRT
030      CLEAR SCREEN
040      DISP "PROGRAM TERMINATED"
050      WAIT 1
060      DISP "OR IS IT?????"
070      STOP
080 Rn:  !                               RUN PROGRAM WITH DATA AS IT IS
090      Flag=11
100      GOTO Menu
110 Fm:  !                               INPUT WEIGHT OF FEMUR
120      DISP "FEMUR WEIGHT (IN POUNDS FORCE) ";
130      INPUT Mfemur
140      Mfemur=Mfemur/G
150      GOTO Menu
160 Llen: !                               INPUT LENGTH OF LEGS
170      DISP "LEG LENGTH (IN INCHES)";
180      INPUT Flen
190      Flen=Flen/12
200      Tlen=Flen
210      GOTO Menu
220 Tm:  !                               INPUT WEIGHT OF TIBIA
230      DISP "TIBIA WEIGHT (IN POUNDS FORCE) ";
240      INPUT Mtibia
250      Mtibia=Mtibia/G
260      GOTO Menu
270 Fang: !                               INPUT INITIAL ANGLE OF FEMUR
280      DISP "INITIAL ANGLE BETWEEN FEMUR AND HORIZONTAL (IN DEGREES)";
290      INPUT Beta
300      Beta=Beta*PI/180
310      GOTO Menu
320 Tang: !                               INPUT INITIAL ANGLE BETWEEN
330      !                               FEMUR AND TIBIA
340      DISP "INITIAL ANGLE BETWEEN FEMUR AND TIBIA (IN DEGREES)";
350      INPUT Iota
360      Iota=Iota*PI/180
370      GOTO Menu
380 Mm:  !                               GO BACK TO MAIN MENU
390      Flag=1
400      GOTO Menu
410 Bal: !                               CHANGE PNT. A
420      DISP "POINT A - X COORDINATE (DEFAULT = ",A(1)*12,"IN.)";
430      LINPUT Temp$
440      IF Temp$="" THEN
450      ELSE
460          A(1)=VAL(Temp$)
470          A(1)=A(1)/12
480      END IF
490      DISP "POINT A - Y COORDINATE (DEFAULT = ",A(2)*12,"IN.)";
500      LINPUT Temp$
510      IF Temp$="" THEN
520      ELSE
530          A(2)=VAL(Temp$)
540          A(2)=A(2)/12
550      END IF
560      DISP "POINT A - Z COORDINATE (DEFAULT = ",A(3)*12,"IN.)";
570      LINPUT Temp$
580      IF Temp$="" THEN

```

```

1590     ELSE
1600         A(3)=VAL(Temp$)
1610         A(3)=A(3)/12
1620     END IF
1630     GOTO Menu
1640 Ba2:   !                               CHANGE PNT. B
1650     DISP "POINT B - X COORDINATE (DEFAULT = ",B(1)*12,"IN.)";
1660     LINPUT Temp$
1670     IF Temp$="" THEN
1680     ELSE
1690         B(1)=VAL(Temp$)
1700         B(1)=B(1)/12
1710     END IF
1720     DISP "POINT B - Y COORDINATE (DEFAULT = ",B(2)*12,"IN.)";
1730     LINPUT Temp$
1740     IF Temp$="" THEN
1750     ELSE
1760         B(2)=VAL(Temp$)
1770         B(2)=B(2)/12
1780     END IF
1790     DISP "POINT B - Z COORDINATE (DEFAULT = ",B(3)*12,"IN.)";
1800     LINPUT Temp$
1810     IF Temp$="" THEN
1820     ELSE
1830         B(3)=VAL(Temp$)
1840         B(3)=B(3)/12
1850     END IF
1860     GOTO Menu
1870 Ta1:   !                               CHANGE PNT. D
1880     DISP "POINT D - X COORDINATE (DEFAULT = ",D(1)*12,"IN.)";
1890     LINPUT Temp$
1900     IF Temp$="" THEN
1910     ELSE
1920         D(1)=VAL(Temp$)
1930         D(1)=D(1)/12
1940     END IF
1950     DISP "POINT D - Y COORDINATE (DEFAULT = ",D(2)*12,"IN.)";
1960     LINPUT Temp$
1970     IF Temp$="" THEN
1980     ELSE
1990         D(2)=VAL(Temp$)
2000         D(2)=D(2)/12
2010     END IF
2020     DISP "POINT D - Z COORDINATE (DEFAULT = ",D(3)*12,"IN.)";
2030     LINPUT Temp$
2040     IF Temp$="" THEN
2050     ELSE
2060         D(3)=VAL(Temp$)
2070         D(3)=D(3)/12
2080     END IF
2090     GOTO Menu
2100 Ta2:   !                               CHANGE PNT. E
2110     DISP "POINT E - X COORDINATE (DEFAULT = ",E(1)*12,"IN.)";
2120     LINPUT Temp$
2130     IF Temp$="" THEN
2140     ELSE
2150         E(1)=VAL(Temp$)
2160         E(1)=E(1)/12
2170     END IF
2180     DISP "POINT E - Y COORDINATE (DEFAULT = ",E(2)*12,"IN.)";

```

```

4190     LINPUT Temp$
4200     IF Temp$="" THEN
4210     ELSE
4220         Z(2)=VAL(Temp$)
4230         E(2)=E(2)/12
4240     END IF
4250     DISP "POINT E - Z COORDINATE (DEFAULT = ",E(3)*12,"IN.)";
4260     LINPUT Temp$
4270     IF Temp$="" THEN
4280     ELSE
4290         E(3)=VAL(Temp$)
4300         E(3)=E(3)/12
4310     END IF
4320     GOTO Menu
4330 Cf:   !           CREATE DATA FILES FOR FEMUR
4340         Flag=5
4350     GOTO Menu
4360 Tibfiles: !           CREATE DATA FILES FOR TIBIA
4370         Flag=6
4380     GOTO Menu
4390 Clf:   !           CLOSE ALL DATA FILES
4400     ASSIGN @Path1 TO *
4410     ASSIGN @Path2 TO *
4420     ASSIGN @Path3 TO *
4430     ASSIGN @Path4 TO *
4440     ASSIGN @Path5 TO *
4450     ASSIGN @Path6 TO *
4460     ASSIGN @Pathsim TO *
4470     FOR I=1 TO 6
4480         Oflag(I)=0
4490     NEXT I
4500     GOTO Menu
4510 Simfil: !           THIS CREATES A BDAT FILE THE OUTPUT OF THE
4520         !           ANGLES FOR THE SIMULATION OF SKITTER
4530     DISP "FILE NAME FOR THE SIMULATION FILE ";
4540     LINPUT Sim$
4550     CREATE BDAT Sim$,200
4560     ASSIGN @Pathsim TO Sim$
4570     Simflag=1
4580     GOTO Menu
4590 Ftj:   !           CREATE TORQUE vs JUMP DISTANCE FILE FOR FEMUR
4600     DISP "FILE NAME FOR TORQUE VS JUMP DISTANCE ";
4610     LINPUT Tvj$
4620     CREATE ASCII Tvj$,25
4630     ASSIGN @Path1 TO Tvj$
4640     Oflag(1)=1
4650     GOTO Menu
4660 Foj:   !           CREATE OMEGA vs JUMP DISTANCE FILE FOR FEMUR
4670     DISP "FILE NAME FOR OMEGA VS JUMP DISTANCE ";
4680     LINPUT Ovj$
4690     CREATE ASCII Ovj$,25
4700     ASSIGN @Path2 TO Ovj$
4710     Oflag(2)=1
4720     GOTO Menu
4730 Fhj:   !           CREATE HP vs JUMP DISTANCE FILE FOR FEMUR
4740     DISP "FILE NAME FOR HORSE POWER VS JUMP DISTANCE ";
4750     LINPUT Hvj$
4760     CREATE ASCII Hvj$,25
4770     ASSIGN @Path3 TO Hvj$
4780     Oflag(3)=1

```

```

4790      GOTO Menu
4800 Ttj:      !                      CREATE TORQUE vs JUMP DISTANCE FILE FOR TIBIA
4810      DISP "FILE NAME FOR TORQUE VS JUMP DISTANCE ";
4820      LINPUT Tvj$
4830      CREATE ASCII Tvj$,25
4840      ASSIGN @Path4 TO Tvj$
4850      Oflag(4)=1
4860      GOTO Menu
4870 Toj:      !                      CREATE OMEGA vs JUMP DISTANCE FILE FOR TIBIA
4880      DISP "FILE NAME FOR OMEGA VS JUMP DISTANCE ";
4890      LINPUT Ovj$
4900      CREATE ASCII Ovj$,25
4910      ASSIGN @Path5 TO Ovj$
4920      Oflag(5)=1
4930      GOTO Menu
4940 Thj:      !                      CREATE HP vs JUMP DISTANCE FILE FOR TIBIA
4950      DISP "FILE NAME FOR HORSE POWER VS JUMP DISTANCE ";
4960      LINPUT Hvj$
4970      CREATE ASCII Hvj$,25
4980      ASSIGN @Path6 TO Hvj$
4990      Oflag(6)=1
5000      GOTO Menu
5010 Actf:      !                      INPUT MAXIMUM FORCE ACTUATOR CAN EXERT
5020      DISP "MAXIMUM FORCE ACTUATOR CAN EXERT (IN POUNDS FORCE) ";
5030      INPUT Aforce
5040      GOTO Menu
5050 Actv:      !                      INPUT MAXIMUM VELOCITY ACTUATOR CAN ACHIEVE
5060      DISP "MAXIMUM ACTUATOR VELOCITY ACHIEVE (IN IN/SEC) ";
5070      INPUT Avel
5080      Avel=Avel/12
5090      GOTO Menu
5100 Actt:      !                      INPUT MAXIMUM TORQUE ACTUATOR CAN EXERT
5110      DISP "MAXIMUM TORQUE ACTUATOR CAN EXERT (IN FT-LBS) ";
5120      INPUT Atorque
5130      GOTO Menu
5140 Acto:      !                      INPUT MAXIMUM VELOCITY ACTUATOR CAN ACHIEVE
5150      DISP "MAXIMUM ACTUATOR OMEGA ACHIEVE (IN RAD/SEC) ";
5160      INPUT Aomega
5170      GOTO Menu
5180 Pcrt:      !                      MAKE PRINTER CRT
5190      PRINTER IS CRT
5200      Flag=1
5210      GOTO Menu
5220 Las:      !                      MAKE PRINTER THE LASER JET
5230      PRINTER IS 9
5240      Flag=1
5250      GOTO Menu
5260 Fmv:      !                      RUNS THE PROGRAM TO FIND MAX TORQUE & OMEGA
5270      DISP "NUMBER OF STEPS PER INCH OF ACCELERATION ";
5280      INPUT Delstep
5290      Delstep=1/Delstep
5300      Rflag=2
5310      Flag=1
5320      GOTO Rest
5330 Inc:      !                      RUNS THE PROGRAM ON AN INCREMENTAL SETTING
5340      Rflag=1
5350      Flag=1
5360      GOTO Rest
5370 Rest:      !                      RUN THE PROGRAM WITH THE VALUES AS SET
5380      Flag=1

```

```

5390          CLEAR SCREEN
5400 !
5410 !   IF NO FILE ASSIGNMENTS HAVE BEEN MADE, THEN OUTPUT IS TO THE CRT
5420 !
5430 IF Oflag(1)=0 THEN
5440     ASSIGN @Path1 TO CRT
5450 END IF
5460 IF Oflag(2)=0 THEN
5470     ASSIGN @Path2 TO CRT
5480 END IF
5490 IF Oflag(3)=0 THEN
5500     ASSIGN @Path3 TO CRT
5510 END IF
5520 IF Oflag(4)=0 THEN
5530     ASSIGN @Path4 TO CRT
5540 END IF
5550 IF Oflag(5)=0 THEN
5560     ASSIGN @Path5 TO CRT
5570 END IF
5580 IF Oflag(6)=0 THEN
5590     ASSIGN @Path6 TO CRT
5600 END IF
5610 !*****
5620 !
5630 !           START THE PROGRAM
5640 !
5650 !*****
5660 !
5670 !
5680 !   DEFINE YOUR END CONDITIONS
5690 !
5700 Maxvel=SQR(2*G*Jdist)
5710 Totaltime=ABS(2*Adist/Maxvel)
5720 Accel=Maxvel/Totaltime
5730 !
5740 !   DETERMINE ANGLES FOR THIS POSITION OF THE FOOT
5750 !
5760 CALL Findfoot(-1*Beta,Iota,Femur(*),Foot(*),Newfoot(*),Newfem(*))
5770 !
5780 !
5790 !   DEFINE ORIGINAL LOCATION OF ACTUATOR CONNECT POINTS FOR THIS
5800 !   CONFIGURATION OF THE LEGS
5810 !
5820 CALL Rot(-1*Beta,Trans1(*))
5830 MAT Origb= Trans1*B
5840 MAT Origd= Trans1*D
5850 MAT Temp2= E-Femur
5860 Temp2(4)=1
5870 MAT Temp1= Trans3*Temp2
5880 Xnew=Newfoot(1)
5890 Ynew=Newfoot(2)
5900 Lac=SQR(Xnew^2+(Ynew)^2)
5910 CALL Sss(Tlen,Lac,Flen,A3)
5920 A3=A3-PI/2
5930 CALL Rot(A3,Trans2(*))
5940 CALL Trans(Newfem(*),Trans2(*))
5950 MAT Orige= Trans2*Temp1
5960 !
5970 !   SET END POINTS FOR LOOPING OVER SLIDER CRANK
5980 !

```



```

5990 I=1 ! COUNTER
6000 IF Rflag=1 THEN
6010 Delstep=1
6020 Y0=Delstep
6030 ELSE
6040 Y0=Delstep
6050 END IF
6060 Endpnt=Adist*12
6070 Theta2=-1*Beta
6080 Iota2=Iota
6090 !
6100 ! BEGIN LOOP OVER ACCELERATION DISTANCE
6110 !
6120 FOR Y=Y0 TO Endpnt STEP Delstep
6130 Y1=Y/12
6140 T=ABS(SQR(2*Y1/Accel))
6150 V=Accel*T
6160 !
6170 ! DETERMINE JOINT ANGLES
6180 !
6190 Xnew=Newfoot(1)
6200 Ynew=Newfoot(2)
6210 Lac=SQR(Xnew^2+(Ynew-(Y1))^2)
6220 Psi=ATN((Ynew-(Y1))/Xnew)
6230 CALL Sss(Flen,Tlen,Lac,A3)
6240 IF Psi<0. THEN A3=-1*A3
6250 Theta=Psi-A3
6260 IF Psi>0. THEN
6270 Gamma=PI/2-Psi
6280 ELSE
6290 Gamma=Psi+PI/2
6300 END IF
6310 CALL Sss(Lac,Flen,Tlen,A3)
6320 IF Gamma>0. THEN
6330 Phi=Gamma-A3
6340 ELSE
6350 Phi=A3-Gamma
6360 END IF
6370 !
6380 ! NOW OUTPUT ANGLES FOR SIMULATION IF SIMFLAG=1
6390 !
6400 IF Simflag=1 THEN
6410 CALL Sss(Tlen,Lac,Flen,Iota1)
6420 Theta1=Theta
6430 Angle1=(Theta1-Theta2)*180/PI
6440 Angle2=(Iota1-Iota2)*180/PI
6450 Free=2.0
6460 OUTPUT @Pathsim;T
6470 OUTPUT @Pathsim;Free
6480 OUTPUT @Pathsim;Angle1
6490 OUTPUT @Pathsim;Angle2
6500 Theta2=Theta1
6510 Iota2=Iota1
6520 END IF
6530 !
6540 ! NOW CALCULATE THE PHYSICAL PARAMETERS
6550 !
6560 Fomega(I)=ABS((V*COS(Phi))/(Flen*(COS(Theta)*COS(Phi)-SIN(Theta)*SIN(P
hi))))
6570 Tomega(I)=ABS((Fomega(I)*SIN(Theta))/COS(Phi))

```

```

6580      Numbbc=-1*Accel*Tlen*SIN(Theta)-Tomega(I)^2*Tlen^2*(COS(Phi)*SIN(Theta
)+SIN(Phi)*COS(Theta))+Fomega(I)^2*Flen^2*(COS(Theta)^2-SIN(Theta)^2)
6590      Talpha=Numbbc/(-1*Tlen^2*(COS(Phi)*COS(Theta)+SIN(Phi)*SIN(Theta))
6600      Falpha=(Talpha*Tlen*COS(Phi)/2-Tomega(I)^2*Tlen*SIN(Phi)+Fomega(I)^2*F
len*COS(Theta))/(Tlen*SIN(Theta))
6610      Axtibia=-1*Talpha*Tlen*COS(Phi)/2+Tomega(I)^2*Tlen*SIN(Phi)/2
6620      Aytibia=-1*Accel-Talpha*SIN(Phi)*Tlen/2-Tomega(I)^2*Tlen*COS(Phi)/2
6630      Fbx=Mtibia*Axtibia
6640      Fby=Wgt/3-Mtibia*Aytibia-Mtibia*G
6650      Ftorque(I)=ABS(Ifemur*Falpha+Fbx*Flen*SIN(Theta)+Mfemur*G*Flen*COS(The
ta)/2-Fby*Flen*COS(Theta))
6660      Ttorque(I)=ABS(-1*Itibia*Talpha-Mtibia*G*Tlen*SIN(Phi)/2+Wgt*Tlen*SIN(
Phi)/2)
6670      Fhp(I)=ABS(Ftorque(I)*Fomega(I)/550)
6680      Thp(I)=ABS(Ttorque(I)*Tomega(I)/550)
6690      !
6700      !       IF ACTUATOR IS A LINEAR ONE, THEN CALCULATE THE MOMENT ARM
6710      !       IF IT IS A ROTARY ONE, THEN COMPARE TORQUE'S AND OMEGA'S NEEDED
6720      !       WITH THE ONES YOU CAN SUPPLY
6730      !
6740      Cnt=0      !CNT COUNTS THE NUMBER OF LINES PRINTED ON THE SCREEN SO THAT
6750      !           SCROLLING OF THE SCREEN CAN BE PREVENTED
6760      !       IF Actflag=2 THEN
6770      !
6780      !       NOW, CALCULATE THE MOMENT ARM FOR THE LINEAR ACTUATOR
6790      !
6800      MAT Trans3= IDN
6810      MAT Trans2= IDN
6820      CALL Rot(Theta,Trans3(*))
6830      MAT Newfem= Trans3*Femur
6840      MAT Newb= Trans3*B
6850      MAT Newd= Trans3*D
6860      MAT Temp2= E-Femur
6870      Temp2(4)=1
6880      MAT Temp1= Trans3*Temp2
6890      CALL Sss(Tlen,Lac,Flen,A3)
6900      A3=A3-PI/2
6910      CALL Rot(A3,Trans2(*))
6920      CALL Trans(Newfem(*),Trans2(*))
6930      MAT Newe= Trans2*Temp1
6940      CALL Eqline(A(*),Newb(*),C(*),Farm)
6950      CALL Eqline(Newd(*),Newe(*),Newfem(*),Tarm)
6960      A3=A3+PI/2
6970      CALL Rot(A3,Trans2(*))
6980      MAT Temp1= Trans3*Foot
6990      MAT Newfoot2= Trans2*Temp1
7000      !
7010      !       CAN GIVEN ACTUATOR SUPPLY POWER?
7020      !
7030      Test=ABS(Ftorque(I)/Farm)
7040      Test2=ABS(Fomega(I)*Farm)
7050      Testf=ABS(Ttorque(I)/Tarm)
7060      Testv=ABS(Tomega(I)*Tarm)
7070      IF Test>Aforce THEN
7080          IF Rflag=1 THEN
7090              PRINT CHR$(129)
7100              PRINT "FEMUR ACTUATOR IS NOT GOOD ENOUGH - FORCE NOT ENOUGH!!!"
7110              PRINT CHR$(128)
7120              Minarm=Ftorque(I)/Aforce
7130              PRINT USING "29A,DDDD.DD";"MINIMUM MOMENT ARM NEEDED IS ";Minarm*1

```

```

2;" IN."
7140 PRINT USING "29A,DDD.DD";"PRESENTLY, HAVE MOMENT ARM = ";Farm*12;"
IN."
7150 PRINT USING "30A,DDDDDD.DD,17A,DDDD.DD";"OR INCREASE ACTUATOR FORC
E TO ";Test;" LBF. INSTEAD OF ";Aforce;" LBF"
7160 PRINT
7170 Cnt=Cnt+7
7180 ELSE
7190 Femfflag=2
7200 END IF
7210 END IF
7220 IF Test2>Avel THEN
7230 IF Rflag=1 THEN
7240 PRINT CHR$(129)
7250 PRINT "FEMUR ACTUATOR IS NOT GOOD ENOUGH - VELOCITY NOT ENOUGH!!!"
7260 PRINT CHR$(128)
7270 Maxarm=Avel/Fomega(I)
7280 PRINT USING "49A,DDD.DD";"MAXIMUM MOMENT ARM POSSIBLE FOR THIS ACTU
ATOR IS ";Maxarm*12;" IN."
7290 PRINT USING "29A,DDD.DD";"PRESENTLY, HAVE MOMENT ARM = ";Farm*12;"
IN."
7300 PRINT USING "33A,DDD.DD,19A,DDD.DD";"OR INCREASE ACTUATOR VELOCITY
TO ";Test2*12;" IN/SEC INSTEAD OF ";Avel*12;" IN/SEC"
7310 PRINT
7320 Cnt=Cnt+7
7330 ELSE
7340 Femvflag=2
7350 END IF
7360 END IF
7370 IF Cnt>9 THEN
7380 DISP "HIT ANY KEY TO CONTINUE";
7390 ON KBD GOTO 7410
7400 GOTO 7400
7410 Cnt=0
7420 CLEAR SCREEN
7430 END IF
7440 IF Testf>Aforce THEN
7450 IF Rflag=1 THEN
7460 PRINT CHR$(129)
7470 PRINT "TIBIA ACTUATOR IS NOT GOOD ENOUGH - FORCE NOT ENOUGH!!!"
7480 PRINT CHR$(128)
7490 Minarm=Ttorque(I)/Aforce
7500 PRINT USING "29A,DDD.DD";"MINIMUM MOMENT ARM NEEDED IS ";Minarm*12;
IN."
7510 PRINT USING "29A,DDD.DD";"PRESENTLY, HAVE MOMENT ARM = ";Tarm*12;"
IN."
7520 PRINT USING "30A,DDDDDD.DD,17A,DDD.DD";"OR INCREASE ACTUATOR FORCE
TO ";Testf;" LBF. INSTEAD OF ";Aforce;" LBF"
7530 PRINT
7540 Cnt=Cnt+7
7550 ELSE
7560 Tibfflag=2
7570 END IF
7580 END IF
7590 IF Cnt>9 THEN
7600 DISP "HIT ANY KEY TO CONTINUE";
7610 ON KBD GOTO 7630
7620 GOTO 7620
7630 Cnt=0
7640 CLEAR SCREEN

```

```

650     END IF
660     IF Testv>Avel THEN
670         IF Rflag=1 THEN
680             PRINT CHR$(129)
690             PRINT "TIBIA ACTUATOR IS NOT GOOD ENOUGH - VELOCITY NOT ENOUGH!!!"
700             PRINT CHR$(128)
710             Maxarm=Avel/Tomega(I)
720             PRINT USING "49A,DDD.DD";"MAXIMUM MOMENT ARM POSSIBLE FOR THIS ACTU
FOR IS ";Maxarm*12;" IN."
730             PRINT USING "29A,DDD.DD";"PRESENTLY, HAVE MOMENT ARM = ";Tarm*12;"
v."
740             PRINT USING "33A,DDD.DD,19A,DDD.DD";"OR INCREASE ACTUATOR VELOCITY
) ";Testv*12;" IN/SEC INSTEAD OF ";Avel*12;" IN/SEC"
750             PRINT
760             Cnt=Cnt+7
770             ELSE
780             Tibvflag=2
790             END IF
300     END IF
310     !
320     !     NOW, FOR A ROTARY ACTUATOR
330     !
340     ELSE
350         IF Ftorque(I)>Atorque THEN
360             IF Rflag=1 THEN
370                 PRINT CHR$(129)
380                 PRINT "FEMUR ACTUATOR IS NOT GOOD ENOUGH - TORQUE NOT ENOUGH"
390                 PRINT CHR$(128)
400                 PRINT USING "15A,DDD.DD,35A,DDD.DD";"NEED TORQUE OF ";Ftorque(I);"
r-LBS,BUT ACTUATOR ONLY SUPPLIES ";Atorque
410                 PRINT
420                 Cnt=Cnt+5
430                 ELSE
440                 Femfflag=2
450                 END IF
460             END IF
470             IF Fomega(I)>Aomega THEN
480                 IF Rflag=1 THEN
490                     PRINT CHR$(129)
500                     PRINT "FEMUR ACTUATOR IS NOT GOOD ENOUGH - OMEGA NOT ENOUGH"
510                     PRINT CHR$(128)
520                     PRINT USING "14A,DDD.DD,36A,DDD.DD";"NEED OMEGA OF ";Fomega(I);" RA
SEC,BUT ACTUATOR ONLY SUPPLIES ";Aomega
530                     PRINT
540                     Cnt=Cnt+5
550                     ELSE
560                     Femtflag=2
570                     END IF
580                 END IF
590                 IF Ttorque(I)>Atorque THEN
600                     IF Rflag=1 THEN
610                         PRINT CHR$(129)
620                         PRINT "TIBIA ACTUATOR IS NOT GOOD ENOUGH - TORQUE NOT ENOUGH"
630                         PRINT CHR$(128)
640                         PRINT USING "15A,DDD.DD,35A,DDD.DD";"NEED TORQUE OF ";Ttorque(I);"
-LBS,BUT ACTUATOR ONLY SUPPLIES ";Atorque
650                         PRINT
660                         Cnt=Cnt+5
670                         ELSE
680                         Tibfflag=2

```

```

8190         END IF
8200     END IF
8210     IF Cnt>11 THEN
8220         DISP "HIT ANY KEY TO CONTINUE";
8230         ON KBD GOTO 8250
8240         GOTO 8240
8250         Cnt=0
8260         CLEAR SCREEN
8270     END IF
8280     IF Tomega(I)>Aomega THEN
8290         IF Rflag=1 THEN
8300             PRINT CHR$(129)
8310             PRINT "TIBIA ACTUATOR IS NOT GOOD ENOUGH - OMEGA NOT ENOUGH"
8320             PRINT CHR$(128)
8330             PRINT USING "14A,DDD.DD,36A,DDD.DD";"NEED OMEGA OF ";Tomega(I);" RA
D/SEC,BUT ACTUATOR ONLY SUPPLIES ";Aomega
8340             PRINT
8350             Cnt=Cnt+5
8360             ELSE
8370                 Tibvflag=2
8380             END IF
8390         END IF
8400     END IF
8410     !
8420     !             LOOP BACK FOR ACCELERATION DISTANCE
8430     !
8440         IF Rflag=1 THEN
8450             CALL Outdata(Jdist,Adist,Ftorque(*),Ttorque(*),Fomega(*),Tomega(*),Fhp
(*) ,Thp(*),@Path1,@Path2,@Path3,@Path4,@Path5,@Path6,Oflag(*),Y,Cnt,I)
8460             END IF
8470             I=I+1
8480     NEXT Y
8490     !
8500     !             WRITE LINE TO END SIMULATION FILE
8510     !
8520     IF Simflag=1 THEN
8530         Endnum=999.0
8540         OUTPUT @Pathsim;Endnum
8550         OUTPUT @Pathsim;Free
8560         OUTPUT @Pathsim;Free
8570         OUTPUT @Pathsim;Free
8580     END IF
8590     !
8600     !             NOW CALCULATE THE STROKE LENGTH FOR THE ACTUATOR
8610     !
8620     Fsl=ABS(SQR((Origb(1)-A(1))^2+(Origb(2)-A(2))^2)-SQR((Newb(1)-A(1))^2+(New
b(2)-A(2))^2))
8630     Tsl=ABS(SQR((Origd(1)-Orige(1))^2+(Origd(2)-Orige(2))^2)-SQR((Newd(1)-Newe
(1))^2+(Newd(2)-Newd(2))^2))
8640     !
8650     CLEAR SCREEN
8660     !
8670     !             IF THE MAXIMUM VALUES WERE DESIRED, THEN THIS SECTION
8680     !             DETERMINES THE MAXIMUM TORQUE AND ANGULAR VELOCITY
8690     !
8700     IF Actflag=2 THEN
8710         PRINT USING "22A,DD.DD";"FEMUR STROKE LENGTH = ";Fsl*12;" IN."
8720         PRINT USING "22A,DD.DD";"TIBIA STROKE LENGTH = ";Tsl*12;" IN."
8730         PRINT
8740     END IF

```

```

8750     IF Femfflag=2 THEN
8760     PRINT "FEMUR ACTUATOR NOT GOOD ENOUGH !!  NEED MORE TORQUE ABOUT JOI
NT !!"
8770     END IF
8780     IF Femvflag=2 THEN
8790     PRINT "FEMUR ACTUATOR NOT GOOD ENOUGH !!  NEED MORE ANGULAR VELOCITY
ABOUT JOINT !!"
8800     END IF
8810     IF Tibfflag=2 THEN
8820     PRINT "TIBIA ACTUATOR NOT GOOD ENOUGH !!  NEED MORE TORQUE ABOUT JOI
NT !!"
8830     END IF
8840     IF Tibvflag=2 THEN
8850     PRINT "TIBIA ACTUATOR NOT GOOD ENOUGH !!  NEED MORE ANGULAR VELOCITY
ABOUT JOINT !!"
8860     END IF
8870     PRINT
8880     MAT SEARCH Ftorque,MAX;Maxft
8890     MAT SEARCH Ttorque,MAX;Maxtt
8900     MAT SEARCH Fomega,MAX;Maxfo
8910     MAT SEARCH Tomega,MAX;Maxto
8920     MAT SEARCH Fhp,MAX;Maxfhp
8930     MAT SEARCH Thp,MAX;Maxthp
8940     PRINT USING "35A,DDDD.DD";"MAXIMUM TORQUE ABOUT THE FEMUR WAS ";Maxft;"
FT-LBS"
8950     PRINT USING "45A,DD.DDD";"MAXIMUM ANGULAR VELOCITY ABOUT THE FEMUR WAS
";Maxfo;" RAD/SEC"
8960     PRINT USING "35A,DDDD.DD";"MAXIMUM TORQUE ABOUT THE TIBIA WAS ";Maxtt;"
FT-LBS"
8970     PRINT USING "45A,DD.DDD";"MAXIMUM ANGULAR VELOCITY ABOUT THE TIBIA WAS
";Maxto;" RAD/SEC"
8980     PRINT USING "42A,D.DDD";"MAXIMUM HORSE POWER ABOUT THE FEMUR WAS - ";Ma
xfhp;" HP"
8990     PRINT USING "42A,D.DDD";"MAXIMUM HORSE POWER ABOUT THE TIBIA WAS - ";Ma
xthp;" HP"
9000     DISP "HIT ANY KEY TO CONTINUE";
9010     ON KBD GOTO 9030
9020     GOTO 9020
9030     !
9040     !   LOOP BACK TO THE MENU
9050     !
9060     GOTO Menu
9070     !
9080     !   STOP PROGRAM
9090     !
9100     END
9110     !
9120     !
9130     !
9140     SUB Printvar(Jdist,Adist,Wgt,G,Mfemur,Flen,Mtibia,Tlen,Angf,Angt,A(*),B(*),
.C(*),D(*),E(*),Beta,Iota,Aforce,Avel,Actflag,Actt,Acto)
9150     !
9160     !   THIS SUBROUTINE SIMPLY PRINTS THE VARIABLES THAT THE USER CAN
9170     !   CHNAGE TO THE SCREEN
9180     !
9190     PRINT "HERE ARE THE PRESET PARAMETERS:"
9200     PRINT " "
9210     PRINT "INITAL ANGLE FEMUR AND HORIZONTAL = ";Beta*180/PI;"DEG"
9220     PRINT "INITAL ANGLE FEMUR AND TIBIA = ";Iota*180/PI;"DEG"
9230     PRINT "JUMP DISTANCE = ";Jdist*12;"IN."

```

```

9240 PRINT "ACCELERATION DISTANCE = ";Adist*12;"IN."
9250 PRINT "WEIGHT OF SKITTER = ";Wgt;"LBF"
9260 PRINT "FEMUR'S WEIGHT = ";Mfemur*G;"LBF"
9270 PRINT "FEMUR'S LENGTH = ";Flen*12;"IN."
9280 PRINT "TIBIA'S WEIGHT = ";Mtibia*G;"LBF"
9290 PRINT "TIBIA'S LENGTH = ";Tlen*12;"IN."
9300 IF Actflag=2 THEN
9310 PRINT USING "37A,DDD.DD,2X,DDD.DD,2X,DDD.DD";"POINT A COORDINATES ARE
(IN INCHES): ";A(1)*12,A(2)*12,A(3)*12
9320 PRINT USING "37A,DDD.DD,2X,DDD.DD,2X,DDD.DD";"POINT B COORDINATES ARE
(IN INCHES): ";B(1)*12,B(2)*12,B(3)*12
9330 PRINT USING "37A,DDD.DD,2X,DDD.DD,2X,DDD.DD";"POINT C COORDINATES ARE
(IN INCHES): ";C(1)*12,C(2)*12,C(3)*12
9340 PRINT USING "37A,DDD.DD,2X,DDD.DD,2X,DDD.DD";"POINT D COORDINATES ARE
(IN INCHES): ";D(1)*12,D(2)*12,D(3)*12
9350 PRINT USING "37A,DDD.DD,2X,DDD.DD,2X,DDD.DD";"POINT E COORDINATES ARE
(IN INCHES): ";E(1)*12,E(2)*12,E(3)*12
9360 END IF
9370 IF Actflag=2 THEN
9380 PRINT USING "17A,DDDD.DD";"ACTUATOR FORCE = ";Aforce;" LBF"
9390 PRINT USING "20A,DDD.DD";"ACTUATOR VELOCITY = ";Avel*12;" IN/SEC"
9400 ELSE
9410 PRINT USING "18A,DDDD.DD";"ACTUATOR TORQUE = ";Actt;" FT-LBS"
9420 PRINT USING "17A,DDD.DD";"ACTUATOR OMEGA = ";Acto;" RAD/SEC"
9430 END IF
9440 !
9450 ! END SUBROUTINE
9460 !
9470 SUBEND
9480 !
9490 !
9500 !
9510 SUB Invar(Mfemur,Ifemur,Flen,Mtibia,Itibia,Tlen)
9520 !
9530 ! THIS SUBROUTINE DEFINES THE VARIABLES THAT NEED BE CALCULATED
9540 ! FROM OTHER INPUT VARIABLES
9550 !
9560 Ifemur=.0252+Mfemur*(Flen/2)^2
9570 Itibia=.01495+Mtibia*(Tlen/2)^2
9580 !
9590 ! END SUBROUTINE
9600 !
9610 SUBEND
9620 !
9630 !
9640 !
9650 SUB Outdata(Jdist,Adist,Ftorque(*),Ttorque(*),Fomega(*),Tomega(*),Fhp(*),T
hp(*),@Path1,@Path2,@Path3,@Path4,@Path5,@Path6,Oflag(*),Y,Cnt,I)
9660 !
9670 ! THIS SUBROUTINE OUTPUTS THE DATA
9680 !
9690 Temp=0
9700 FOR J=1 TO 6
9710 IF Oflag(J)=0 THEN Temp=1
9720 NEXT J
9730 IF Cnt>13 THEN
9740 DISP "HIT ANY KEY TO CONTINUE";
9750 ON KBD GOTO 9770
9760 GOTO 9760
9770 Cnt=0

```

```

9780     CLEAR SCREEN
9790     END IF
9800     IF Temp=1 THEN
9810         PRINT USING "DDD.D,32A";Y/(Adist*12)*100;" % THROUGH ACCELERATION DISTA
NCE"
9820         PRINT
9830         PRINT
9840         Cnt=Cnt+3
9850     END IF
9860     IF Cnt>14 THEN
9870         DISP "HIT ANY KEY TO CONTINUE";
9880         ON KBD GOTO 9900
9890         GOTO 9890
9900         Cnt=0
9910         CLEAR SCREEN
9920     END IF
9930     IF Temp=1 THEN
9940         PRINT USING "16A,DDD.DD";"JUMP DISTANCE = ";Jdist*12;" IN."
9950         PRINT USING "24A,DDD.DD";"ACCELERATION DISTANCE = ";Adist*12;" IN"
9960         Cnt=Cnt+2
9970     END IF
9980     IF Oflag(1)=1 THEN
9990         OUTPUT @Path1;Jdist*12,Ftorque(I)
10000    ELSE
10010        IF Cnt>15 THEN
10020            DISP "HIT ANY KEY TO CONTINUE";
10030            ON KBD GOTO 10050
10040            GOTO 10040
10050            Cnt=0
10060            CLEAR SCREEN
10070        END IF
10080        PRINT USING "15A,DDDDDD.DD";"FEMUR TORQUE = ";Ftorque(I);" FT-LB"
10090        Cnt=Cnt+1
10100    END IF
10110    IF Oflag(2)=1 THEN
10120        OUTPUT @Path2;Jdist*12,Fomega(I)
10130    ELSE
10140        IF Cnt>15 THEN
10150            DISP "HIT ANY KEY TO CONTINUE";
10160            ON KBD GOTO 10180
10170            GOTO 10170
10180            Cnt=0
10190            CLEAR SCREEN
10200        END IF
10210        PRINT USING "14A,DDDD.DD";"FEMUR OMEGA = ";Fomega(I);" RAD/SEC"
10220        Cnt=Cnt+1
10230    END IF
10240    IF Oflag(3)=1 THEN
10250        OUTPUT @Path3;Jdist*12,Fhp(I)
10260    ELSE
10270        IF Cnt>15 THEN
10280            DISP "HIT ANY KEY TO CONTINUE";
10290            ON KBD GOTO 10310
10300            GOTO 10300
10310            Cnt=0
10320            CLEAR SCREEN
10330        END IF
10340        PRINT USING "11A,DDDD.DD";"FEMUR HP = ";Fhp(I);" HP"
10350        Cnt=Cnt+1
10360    END IF

```



```

10370 IF Oflag(4)=1 THEN
10380   OUTPUT @Path4;Jdist*12,Ttorque(I)
10390 ELSE
10400   IF Cnt>15 THEN
10410     DISP "HIT ANY KEY TO CONTINUE";
10420     ON KBD GOTO 10440
10430     GOTO 10430
10440     Cnt=0
10450     CLEAR SCREEN
10460   END IF
10470   PRINT USING "15A, DDDD.DDDD";"TIBIA TORQUE = ";Ttorque(I);" FT-LB"
10480   Cnt=Cnt+1
10490 END IF
10500 IF Oflag(5)=1 THEN
10510   OUTPUT @Path5;Jdist*12,Tomega(I)
10520 ELSE
10530   IF Cnt>15 THEN
10540     DISP "HIT ANY KEY TO CONTINUE";-
10550     ON KBD GOTO 10570
10560     GOTO 10560
10570     Cnt=0
10580     CLEAR SCREEN
10590   END IF
10600   PRINT USING "14A, DDDD.DDDD";"TIBIA OMEGA = ";Tomega(I);" RAD/SEC"
10610   Cnt=Cnt+1
10620 END IF
10630 IF Oflag(6)=1 THEN
10640   OUTPUT @Path6;Jdist*12,Thp(I)
10650 ELSE
10660   IF Cnt>15 THEN
10670     DISP "HIT ANY KEY TO CONTINUE";
10680     ON KBD GOTO 10700
10690     GOTO 10690
10700     Cnt=0
10710     CLEAR SCREEN
10720   END IF
10730   PRINT USING "11A, DDDD.DDDD";"TIBIA HP = ";Thp(I);" HP"
10740   Cnt=Cnt+1
10750 END IF
10760 !
10770 IF Temp=1 THEN
10780   DISP "HIT ANY KEY TO CONTINUE";
10790   ON KBD GOTO 10810
10800   GOTO 10800
10810   CLEAR SCREEN
10820 END IF
10830 !
10840 !   END SUBROUTINE
10850 !
10860 SUBEND
10870 !
10880 !
10890 !
10900 SUB Rot(Angle,Matrix(*))
10910 !
10920 !   THIS SUBROUTINE FORMS THE TRANSFORMATION MATRIX FOR A ROTATION
10930 !   ABOUT THE Z-AXIS.  THE MATRIX IS A 4x4.
10940 !
10950 OPTION BASE 1
10960 Matrix(1,1)=COS(Angle)

```

```

10970 Matrix(2,2)=COS(Angle)
10980 Matrix(1,2)=-1*SIN(Angle)
10990 Matrix(2,1)=SIN(Angle)
11000 FOR I=1 TO 4
11010     FOR J=1 TO 4
11020         IF ABS(Matrix(I,J))<.0000001 THEN Matrix(I,J)=0.
11030     NEXT J
11040 NEXT I
11050 !
11060 !     END SUBROUTINE
11070 !
11080 SUBEND
11090 !
11100 !
11110 !
11120 SUB Trans(Dist(*),Matrix(*))
11130 !
11140 !     THIS FORMS A TRANSLATION MATRIX THAT OFFSETS A ROTATION MATRIX
11150 !
11160 OPTION BASE 1
11170 Matrix(1,4)=Dist(1)
11180 Matrix(2,4)=Dist(2)
11190 Matrix(3,4)=Dist(3)
11200 !
11210 !     END SUBROUTINE
11220 !
11230 SUBEND
11240 SUB Findfoot(Beta,Iota,Femur(*),Foot(*),Newfoot(*),Newfem(*))
11250 !
11260 !     THIS SUBROUTINE WILL FIND THE FOOT POSITION GIVEN THE ANGLES
11270 !     BETWEEN THE FEMUR & THE HORIZONTAL & THE ANGLE BETWEEN THE FEMUR
11280 !     AND THE TIBIA
11290 !
11300 OPTION BASE 1
11310 DIM Trans1(4,4),Trans2(4,4),Temp1(4)
11320 MAT Trans1= IDN
11330 MAT Trans2= IDN
11340 CALL Rot(Beta,Trans1(*))
11350 MAT Newfem= Trans1*Femur
11360 MAT Temp1= Trans1*Foot
11370 CALL Rot(Iota,Trans2(*))
11380 CALL Trans(Newfem(*),Trans2(*))
11390 MAT Newfoot= Trans2*Temp1
11400 !
11410 !     END SUBROUTINE
11420 !
11430 SUBEND
11440 !
11450 !
11460 !
11470 SUB Sss(S1,S2,S3,A3)
11480 !
11490 !     THIS SUBROUTINE DETERMINES ANGLE 3 FOR A TRIANGLE WHERE YOU
11500 !     KNOW THE 3 SIDES
11510 !
11520 P=(S1+S2+S3)/2
11530 Temp=SQR(P*(P-S2)/(S1*S3))
11540 IF ABS(Temp)>1.0 THEN
11550     PRINT "IMPOSSIBLE POSITION TO REACH!!!"
11560 ELSE

```

```

11570     A3=2*ACS(Temp)
11580 END IF
11590 !
11600 !   END SUBROUTINE
11610 !
11620 SUBEND
11630 !
11640 !
11650 !
11660 SUB Eqline(J(*),K(*),L(*),Marm)
11670 !
11680 !   THIS SUBROUTINE DETERMINES THE MOMENT ARM
11690 !
11700     OPTION BASE 1
11710     DIM R(4)
11720     M=(J(2)-K(2))/(J(1)-K(1))
11730     B1=J(2)-M*J(1)
11740     B2=L(2)+L(1)/M
11750     R(1)=(B2-B1)/(M+1/M)
11760     R(2)=M*R(1)+B1
11770     Marm=SQR((R(1)-L(1))^2+(R(2)-L(2))^2)
11780 !
11790 !   END SUBROUTINE
11800 !
11810 SUBEND

```

```

10  REM *****
20  !
30  !   THIS PROGRAM WILL DETERMINE THE TORQUE NEEDED TO LEAN
40  !   SKITTER THROUGH A DESIRED NUMBER OF DEGREES
50  !
60  !   THE NECESSARY TORQUE AND ANGULAR VELOCITY AT THE HIP
70  !
80  !   PROGRAM WRITTEN BY:
90  !   BRICE MACLAREN
100 !   GARY MCMURRAY
110 !
120 REM *****
130 OPTION BASE 1
140 RAD
150 DIM A(9,10),B(9),X(9)
160 !
170 !   THIS DATA DEFINES THE SYSTEM MATRIX THAT
180 !   MUST BE SOLVED IN THE DYNAMICS TO CALCULATE THE
190 !   INPUT TORQUE.
200 !
210 DATA 0,0,-1,0,1,0,0,0,0,0
220 DATA 0,0,0,-1,0,1,0,0,0,0
230 DATA 0,0,1,-1,0,0,0,0,0,0
240 DATA -1,0,1,0,0,0,0,0,0,0
250 DATA 0,-1,0,1,0,0,0,0,0,0
260 DATA 1,-1,0,0,0,0,0,0,1,0
270 DATA 1,0,0,0,0,0,1,0,0,0
280 DATA 0,1,0,0,0,0,0,1,0,0
290 DATA -1,-1,0,0,0,0,0,0,0,0
300 N=9
310 !
320 !   USER PROMPTED FOR VARIABLES
330 !
340 DISP "INPUT ANGLE OF ROTATION IN DEGREES";
350 INPUT Rotangle
360 Rotangle=Rotangle*PI/180
370 DISP "INPUT ACCELERATION ANGLE IN DEGREES";
380 INPUT Aangle
390 Aangle=Aangle*PI/180
400 !
410 !   DEFINE LINK LENGTHES OF 4-BAR
420 !
430 R=SQR(((9.1602+15.194)/12)^2+(20/12)^2)
440 R1=44.35
450 R2=R*12
460 R3=20
470 R4=20
480 Tlen=20/12
490 Flen=20/12
500 L=Flen
510 G=32.2
520 Mfemur=3/G
530 Mtibia=2/G
540 Wgt=100
550 Mbody=85/G
560 !
570 !   DEFINE INERTIA'S
580 !
590 CALL Invar(Mfemur,Ifemur,Flen,Mtibia,Itibia,Tlen,R,Mbody,Ibody)

```

```

500  !
610  !   CALCULATE NECESSERY ANGULAR VELOCITES AND ACCELERATION
620  !
630  Phi=ASN(20/(R*12))
640  H=R*SIN(Rotangle+Phi+Angle)-R*SIN(Phi+Angle)
650  Finalomegad=-1*SQR(2*Wgt*r/Ibody)
660  Alphaad=-1*Finalomegad^2/(2*Angle)
670  Totaltime=ABS(Finalomegad/Alphaad)
680  !
690  !   DEFINE INITIAL ANGLES
700  !
710  Theta4=PI-ASN(20/(R*12))
720  PRINT "INITIAL THETA4 =" ;Theta4*180/PI
730  CALL Findthetas(R1,R2,R3,R4,Theta2,Theta3,Theta4)
740  Inittheta4=Theta4
750  Inittheta3=Theta3
760  Inittheta2=Theta2
770  Finalangle=Theta4-Rotangle-Angle
780  !
790  !   BEGIN LOOPING OVER THETA4
800  !
810  Deltatheta=Angle/10
820  FOR Temp=(Inittheta4-Deltatheta) TO (Inittheta4-Angle) STEP -1*Deltatheta
830  Theta4=Temp
840  !
850  !   UPDATE ANGLES
860  !
870  CALL Findthetas(R1,R2,R3,R4,Theta2,Theta3,Theta4)
880  !
890  !   CALCULATE TIME
900  !
910  T=SQR(ABS((Temp-Inittheta4)*2/Alphaad))
920  !
930  !   CALCULATE ANGULAR VELOCITIES
940  !
950  Omegaad=Alphaad*T
960  Omegaab=R*Omegaad*SIN(Theta3-Theta4)/(L*SIN(Theta3-Theta2))
970  Omegabc=(Omegaad*R*SIN(Theta4)-L*Omegaab*SIN(Theta2))/(L*SIN(Theta3))
980  !
990  !   CALCULATE ANGULAR ACCELERATIONS
1000 !
1010 Num1=R*Alphaad*SIN(Theta3-Theta4)-R*(Omegaad^2)*COS(Theta4-Theta3)+L*(
Omegaab^2)*COS(Theta2-Theta3)+L*(Omegabc^2)
1020 Alphaab=Num1/(L*SIN(Theta3-Theta2))
1030 Num2=Alphaad*R*SIN(Theta4)+Omegaad^2*R*COS(Theta4)-Omegabc^2*L*COS(Theta3)-Alphaab*L*SIN(Theta2)-Omegaab^2*L*COS(Theta2)
1040 Alphabc=Num2/(L*SIN(Theta3))
1050 !
1060 !   CALCULATE CENTER OF MASSES ACCELERATIONS
1070 !
1080 Accelbx=-1*Alphabc*L*SIN(Theta3)-Omegabc^2*L*COS(Theta3)
1090 Accelby=Alphabc*L*COS(Theta3)-Omegabc^2*L*SIN(Theta3)
1100 Accelax=-1*Alphaad*R*SIN(Theta4)-Omegaad^2*R*COS(Theta4)
1110 Accelay=Alphaad*R*COS(Theta4)-Omegaad^2*R*SIN(Theta4)
1120 Acceltibx=Accelbx/2
1130 Acceltiby=Accelby/2
1140 Accelfemx=Accelbx-Alphaab*L*SIN(Theta2)/2-Omegaab^2*L*COS(Theta2)/2
1150 Accelfemy=Accelby+Alphaab*L*COS(Theta2)/2-Omegaab^2*L*SIN(Theta2)/2
1160 Accelbody=Accelax/2
1170 Accelbody=Accelay/2

```

```

1180 !
1190 NEXT Temp
1200 !
1210 READ A(*)
1220 A(3,3)=L*ABS(SIN(Theta3))
1230 A(3,4)=-1*L*ABS(COS(Theta3))
1240 A(6,1)=L*ABS(SIN(Theta2))
1250 A(6,2)=-1*L*AF3(COS(Theta2))
1260 A(9,1)=-1*ABS(R*SIN(Theta4))
1270 A(9,2)=-1*ABS(R*COS(Theta4))
1280 B(1)=Mtibia*Acceltibx
1290 B(2)=Mtibia*Acceltiby+Mtibia*G
1300 B(3)=-Itibia*Alphabc+Mtibia*G*Tlen*COS(Theta3)/2
1310 B(4)=Mfemur*Accelfemx
1320 B(5)=Mfemur*Accelfemy+Mfemur*G
1330 B(6)=-Ifemur*Alphaab+Mfemur*G*Flen*COS(Theta2)/2
1340 B(7)=Mbody*Accelbodyx
1350 B(8)=Mbody*Accelbody+Mbody*G
1360 B(9)=-Ibody*Alphaad-Mbody*R*COS(Theta4)/2
1370 !
1380 CALL Gauss(N,A(*),B(*),X(*))
1390 !
1400 Power=X(9)*Omegaab
1410 Hp=Power/550
1420 !
1430 !     STOP PROGRAM
1440 !
1450 END
1460 !
1470 !
1480 !     SUB Gauss(N,A(*),B(*),X(*))
1490 !
1500 !     THIS SUBROUTINE PERFORMS GAUSSIAN ELIMINATION
1510 !
1520 OPTION BASE 1
1530 !
1540 !     FIRST, REPLACE THE LAST COLUMN OF THE A MATRIX WITH THE
1550 !     B MATRIX
1560 !
1570 FOR I=1 TO N
1580     A(I,10)=B(I)
1590 NEXT I
1600 !
1610 FOR K=1 TO N-1
1620     Jj=K
1630     Big=ABS(A(K,K))
1640     Temp2=K+1
1650     FOR I=Temp2 TO N
1660         Ab=ABS(A(I,K))
1670         IF Big-Ab<0 THEN
1680             Big=Ab
1690             Jj=I
1700         END IF
1710     NEXT I
1720     IF Jj-K<>0 THEN
1730         FOR J=K TO N+1
1740             Temp=A(Jj,J)
1750             A(Jj,J)=A(K,J)
1760             A(K,J)=Temp
1770         NEXT J

```

```

1780     END IF
1790     FOR I=Temp2 TO N
1800         Quot=A(I,K)/A(K,K)
1810         FOR J=Temp2 TO N+1
1820             A(I,J)=A(I,J)-Quot*A(K,J)
1830         NEXT J
1840     NEXT I
1850     FOR I=Temp2 TO N
1860         A(I,K)=0.
1870     NEXT I
1880 NEXT K
1890 X(N)=A(N,N+1)/A(N,N)
1900 FOR Nn=1 TO N-1
1910     Sum1=0
1920     I=N-Nn
1930     Ip1=I+1
1940     FOR J=Ip1 TO N
1950         Sum1=Sum1+A(I,J)*X(J)
1960     NEXT J
1970     X(I)=(A(I,N+1)-Sum1)/A(I,I)
1980 NEXT Nn
1990 SUBEND
2000 !
2010 !
2020 !
2030     SUB Printmat(Array(*))
2040 !
2050     THIS SUBROUTINE PRINTS OUT THE INPUT MATRIX
2060 !
2070     OPTION BASE 1
2080     FOR Row=BASE(Array,1) TO SIZE(Array,1)+BASE(Array,1)-1
2090         FOR Column=BASE(Array,2) TO SIZE(Array,2)+BASE(Array,2)-2
2100             PRINT USING "DDD.DD,XX,#";Array(Row,Column)
2110         NEXT Column
2120     PRINT
2130     NEXT Row
2140     SUBEND
2150 !
2160 !
2170 SUB Invar(Mfemur,Ifemur,Flen,Mtibia,Itibia,Tlen,R,Mbody,Ibody)
2180 !
2190     THIS SUBROUTINE DETERMINES THE INERTIA FOR THE VARIOUS MEMBERS
2200     ABOUT THEIR AXES
2210 !
2220 Ifemur=.0252+Mfemur*(Flen/2)^2
2230 Itibia=.01495+Mtibia*(Tlen/2)^2
2240 !
2250     IBODY IS DETERMINED ASSUMING THE BODY OF SKITTER IS SPHERICAL
2260     AND THAT THE LEGS DO NOT CONTRIBUTE SIGNIFICANTLY TO THE INERTIA
2270 !
2280 Ibody=2*Mbody*(9.16/12)^2/5+Mbody*((15.194/12)^2+(20.9817/12)^2)
2290 !
2300     RETURN TO PROGRAM
2310 !
2320 SUBEND
2330 !
2340 !
2350     SUB Findthetas(R1,R2,R3,R4,Theta2,Theta3,Theta4)
2360 !
2370     THIS SUBROUTINE FINDS THE OTHER JOINT ANGLES GIVEN ONE ANGLE

```

```
2380 !           FOR A 4-BAR LINK
2390 !
2400 L=SQR(R1^2+R2^2-2*R1*R2*COS(PI-Theta4))
2410 Beta=ACS((R1^2+L^2-R2^2)/(2*R1*L))
2420 Psi=ACS((R3^2+L^2-R4^2)/(2*R3*L))
2430 Lamda=ACS((R4^2+L^2-R3^2)/(2*R4*L))
2440 IF (PI-Theta4)>=0 AND (PI-Theta4)<=PI THEN
2450     Theta2=-1*(Psi-Beta)
2460     Theta3=PI-(PI-Lamda-Beta)
2470 ELSE
2480     Theta2=-1*(Psi+Beta)
2490     Theta3=PI-(PI-Lamda+Beta)
2500 END IF
2510 !
2520 !           RETURN TO PROGRAM
2530 !
2540 SUBEND
```



