Chapter 2

McGeer's Recipe For Walking Analysis

Besides pioneering the passive-dynamic approach to gait study, Tad McGeer also used a *Poincaré map* to analyze gait simulations. This robotics approach has also been used by Rubanovich and Formalskii (1981), Hurmuzlu (1987a), Hurmuzlu (1987b), and others; this chapter is an outline of that analysis procedure. Although McGeer did not invent any of the individual elements of this analysis, he adapted it effectively to the study of passive models.

The technique presented is quite useful and is completely independent of the model's details (i.e., it is independent of the "passive-ness" of the models). Poincaré maps and other nonlinear dynamics tools have been used by Hurmuzlu et al. (1993) and Hurmuzlu et al. (1996) in the study of post-polio gait, and also by Holt et al. (1991) in the assessment of human walking stability. The same techniques could be used, for example, to find gaits in a multi-link android with a complicated control scheme.

The reader is assumed to have some knowledge of nonlinear dynamics, in particular, Poincaré maps and the notion of phase space; the basic ideas behind can be found in texts by Strogatz (1994), Devaney (1989), and Guckenheimer and Holmes (1983), and Moon (1992) for instance. Similar procedures are used by Andronov et al. (1966) in their analysis of clocks and escapement mechanisms with impulses.

2.1 Summary Of Procedure

The procedure can be summarized as follows.

1. Create a mechanically complete model (see section 2.2 and section 4.2).
2. Find ODEs governing the smooth portion(s) of the model's motion (see section 2.3).
3. Imbed the ODEs in a simulation environment (see section 2.4).
4. Determine collision conditions and create one or more maps which map the state of the walker just before a certain collision to the state of the walker just after that collision. Imbed these in the code, along with a procedure to settle down onto the surface(s) of section (see section 2.5).
5. Define a function which maps the state of the walker from just after a heelstrike to just after the next heelstrike (see section 2.6).
6. Using Newton's method, find a root of this function (see section 2.8).
7. At the root, numerically approximate the Jacobian of the function, or analyze other characteristics of the gait (see section 2.9).

8. Adjust parameters as needed and repeat repeat steps 6-8 until desirable performance is obtained. If the goal is a physical model, it can then be built (see section 2.12).

These items are explained below.

2.2 Preliminaries: Model Assumptions

Before beginning an analysis of a walking machine, we need to design a model! Although this step may seem trivial, there are numerous subtle judgements to be made here. For example, at present Coleman and Ruina (1998) has a physical walker that balances in 3-D but does not yet know exactly which aspects of its physical description are needed to theoretically predict its stability with computer simulation.

So, to paraphrase Einstein, the model should be as simple as possible, but not too simple; design choices involve the nature of connections, and contact. The theoretical models here are simultaneously based on previous results, imitation of human design, and imitation of other models. Assuming that a complete mechanical model has been formulated for the walker of interest (see section 4.2), one can follow McGeer's recipe for analyzing and/or optimizing the model, based on parameter variations.

2.3 Derivation of Equations of Motion for Linked Rigid Bodies

Consider a system of 2D or 3D rigid bodies joined by rotational joints as shown in Figure 2.1. The goal is to derive the ordinary differential equations which govern the motion of this linkage. Note that these equations only govern the smooth parts of the motions of the walking mechanisms – accounting for kneestrike and heelstrike in the modelling is described in Section 2.5.

In Figure 2.1, the foot radius is zero, the stance leg is connected to the ground by a 2D rotational joint with one degree of freedom (DOF), and the joints are frictionless. The walkers here can also have rolling contact in 2-D or 3-D and torsional springs and dampers at the joints. The procedure described here is essentially the one described in Craig (1989), which is based on the method of Luh et al. (1990), with some modifications. Some advantages of this approach are as follows:

- The algorithm is well-known and is reasonably robust and efficient.
- The algorithm is generalizable to contact conditions such as disk feet, other foot shapes, and to systems with tree-like branches.
- Applied torques can be easily added to the equations.
- The inter-link forces are part of the equations and so can be analyzed, if desired, with minimal additional effort.
- The algorithm lends itself to being solved "on-line" instead of analytically, and so saves time in writing the equations of motion, although at the slight expense of computation time during simulation.

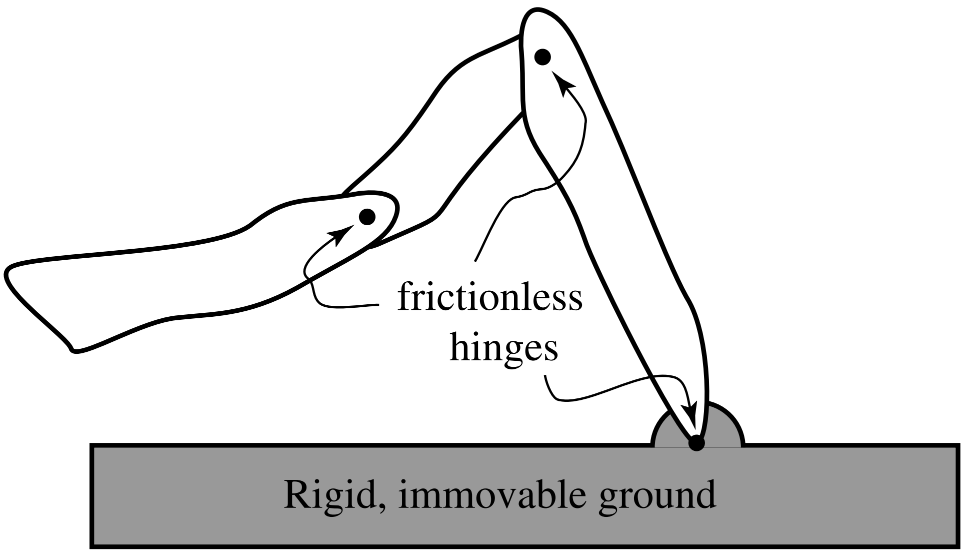

Figure 2.1: A simple linkage.

2.3.1 Configuration Angles And Reference Frames

To write the equations of motion for 2-D and 3-D walkers, first define reference frames using a convention like the ones shown in Figures 2.2 or 2.3. Each degree of freedom (DOF) has a reference frame and angle θ associated with it. For 2-D models, there are the same number of links n as reference frames k, so $n = k = 2$ for straight-legged models and $n = k = 3$ for kneed models. For a 3-D straight-legged model, there are only two links (stance and swing leg) but the stance leg has 3 DOF and the swing leg has 1 DOF, so $n = 2$ and $k = 4$ in 3-D. Another approach is to create "virtual links" with zero mass and inertia so that each link has a reference frame and DOF associated with it and $n = k$ always.

Cases where there are more links than degrees of freedom are not considered; this would occur, for example, during a double support phase in a 2-D model with flexible ankles, where both feet were constrained to touch the ground over a finite period of time.

Reference frame i is generally oriented as follows: $\hat{\mathbf{z}}_i$ points along the direction of the axis of rotation of the joint, $\hat{\mathbf{x}}_i$ points from the origin of frame i to the origin of frame $i+1$, and $\hat{\mathbf{y}}_i$ points in such a way as to complete a right-handed cartesian frame. Frame i is fixed to link i at the rotational joint where link i touches link $i-1$. If a joint has more than one rotational degree of freedom (a ball and socket joint, say) then several frames are used in an Euler-angle scheme, all with the same point as their origin. If several frames share the same origin, then the choice of some $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ directions may be arbitrary – it is generally most convenient to make parallel as many $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ directions as possible. In any case, because of the generality of the rotation matrices, the only necessary requirement for using the algorithm below is that frame i rotate about $\hat{\mathbf{z}}_i$, and even that requirement can be waived with some slight modification of the algorithm details.

Figure 2.2 shows the frame convention and notation for using circular feet in 2D. Frame 0 translates with the center of the foot but does not rotate with it.

Figure 2.3 shows the notation for disk feet in 3D. Reference frame 0 will be attatched to the ground. Reference frames 1, 2, and 3 are at the foot center and form an Euler angle system for the stance foot. The stance leg is now link 3, and the swing leg is now link 4. Frames 0, 1, and 2 are associated with massless virtual links. θ_1 is the "heading" or "steering" angle about the $\hat{\mathbf{z}}_1$ axis, which is perpendicular to the slope. $\theta_1 = 0$ corresponds to straight downhill. θ_2 is the "bank" or "lean" angle, measured about the $\hat{\mathbf{z}}_2$ axis. $\theta_2 = \pi/2$ means that the walker is not leaning to the left or the right and that the hip axis is parallel to its projection on the slope surface. θ_3 is the "pitch" or "stance" angle corresponding to the stance angle in 2D. It is measured about the $\hat{\mathbf{z}}_3$ axis, which is collinear with the hip axis. $\theta_3 = 0$ corresponds to the stance leg not leaning forward or backward and means that the plane formed by the hip axis and the stance leg is perpendicular to the slope surface.

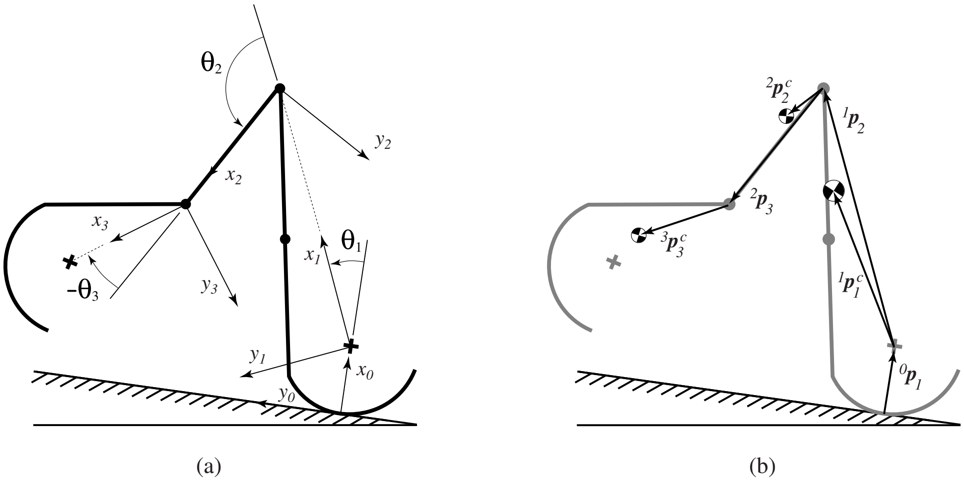

Figure 2.2: 2-D kneed walker with (a) directions and angles, and (b) vectors defined. In the straight-legged version, $\theta_3 \equiv 0$. To convert to McGeer's usual 2-D angle convention in his figures, $\theta_{stance} = -\theta_1$, $\theta_{thigh} = \pi - \theta_1 - \theta_2 - \varepsilon_T$, and $\theta_{shank} = \theta_{thigh} - \theta_3 + \varepsilon_K$

θ_i is the angle of rotation of link i with respect to link $i-1$ about the $\hat{\mathbf{z}}_i$ axis, with link 0 either fixed to the ground, or at least not rotating with respect to the ground. Together, $\boldsymbol{\theta} \equiv \theta_1 \ldots \theta_k$ make up the configuration space, or generalized coordinates, for the system.

Rotation matrices, denoted by $\mathbf{R}$, are then defined so that a vector written in frame i (denoted by a pre-superscript) ${}^i\mathbf{p}$ could be written in frame $i+1$ as

$$ {}^{i+1}\mathbf{p} = {}^{i+1}_{\;\;i}\mathbf{R}\,{}^i\mathbf{p} \tag{2.1} $$

Note that ${}^{i+1}\mathbf{p}$ and ${}^i\mathbf{p}$ are really the same vector, just written in different coordinate systems.

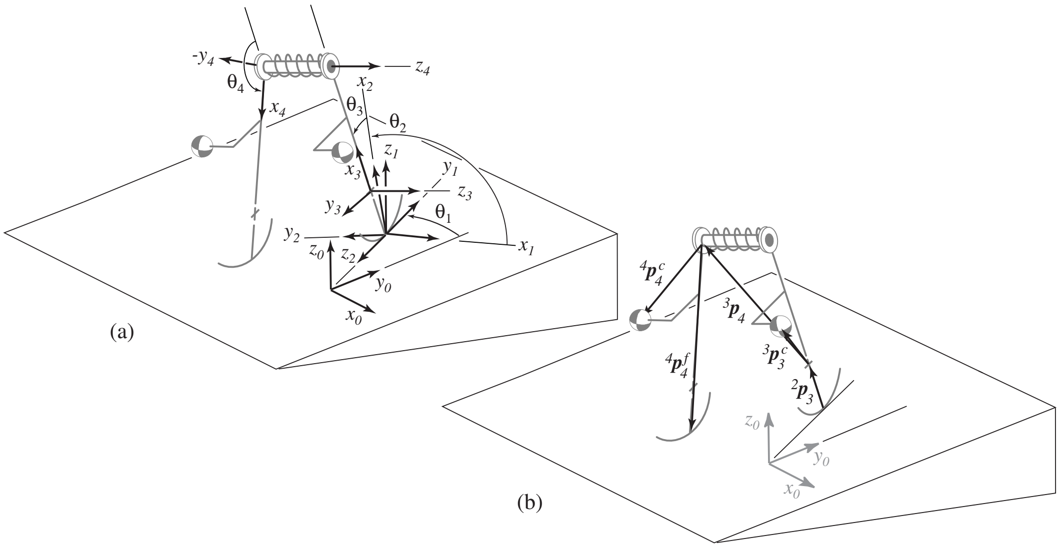

Figure 2.3: 3-D walker with (a) directions and angles, and (b) vectors defined. To convert to McGeer's usual 3-D angle convention in his figures, $\psi = -\theta_1$, $\phi = \pi/2 - \theta_2$, $\theta_{stance} = \theta_3$, and $\theta_{swing} = \theta_3 + \theta_4 - \pi$

.

2.3.2 Dynamics Algorithm

With reference frames and rotation matrices defined, the next step is to define the angular velocity $\boldsymbol{\omega}_i$ and acceleration $\dot{\boldsymbol{\omega}}_i$ of frame i (same as link i using virtual links and $n = k$). In the notation here, the subscript identifies the link of interest, while the superscript denotes the frame in which the components are written. Beginning with the base frame and proceed outwards, $i = 0 \to n - 1$.

$$^{i+1}\boldsymbol{\omega}_{i+1} = {}^{i+1}_{i}\mathbf{R}\,{}^{i}\boldsymbol{\omega}_i + \dot{\theta}_{i+1}\,\hat{\mathbf{z}}_{i+1} \tag{2.2}$$

$$^{i+1}\dot{\boldsymbol{\omega}}_{i+1} = {}^{i+1}_{i}\mathbf{R}\,{}^{i}\dot{\boldsymbol{\omega}}_i + {}^{i+1}_{i}\mathbf{R}\,{}^{i}\boldsymbol{\omega}_i \times \dot{\theta}_{i+1}\,\hat{\mathbf{z}}_{i+1} + \ddot{\theta}_{i+1}\,\hat{\mathbf{z}}_{i+1} \tag{2.3}$$

Next, define the velocity and acceleration of each frame ($\mathbf{v}$, $\dot{\mathbf{v}}$) and center of mass ($\mathbf{v}^c$, $\dot{\mathbf{v}}^c$), in terms of the θ_i, $\dot{\theta}_i$, and $\ddot{\theta}_i$, assuming all links are joined by pin joints and $R = 0$. If the walker has round or disk feet, see Section 2.3.3 for the correct stance leg equations.

$$ {}^{i+1}\mathbf{v}_{i+1} = {}^{i+1}_{i}\mathbf{R}\ ({}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{p}_{i+1} + {}^{i}\mathbf{v}_i) \tag{2.4} $$

$$ {}^{i+1}\mathbf{v}^{c}{}_{i+1} = {}^{i+1}\boldsymbol{\omega}_i \times {}^{i+1}\mathbf{p}^{c}_{i+1} + {}^{i+1}\mathbf{v}_{i+1}) \tag{2.5} $$

$$ {}^{i+1}\dot{\mathbf{v}}_{i+1} = {}^{i+1}_{i}\mathbf{R}\ ({}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\mathbf{p}_{i+1} + {}^{i}\boldsymbol{\omega}_i \times {}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{p}_{i+1} + {}^{i}\dot{\mathbf{v}}_i) \tag{2.6} $$

$$ {}^{i+1}\dot{\mathbf{v}}^{c}_{i+1} = {}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} \times {}^{i+1}\mathbf{p}^{c}{}_{i+1} + {}^{i+1}\boldsymbol{\omega}_{i+1} \times {}^{i+1}\boldsymbol{\omega}_{i+1} \times {}^{i+1}\mathbf{p}^{c}_{i+1} + {}^{i+1}\dot{\mathbf{v}}_{i+1} \tag{2.7} $$

$$ \tag{2.8} $$

where ${}^{i}\mathbf{p}_{i+1}$ is a vector from the origin of frame i to the origin of frame $i+1$, written in terms of frame i, and ${}^{i+1}\mathbf{p}^{c}_{i+1}$ is a vector from the origin of frame i to the center of mass of link i, written in frame $i+1$. The velocities (Equations 2.4 and 2.5) do not necessarily enter into the momentum conservation equations shown below (Equations 2.11 and 2.12), but they are included here for completeness and also because in some cases it is convenient to use or track them.

So-called “inertial forces” **F** and “inertial torques” **N** are then defined for each link, also in terms of the θ_i, $\dot{\theta}_i$, and $\ddot{\theta}_i$. Although we do not usually advertise this terminology, we use it here in order to follow the derivations in Craig (1989) and Luh et al. (1990).

$$ {}^{i+1}\mathbf{F}_{i+1} = m_{i+1}{}^{i+1}\dot{\mathbf{v}}^{c}_{i+1} \tag{2.9} $$

$$ {}^{i+1}\mathbf{N}_{i+1} = {}^{i+1}\mathbf{I}^{c}{}_{i+1}{}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} + {}^{i+1}\boldsymbol{\omega}_{i+1} \times {}^{i+1}\mathbf{I}^{c}{}_{i+1}{}^{i+1}\boldsymbol{\omega}_{i+1} \tag{2.10} $$

Here, m_{i+1} is the mass of link $i+1$ and ${}^{i+1}\mathbf{I}^{c}{}_{i+1}$ is the moment of inertia matrix about the center of mass of link $i+1$, written in the $i+1$ frame. Obviously, **F** and **N** will be zero for virtual links.

At this point, the so-called “outward iterations” are complete. This is more or less a copy of the notation in Craig (1989), where ${}^{i+1}\mathbf{F}_{i+1}$ and ${}^{i+1}\mathbf{N}_{i+1}$ are vectors.

Generally speaking, uppercase boldface is reserved for matrices and/or tensors like ${}^{i+1}\mathbf{I}^{c}{}_{i+1}$, but this rule isn't followed strictly .

Note that the kinematic constraints are embedded in equations 2.9 and 2.10. For more complicated contact conditions, the equations must be modified to reflect the appropriate constraints. Section 2.3.3 demonstrates code modifications to include disk feet.

The so-called "inward iterations" consist of writing Newton-Euler equations for each link. Assuming pin joints as before, a free-body diagram (FBD) of each link (the two outermost links are shown in Figure 2.4), starting with the outermost link and proceeding inwards, $i = n \rightarrow 1$, will yield the following equations, after some rearrangement.

$$ {}^{i}\mathbf{f}_i = {}^{i}_{i+1}\mathbf{R}\,{}^{i+1}\mathbf{f}_{i+1} + {}^{i}\mathbf{F}_i \tag{2.11} $$

$$ {}^{i}\mathbf{n}_i = {}^{i}\mathbf{N}_i + {}^{i}_{i+1}\mathbf{R}\,{}^{i+1}\mathbf{n}_{i+1} + {}^{i}\mathbf{p}^{c}{}_{i} \times {}^{i}\mathbf{F}_i + {}^{i}\mathbf{p}_{i+1} \times {}^{i}_{i+1}\mathbf{R}\,{}^{i+1}\mathbf{f}_{i+1} \tag{2.12} $$

$$ {}^{i}\tau_i = {}^{i}\mathbf{n}_i \cdot \hat{\mathbf{z}}_i \tag{2.13} $$

Above, ${}^{i}\tau_i$ is the applied torque about axis $\hat{\mathbf{z}}_i$, which is zero for passive models. ${}^{i}\mathbf{f}_i$ is the force exerted on link i by link $i-1$, and ${}^{i}\mathbf{n}_i$ is the torque exerted on link i by link $i-1$, each written in frame i. If the stance leg has round or disk feet, then one needs to modify the stance leg angular momentum balance equation(s) as shown in Section 2.3.3.

The effect of gravity is included most simply by accelerating the base of the mechanism ${}^{0}\dot{\mathbf{v}}_0$ upward by g, as described in Craig (1989), and so one does not need to compute the forces and torques due to gravity in the equations above (see also Section 2.3.3). Also note that ${}^{n+1}\mathbf{f}_{n+1} = {}^{n+1}\mathbf{n}_{n+1} = \mathbf{0}$ if the outermost link is swinging freely, which is almost always the assumption. One exception to this would be if there was some torque on the leg about the hip axis due to a torsional spring or damper.

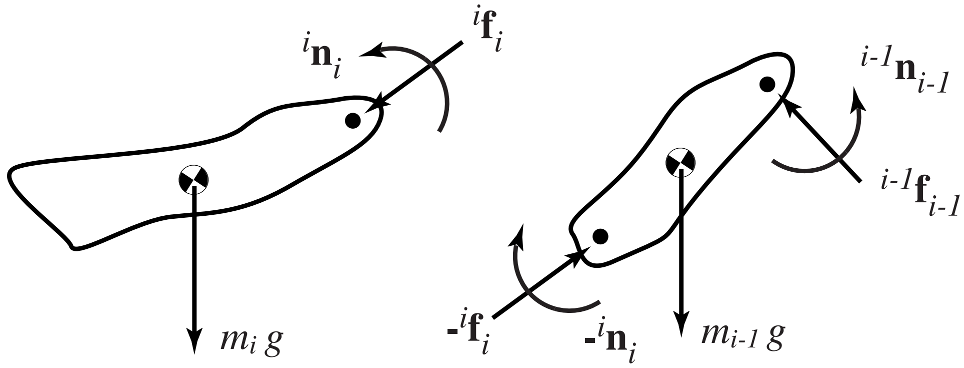

Figure 2.4: Free body diagrams of the two outermost links. Gravity is shown on the FBDs but can be accounted for in the algorithm by simply accelerating the base link upwards by g, so it does not enter explicitly into each force and torque balance.

2.3.3 Adapting the Algorithm to Other Models

Some models are not properly described by Figure 2.1. Most commonly, the models will have round or disk feet or other kinematic contact conditions at the foot. They can also have torsional springs, dampers, or rotary actuators at the joints, branching links, or separate contact conditions at the outer links. In this section, some of these variations are considered.

Torsional Springs, Dampers and Actuators

$\boldsymbol{\tau}$ is a vector of applied joint torques, which can include torsional springs, dampers, and actuators. A linear torsional spring at joint i will provide a restoring torque proportional to θ_i. A linear torsional damper at joint i will provide a negative torque proportional to $\dot{\theta}_i$. Most generally, τ_i can be some function of time or state or both, which is solved for independently at each time step.

Consider the following example for a planar double pendulum. The middle joint has an actuator which provides a constant torque of 1 about its $\hat{\mathbf{z}}$ axis (all $\hat{\mathbf{z}}$ axes are parallel and point out of the manipulator's plane in this case). The base joint has a linear torsional spring of stiffness k which exerts no torque on the pendulum when $\theta_1 = 0$. Both joints have some friction which is modeled by a torsional damper with coefficient c. The vector of joint torques at each time step in this case would be

$$\boldsymbol{\tau} = \begin{bmatrix} -c\dot{\theta}_1 - k\theta_1 \\ 1 - c\dot{\theta}_2 \end{bmatrix} \tag{2.14}$$

which would then be inserted into Equation 2.19 to solve for $\ddot{\boldsymbol{\theta}}$.

Circular Or Disk Feet

The equations presented previously need to be slightly modified if round or disk feet are used in the model. These modifications amount to (1) writing the correct accelerations for the center of mass of the stance leg and associated frames, and (2) correcting the angular momentum balance equation(s) for the stance leg. Note that the modified equations will reduce to the original pointfoot equations when R is set to 0, so only one derivative file is needed for point or round feet.

Circular Feet in 2D Figure 2.2 shows the frame convention and notation for using circular feet in 2D. The acceleration of frame 0 is non-zero in general (even for point-feet because it is used to account for gravity) and is denoted by ${}^{0}\dot{\mathbf{v}}_0$ Equation

2.6 for ${}^1\dot{\mathbf{v}}_1$ is modified in the above algorithm as follows.

$$ {}^0\dot{\mathbf{v}}_0 = [g\cos\gamma,\ R\ddot{\theta}_1 - g\sin\gamma,\ 0] \tag{2.15}$$

$$ {}^1\dot{\mathbf{v}}_1 = {}^1_0\mathbf{R}\,{}^0\dot{\mathbf{v}}_0 \tag{2.16}$$

The $g\cos\gamma$ and $-g\sin\gamma$ terms result from accelerating the base upwards to account for the effect of gravity, as described above.

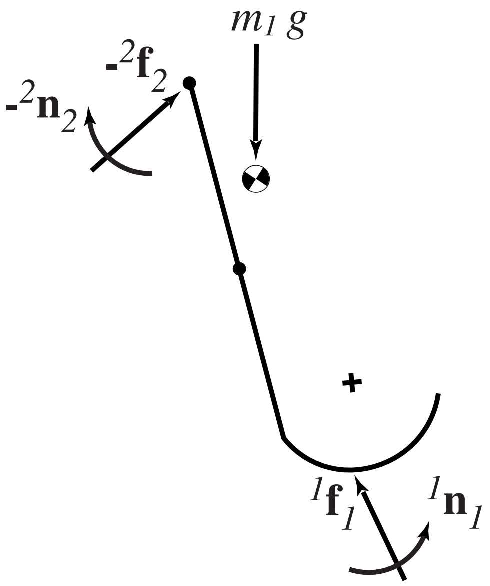

Figure 2.5: Free-body diagram of the stance leg for a walker with circular feet.

Figure 2.5 shows a free-body diagram of the stance leg with a circular foot. Because of the circular foot, one needs to account for the torque of the contact force ${}^1\mathbf{f}_1$ from the ground about the center of the stance foot. Lastly, Equation 2.12 for the stance leg is modified as follows.

$$ {}^1\mathbf{n}_1 = {}^1\mathbf{N}_1 + {}^1_2\mathbf{R}\,{}^2\mathbf{n}_2 + {}^1\mathbf{p}^c{}_1 \times {}^1\mathbf{F}_1 + {}^1\mathbf{p}_2 \times {}^1_2\mathbf{R}\,{}^2\mathbf{f}_2 - {}^1_0\mathbf{R}\,{}^0\mathbf{p}_1 \times {}^1\mathbf{f}_1 \tag{2.17}$$

where ${}^0\mathbf{p}_1 = [R\ 0\ 0]^T$ is the vector from the contact point to the center of the foot. Gravity is included in the FBD but it is not explicitly included in Equation 2.17 for reasons stated previously.

Disk Feet in 3D Another possibility is to have disk feet, as McGeer (1991) and Fowble and Kuo (1996) did in their 3D models. Modifying the code to include the case of disk feet involves following procedure as done previously for circular feet, except that the calculation of the foot center acceleration ${}^3\dot{\mathbf{v}}_3$ is slightly more involved. The formulas which govern a rolling disk are well known, however (see Rand (1994) or Greenwood (1988) for example), so there is no fundamental obstacle, especially if the equations are first derived symbolically.

Because not all of the equations are derived symbolically, including the acceleration of the stance leg center of mass and associated frames for disk feet is a slightly convoluted procedure and is best postponed until the discussion of the *on-line scheme* in Section 2.4. Briefly, Maple ® symbolic software is used to derive and substitute some extra terms for ${}^3\dot{\mathbf{v}}_3$ in the algorithm (Equation 2.6). When they are added to the existing algorithm, the resulting formula for ${}^3\dot{\mathbf{v}}_3$ is that for the acceleration of the center of a rolling disk. If the foot radius R is set to zero, the formula reduces to the previous one for a 3 DOF rotational joint. For more details on the modification to Equation 2.6 in 3D, see Section 2.4.1.

The second modification for disk feet in 3-D, as in the 2-D case, is to account for the torque of the contact force (in this case, ${}^3\mathbf{f}_3$) on the stance leg. Paralleling Equation 2.17, and using Figure 2.6,

$$ {}^3\mathbf{n}_3 = {}^3\mathbf{N}_3 + {}^3_4\mathbf{R}\,{}^4\mathbf{n}_4 + {}^3\mathbf{p}^c{}_3 \times {}^3\mathbf{F}_3 + {}^3\mathbf{p}_4 \times {}^3_4\mathbf{R}\,{}^4\mathbf{f}_4 - {}^3_2\mathbf{R}\,{}^2\mathbf{p}_3 \times {}^3\mathbf{f}_3 \tag{2.18} $$

where ${}^2\mathbf{p}_3 = [R\ 0\ 0]^T$ is the vector from the contact point (origin of frame 2) to the center of the foot (origin of frame 3).

Note that in the usual case of rotational joints, one writes the angular momentum balance for link i about the $\hat{\mathbf{z}}_i$ axis, which passes through the frame origin. But the angular momentum balance equations for frames 1 and 2 are written about axes

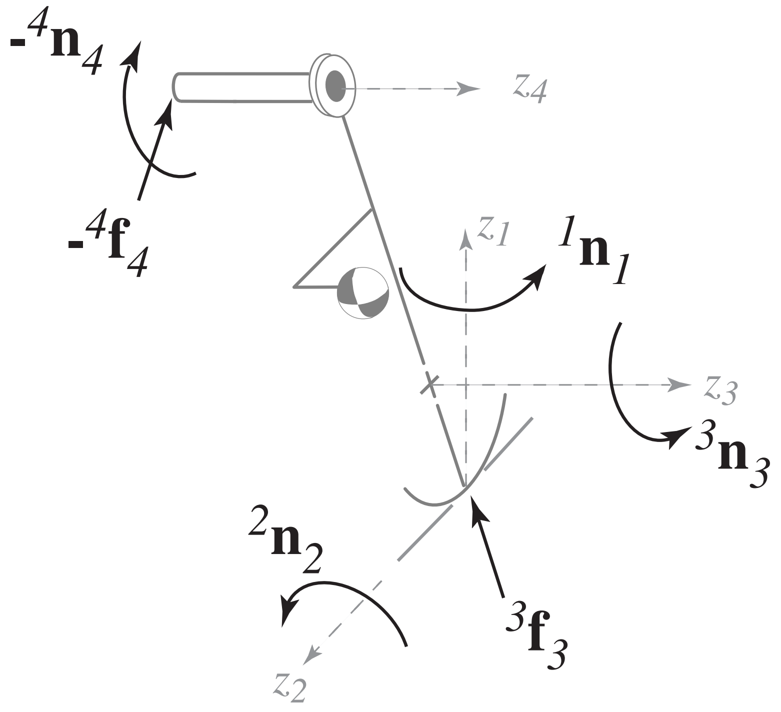

Figure 2.6: Free-body diagram of the stance leg for a 3-D walker with disk feet.

through the foot center which are parallel to $\hat{\mathbf{z}}_1$ and $\hat{\mathbf{z}}_2$ and do not pass through the origins of frames 1 or 2. The equations would yield the same result regardless of the choice of axis.

2.3.4 Form Of Equations Of Motion

In the way shown above, one can generate equations which govern the motion of the linkage. With some separation and factorization, they can be re-written in the form

$$\boldsymbol{\tau} = \mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) \tag{2.19}$$

where $\boldsymbol{\tau}$ is an $n \times 1$ vector of joint torques, $\mathbf{M}(\boldsymbol{\theta})$ is the symmetric $n \times n$ *mass* *matrix* of the linkage, and $\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$, and $\ddot{\boldsymbol{\theta}}$ are the $n \times 1$ vectors of configuration angles, angular rates, and angular accelerations, respectively. Furthermore, $\mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is an $n \times 1$ vector of centrifugal and coriolis terms, and $\mathbf{G}(\boldsymbol{\theta})$ is an $n \times 1$ vector with gravity terms only Craig (1989). Note that there are no additional constraint equations; the constraints are embedded in equation 2.19 when the accelerations of the link centers of mass are written as functions of the θ, $\dot{\theta}$, and $\ddot{\theta}$ in Equations 2.6 and 2.7. It is important to understand the format above in order to use the scheme which is presented in Section 2.4.

Typically, in dynamic simulation, the goal is to solve for the angular accelerations at each time step as functions of the angles and angular velocities. To do so, one simply inverts $\mathbf{M}(\boldsymbol{\theta})$ in Equation 2.19 above.

$$\ddot{\boldsymbol{\theta}} = \mathbf{M}^{-1}(\boldsymbol{\theta})[\boldsymbol{\tau} - \mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) - \mathbf{G}(\boldsymbol{\theta})] \tag{2.20}$$

Note that the inversion is never done symbolically; most often, $\mathbf{M}(\boldsymbol{\theta})$ is calculated and inverted numerically.

Neglecting $\mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$

If the angular velocities are small enough during the motions of interest, and if some loss of accuracy is acceptable in the interests of computation time, one can neglect the coriolis and centrifugal terms in $\mathbf{V}$. In this analysis, no terms are neglected. However, McGeer (1990a) used linearized equations for a straight-legged walker. In kneed walkers, the the swing shank reaches angluar velocities that are high enough to render $\mathbf{V}$ non-negligible in general.

Two Advantages To The Above Scheme

Two fortuitous side-effects occur with the abovementioned notation scheme(s). One is that the kinetic energy of the walker, a useful scalar quantity, can be written compactly as

$$KE = \frac{1}{2}\dot{\boldsymbol{\theta}} \cdot \mathbf{M}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{2.21}$$

as described by Craig (1989). Here, $\mathbf{M}(\boldsymbol{\theta})$ is, of course, the mass matrix described previously and $\dot{\boldsymbol{\theta}}$ is a vector of angular rates $[\dot{\theta}_1 ... \dot{\theta}_i]^T$.

The second, and possibly more useful side-effect is that the expression

$$\mathbf{M}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{2.22}$$

is a vector whose entries are the angular momenta of certain subsystems of the walker. The last entry is the angular momentum of the outermost link about its hinge axis. The next-to-last entry is the angular momentum of the *two* outermost links about the *next-to-last* hinge axis, and so on. The first entry in Equation 2.22 (or the first three entries, in the case of a 3D walker) is the angular momentum of the whole walker about its contact point with the ground. This will prove useful in the heelstrike and kneestrike calculations, as is described in Section 2.5.

2.4 Numerical Integration of Equations of Linked Rigid Bodies

Using the equation-generating algorithm described previously, there is a choice of methods for simulating a given mechanism. "Simulation" means numerical solution of the ordinary differential equations which govern the motion of the mechanism, subject to specific initial conditions (the swing phase), along with the calculation of pre- and post-collision velocities at kneestrike and heelstrike.

2.4.1 On-Line Scheme for the Swing Phase

The equations for the swing phase of motion are presented in the Section 2.3. The straightforward method is to use the algorithm to derive symbolic code for the elements of $\mathbf{M}(\boldsymbol{\theta})$, $\mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, $\mathbf{G}(\boldsymbol{\theta})$, and possibly $\boldsymbol{\tau}$, calculate each element at each time step, and then solve Equation 2.20 numerically at each time step or partial time step.

Another simulation technique is known as *on-line* simulation Luh et al. (1990). In this method, symbolic code is not written in the usual sense, but Equations 2.2-2.13 are solved directly at each time step. If you like, Equations 2.2-2.13 are themselves the symbolic code, and the parts of Equation 2.19 are solved for separately, and then put together.

Solving for V and G

To solve Equations 2.2-2.13 for $\ddot{\boldsymbol{\theta}}$, one first sets all the $\ddot{\theta}_i$ to 0 and sweep through the inward and outward Newton-Euler iterations (Equations 2.2-2.13). The only non-zero terms which remain are those due to gravity and velocity products, since $\ddot{\theta}_i = 0$. The result will be of the form

$$\boldsymbol{\tau} = \mathbf{0} = \mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) \tag{2.23}$$

where $\mathbf{V}$ and $\mathbf{G}$ are terms sought as part of Equation 2.19. If there are applied torques and $\boldsymbol{\tau} \neq \mathbf{0}$, the above structure is not changed and $\boldsymbol{\tau}$ can be specified at a later step.

Solving for M and then $\ddot{\boldsymbol{\theta}}$

Now the task remains to find $\mathbf{M}(\boldsymbol{\theta})$ (or more simply, just $\mathbf{M}$) in order to solve for the components of Equation 2.19. It is helpful to write out $\mathbf{M}\ddot{\boldsymbol{\theta}}$ as follows, with $\mathbf{M}$ of size $n \times n$.

$$\begin{bmatrix} M_{11} & \dots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \dots & M_{nn} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \vdots \\ \ddot{\theta}_n \end{bmatrix} \tag{2.24}$$

It is clear that the first column of $\mathbf{M}$ ican be generated by setting $\ddot{\theta}_1 = 1$, $\ddot{\theta}_2 \dots \ddot{\theta}_n = 0$, and sweeping through the inward and outward iterations (Equations 2.2-2.13) as done above for $\mathbf{V}$ and $\mathbf{G}$, but now with $g = 0$ and all the $\dot{\theta}_i = 0$. Similarly, one can solve for column i of $\mathbf{M}$ by setting $\ddot{\theta}_i = 1$ and all other $\ddot{\theta} = 0$ for $i = 2 \dots n$. In this manner, one constructs the matrix $\mathbf{M}$ of the coefficients of the $\ddot{\theta}_i$ in the terms of the form $\dot{\boldsymbol{\omega}} \times \mathbf{p}$ and, together with $\mathbf{V}$ and $\mathbf{G}$ from above, solves Equation 2.19 for the new $\ddot{\boldsymbol{\theta}}$.

On-Line Scheme for Disk Feet in 3D

In the previous section, the Newton-Euler iterative scheme was modified to compute the acceleration of the stance leg and related frames for the case of disk feet. This can also be done with the on-line approach. First, note that ${}^{3}\dot{\mathbf{v}}_3$ is written as follows.

$${}^{3}\dot{\mathbf{v}}_3 = {}^{3}\dot{\boldsymbol{\omega}}_3 \times {}^{2}\mathbf{p}_3 + {}^{3}\boldsymbol{\omega}_3 \times {}^{3}\boldsymbol{\omega}_3 \times {}^{2}\mathbf{p}_3 + {}^{3}_{0}\mathbf{R}\,{}^{0}\dot{\mathbf{v}}_0 \tag{2.25}$$

where ${}^0\dot{\mathbf{v}}_0$ is the upwards acceleration to account for gravity.

The gravity terms and the $\boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{p}$ terms are calculated during the first Newton-Euler iteration, when all the $\ddot{\theta}_i$ are zero. The expressions that remain are in the $\dot{\boldsymbol{\omega}} \times \mathbf{p}$ term, and consist of "second-derivative" expressions with coefficients multiplying $\ddot{\theta}_i$ and "product" expressions containing $\dot{\theta}_i\dot{\theta}_j$. The second-derivative coefficients are calculated during the mass-matrix computations, by setting

$$ {}^3\dot{\mathbf{v}}_3 = {}^3\dot{\boldsymbol{\omega}}_3 \times {}^2\mathbf{p}_3 \tag{2.26} $$

with all the $\dot{\theta}_i = 0$ and one $\ddot{\theta} = 1$ at a time while all the others are zero, as described above in Section 2.4.1.

So, it remains to calculate the product expressions in the $\dot{\boldsymbol{\omega}} \times \mathbf{p}$ term. MAPLE ® symbolic algebra software was used to differentiate the elements of the velocity vector of the foot center (providing the $\dot{\boldsymbol{\omega}} \times \mathbf{p}$ term) and then set all the $\ddot{\theta}_i = 0$. The terms that remain are added to ${}^3\dot{\mathbf{v}}_3$ during the calculation of the $\boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{p}$ terms.

So the lines in the code to calculate the acceleration of the foot center *without* the $\ddot{\theta}_i$ (which are calculated during the iterations for the mass matrix) are as follows.

$$ {}^3\mathbf{v}_3 = {}^3\boldsymbol{\omega}_3 \times {}^2\mathbf{p}_3 \tag{2.27} $$

$$ {}^3\dot{\mathbf{v}}_3 = {}^3\boldsymbol{\omega}_3 \times {}^3\mathbf{v}_3 + {}^3_0\mathbf{R}\,{}^0\dot{\mathbf{v}}_0 + \begin{bmatrix} R\sin\theta_3\sin\theta_2\dot{\theta}_2\dot{\theta}_1 + (\dot{\theta}_3 - \dot{\theta}_1\cos\theta_2)R\cos\theta_3\dot{\theta}_3 \\ R\cos\theta_3\sin\theta_2\dot{\theta}_2\dot{\theta}_1 - (\dot{\theta}_3 - \dot{\theta}_1\cos\theta_2)R\sin\theta_3\dot{\theta}_3 \\ 0 \end{bmatrix} \tag{2.28} $$

where, as before, ${}^0\dot{\mathbf{v}}_0$ is the upwards acceleration of the base to mimic gravity.

Comments On Efficiency

The efficiency of algorithms for generating and/or solving equations of motion is typically measured by the number of floating-point operations, or "flops", that they require in order to generate coefficients in the ODEs, as a function of the number of links or degrees of freedom (DOF) of the mechanism in question. As presented, the Equations 2.2-2.13 constitute a so-called "order-n algorithm", meaning that the number of operations needed to solve for the elements of Equation 2.19 (but not solve for $\ddot{\boldsymbol{\theta}}$) grows linearly with the number of links or DOF.

By making an on-line scheme, an order-n algorithm is performed a total of $n+1$ times (once for $\mathbf{V}$ and $\mathbf{G}$ and n times for $\mathbf{M}$), which creates an algorithm of order n^2. Although it might seem that using a slower algorithm is pointless, the following things should be kept in mind.

- These models have relatively few links ($n < 3$) and degrees of freedom ($k < 4$), and so efficiency considerations are less important here, although certainly not trivial.
- There is no need to use a separate symbolic algebra package for writing equations, except for the disk feet modifications.
- The symmetry of $\mathbf{M}$ reduces the number of operations needed to generate its columns (this is true for both symbolic and on-line schemes).
- The algorithm is the same for any linkage of the type described by Figure 2.1; the only things that change are the user-defined rotation matrices. So, once the code is written (for a variable number of links), it can be used for many different models.

- The code is modular and can be modified to include different contact conditions or branching structures.

In short, the flexibility and simplicity of the above algorithm tends to outweigh its efficiency disadvantages, since it can be used for all of these (simple) models. One other thing to keep in mind when discussing algorithm efficiency is that the efficiency of the symbolic code generated is dependent on the strategy used to optimize it and not just on the mechanics approach. An otherwise-efficient approach can still result in inefficient code if there are many redundant calculations or a lack of trigonometric simplification.

Another option is to use commercial multibody codes such as Working Model®, ADAMS, or DADS to study the motions of the models. But for systematically varying parameters and finding limit cycles (including unstable limit cycles), simulation packages themselves are generally not sufficient. However, as a simulation check, some undergraduates in the Ruina lab developed simulations of a 2D point-foot model in DADS, and of a 2D kneed model in Working Model® whose results agreed with the numerical simulations.

2.5 Collision Calculations

This section describes the computations that take place between different smooth motion phases of the walker; in general, these phases are separated by sticking (plastic) collisions (heelstrike or kneestrike). The smooth ODEs are generally integrated forward in time until the state vector approaches some collision condition which can be written as

$$c(\boldsymbol{\theta}) = 0 \tag{2.29}$$

Note that c is a scalar function of the state (more specifically, of the configuration). Approaching heelstrike, the function $c(\boldsymbol{\theta})$ often corresponds to the swing foot height above the ground, or some similar calculation, as in Equation 3.3 for a point-foot walker. Approaching kneestrike, $c(\boldsymbol{\theta})$ corresponds to the difference between the swing knee angle and the locked knee angle (see Figure 4.1).

Note, in the case of heelstrike, there is the possibility that the swing foot might scuff (i.e. temporarily pass underground) and Equation 2.29 might be true for some short period of time at mid-stride; this is generally ignored. For instance, straight-legged walkers in 2-D always scuff when both legs are parallel. So the while-loop must be able to tell the difference between a scuff and a true heelstrike.

With the above caveat stated, and ignoring scuffing, the smooth ODEs are integrated in a while-loop for as long as

$$c(\boldsymbol{\theta}) + h\dot{c}(\boldsymbol{\theta}) > 0 \tag{2.30}$$

where h is the time step of the numerical integration, and c and $\dot{c}$ are evaluated at each time step.

2.5.1 Getting Onto The Poincaré Section

The simulation exits the while-loop when the system is less than h time units away from a collision. Before the collision transition equations can be applied, the collision condition in Equation 2.29 must hold true to at least the integration tolerance, and to higher precision if possible. To do this, one can use a Newton method to find zeros of c.

A simple version of the algorithm is as follows.

- Exit ODE while-loop with current c and $\dot{c}$ and store current time step h_0.
- While abs(c) > numerical tolerance/1000

  1. Calculate new time step $h = -c/\dot{c}$.
  2. Evaluate state derivatives and compute new state with Runge-Kutta method.
  3. Compute new values for c and $\dot{c}$ based on new state.

- Draw final configuration after c converges to near-zero and restore original time step h_0.

Typically, the numerical tolerance is on the order of 1e-10, so the algorithm should converge quadratically to $c < 1e - 13$. The factor of 1000 is somewhat arbitrary.

After the algorithm is complete, one can apply the approporiate collision calculation (heelstrike or kneestrike) described below.

2.5.2 Kneestrike in 2-D

Using McGeer's angles, just before kneestrike, the state space consists of a stance angle θ_{st}^{-}, swing thigh angle θ_{th}^{-}, swing shank angle θ_{sh}^{-}, and their respective rates (see Figure 4.1). The "-" superscript indicates that the state is immediately before kneestrike; this state is known from the numerical integration and the algorithm described previously. McGeer's angles are used here to explain concepts in 2-D kneestrike and heelstrike, but the code actually uses the angles described by Figure 2.2.

Just after kneestrike, the state space will consist of a stance angle θ_{st}^{+}, swing leg angle θ_{sw}^{+} (or θ_{th}^{+}), and their respective rates. The post-kneestrike angles are basically the same as the pre-kneestrike angles, except that the swing thigh and swing shank are referred to collectively as the swing leg. For both kneestrike and heelstrike, the principles described and the essence of the algorithm will hold true regardless of the particular angles used for the configuration.

$$\theta_{st}^{+} = \theta_{st}^{-} \tag{2.31}$$

$$\theta_{sw}^{+} = \theta_{th}^{-} = \theta_{sh}^{-} \tag{2.32}$$

The post-collision angular rates $\dot{\theta}_{st}^{+}$ and $\dot{\theta}_{sw}^{+}$ are calculated by considering the two free-body diagrams (FBDs) shown in Figure 2.7, at the instant of kneestrike. The impulse from the collision is assumed to dominate all other forces/impulses at the instant of the collision; this is a standard assumption in rigid-body dynamics.

From the FBDs, angular momentum will be conserved through the collision for the whole walker about the stance foot contact point ($\mathbf{H}_{tot/cp}^{-} = \mathbf{H}_{tot/cp}^{+}$), and the swing leg about the hip ($\mathbf{H}_{sw/h}^{-} = \mathbf{H}_{sw/h}^{+}$). Since this is a 2-D case, there are equations for angular momentum conservation through kneestrike with which to solve for the two unknowns $\dot{\theta}_{st}^{+}$ and $\dot{\theta}_{sw}^{+}$.

$\mathbf{H}_{sw/h}^{-}$ and $\mathbf{H}_{tot/cp}^{-}$ are calculated as follows.

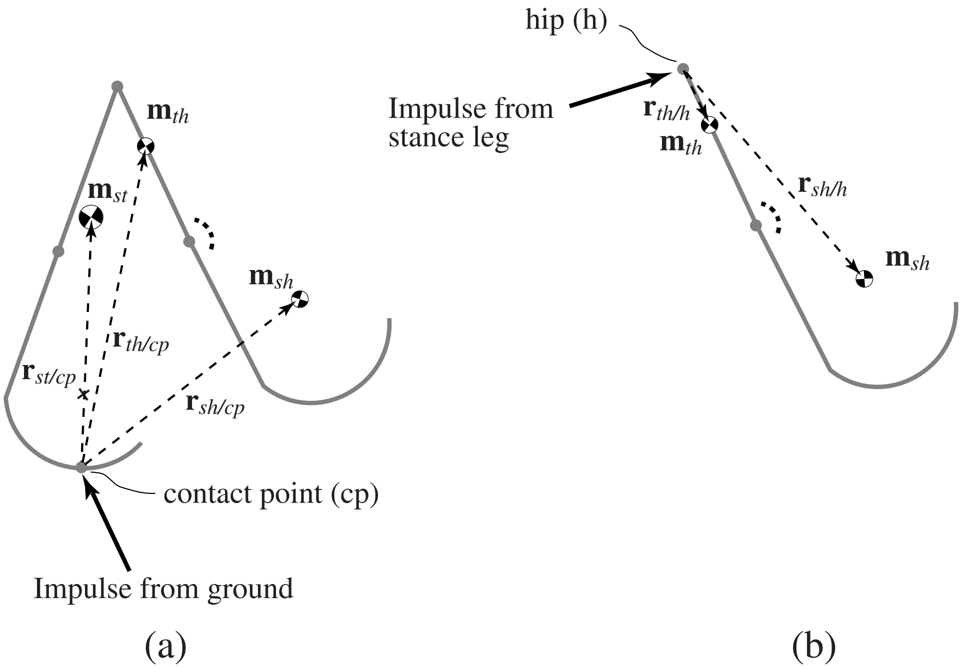

Figure 2.7: Free-body diagrams of (a) the entire walker and (b) the swing leg, each showing the impulses acting at the instant of collision. Angular momentum is conserved at this instant for the whole walker about the stance foot contact point, and for the swing leg about the hip.

$$\begin{aligned}\mathbf{H}^-_{tot/cp} = \mathbf{r}_{th/cp} \times m_{th}\mathbf{v}^-_{th} + \mathbf{I}^c_{th}\boldsymbol{\omega}^-_{th} + \\ \mathbf{r}_{sh/cp} \times m_{sh}\mathbf{v}^-_{sh} + \mathbf{I}^c_{sh}\boldsymbol{\omega}^-_{sh} + \\ \mathbf{r}_{st/cp} \times m_{st}\mathbf{v}^-_{st} + \mathbf{I}^c_{st}\boldsymbol{\omega}^-_{st}\end{aligned} \tag{2.33}$$

$$\begin{aligned}\mathbf{H}^-_{sw/h} = \mathbf{r}_{th/h} \times m_{th}\mathbf{v}^-_{th} + \mathbf{I}^c_{th}\boldsymbol{\omega}^-_{th} + \\ \mathbf{r}_{sh/h} \times m_{sh}\mathbf{v}^-_{sh} + \mathbf{I}^c_{sh}\boldsymbol{\omega}^-_{sh}\end{aligned} \tag{2.34}$$

where $\mathbf{r}_{X/A}$ is the vector from point A to the center of mass of segment X, m_X is the mass of segment X, $\mathbf{v}^-_X$ is the velocity of segment X just before kneestrike, $\mathbf{I}^c_X$ is the moment of inertia of segment X about its center of mass, and $\boldsymbol{\omega}^-_X$ is the angular velocity of segment X just before kneestrike. Note that $\mathbf{v}^-_X$ and $\boldsymbol{\omega}^-_X$ are functions of $\dot{\theta}^-_{st}$ and $\dot{\theta}^-_{sw}$. Recall also that in the 2-D case, all the angular momentum is along the $\hat{\mathbf{z}}$ direction (out of the page).

The equations are identical for $\mathbf{H}^+_{sw/h}$ and $\mathbf{H}^+_{tot/cp}$ (the angular momenta of the same two systems just after heelstrike), except that the "-" superscripts become "+" superscripts. The complication, however, is that $\dot{\theta}^+_{st}$ and $\dot{\theta}^+_{sw}$ are unknowns, so one actually needs to derive the matrix of coefficients $\mathbf{K}$ in the following equation.

$$\begin{bmatrix}\{\mathbf{H}^-_{tot/cp}\} \cdot \hat{\mathbf{z}} \\ \{\mathbf{H}^-_{sw/h}\} \cdot \hat{\mathbf{z}}\end{bmatrix} = \underbrace{\begin{bmatrix}K_{11} & K_{12} \\ K_{21} & K_{22}\end{bmatrix}}_{\mathbf{K}} \begin{bmatrix}\dot{\theta}^+_{st} \\ \dot{\theta}^+_{sw}\end{bmatrix} \tag{2.35}$$

This can be done numerically with a procedure like that of Section 2.4.1 by setting $\dot{\theta}^+_{st}$ and $\dot{\theta}^+_{sw}$ alternately to 1 while the other is zero and computing $\mathbf{H}^+_{tot/cp}$ and $\mathbf{H}^+_{sw/h}$ each time, with computations like those in Equations 2.33 and 2.34. Another method is to generate symbolic code for $\mathbf{H}^+_{tot/cp}$ and $\mathbf{H}^+_{sw/h}$ and pick out the coefficients of $\dot{\theta}_{st}^{+}$ and $\dot{\theta}_{sw}^{+}$ using a symbolic algebra package such as Maple ®. The method generally used here is based on a numerical shortcut, and is described in Section 2.5.5 below.

Once $\mathbf{K}$ is found, $\dot{\theta}_{st}^{+}$ and $\dot{\theta}_{sw}^{+}$ are computed by inverting Equation 2.35; this gives a set of initial conditions for the smooth, 2-link ODEs.

2.5.3 Heelstrike in 2-D

Although the principles are the same for heelstrike in 2-D as for kneestrike in 2-D, heelstrike is slightly more complicated because it involves swapping the stance and swing legs. In this section, the reader is assumed to be comfortable with the principles outlined above in Section 2.5.2, so some of the explanations will not be repeated. The model here is assumed to have knees; simplification to a straight-legged model is straightforward and will be summarized below.

Calculating New Angles

Just before heelstrike, the state space consists of a stance angle θ_{st}^{-}, swing leg angle θ_{sw}^{-}, and their respective rates (see Figure 4.1). Just after heelstrike, the state space will consist of a stance angle θ_{st}^{+}, swing thigh angle θ_{th}^{+}, swing shank angle θ_{sh}^{+}, and their respective rates. The pre- and post-collision angles are related by the following equations.

$$\theta_{st}^{+} = -\theta_{st}^{-} \tag{2.36}$$

$$\theta_{th}^{+} = -\theta_{sw}^{-} \tag{2.37}$$

$$\theta_{sh}^{+} = -\theta_{sw}^{-} = \theta_{th}^{+} \tag{2.38}$$

Calculating New Angular Rates

The post-collision angular rates $\dot{\theta}_{st}^{+}$, $\dot{\theta}_{th}^{+}$, and $\dot{\theta}_{sh}^{+}$ are calculated by considering the three FBDs shown in Figure 2.8, at the instant of heelstrike. As before, one assumes that the collision impulse at the new contact point dominates all other forces/impulses and that there is no impulse at the trailing (former stance) leg. While not provably correct, this is a self-consistent assumption producing simulation results which correctly predict the behavior of physical models. This means that the simulation results can be confidently used as a guide to construct stable physical models with appropriate foot clearance, etc.

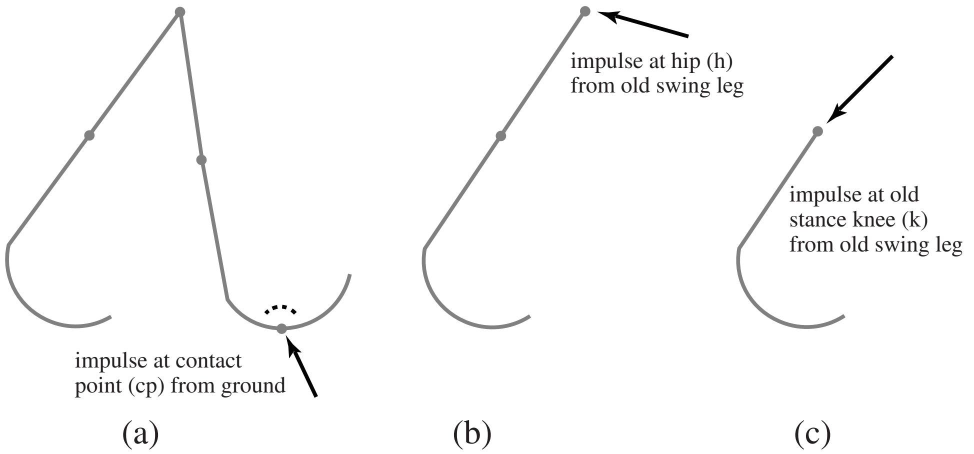

Figure 2.8: Free-body diagrams of (a) the entire walker, (b) the former stance leg (new swing leg), and (c) the former stance shank (new swing shank) showing the impulses acting at the instant of collision. Angular momentum is conserved at this instant for the whole walker about the new stance foot contact point, for the new swing leg about the hip, and for the new swing shank about the new swing knee.

From the FBDs, angular momentum will be conserved through the collision for the whole walker about the new contact point ($\mathbf{H}_{tot/cp}^{-} = \mathbf{H}_{tot/cp}^{+}$), for the new swing leg about the hip ($\mathbf{H}_{sw/h}^{-} = \mathbf{H}_{sw/h}^{+}$), and for the new swing shank about the new swing knee ($\mathbf{H}^-_{sh/k} = \mathbf{H}^+_{sh/k}$). Since this is a 2-D case, there are three equations for angular momentum conservation through heelstrike with which to solve for the three unknowns $\dot{\theta}^+_{st}$ and $\dot{\theta}^+_{th}$, and $\dot{\theta}^+_{sh}$.

As with kneestrike, $\mathbf{H}^-_{tot/cp}$, $\mathbf{H}^-_{sw/h}$, and $\mathbf{H}^-_{sw/k}$ are calculated directly, while the components of $\mathbf{H}$ in the following equation must be obtained with some indirect method; the method of choice is described in Section 2.5.5 below. Once the components of $\mathbf{H}$ are determined, one can solve for $\dot{\theta}^+_{st}$, $\dot{\theta}^+_{th}$, and $\dot{\theta}^+_{sh}$ by inverting $\mathbf{H}$ in the following equation.

$$\begin{bmatrix} \{\mathbf{H}^-_{tot/cp}\} \cdot \hat{\mathbf{z}} \\ \{\mathbf{H}^-_{sw/h}\} \cdot \hat{\mathbf{z}} \\ \{\mathbf{H}^-_{sh/k}\} \cdot \hat{\mathbf{z}} \end{bmatrix} = \underbrace{\begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} \dot{\theta}^+_{st} \\ \dot{\theta}^+_{th} \\ \dot{\theta}^+_{sh} \end{bmatrix} \tag{2.39}$$

Heelstrike for Straight-Legged Walkers

If the walker has no knees, simply ignore FBD c in Figure 2.8 and the corresponding equation ($\mathbf{H}^-_{sh/k} = \mathbf{H}^+_{sh/k}$). Instead, solve for $\dot{\theta}^+_{st}$ and $\dot{\theta}^+_{th}$ with $\mathbf{H}$ of size 2×2 (the upper left 2×2 block of $\mathbf{H}$ in Equation 2.39).

2.5.4 Heelstrike in 3-D

Heelstrike in 3-D follows the same principles as heelstrike in 2-D except that there are more equations and unknowns, and the post-collision angle calculation and leg switch are slightly more complicated. It is assumed that the walker has some non-zero hip spacing and foot radius, as in Figure 5.1; it is also assumed that the walker has no knees, but extending this procedure to kneed 3-D walkers is straightforward.

Calculating New Angles

Just before heelstrike, the state space consists of a heading (or steer) angle θ_1^-, bank (or lean) angle θ_2^-, stance leg angle θ_3^-, relative swing leg angle θ_4^-, and their respective rates (see Figure 5.1). Note that θ_4 is measured differently than θ_{sw} in the previous cases.

Just after heelstrike, the state space will consist of nominally the same angles and rates, but with different values. At heelstrike, the base reference frames (Euler-angle frames) are moved from the old contact point to the new contact point. Frames 0, 1, and 2 do not change their orientation through the heelstrike; thus the heading angle θ_1 and the bank angle θ_2 remain unchanged through the collision. The stance angle and relative swing angle must be re-calculated. The angles change according to the following equation.

$$\theta_1^+ = \theta_1^- \tag{2.40}$$

$$\theta_2^+ = \theta_2^- \tag{2.41}$$

$$\theta_3^+ = \theta_4^- - \pi + \theta_3^- \tag{2.42}$$

$$\theta_4^+ = \pi - \theta_3^+ + \theta_3^- \tag{2.43}$$

$$\tag{2.44}$$

See Figure 2.9a for reference. In the case of zero hip spacing, these equations reduce to the equations for the 2-D case, since the walker becomes a planar object.

Resetting Parameters

As stated above, the $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$ directions of frames 0, 1, and 2 remain constant through heelstrike. For frames 3 and 4 (fixed to the stance and swing legs, respectively) only the $\hat{\mathbf{z}}$ directions remain constant; the $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ directions change, since angles θ_3 and θ_4 change.

Resetting Position Vectors Since the reference frames switch legs at heelstrike, the stance and swing legs have switched their relative orientation in the $\hat{\mathbf{z}}_3$ direction. If the swing leg had been, say, in the positive $\hat{\mathbf{z}}_3$ direction with respect to the stance leg prior to heelstrike, it will be in the negative $\hat{\mathbf{z}}_3$ direction with respect to the stance leg after heelstrike. So, the $\hat{\mathbf{z}}$ components of the vectors ${}^2\mathbf{p}_3$, ${}^3\mathbf{p}_3^c$, and ${}^4\mathbf{p}_4^c$, will change in sign at each heelstrike, as shown in Figure 2.9 (also see Figure 5.1).

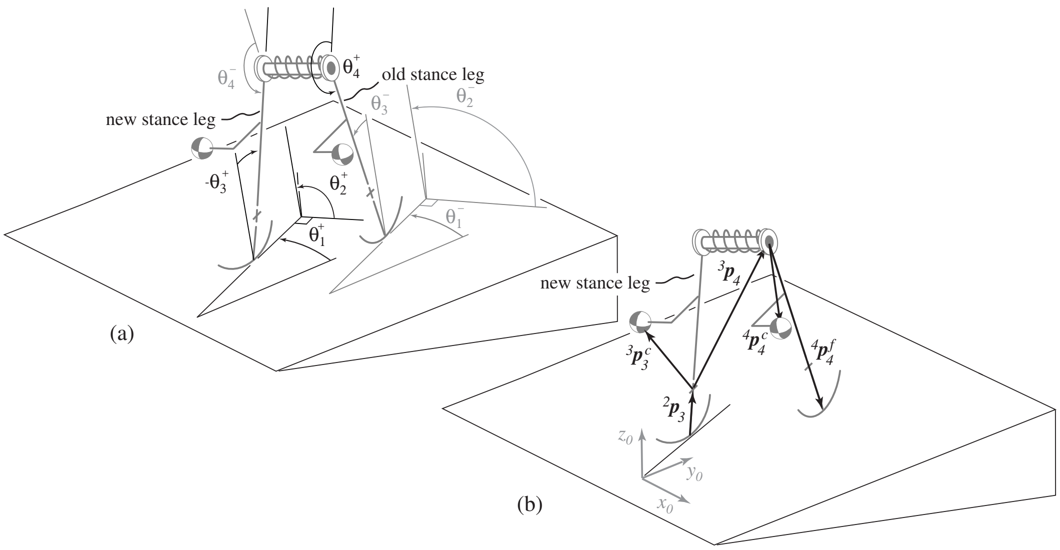

Figure 2.9: Diagrams showing (a) how angles are defined immediately after heelstrike, and (b) how the vectors ${}^3\mathbf{p}_4$, ${}^3\mathbf{p}_3^c$, and ${}^4\mathbf{p}_4^c$ are defined after heelstrike.

Resetting I matrices With respect to the legs, frames 3 and 4 have the opposite orientation for each of their $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$ directions, meaning that $\hat{\mathbf{x}}_3$ points from foot to hip while $\hat{\mathbf{x}}_4$ points from hip to foot, $\hat{\mathbf{y}}_3$ points forward while $\hat{\mathbf{y}}_4$ points backward, and $\hat{\mathbf{z}}_3$ points laterally while $\hat{\mathbf{z}}_4$ points medially. One consequence of this (also see Figure 2.3) is that the moment of inertia matrices about their centers of mass are equal (${}^3\mathbf{I}^c{}_3 = {}^4\mathbf{I}^c{}_4$). However, when switching $\hat{\mathbf{z}}_3$ and $\hat{\mathbf{z}}_4$ directions through heelstrike, the off-diagonal $\hat{\mathbf{z}}$ terms in ${}^3\mathbf{I}^c{}_3$ and ${}^4\mathbf{I}^c{}_4$ will also switch sign.

Summary of Parameter Changes The changes in signs of certain parameter vectors and matrices are summarized below. The numbers in parenthesis denote vector and matrix indices.

$$ {}^2\mathbf{p}_3(3)^+ = -{}^2\mathbf{p}_3(3)^- \tag{2.45}$$

$$ {}^3\mathbf{p}_3^c(3)^+ = -{}^3\mathbf{p}_3^c(3)^- \tag{2.46}$$

$$ {}^4\mathbf{p}_4^c(3)^+ = -{}^4\mathbf{p}_4^c(3)^- \tag{2.47}$$

$$ {}^3\mathbf{I}^c{}_3(1,3)^+ = -{}^3\mathbf{I}^c{}_3(1,3)^- \quad {}^3\mathbf{I}^c{}_3(2,3)^+ = -{}^3\mathbf{I}^c{}_3(2,3)^- \tag{2.48}$$

$$ {}^3\mathbf{I}^c{}_3(3,2)^+ = -{}^3\mathbf{I}^c{}_3(3,2)^- \quad {}^3\mathbf{I}^c{}_3(3,1)^+ = -{}^3\mathbf{I}^c{}_3(3,1)^- \tag{2.49}$$

$$ {}^4\mathbf{I}^c{}_4(1,3)^+ = -{}^4\mathbf{I}^c{}_4(1,3)^- \quad {}^4\mathbf{I}^c{}_4(2,3)^+ = -{}^4\mathbf{I}^c{}_4(2,3)^- \tag{2.50}$$

$$ {}^4\mathbf{I}^c{}_4(3,2)^+ = -{}^4\mathbf{I}^c{}_4(3,2)^- \quad {}^4\mathbf{I}^c{}_4(3,1)^+ = -{}^4\mathbf{I}^c{}_4(3,1)^- \tag{2.51}$$

$$ \tag{2.52}$$

Before and after heelstrike, ${}^3\mathbf{I}^c{}_3 = {}^4\mathbf{I}^c{}_4$.

Calculating New Angluar Rates

The post-collision angular rates are calculated by considering the FBDs shown in Figure 2.10, at the instant of heelstrike. As before, it is assumed that the collision impulse at the new contact point dominates all other forces/impulses and that there is no impulse at the trailing (former stance) leg.

Angular momentum is conserved through the collision for the whole walker about

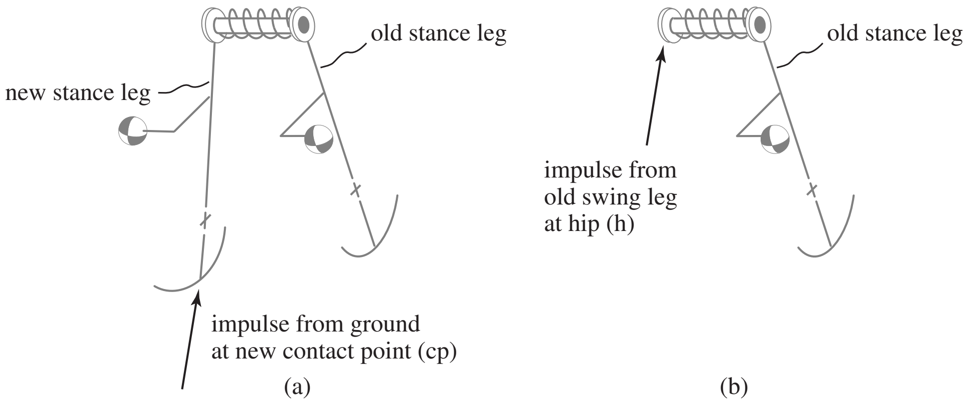

Figure 2.10: Free-body diagrams of (a) the entire walker and (b) the former stance leg (new swing leg), showing the impulses acting at the instant of collision. Angular momentum is conserved at this instant for the whole walker about the new stance foot contact point, and for the new swing leg about the hip axis.

the new contact point ($\mathbf{H}^-_{tot/cp} = \mathbf{H}^+_{tot/cp}$), and for the new swing leg about the hip ($\mathbf{H}^-_{sw/h} = \mathbf{H}^+_{sw/h}$). The first equation is a vector equation in 3-D, and yields three scalar equations, while the second equation is only along the hip axis ($\hat{\mathbf{z}}_3$). As before, $\mathbf{H}^-_{tot/cp}$ and $\mathbf{H}^-_{sw/h}$ are known (vector) quantities. Thus, there are four equations from angular momentum conservation through heelstrike with which to solve for the four unknowns $\dot{\theta}_1^+$, $\dot{\theta}_2^+$, $\dot{\theta}_3^+$, and $\dot{\theta}_4^+$. They are found by solving for the components of $\mathbf{H}$ in Equation 2.53 and then inverting it as before.

$$\begin{bmatrix} \{\mathbf{H}^-_{tot/cp}\} \cdot \hat{\mathbf{x}}_3 \\ \{\mathbf{H}^-_{tot/cp}\} \cdot \hat{\mathbf{y}}_3 \\ \{\mathbf{H}^-_{tot/cp}\} \cdot \hat{\mathbf{z}}_3 \\ \{\mathbf{H}^-_{sw/h}\} \cdot \hat{\mathbf{z}}_3 \end{bmatrix} = \underbrace{\begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \\ H_{41} & H_{42} & H_{43} & H_{44} \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} \dot{\theta}_1^+ \\ \dot{\theta}_2^+ \\ \dot{\theta}_3^+ \\ \dot{\theta}_4^+ \end{bmatrix} \tag{2.53}$$

2.5.5 A Shortcut for K and H

In each case considered above (kneestrike, 2-D heelstrike, 3-D heelstrike), the post-collision angular momenta are derived by considering a series of FBDs starting with the whole walker about its ground contact point and ending with the outermost link (the swing leg or swing knee) about its hinge axis.

Using the notation and procedure developed in Section 2.3, one can write the angular momentum of each system about its contact point or hinge point, immediately after collision, dotted with the appropriate axis, as follows.

$$ {}^{i+1}h_{i+1} = \{m_{i+1}({}^{i+1}\mathbf{p}^{c}_{i+1} \times {}^{i+1}\mathbf{v}^{c}{}_{i+1}) + {}^{i+1}\mathbf{I}^{c}{}_{i+1}{}^{i+1}\boldsymbol{\omega}_{i+1}\} \cdot \hat{\mathbf{z}}_{i+1} \tag{2.54} $$

h_i can be written as a vector function of the θ_i as done in Equations 2.35, 2.39, and 2.53.

$$ \begin{bmatrix} {}^{1}h_1 \\ \vdots \\ {}^{i}h_i \end{bmatrix} = \underbrace{\begin{bmatrix} H_{11} & \dots & H_{1i} \\ \vdots & \ddots & \vdots \\ H_{i1} & \dots & H_{ii} \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_i \end{bmatrix} \tag{2.55} $$

The matrix $\mathbf{H}$ in this case is comprised of the same coefficients as the mass matrix $\mathbf{M}$ in the angular momentum balance algorithm in Sections 2.3 and 2.4. That is to say, $\mathbf{H} = \mathbf{M}$ (see also Section 2.3.4) So, if there is a procedure in place to find the coefficients of the mass matrix (as described in Section 2.4.1), one can simply calculate $\mathbf{M}$ (after resetting angles and parameters) and substitute it for $\mathbf{H}$.

Note that the use of this shortcut depends on using angles as defined in Section 2.3. For some discussions, these exact angles are not used (for example, θ_{th} is measured from the slope-normal instead of from the foot-hip reference line in Figure 4.1). This is because defining angles like θ_{th} in Figure 4.1 can be more intuitive in some cases, and makes plots easier to follow. However, in the heart of the actual integration procedure, relative angles are always used as defined in Section 2.3; this shortcut saves some time in equation-writing and helps to eliminate sources of possible error.

2.6 The Walking Map Or Stride Function

A *step* can be thought of as a piecewise function $\mathbf{f}({}^{i}\boldsymbol{\theta}^{+})$, the "stride function," which takes as input the list of values of the various angles and rates (the state variable vector $\boldsymbol{\theta}$) at a definite point in the motion, usually just after heelstrike of step i (denoted by the "+" superscript), and returns the values of $\boldsymbol{\theta}$ just after heelstrike $i+1$, or ${}^{i+1}\boldsymbol{\theta}^{+}$. $\mathbf{f}$ is also a function of some invariant set of model parameters (mass distribution, gravity, etc.), but for simplicity, this dependence is excluded from the notation.

The function $\mathbf{f}(\boldsymbol{\theta})$ can be generically dissected into the following parts, each of which has been described previously.

- **Numerical integration of smooth motion governed by ordinary differential equations (ODEs).** The ODEs are generally those of a multi-link pendulum, possibly with rolling ground contact; they are derived in section 2.3, and procedures for integration in section 2.4. ODEs are integrated until a collision (heelstrike or kneestrike) is imminent.

- **Detection of kneestrike or heelstrike and settling onto the Poincaré section.** Before the effects of any collisions can be computed, the integration procedure must home in on the pre-collision configuration to ensure that the state is exactly on the Poincaré section (at least to numerical tolerance, and preferrably better). This will usually involve finding a zero of some function of the configuration variables, known as a "collision condition". This procedure is described in section 2.5.

- **Calculation of collision effects and renaming variables.** Post-collision velocities are calculated by assuming angular momentum conservation before and after impact, about various points, for various sub-systems of the robot links, in much the same way as the equations of motion were derived. In the case of kneestrike, there are velocity discontinuities, but no renaming of variables, and the above two steps are repeated until heelstrike is detected. In the case of heelstrike, the stance and swing legs exchange roles, and so some renaming of configuration variables must occur, along with a change in angular velocities. Collisions are assumed to be plastic, meaning that there is no rebound (not that there is plastic deformation); these conditions are enforced in physical models by dead rubber on the feet and leaky suction cups at the knee joints. The collision calculations are described in more detail in section 2.5.

The above parts are almost always solved numerically, although one analytical scheme is presented in Chapter 3 for a very simple model. The final result after the last collision calculation, starting from a given set of initial conditions $\boldsymbol{\theta}$, yields one evaluation of the function $\mathbf{f}(\boldsymbol{\theta})$.

Figure 2.11 shows a plot of the evolution of one step as a stride function, starting with ${}^{i}\boldsymbol{\theta}^{+}$, and ending with ${}^{i+1}\boldsymbol{\theta}^{+}$.

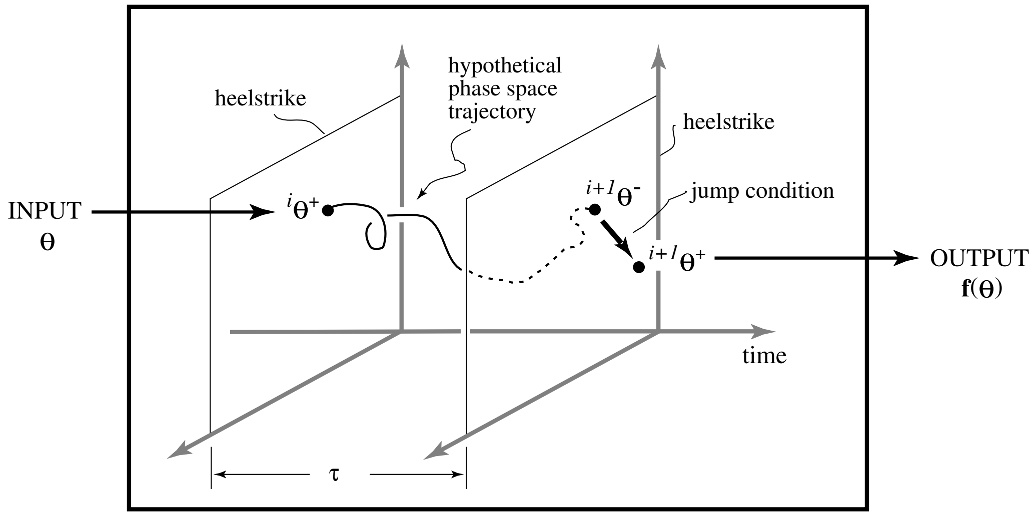

Figure 2.11: One evaluation of **f**. Time is shown on the horizontal axis, while the phase space variables are represented by the other axes. In general, the phase space can be multi-dimensional but in the plot, it is only two-dimensional. The step period is denoted by τ.

In the language of dynamical systems, McGeer's stride function is a Poincaré map. Many questions about the dynamics of a given walking model are reduced to questions about the mapping function $\mathbf{f}(\boldsymbol{\theta})$.

2.7 Gait Cycles: Fixed Points of the Map

A simple (period-one)*gait cycle*, if it exists, corresponds to a set of initial values for the angles and rates which lead back to the same angles and rates after one step. In the language of nonlinear dynamics, this repeated motion is called a *limit cycle.* $\boldsymbol{\theta}^*$ is a "fixed point" of the function $\mathbf{f}(\boldsymbol{\theta})$.

$$\mathbf{f}(^{i}\boldsymbol{\theta}^{+}) = {}^{i}\boldsymbol{\theta}^{+} =^{i+1} \boldsymbol{\theta}^{+} = \boldsymbol{\theta}^{*}. \tag{2.56}$$

This corresponds to a zero of the difference function $\mathbf{g}(\boldsymbol{\theta})$.

$$\mathbf{g}({}^{i}\boldsymbol{\theta}) \equiv \mathbf{f}({}^{i}\boldsymbol{\theta}) - {}^{i}\boldsymbol{\theta} \equiv^{i+1} \boldsymbol{\theta} - {}^{i}\boldsymbol{\theta} \tag{2.57}$$

In three dimensions, like McGeer, one searches for period-one gait cycles that switch heading and bank after one step but are otherwise identical. Specifically, using the angles in Figure 2.3,

$$\begin{aligned}
{}^{i}g_1(\boldsymbol{\theta}) &=^{i+1} {\theta_1}^{+} + {}^{i}{\theta_1}^{+} \\
{}^{i}g_2(\boldsymbol{\theta}) &=^{i+1} {\theta_2}^{+} + {}^{i}{\theta_2}^{+} - \pi \\
{}^{i}g_3(\boldsymbol{\theta}) &=^{i+1} {\theta_3}^{+} - {}^{i}{\theta_3}^{+} \\
{}^{i}g_4(\boldsymbol{\theta}) &=^{i+1} {\theta_4}^{+} - {}^{i}{\theta_4}^{+} \\
{}^{i}g_5(\boldsymbol{\theta}) &=^{i+1} {\dot{\theta}_1}^{+} + {}^{i}{\dot{\theta}_1}^{+} \\
{}^{i}g_6(\boldsymbol{\theta}) &=^{i+1} {\dot{\theta}_2}^{+} + {}^{i}{\dot{\theta}_2}^{+} \\
{}^{i}g_7(\boldsymbol{\theta}) &=^{i+1} {\dot{\theta}_3}^{+} - {}^{i}{\dot{\theta}_3}^{+} \\
{}^{i}g_8(\boldsymbol{\theta}) &=^{i+1} {\dot{\theta}_4}^{+} - {}^{i}{\dot{\theta}_4}^{+}
\end{aligned} \tag{2.58}$$

Note that this is not the only feasible definition of a gait cycle in 3-D. Coleman et al. (1997) defined a 3-D cycle to include two steps (as defined here), and used the standard 2-D definition $\mathbf{g}(\boldsymbol{\theta}) \equiv \mathbf{f}(\boldsymbol{\theta}) - \boldsymbol{\theta}$.

Besides period-one cycles, higher order cycles (limping gaits) can exist; a *period-two* gait cycle returns the same variable values after *two* steps, for instance; by definition, a period-one gait cycle is also a period-n gait cycle. The term "gait cycle" by itself implies a period-one gait cycle. Period-one motions are of central interest because they correspond to the important task of steady walking.

2.8 Searching For Gait Cycles

To find gait cycles, McGeer coupled the idea of a stride function with a multidimensional Newton-Raphson method. To follow his procedure, one searches, more or less systematically, for $\boldsymbol{\theta}^*$ such that $\mathbf{g}(\boldsymbol{\theta}^*) = \mathbf{0}$. In other words, the difference between ${}^{i+1}\boldsymbol{\theta}^+$ (the state after heelstrike $i+1$) and ${}^{i}\boldsymbol{\theta}^+$ (the state after heelstrike i) should be zero at a gait cycle. A Newton-Raphson root-finding procedure for $\mathbf{g}$ generally converges to the desired numerical accuracy.

One major advantage of the root-finding algorithm over straightforward simulation in finding gait cycles is that the root finding algorithm will locate both *stable* and *unstable* limit cycles, while simulation will only locate *stable* gait (see below).

2.8.1 Finding Zeros of the Difference Function g

Close to the fixed point $\boldsymbol{\theta}^*$, the difference function $\mathbf{g}(\boldsymbol{\theta})$ can be linearized as follows.

$$\mathbf{g}(\boldsymbol{\theta}_0) \approx \mathbf{g}(\boldsymbol{\theta}^*) + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \boldsymbol{\Delta\theta_0} \tag{2.59}$$

where $\boldsymbol{\theta}_0$ is the initial guess at a fixed point. $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ is the Jacobian of the difference function $\mathbf{g}$ (not to be confused with $\mathbf{J}$, the Jacobian of $\mathbf{f}$), which is evaluated numerically using forward difference as follows.

- First, evaluate $\mathbf{g}$ at $\boldsymbol{\theta_0}$ (call this $\mathbf{g}_0$).
- Then perturb the first element of $\boldsymbol{\theta}$ by ζ and re-evaluate $\mathbf{g}$. Call this $\mathbf{g}_p$. An estimate of the first column of $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$, then, is given by

$$\frac{\mathbf{g}_p - \mathbf{g}_0}{\zeta} \tag{2.60}$$

  where a good value for ζ is the square root of the integration tolerance.

- The remaining columns of $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ are formed in a similar way, perturbing the second, third, etc. elements of $\boldsymbol{\theta_0}$, re-evaluating $\mathbf{g}$ each time, and repeating the previous calculation in Equation 2.60.

Assuming that $\mathbf{g}(\boldsymbol{\theta}^*) = 0$ and $\mathbf{g}(\boldsymbol{\theta}_0)$ is close to zero but not less than numerical tolerance, Newton's Method provides the adjustment to $\boldsymbol{\theta}_0$,

$$\boldsymbol{\Delta\theta_0} = \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}^{-1} \mathbf{g}(\boldsymbol{\theta}_0) \tag{2.61}$$

and so the new guess for $\boldsymbol{\theta}^*$) is

$$\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 + \boldsymbol{\Delta\theta_0} \tag{2.62}$$

Together, Equations 2.61 and 2.62 constitute the iterative procedure known as a Newton or Newton-Raphson method. If the initial guess for $\boldsymbol{\theta}$ is close enough to a fixed point, and one is in a parameter region where the numerics behave appropriately, Newton's Method will converge quadratically to the fixed point value $\boldsymbol{\theta}^*$. Although Newton's method has no guarantee of global convergence, in practice, initial guesses can be based on results from simpler models, for which a solution is known, and slowly parameters can then be varied, solving for intermediate solutions, until the solution family disappears or a gait solution is found for the mechanism of interest. This is a heuristic explanation of the bifurcation technique of homotopy. Using this technique, gait cycles can usually be found if they exist. So the root finding aspects of the work involve a mixture of intuitively based theoretical model definition, on starting new searches on known solutions, and on various numerical methods.

Assuming that one's code is free of error and the model is well-posed, failure of Newton's method to converge generally has one of the following causes.

1. The initial guess $\boldsymbol{\theta}_0$ is not close enough to $\boldsymbol{\theta}^*$. This can be a problem when dealing with extremely unstable walkers.

2. The slope of one of the state variable vs. parameter plots is approaching infinite slope. This is known in the bifurcation literature as a "turning point," and is indicated by an unexpected zero value in the stride function Jacobian $\mathbf{J}$. The solution is to temporarily augment the $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ matrix, or parameterize the solution path by arclength, as described by Seydel (1988).

3. There is no fixed point for the parameter combination specified, or the solution path is approaching its terminus. This problem has the same symptoms as the previous one, but cannot be fixed by augmenting $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$.

4. There is some sort of family of solutions. The solution is to augment $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ to choose only one particular solution from the family. For instance, this can occur with planar solutions in 3-D; one fix is to only allow solutions which head straight downslope.

With a experience, one begins to get a little intuition for path-following and this makes it easier to recognize one of the above situations.

2.9 Stability of the Gait

Near a fixed point, one can linearize the stride function $\mathbf{f}$ as follows.

$$\mathbf{f}(\boldsymbol{\theta}^* + \hat{\boldsymbol{\theta}}) \approx \mathbf{f}(\boldsymbol{\theta}^*) + \mathbf{J}\hat{\boldsymbol{\theta}} \tag{2.63}$$

where $\mathbf{J}$ is the Jacobian matrix of partial derivatives of $\mathbf{f}$ with respect to each angle and rate.

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \qquad \text{with components } \frac{\partial f_i}{\partial \theta_j} \tag{2.64}$$

After finding a limit cycle, approximate $\mathbf{J}$ by numerically evaluating $\mathbf{f}$ a number of times in a small neighborhood of $\boldsymbol{\theta}^*$, or use Equation 2.66.

Since $\mathbf{f}(\boldsymbol{\theta}^*) = \boldsymbol{\theta}^*$, small perturbations $\hat{\boldsymbol{\theta}}$ to the limit cycle state vector $\boldsymbol{\theta}^*$ at the start of a step will grow or decay from the ith step to the $i+1$th step approximately according to $\hat{\boldsymbol{\theta}}^{i+1} \approx \mathbf{J}\hat{\boldsymbol{\theta}}^{i}$.

If the map Jacobian $\mathbf{J}$ has all of its eigenvalues inside the unit circle, all sufficiently small perturbations $\hat{\boldsymbol{\theta}}$ will decay to $\mathbf{0}$, the system will return to its limit cycle, and the cycle is *asymptotically stable*. If the Jacobian has any eigenvalues outside the unit circle, any perturbation with a component along the corresponding eigenvector will bump the system off the limit cycle, and the cycle is *unstable*. If an eigenvalue has magnitude of one, then the cycle is *neutrally stable* for infinitesimal perturbations along the corresponding eigenvector, and such perturbations will neither shrink nor grow (to first order). Many times, persistent eigenvalues of magnitude one have some obvious physical significance; they can signify a one-parameter family of gait solutions, for instance.

The eigenvalues of the linearization $\mathbf{J}$ are a surprisingly useful characterization of stability, with one drawback described in 5.1.1.

2.9.1 Relationship Between J And $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ in 2-D

Instead of evaluating $\mathbf{J}$ column-by-column at a fixed point in 2-D, which is time-consuming, one can use the relationship

$$\mathbf{g}(\boldsymbol{\theta}) = \mathbf{f}(\boldsymbol{\theta}) - \boldsymbol{\theta} \tag{2.65}$$

to arrive at the relationship between $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}$, namely

$$\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} - \mathbf{I} \tag{2.66}$$

Note that this is only true if Equation 2.65 holds true.

2.9.2 Relationship Between J And $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ in 3-D

Parallelling Equation 2.66, one can use Equation 2.58 to arrive at the relationship between $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}$ in three dimensions.

$$\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} - \mathbf{I}^* \tag{2.67}$$

where $\mathbf{I}^* = \mathrm{diag}(-1, -1, 1, 1, -1, -1, 1, 1)$.

2.10 Other Gait Characteristics

Although stability is usually the most interesting gait characteristic, there are other characteristics such as step period, efficiency (which can be defined in various ways), minimum swing foot clearance, etc. These characteristics are functions of the model parameters and of the particular fixed point $\boldsymbol{\theta}^*$, if more than one exists.

2.11 A Graphical Summary

A graphical summary of the above terminology is provided in Figure 2.12. In Figure 2.12, Figure 2.11 is mapped back onto itself so that time is no longer an axis (note that time does not explicitly enter into the stride function **f**); this is a common way of graphically portraying a Poincaré map.

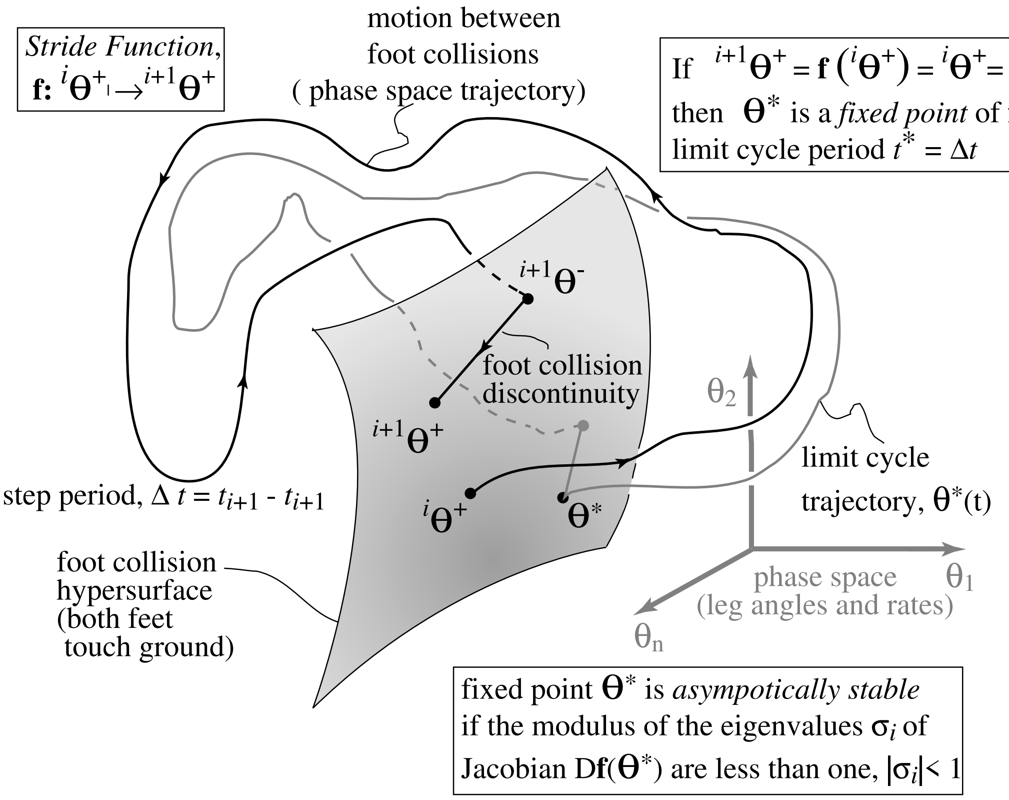

Figure 2.12: Schematic of a generic stride function evaluation from ${}^{i}\boldsymbol{\theta}^{+}$ to ${}^{i+1}\boldsymbol{\theta}^{+}$ and a limit cycle and fixed point $\boldsymbol{\theta}^{*}$ (in gray). Kneestrikes may also be seen as discontinuities during the cycle. Summary of jargon is provided in the text boxes. This figure is based on a similar one in Coleman (1998b).

2.12 Parameter Variations: Embedding the Loop

The gaits of a mechanism can have different stability, depending on the initial state vector, mass distribution, ramp slope, etc. Sometimes, one would like to see how certain gait characteristics change as different parameters are varied (the fixed point also varies, but this is not of primary interest).

Other times, one's goal is to optimize certain gait characteristics, or combinations of characteristics, by varying parameters. For instance, the design of a successful physical passive-dynamic kneed walker depends on finding fixed points that are stable, have sufficient ground clearance for the swing foot at mid-stride, and sufficient passive locking torque at the stance knee. Parameter variation could also include adjusting feedback gains and actuation strategies in a controlled model.

In all these cases, the overall strategy is to vary parameters in some continuous or automated way, find fixed points along the way, and analyze the resulting gait characteristics of interest at each fixed point.

A superficial counting analysis predicts that if the number of adjustable system parameters exceeds the number of characteristics of interest, one should be able to simultaneously modify all the characteristics as desired. However there is no assurance that this generically-possible process will not terminate at a local minimum or maximum above or below the desired threshold (or a parameter boundary) before a parameter set can be found with the desired characteristics. Simple optimization techniques such as steepest-descent algorithms can be used to find favorable gait characteristics as functions of parameters.

In practice, however, such techniques must be modifed to ensure that Newton's method will be able to locate a fixed point each time it adjusts the parameter set.

Usually, Limiting the size of the parameter jumps ensures that the new fixed point can be found by using the previous fixed point as the initial guess at a new fixed point. More complicated *predictor* schemes are described by Seydel (1988). Another pitfall of automated parameter adjustments, however, is that they can lead into undesirable parameter regions, where solutions are often lost. See Coleman (1998b) and Chapter 5 for more discussion.

One can often use McGeer's algorithm with only a rudimentary knowledge of these tools. (A good introduction to path-following is the book by Seydel (1988).) Simple one-parameter searches can often provide some utility and insight with a miminum investment in code-writing, shown below and in Garcia et al. (1998).

2.12.1 Reality Checks

Our simulation code does not guarantee complete realism for a physical model, as discussed in Chapter 4. A brief example of the use of McGeer's algorithm and related reality checks follows.

2.12.2 Example: Constructing a Working Kneed Walker

Building physical devices may seem unneeded if one has good simulations. Although the physical models studied here are relatively simple, designing, building, and measuring the properties of prototypes often takes some effort.

Consider the Ruina lab's version of the kneed walker of McGeer (1990b). The variables and parameters used in this model are shown in Figure 2.13. (Some notes on the building of this model are contained in Appendix 2.14.) Tad McGeer showed, with simulations and a physical model, that for certain (anthropomorphic) parameter combinations, this model could exhibit stable, human-like gait down shallow slopes.

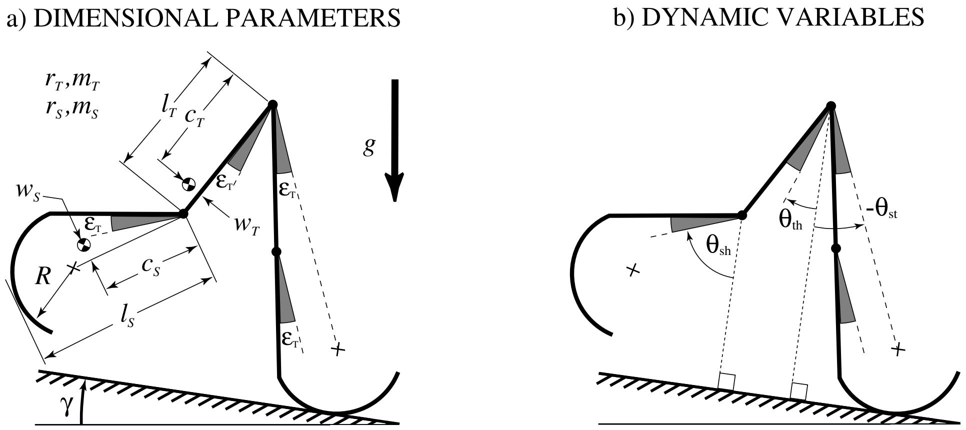

Figure 2.13: Our description of McGeer's kneed walking model. Shown above are (a) model parameters, and (b) dynamic variables. Radii of gyration and masses of thigh and shank are denoted by r_t, m_t, r_s, and m_s, respectively. The foot is a circular arc centered at the "+". ϵ_T is defined to be the angle between the stance thigh and the line connecting the hip to the foot center. Dynamic variable values θ_{st}, θ_{th}, and θ_{sh} are measured from ground-normal to lines offset by ϵ_T from their respective segments. A stop (not shown) at each knee prevents hyperextension of either knee. In straight-legged models, the knee is locked.

To mimic McGeer's work, several undergraduates in the lab [1] first built an imitation of McGeer's machine with similar leg length and proportions, but different mass distribution. After adding some masses to make the two legs dynamically equal, (generic procedure outlined in Appendix 2.14), the parameters were simulated; the result was prediction of an unstable period-1 gait with minimal foot clearance (3mm) and knee-locking characteristics as shown in Table 2.1.

[1] Some of the people directly and indirectly involved were John Camp, Mario Gomes, Lanise Baidas, Maria Hagan, Dan Jung, Jacqui Rodrick, Tony Liu, Jill Startzell, Larry Gosse, Jaime Estupinian, and Yan Yevmenenko. Apologies to those who contributed but whose names were omitted.

Loose reasoning suggested (based on human anatomy and on McGeer's observations) that a heavy thigh might prove beneficial. So, the first parameter that varied in the simulation was the thigh mass m_t. Adding 0.8 kg at the cm of the thigh turned out to stabilize the walker and improve foot clearance slightly (see "1st mod", row 2 of Table 2.1), but not enough to allow for a margin of error in the physical model.

The next goal was to improve foot clearance without adversely affecting stability. We introduced a point mass that we could add at some arbitrary point on the thigh, and varied the position and amount of mass. Adding 0.1718 kg a distance of about 0.0298 m from the hip on each leg produced the characteristics shown in row 3 of Table 2.1, labelled "2nd mod".

Now our model had adequate foot clearance, but we were still having problems with the stance knee unlocking unexpectedly during the gait. The last modification was to add 0.773 kg at the hip joint, which improved the minimum torque on the stance knee during the stride and at kneestrike of the swing leg. After this modification, the walker walked with acceptable robustness. It was later discovered that there was a slight impulsive unlocking torque on the stance knee at kneestrike of the swing leg, but this did not seem to affect the gait appreciably.

A strobe photo of the functional walker is shown in Figure 2.14. A materials list and copies of schematic drawings for constructing a physical model (not exactly the same as the one shown here) are shown in Appendix 2.14, along with some comments on model construction. See Chapter 4 for more experimental results and comparisons to theory for this model. Data from leg angle measurements at three slopes is shown superimposed on simulation results in Figure 4.6.

Table 2.1: Summary of relevant parameters and characteristics for initial kneed model that didn't work, labelled "original", and various stages of modification, until we arrived at a working set of parameters, labelled "final". The parameters m_t, r_t, and c_t are shown in Figure 2.13. The gait characteristics "max eig", "min cl", "min k trq", and "hstr imp" are the largest eigenvalue in modulus, the minimum foot clearance during the gait, the minimum locking torque on the stance knee during gait, and the impulsive torque on the new stance knee at heelstrike, respectively. The gait characteristics are based on simulation results. Parameters that are not shown are the same as those listed in the caption of Figure 4.3. Base units are kilograms, meters, and radians.

walker	m_t	r_t	c_t	max eig	min cl	min k trq	hstr imp
original	0.9868	0.1354	0.116	1.35	0.003	0.217	0.026
1st mod	1.7868	0.1006	0.116	0.52	0.006	0.266	0.011
2nd mod	1.9568	0.0992	0.108	0.50	0.010	0.293	0.011
final	2.3451	0.0992	0.091	0.45	0.012	0.353	0.011

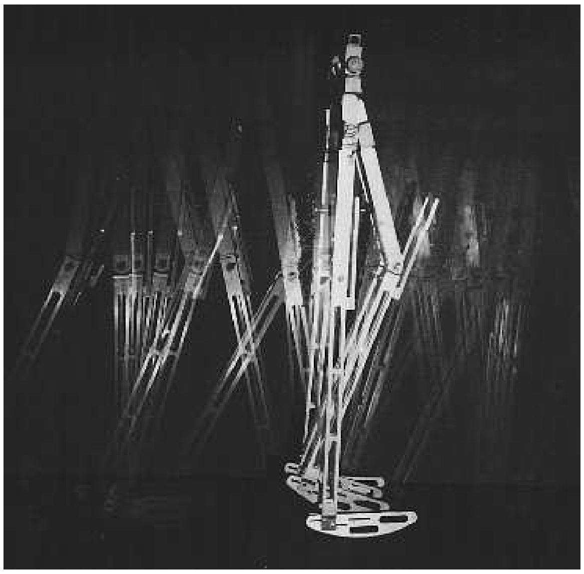

Figure 2.14: Our passive dynamic walker walking down a shallow ramp in the Ruina lab. The double leg-set constrains motions to a plane. The simulation shown in Figure 4.3 uses parameters measured from this walker.

2.13 How Crucial Is Stability?

In this section, several observations are presented which imply that for the most part, stability may not necessarily be critical to locomotion or other repeated tasks.

2.13.1 Mild Instability Seems Acceptable

When we relax our muscles and remain static, we are an unstable collection of (for simplicity) links and hinges. We would certainly collapse in a heap unless we contract some muscles to stand upright and/or initiate locomotion.

Because humans do have control and need to exercise this control to go where they want, slow instabilities may not be important. For example, many bicycles are passively stable in a limited range of speeds Hand (1988). This stability is lost by a very slow instability at high speeds (starting typically at about 18 mph).

Bicycle riders, on the other hand, only sense *increased* stability at higher speeds due apparently to the one decreasing eigenvalue.

Passive instabilities that are easily controlled and have long time constants may have little cost of any kind to controlled biological systems. Although much of this and other work in passive-dynamic locomotion concentrates on finding stable gait, the important essence of this research may end up in finding limit cycles without need for exponential stability.

On the other hand, as mentioned previously, the design of physical, absolutely-uncontrolled passive-dynamic walkers does depend on both motions *and* stability (finding fixed points of $\mathbf{f}$ and having the eigenvalues of the Jacobian $\mathbf{J}$ inside the unit circle on the complex plane). Besides stability, a theoretical model may need to satisfy other performance criteria such as acceptable foot clearance, sufficient knee-locking torques, minimal collisional impulses, high efficiency, high speed, etc.

2.13.2 Stabilizing Unstable Limit Cycles is Easy

The energetic cost associated with stabilizing an unstable limit cycle is minimal. If the perturbations from the limit cycle are infinitessimal in size, then the actuation input necessary to remain on the limit cycle is also infintessimally small. So an unstable limit cycle can in theory be stabilized with 0^+ energy cost. In practice, the energy cost is proportional to the amount of mechanical noise present in the system.

2.13.3 Unstable Period-1 Gaits Don't Always Lead to Falls

As is generally known in nonlinear dynamics, systems which exhibit period-doubling and chaos can have a chaotic attractor which is bounded and stable in some sense, since the system does not leave the attractor if it starts on or near it.

The simplest walker of Garcia et al. (1998) (also Chapter 3) and a kneed version in Garcia et al. (1997) (also Chapter 4) both exhibit this period doubling phenomenon in simulations; in both cases, the walker continues to walk at slopes above those which yield stable period-1 gait. Although the period-1 gait is unstable, the walker does not fall down because of the stability of the higher-period gaits and the chaotic attractor.

In addition, the control problem of several steps of varying length can be more easily dealt with in a chaotic region where many different step lengths and combinations thereof are accessible to the machine, as discussed by McGeer (1993b) and Garcia et al. (1998).

So, stable period-1 gait is not necessarily a prerequisite for stable controlled walking.

2.14 Some Notes On Building A Physical Model

2.14.1 Construction

This section contains excerpts and summary from the undergraduate lab reports by Yevmenenko (1996) and Yevmenenko (1997), which contain instructions and insights into building a physical passive walker.

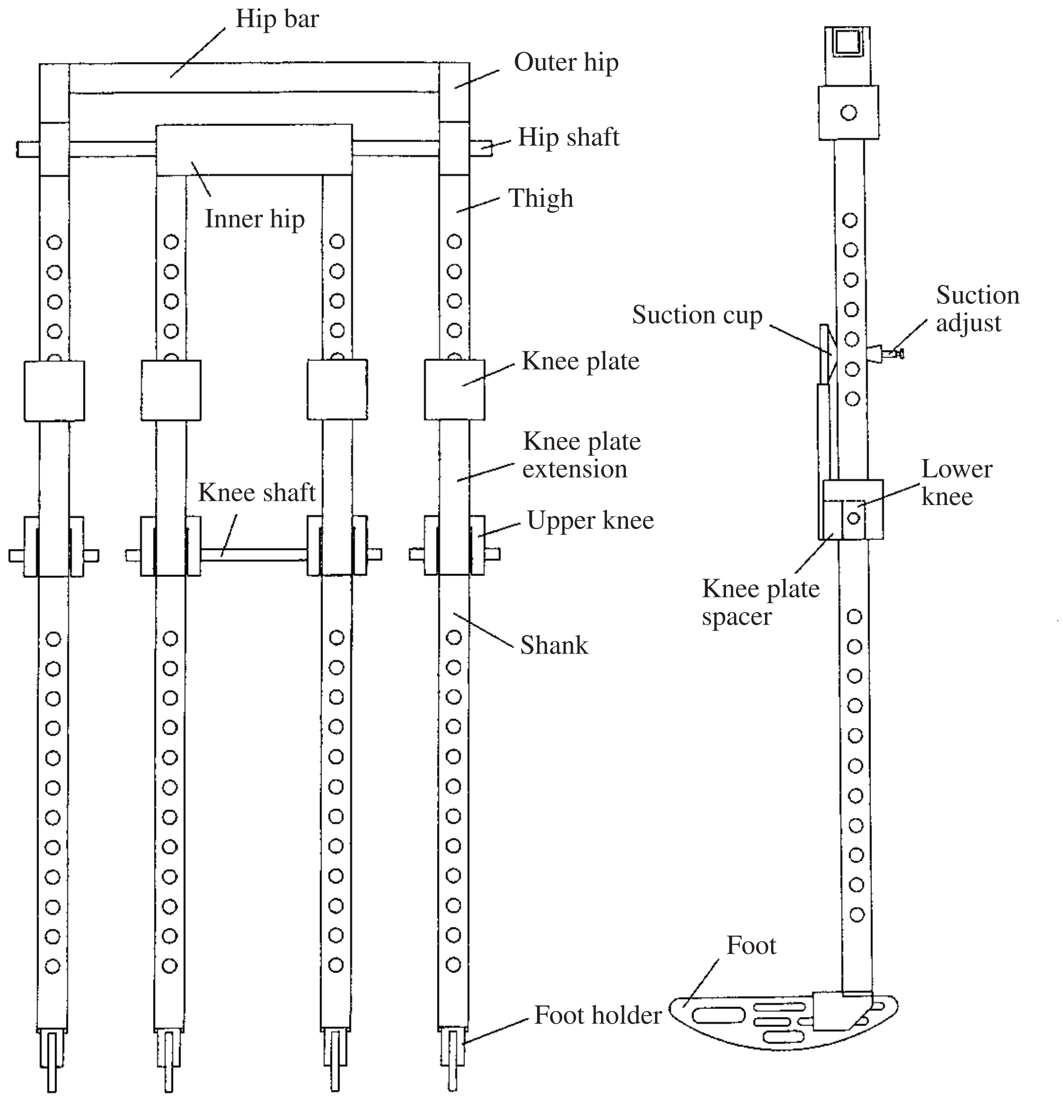

Figure 2.15: A complete schematic of the physical kneed walker with various parts labelled.

Figure 2.15 shows a general overview of the walker, with individial parts labelled. The parts are itemized and described below.

- Hips:

  Outer Hips 1” x 2” x 5.5” Al (2)

  Inner Hip 1” x 3.5” x 7” Al (1)

  Hip Shaft 18” SS tube 0.5” OD (1)

  Hip Bar: 1” x 1” x 14” Al square tube 0.125” wall (1)

  Bearings ABEC 3 single row double-shield 0.5” bore (2)

  SS clamp collars 0.5” bore (2)

  SS shaft spacers 0.5” bore, 0.188” long (2)

  The hips consist of the inner and outer hips, hip shaft, and top hip bar (see Figure 2.16). The inner hip (see Figure 2.17) clamps onto the hip shaft, while the outer hips (see Figure 2.18) house the hip shaft bearings. Both plugs are press-fit into the inner and outer thighs, respectively. The outer hips are bolted and epoxied to the outer thigh connector. The inner hip doubles as a connector; it is machined from a block of Al. The top hip bar connector is the walker’s top-most member; It is epoxied and bolted at each end to an outer hip plug.

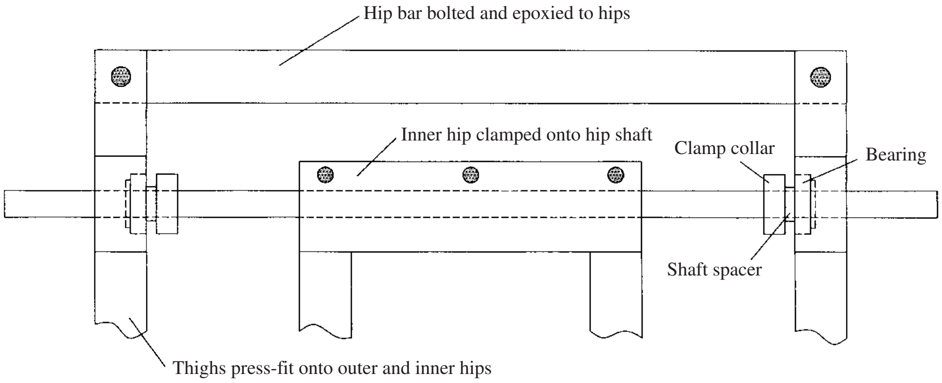

Figure 2.16: A CAD drawing of the inner and outer hip joints.

- Thighs:

  1" x 1" x 12" Al square tube 0.125" wall (4)

  The thighs are square Al tubing with holes drilled to reduce weight and accomodate suction sups for knee stops. The upper ends of the thighs are press-fit into the inner and outer hip plugs. The lower ends are press-fit into the upper knees. See Figure 2.19 for specifications.

- Knees:

  Upper Knees 2" x 2" x 4" Al (4)

  Lower Knees 1" x 1" x 3" Al (4)

  Outer Knee Shafts 3.5" SS tube 0.375" OD (2)

  Inner Knee Shafts 9" SS tube 0.375" OD (1)

  Bearings ABEC 3 flanged single row double shield 0.375" bore (8)

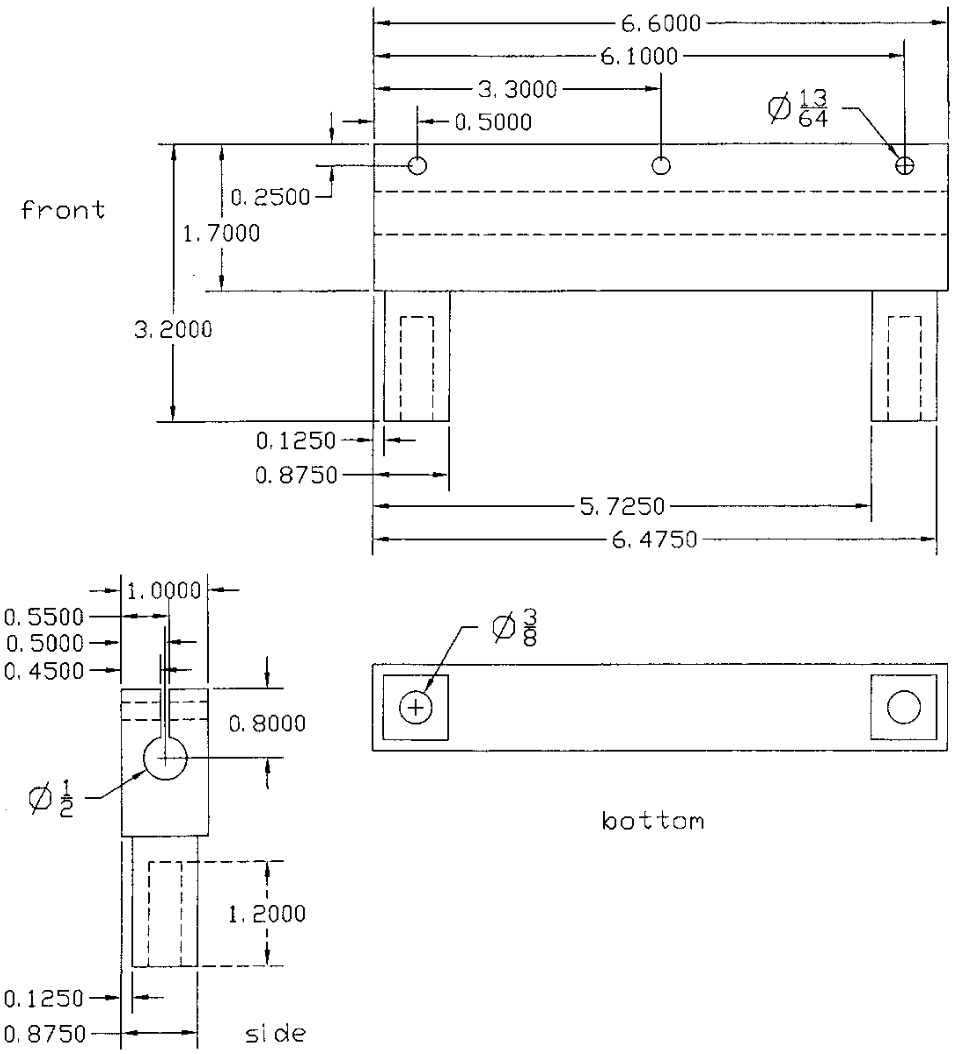

Figure 2.17: CAD details of the inner hip joint. This piece is machined from an aluminum block.

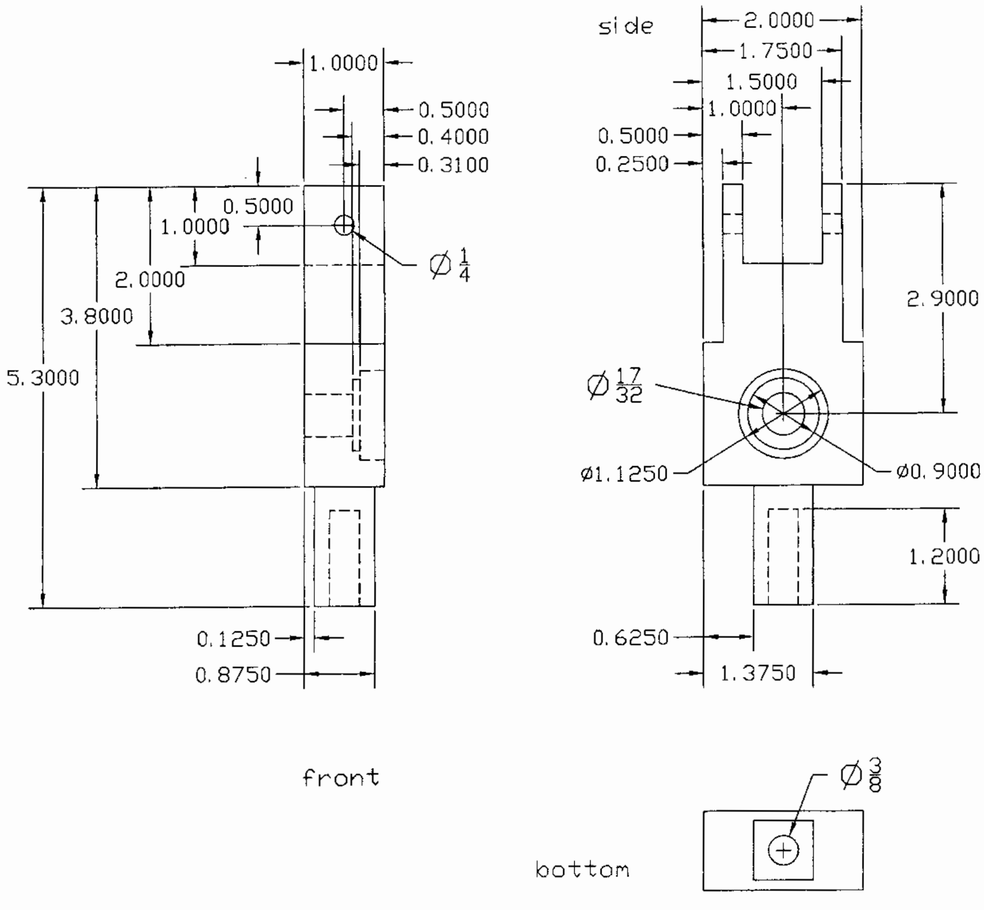

Figure 2.18: CAD details of the outer hips (plugs). These pieces are machined from aluminum block.

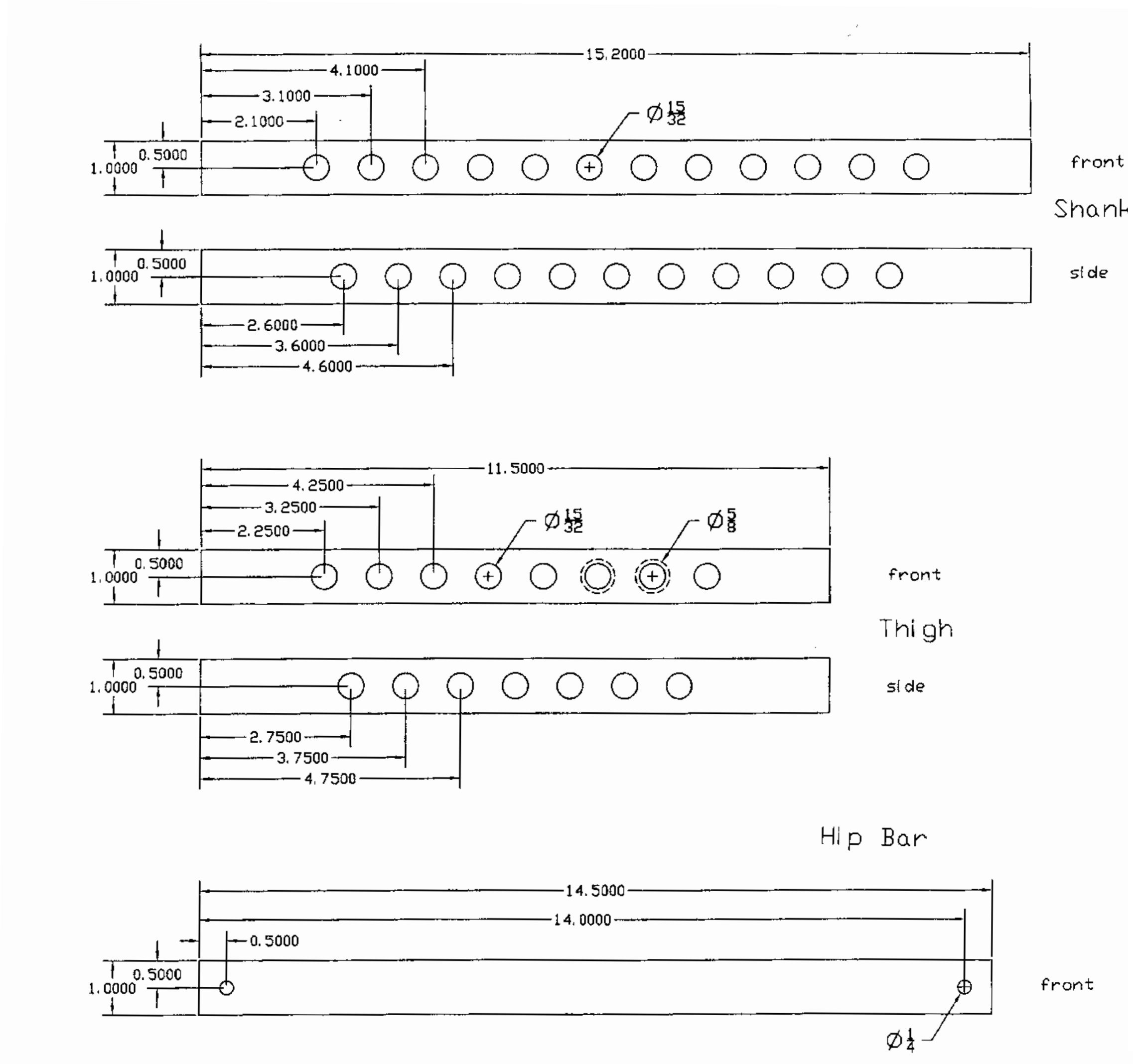

Figure 2.19: CAD details of the shanks, thighs, and top hip bar. All of the parts in this drawing are made from 1 inch square tubing with 0.125 inch wall thickness.

Al clamp collars 0.375” bore (8)

SS shaft spacers 0.375” bore, 0.125” long (8)

Figure 2.20 shows an overview of the knee joints. The upper knees (Figure 2.21) contain the knee bearings and are press-fit into the bottoms of the thighs, while the lower knees (Figure 2.22) clamp onto the knee shafts and are press-fit into the tops of the shanks. The inner knee shaft passes through both inner knees, while each outer knee has its own shaft.

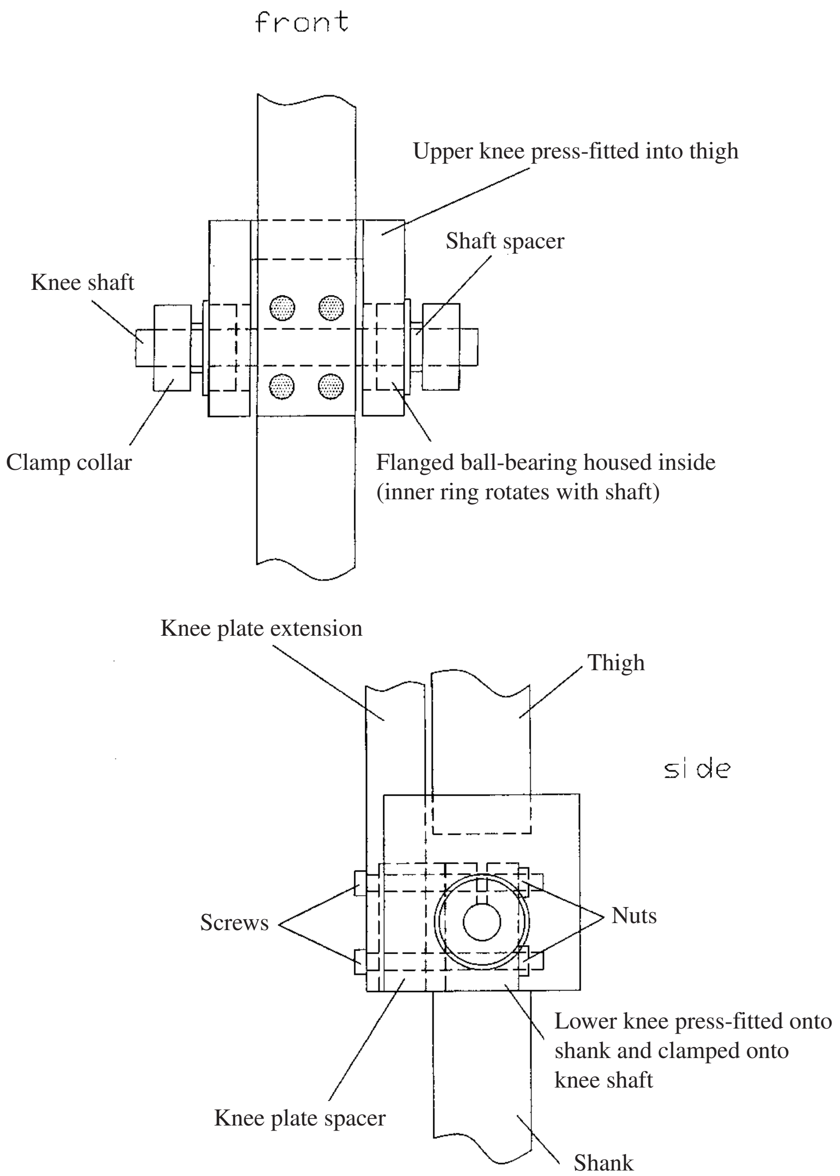

Figure 2.20: CAD overview of the knee joints.

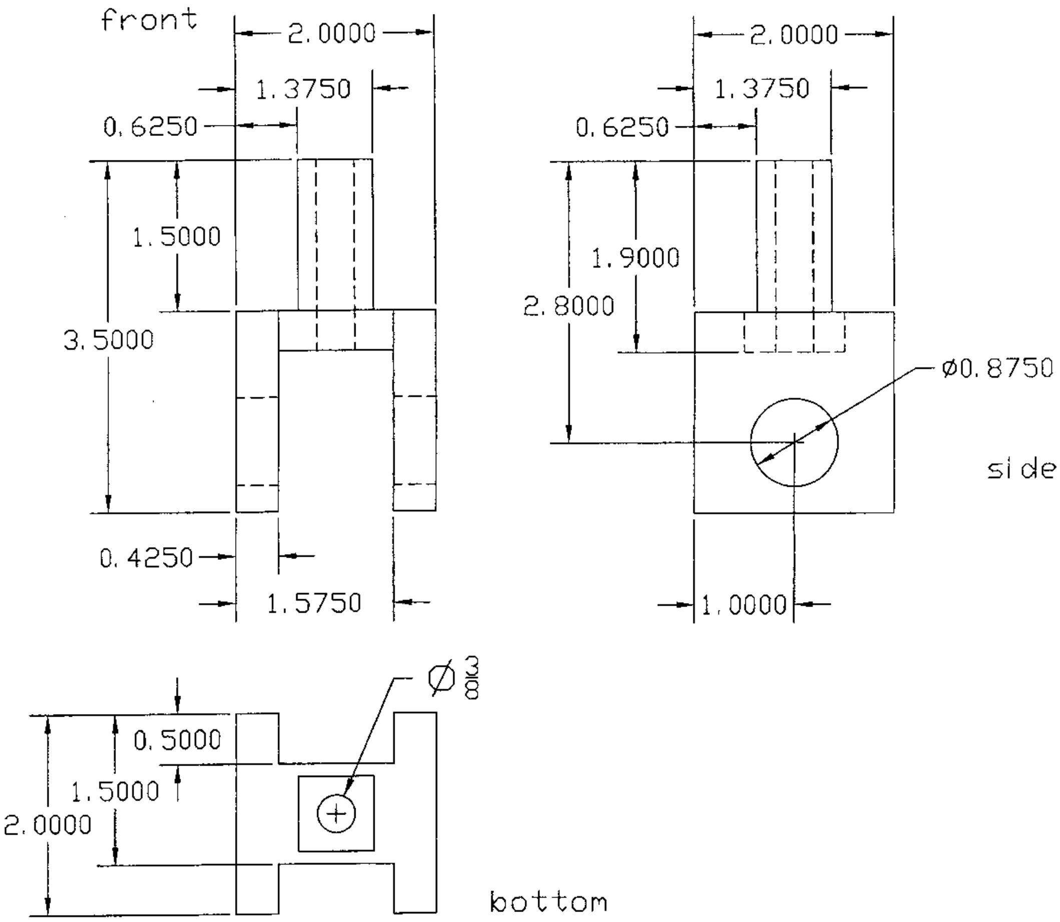

Figure 2.21: CAD details of the upper knees. All parts are machined from aluminum block.

- Knee-Locker:

  Knee-Plate Extensions 1" x 1" x 5.5" Al square tube 0.125" wall (2)

  Knee Plates 0.25" x 2" x 4" Al plate (4)

  Knee Plate Spacers 1" x 1" x 1.5" Delrin blocks (4)

  Suction cups (4)

  Rubber stoppers (4)

  Thin Brass tubing 1" (8)

  Adjustment screws for leaks (4)

  The knee-lockers, or knee-stops are McGeer's mechanism to provide a plastic (no-bounce) collision at kneestrike. The shanks have extensions (described below) which hit suction cups on the thigh. The suction cups have controlled leaks so as not to cause sticking. See Figure 2.23 for a sketch of this mechanism. The suction cups are inserted into the thigh, and pierced by the brass tubing, which runs through the rubber stoppers attatched to the back of the thigh. The back end of the tubing is tapped for the adjustment screws. Several small holes are drilled along the top of the brass tubing in the threaded zone so that by adjusting the screw, the rate of air leakage from the suction cup can be controlled. The knee plate extension (Figure 2.25) is bolted to the lower knee, offset by the knee plate spacer blocks (Figure 2.26). The knee plates (Figure 2.24) are bolted to the top of the knee plate extension.

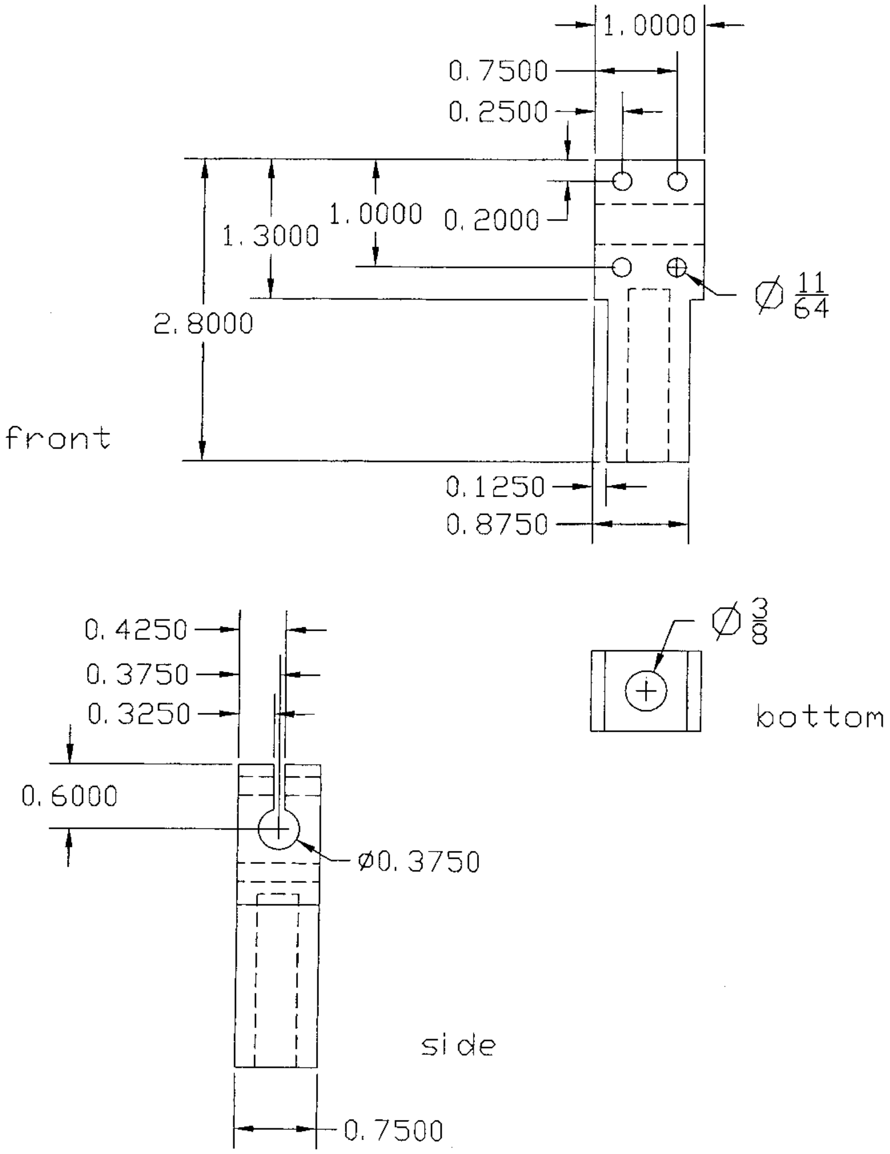

Figure 2.22: CAD details of the lower knees. All parts are machined from aluminum block.

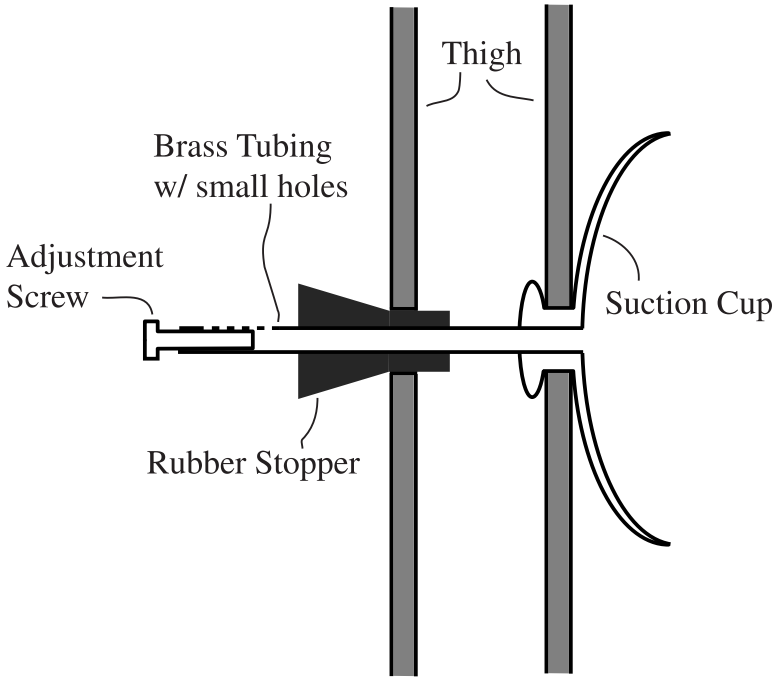

Figure 2.23: Side view of the thigh showing details of the suction-cup mechanism.

- Shanks:

  1” x 1” x 15.5” Al square tube 0.125” wall (4)

  The shanks (Figure 2.19) are square Al tubing with holes drilled to reduce weight. The upper ends of the shanks are press-fit into the lower knees and the lower ends of the shanks are press-fit into the foot holders. The knee-lockers are attatched to the top of each shank.

- Feet:

  Semicircular Feet 0.25” x 2” x 9” Al plate (4)

  Foot Holders 1” x 2” x 3” Al (4)

  The feet (Figure 2.28) are cut from Al plate and material is milled out to provide for adjustment slots and also to save weight. The foot holders (Figure 2.27) are press-fit into the lower ends of the shanks and clamp onto the feet.

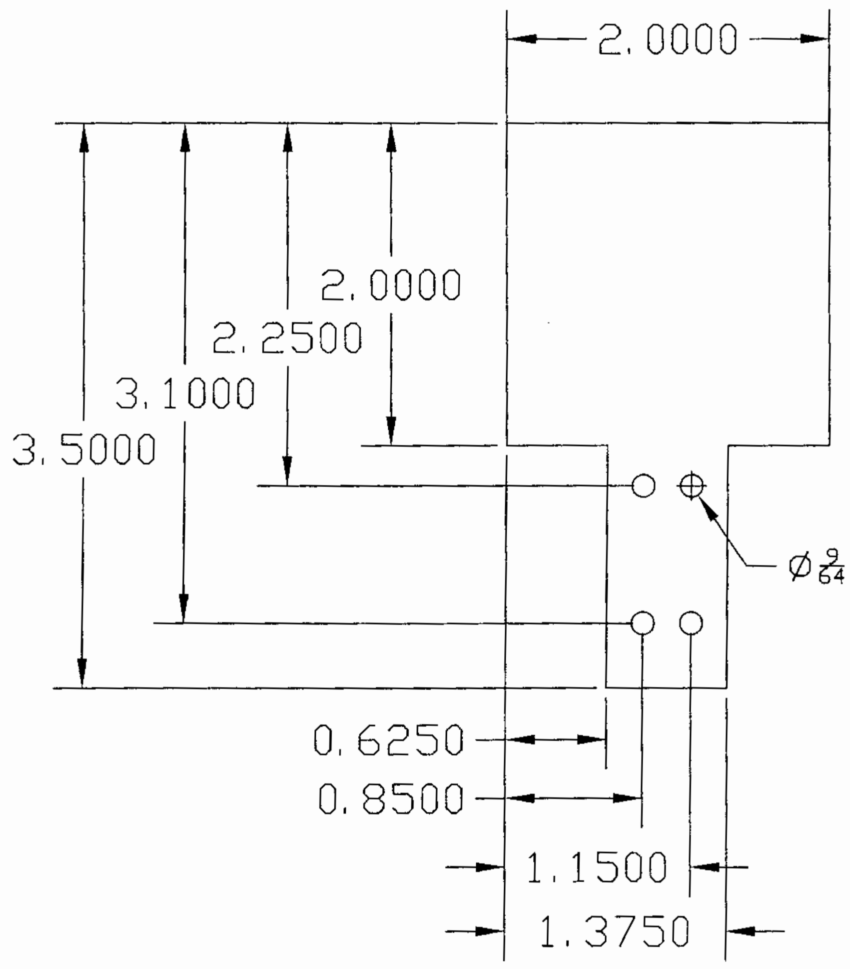

Figure 2.24: Front view of CAD details of the knee plates. They are machined from 0.25 inch thick aluminum plate.

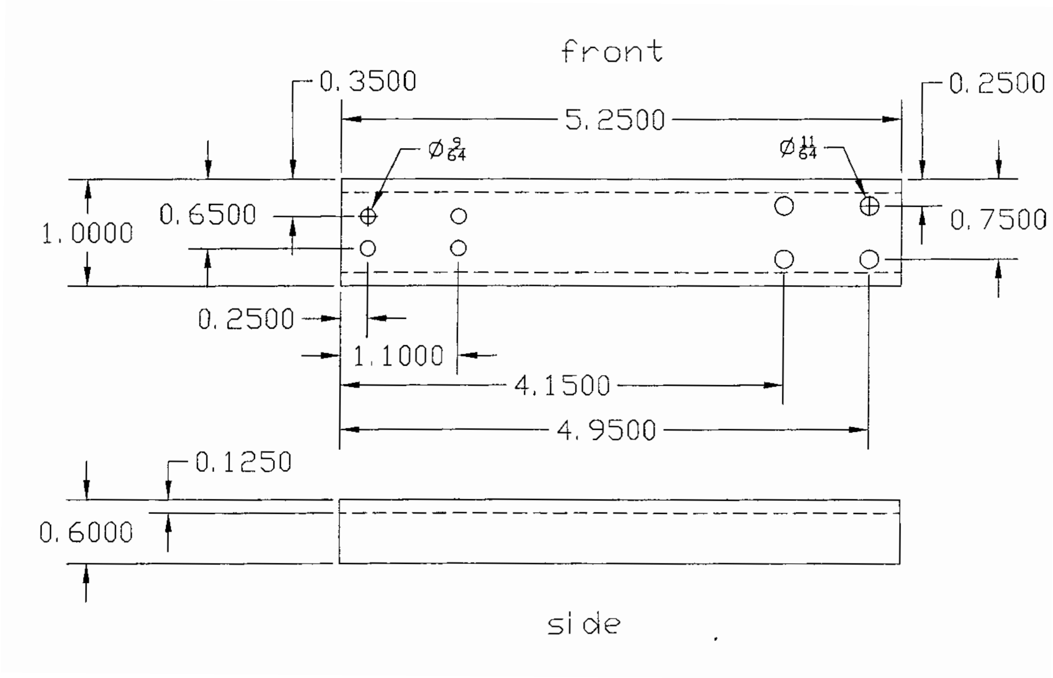

Figure 2.25: CAD details of the knee plate extensions. They are cut from square aluminum tube.

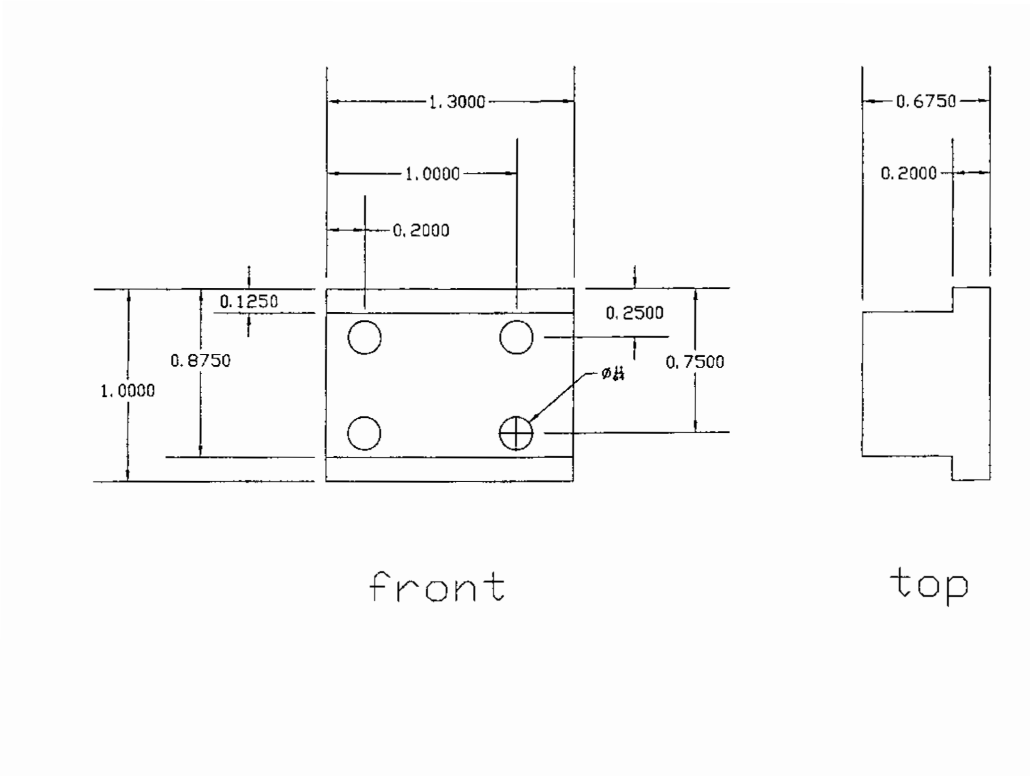

Figure 2.26: CAD details of the knee plate spacers. They are machined from delrin blocks.

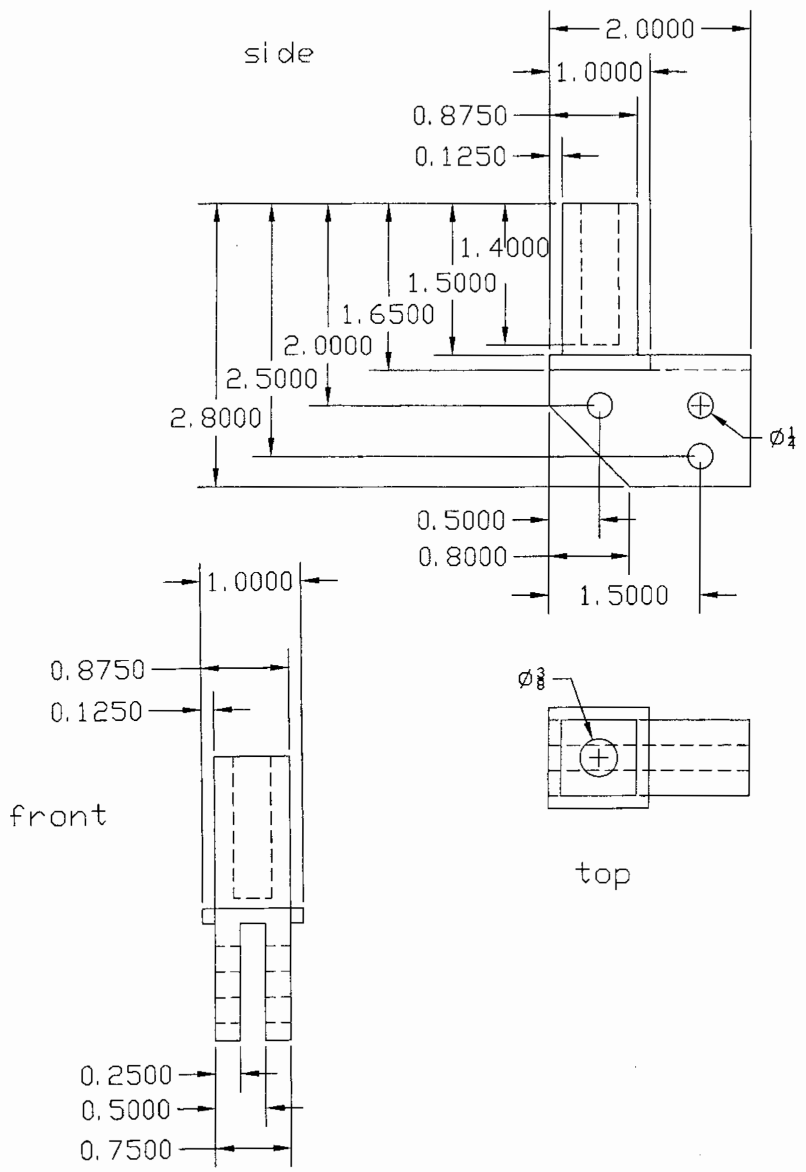

Figure 2.27: CAD details of the foot holders. They are machined from aluminum block.

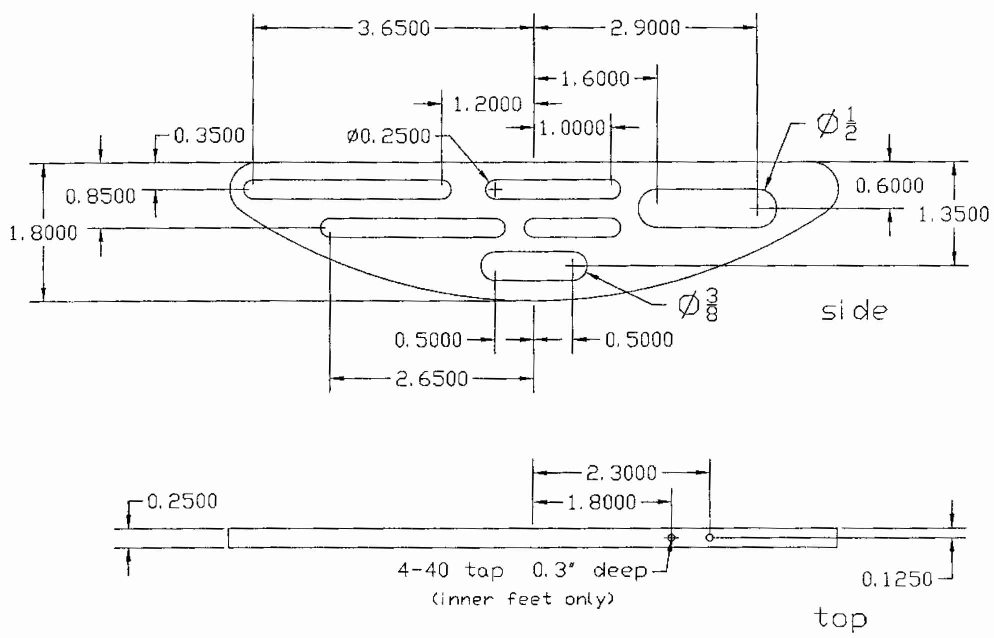

Figure 2.28: CAD details of the feet. They are cut from 0.25 inch thick aluminum plate.

- Screws, Nuts, Washers:

  0.25” x 1” Alloy hex head shoulder screws (36)

  0.25” x 0.625” Alloy hex head shoulder screws (12)

  10-24-1.25” Alloy allen head machine screws (3)

  8-32-1.75” SS allen head machine screws (16)

  6-32-0.5” Alloy allen head machine screws (16)

  10-24 Alloy nuts (51)

  8-32 SS nuts (16)

  6-32 Alloy nuts (16)

  0.25” SS flat washers (48)

  No. 10 SS flat washers (54)

  No. 8 SS flat washers (32)

  No. 6 SS flat washers (32)

  This list represents the approximate amount of hardware that is needed to fasten the above pieces together.

2.14.2 Creating Dynamically Equivalent Legs

When building walking robots with inner and outer legs or leg pairs, one usually cannot make the two legs or leg pairs have identical mass distribution because of practical design considerations. The thigh pairs, for example, are generally connected by cross-members that are at different heights so that they do not collide with each other during the swinging of the legs. In powered models, the positioning of the motors and drivetrain might lead to some asymmetry as well. The last step in model construction is to equalize the parameters of the inner and outer thighs.

This subsection describes one way (there are many different ways) to add masses to two different bodies in 2-D in order to make them dynamically equivalent. Although this technique is straightforward and could be used for any two bodies, attention is restricted to bodies which are of similar (but not identical) mass distribution, because this assures that the extra masses added will be small as compared to the weight of the bodies; thus the process will always be feasible.

Problem Statement and Algorithm

Consider two bodies, body 1 and body 2; each has a mass M, cm location c, and moment of inertia about its cm I. The goal is to add masses so that $M_1^* = M_2^*$, $c_1^* = c_2$, and $I_1^* = I_2^*$, where the "*" superscript denotes a change from the original value.

Equalizing Mass and Center-of-Mass Location

Assume that $M_2 > M_1$. The first step is to add a mass $m_1 = M_2 - M_1$ at a location l_1 to to body 1. After adding m_1, the new cm location of body 1 (denoted by c_1^*) will be at

$$c_1^* = \frac{M_1 c_1 + m_1 l_1}{M_1 + m_1} \tag{2.68}$$

To get $c_1^* = c_2$, set c_1^* equal to c_2 in the previous equation and solve for the distance l_1 at which to put the mass m_1.

$$l_1 = \frac{c_2(M_1 + m_1) - M_1 c_1}{m_1} \tag{2.69}$$

Substituting for m_1 and rewriting in terms of M_1 and M_2 gives

$$l_1 = \frac{M_2 c_2 - M_1 c_1}{M_2 - M_1} \tag{2.70}$$

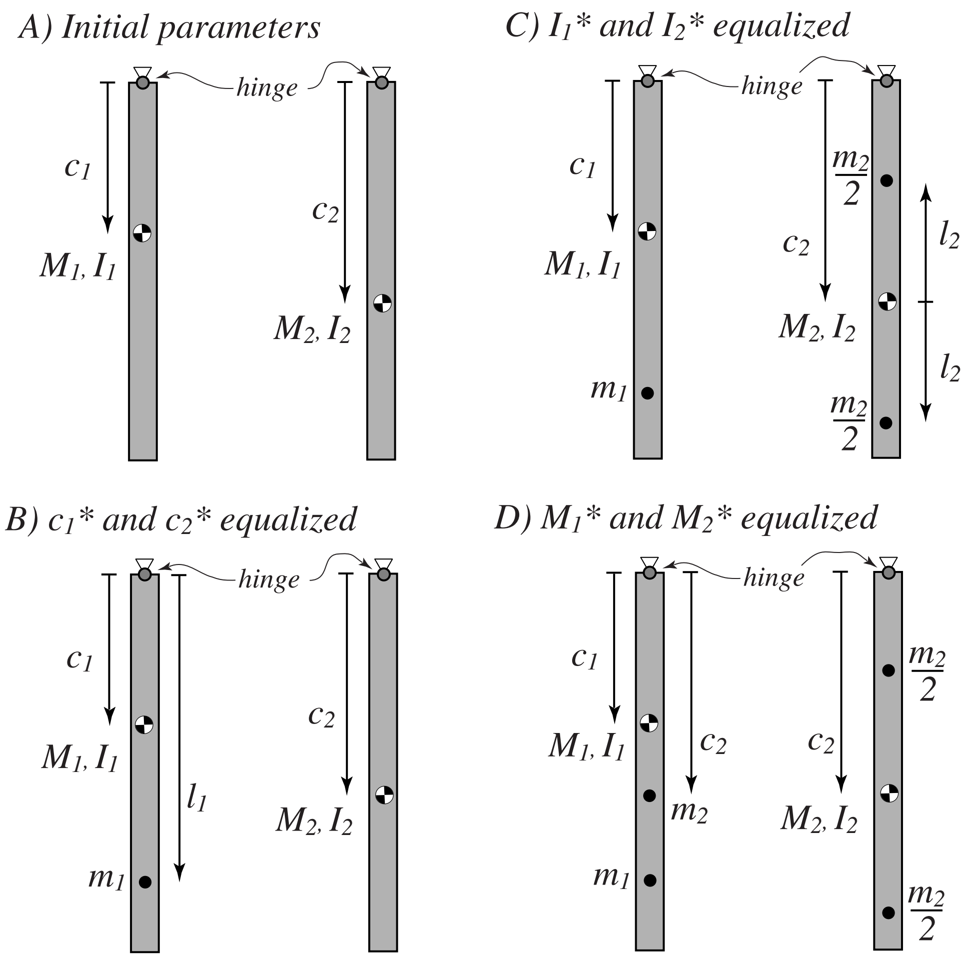

Figure 2.29: A) Two bodies with similar but unequal mass distribution. Each has mass M, cm location c, and moment of inertia about its cm I. B) Mass $m_1 = M_2 - M_1$ is added to body 1 a distance l_2 from the hinge point so that each body has an overall center of mass located at $c_1^* = c_2^* = c_2$. C) Two masses $m_2/2$ added to body 2, each at a distance l_2 from the center of mass of body 2 so that the overall moments of inertia of the bodies are equal, $I_1^* = I_2^*$. D) Mass m_2 is then added to body 1 so that the overall masses of the two bodies are equal, $M_1^* = M_2^*$.

The new moment of inertia for body 1 (denoted by I_1^+) will be

$$I_1^+ = I_1 + M_1(c_1^* - c_1)^2 + m_1(l_1 - c_1^*)^2 \tag{2.71}$$

Equalizing Moments of Inertia

Now body 1 and body 2 have equal masses and cm locations, but unequal moments of inertia. Assume that the new moment of inertia of body 1 (I_1^+) is greater than the moment of inertia of body 2; if this is not the case, body numbers 1 and 2 are arbitrary and can be switched.

The next step is to add two masses (mass $m_2/2$ each) to body 2 symmetrically, each at a distance l_2 from the the center of mass, to increase body 2's moment of inertia while not affecting its cm position.

The distance l_2 is arbitrary. It might be convenient to make l_2 as large as practically possible, so as to minimize the magnitude of m_2. The necessary mass m_2, as a function of l_2, is found from the following formula

$$m_2 = \frac{I_1^+ - I_2}{l_2^2} \tag{2.72}$$

where I_1^+ is given by Equation 2.71 above.

Lastly, add the same total mass m_2 at the cm of body 1 to keep its mass equal to that of body 2.

Summary

By following the above procedure, the inner and outer leg parameters can be made equal (as is assumed in the simulation). Most often, this will be necessary for the inner and outer thighs because of design asymmetries; the inner and outer shanks can usually be constructed identically, and so the procedure is not necessary for the shanks.

The final values for the parameters of the two bodies are as follows.

$$M_1^* = M_2^* = M_2 - M_1 + \frac{I_1^+ - I_2}{l_2^2} \tag{2.73}$$

$$c_1^* = c_2^* = c_2 \tag{2.74}$$

$$I_1^* = I_2^* = I_1 + M_1(c_1^* - c_1)^2 + m_1(l_1 - c_1^*)^2 \tag{2.75}$$

At this point, the walker can be simulated and new parameter adjustments can be attempted.

Aside On Parameter Sets And Virtual Masses

It is well known in robotics and multi-body dynamics that the usual set of parameters (mass, moment of inertia about cm, characteristic length) for a rigid-body (compound) pendulum are actually more than what is necessary to predict the pendulum's behavior. Indeed, a one-parameter family of dynamically equivalent pendula can be constructed which all behave in the same way (identical angles as functions of time with the same initial state).

Some research, such as that of Kawasaki et al. (1996), Khalil et al. (1990), and Lin (1995), focuses on various ways to reduce and describe the parameter space for linked-chain devices such as robots. Most of this work consists of mathematical regrouping of coefficients in the equations of motion. However, some parameter reduction can be achieved intuitively as follows: 1) Nondimensionalization by mass, length, and time can eliminate two structural parameters and g from the model. 2) A *virtual mass* argument can be used to eliminate as many parameters from the model as there are rotational joints, in the following way.

Consider a rigid body pendulum. A point mass added to the body *exactly at* the hinge point will have *no* effect on the motions of the pendulum, and yet the pendulum's parameters will change. In a linked chain, by subtracting (or dis-associating) "virtual mass" from the outer link, and adding (or associating) the same amount to its inner neighbor right at their hinge (beginning with the outermost link), one can set the moment of inertia of every link to 0 without changing the nature of the mechanism. This associates every rigid-body pendulum with an equivalent simple pendulum and eliminates the need for the moment of inertia parameter.

This is one intuitive example of the over-determined nature of rigid-body pendula. Although we do not present such a strategy here, this virtual mass idea can be used to concoct a scheme for making inner and outer legs dynamically equivalent which invloves fewer mass additions than the scheme presented above. In the new scheme, the virtual mass can be "exchanged" at the hip joint.