Chapter 5

Passive Dynamic Walking in 3D

Previously, this work dealt with two-dimensional models that were constrained to a plane. However, since humans and other bipeds move in three dimensions, the obvious extension of the two-dimensional work is to model passive walkers in three dimensions. As it turns out, the modelling and the search for stability become much more challenging in this case.

5.1 The 3D Model

The 3-D model studied here is similar to the model of McGeer (1991), except that a torsional spring and damper at the hip and some torsional steering damping at the contact point of the stance leg are included. The goal is to find model parameters which lead to stability. In particular, the effects of the torsional spring and damper at the hip, which have not been previously investigated, are of interest. McGeer's model is shown in Figure 5.1

Most details on the simulation of the three-dimensional models can be found in Chapter 2, and the actual code is reproduced at the end of this chapter. Some

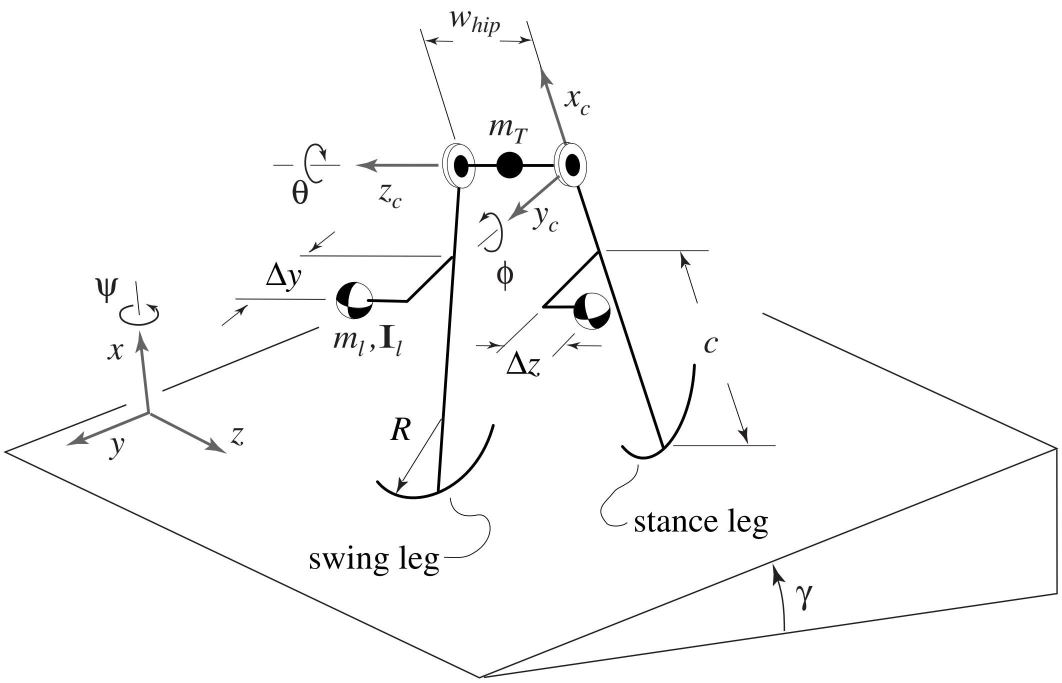

Figure 5.1: McGeer's 3D passive-dynamic walking model: parameters and configuration variables. Like McGeer's 2D straight-legged model, it has two identical straight legs and semi-circular feet. Orientation of the stance leg relative to the ground frame is determined by the heading(ϕ), bank(ψ), and pitch(θ) sequence of rotations about the axes indicated. The swing and stance legs have the same heading and roll angles, but can pitch independently. The leg mass is m_l and has location Δx, Δy, and c (with respect to the stance leg frame) and the leg coordinates are aligned with its principal moments of inertia. The radii of gyration are r_{gyr_x}, r_{gyr_y}, and r_{gyr_z}. The hip has only a point mass, m_T. The radius of curvature of the feet is R. This figure is used with permission from Coleman (1998b).

verification of the three-dimensional simulation code is provided by the following evidence.

- The simulation matches the predicted behavior of a rolling disk when all of the mass is lumped at the center of the stance foot and the swing foot is massless.
- The simulation conserves energy to the appropriate numerical tolerance between foot collisions.
- The simulation results (dynamics *and* stability calculations) match results from another simulation constructed independently by Coleman (1998a).
- The simulation code reproduces (to numerical tolerance) the fixed point and relevant stability characteristics of a two-dimensional model in three dimensions.

5.1.1 Some Comments on Eigenvalues

Other Measures Of Lateral Stability

The simplest measures of stability are the eigenvalues of the linearized Jacobian matrix of the stride function (as in 2-D models); details for calculating the Jacobian are given in Section 3.7.1. As shown by Coleman (1998b), however, this measure can be misleading when considering the lateral stability of the walker. Some parameter changes which superficially appear to benefit stability (since they lower the out-of-plane eigenvalue) only change the time scale of the sideways falling motions and do not affect the inherent tendency of the model to fall sideways.

To address the abovementioned shortcoming, another measure which compares the lateral stability of the walking motion to the stability of the same *frozen* model is also included in the discussions for completeness; see Coleman (1998b) or McGeer (1991) for more details. To briefly summarize this stability measure, the eigenvalue of the walking motion σ, is compared to $e^{p\tau^*}$ where τ^* is the gait cycle period and p is the natural lean frequency of the corresponding frozen planar walker (no hip spacing, all angles except ϕ locked at 0) about the downhill axis (McGeer's y direction).

$$p = \sqrt{\frac{2m_{leg}c + m_T}{2m_{leg}(c^2 + r_{gyr_y}^2) + m_T}} \tag{5.1}$$

For a walker like that of Coleman (1998b), with an out-of-plane center-of-mass position, Equation 5.1 becomes (in the notation in Section2.3)

$$p = \sqrt{\frac{R + pc3(1)}{2I_{yy} + (R + pc3(1))^2 + pc3(3)^2}} \tag{5.2}$$

where I_{yy} is the moment of inertia about the fore-aft axis (McGeer's y_c direction) through the center of mass.

The quantity $\hat{\psi}e^{p\tau^*}$ approximately represents the angle that a planar walker would fall through if it were balanced statically, given an initial bank perturbation $\hat{\psi}$, and allowed to fall for an amount of time equal to one gait cycle period τ^*. We would like to know if the same bank perturbation $\hat{\psi}$, applied at the beginning of a gait cycle, will lead to a higher angular deflection or less of an angular deflection after one step. If the walker falls less during gait than during statically- unstable standing, then the ratio $\sigma/e^{p\tau^*}$ will be less than one, and the walking motions are presumably beneficial to the lateral stability. If the ratio $\sigma/e^{p\tau^*}$ is less than one, then the walking motions are not beneficial to lateral stability. More details on this derivation are given by Coleman (1998b) and McGeer (1991). Note that this is of questionable utility for a three-dimensional model with hip spacing, since adding hip spacing (and foot spacing) produces a non-planar walker. However, it is interesting to observe that hip spacing almost always tends to improve the lateral stability of the planar model. In other words, the stability ratio $\sigma/e^{p\tau^*}$ is almost always less than one in the 3-D simulations presented here.

One drawback to the ratio measure described above is that in the limit as the walker takes very small steps, $\sigma \to 1$, and $\hat{\psi}e^{p\tau^*} \to 1$, since $\tau^* \to 0$. So, the ratio as described above is not as useful in the small-step limit. A third measure of stability which does not have this limitation is the ratio of the characteristic falling frequency p of the walker (as defined above) to the quantity $\ln\sigma/\tau^*$, which is the effective p of the walker during gait. If $p/(\ln\sigma/\tau^*) > 1$, then the walker falls sideways more slowly during walking than when perturbed from standing upright; thus, the walking motions are beneficial to lateral stability. If $p/(\ln\sigma/\tau^*) < 1$, then the walker falls more quickly during walking than when perturbed from standing upright; in this case, the walking motions are not beneficial to lateral stability. So, one can also try to optimize stability by minimizing the ratio $p/(\ln\sigma/\tau^*)$.

In the results below, the former stability ratio is used, since the small-step limit is not approached.

Quasi-Neutral Stability In Heading

For the case of zero hip spacing, one would expect to be able to (and in fact can) reproduce two-dimensional gait cycle motion in three-dimensions; in this case, the motion restricts itself to a plane of progression (i.e. the walker moves in a straight line as viewed from above). We would expect the two-dimensional eigenvalues to remain unchanged, and extra eigenvalues to be introduced which governed lateral (falling out of plane) stability, heading stability, and other three-dimensional effects.

In the case of heading, the corresponding eigenvalue is be exactly 1; the reasoning is as follows. For all known two-dimensional gait cycles at slope γ, there are also gait cycles at slopes of $\gamma + \hat{\gamma}$, where $\hat{\gamma} << \gamma$. For planar motion down a flat ramp in three dimensions, perturbing the heading angle ϕ by $\hat{\phi}$ has the effect of creating a new plane of progression with a slightly different slope (imagine the extreme case of changing the heading from $\phi = 0$, or straight downhill with an effective (original) ramp slope of γ, to $\phi = \pm\pi/2$, which would be sideways across the ramp with an effective slope of 0). The end result is that the walker continues in a planar gait cycle at the new heading with a new effective slope. This a *one-parameter family* of gait solutions in heading.

This reasoning breaks down if the walker has out-of-plane elements in its motion or if it has hip spacing or similar out-of-plane parameters, since all the foot contact points no longer lie along a straight line. However, since the slope is small and the walker is mostly governed by planar dynamics, the effects of this asymmetry will generally be weak, and the result is that one heading eigenvalue is close to one in magnitude.

5.1.2 Parameterization of $\mathbf{I}_{cm}$

Following Coleman (1998b), a parameterization of the moment of inertia matrix (tensor) $\mathbf{I}_{cm}$ is required in order to implement a gradient-search technique, which is described below. Thus will ensure that the components of $\mathbf{I}_{cm}$ will be varied in a physically realizable way. Like Coleman (1998b), $\mathbf{I}_{cm}$ is parameterized with a six-mass "jack" arrangement, which can be rotated in an arbitrary way with respect to the stance leg axes $\hat{\mathbf{x}}_3, \hat{\mathbf{y}}_3, \hat{\mathbf{z}}_3$. The rotation is described by a 1-2-3 Euler angle system with angles $\alpha_x, \alpha_y, \alpha_z$ about the $\hat{\mathbf{x}}_3$, new $\hat{\mathbf{y}}$, and new $\hat{\mathbf{z}}$ axes, respectively (see Figure 5.2).

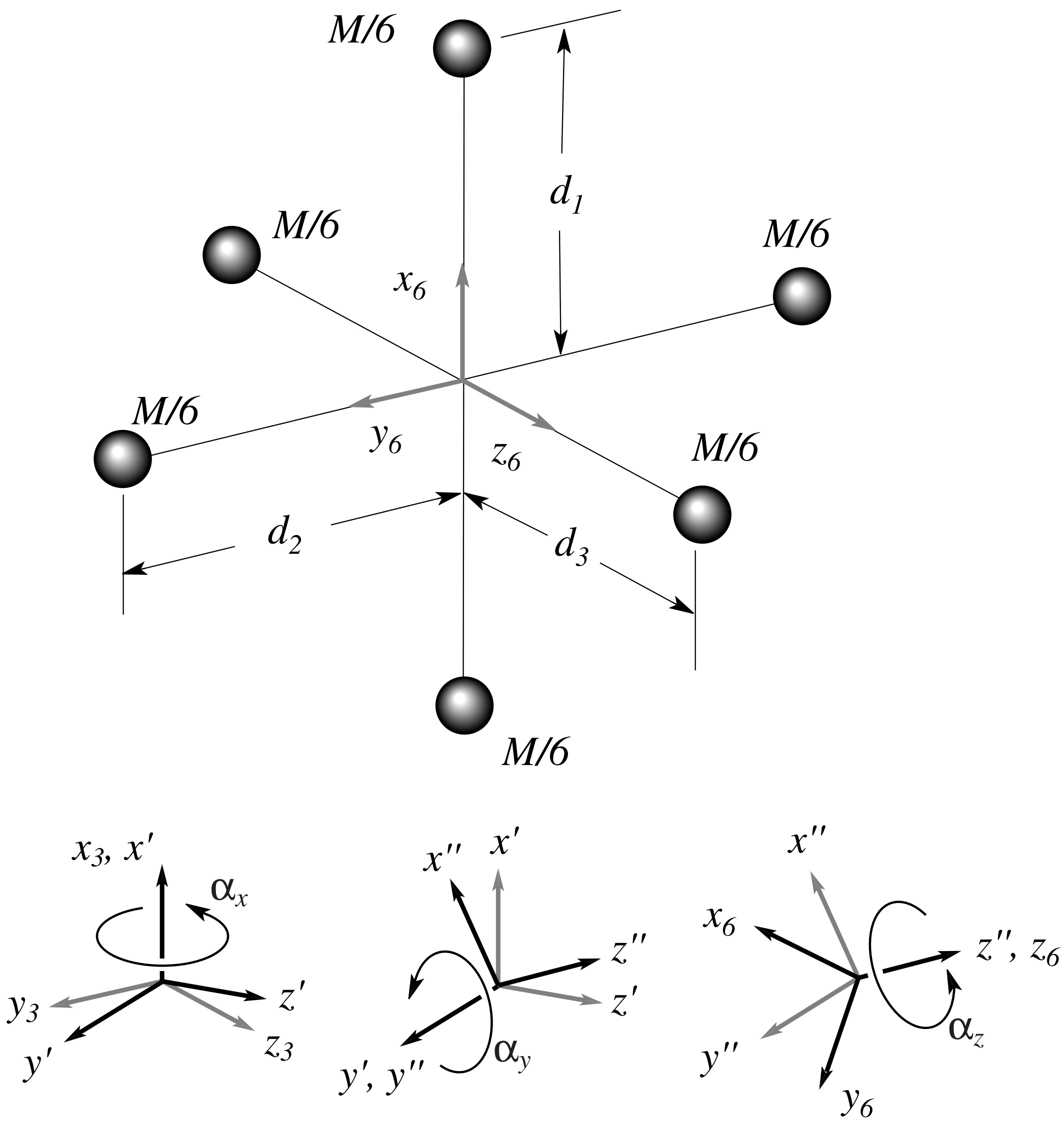

Figure 5.2: Parameterization of the moment of inertia matrix, based on an illustration from Coleman (1998b). The inertia matrix is described by the distances d_1, d_2, d_3 and angles $\alpha_x, \alpha_y, \alpha_z$.

Before the search algorithm is begun, lengths d_1, d_2, d_3 and angles $\alpha_x, \alpha_y, \alpha_z$ are found which parameterize $\mathbf{I}_{cm}$ of the stance leg. Figure 5.2 illustrates this parameterization. If α_x, α_y, and α_z are restricted to be between $-\pi/2$ and $\pi/2$, and none of the d_i are zero, then each moment of inertia matrix $\mathbf{I}_{cm}$ is represented uniquely a set of parameters d_1, d_2, d_3 and $\alpha_x, \alpha_y, \alpha_z$.

The steps for finding d_1, d_2, d_3 and $\alpha_x, \alpha_y, \alpha_z$ are as follows.

1. Diagonalize $\mathbf{I}_{cm}$ into $\mathbf{W}$ and ${}^3_6\mathbf{R}$ where $\mathbf{I}_{cm} = {}^3_6\mathbf{R}\ \mathbf{W}\ {}^3_6\mathbf{R}^T$. The eigenvalues of $\mathbf{I}_{cm}$ are the diagonal elements of $\mathbf{W}$, which can be sorted from lowest to highest: call them I_1, I_2, I_3. The notation is abused here a little by the use of the symbol ${}^3_6\mathbf{R}$. Frame 6 is fixed to the jack; this implies that frame 4 is an intermediate frame between frames 3 and 6, which is not the case; frame 4 is fixed to the swing leg.

2. The values of d can be found as follows (from Coleman (1998b)):

$$\begin{aligned} d_1 &= \sqrt{\frac{3}{2}(-I_1 + I_2 + I_3)}, \\ d_2 &= \sqrt{\frac{3}{2}(I_1 - I_2 + I_3)}, \text{ and} \\ d_3 &= \sqrt{\frac{3}{2}(I_1 + I_2 - I_3)}. \end{aligned} \tag{5.3}$$

   Since the Is are constrained by mechanics to follow a triangle inequality, the d values will always be positive.

3. ${}^3_6\mathbf{R}$ will be of the form

$$\begin{bmatrix} c\alpha_y c\alpha_z & -c\alpha_y s\alpha_z & s\alpha_y \\ c\alpha_x s\alpha_z + s\alpha_x s\alpha_y c\alpha_z & c\alpha_x c\alpha_z - s\alpha_x s\alpha_y s\alpha_z & -s\alpha_x c\alpha_y \\ s\alpha_x s\alpha_z - c\alpha_x s\alpha_y c\alpha_z & s\alpha_x c\alpha_z + c\alpha_x s\alpha_y s\alpha_z & c\alpha_x c\alpha_y \end{bmatrix} \tag{5.4}$$

where $s(x)$ and $c(x)$ denote the sine and cosine of x, respectively. To find $\alpha_x, \alpha_y, \alpha_z$ from Equation 5.4, try the following:

$$\alpha_y = \arcsin {}^3_6\mathbf{R}(1,3) \tag{5.5}$$

$$\alpha_z = \arcsin {}^3_6\mathbf{R}(1,1)/\cos\alpha_y \tag{5.6}$$

$$\alpha_x = \arcsin {}^3_6\mathbf{R}(3,3)/\cos\alpha_y \tag{5.7}$$

Changing the sign of a column of ${}^3_6\mathbf{R}$ does not affect the product ${}^3_6\mathbf{R}\,\mathbf{W}\,{}^3_6\mathbf{R}^T$, so one may need to try different combinations of positive and negative columns to find $\alpha_x, \alpha_y, \alpha_z$. If $\alpha_x, \alpha_y, and \alpha_z$ are all between $-\pi/2$ and $\pi/2$, then this is a unique set of parameters for $\mathbf{I}_{cm}$.

5.2 Numerical Results

Three different numerical searches are presented. The first set of searches are simple one-parameter searches, beginning with the parameters reported by McGeer (1991). The second search involves a gradient-based search method where only the moment of inertia parameters are varied, beginning with parameters like those of McGeer (1991), except that a torional hip spring is added (based on results from the previous search). This gradient search was successful in the sense that the technique worked, but unsucessful in the sense that only a local minimum was found rather than a stable gait. The third set of searches are gradient based with multiple parameters being varied; they were unsucessful in that they lead walkers into parameter regions where no gaits exist.

In two-dimensions, one-parameter searches were relatively successful; thus, this same initial strategy was adopted in 3-D. As a first attempt to discover stability, one parameter is varied until one of the following things occurs:

- The maximum eigenvalue begins to increase.
- An eigenvalue that was previously less than one becomes significantly greater than one in magnitude.
- The gait solution disappears.
- The walker parameters become significantly less anthropomorphic or unappealing for aesthetic reasons.

5.2.1 One-Parameter Searches Near McGeer' Parameters

The results described here are similar to those of Kuo (1998), who did similar independent work about the same time as this work. We first reproduce the solution reported by McGeer (1991) and use this as a starting point for various one-parameter searches. One of McGeer's passive gaits (shown in McGeer (1991)) and the parameters that particular walker are shown in Figure 5.3.

As McGeer observed, the amount of steering motion (yaw) in this model is larger than would be expected in a human biped; perhaps a model with more degrees of freedom and/or counter-swinging arms might mimic human motions more accurately.

A gait cycle for the version of McGeer's model presented here is shown in Figure 5.4. This gait is the starting point for some of the parameter-varying numerical experiments described below.

Tables 5.1 and 5.2 compare eigenvalues and eigenvectors for this walker, as calculated by McGeer (1991) and myself, respectively. The dominant eigenvalue is the one which governs the lateral stability or "out-of-plane" falling motion. The goal of the searches here is to minimize this eigenvalue. Curiously, these eigenvalue

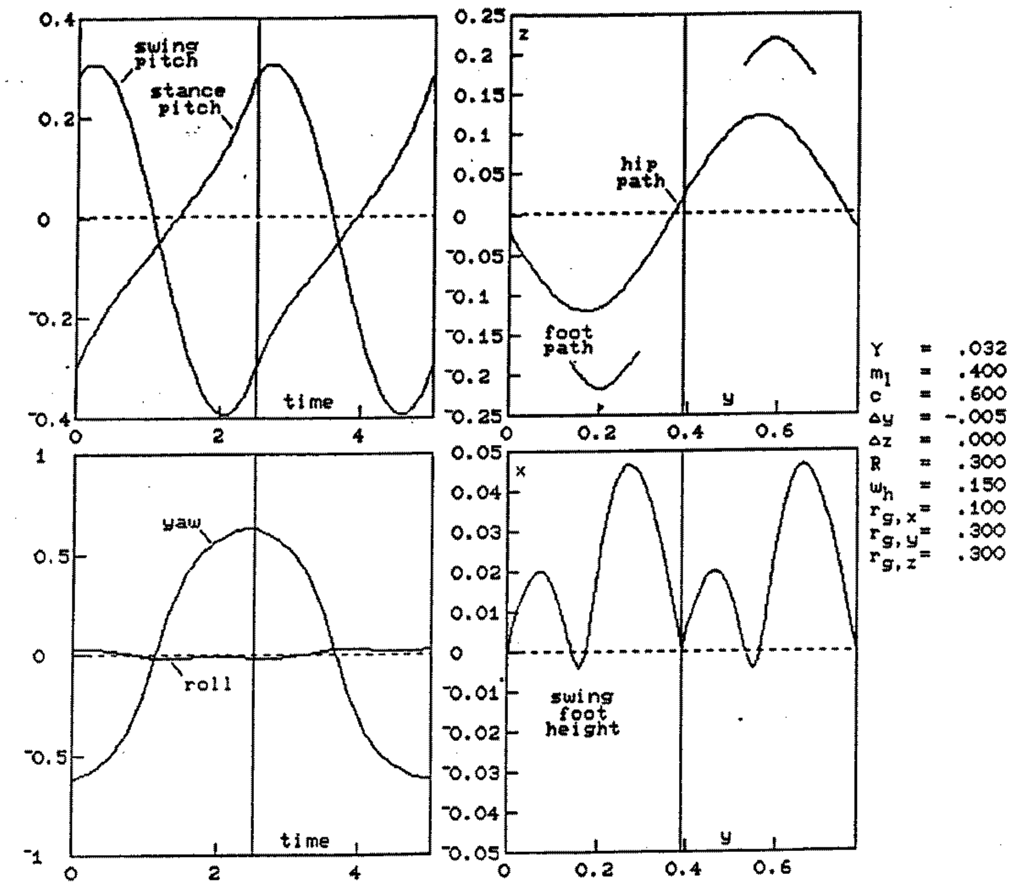

Figure 5.3: Three-dimensional passive cycle, as calculated for a straight-legged biped having legs separated by 15% of leg length. The slope is 0.032; Other parameters are shown on the right side of the plots. (Used with permission from McGeer (1991); also in Coleman (1998b))

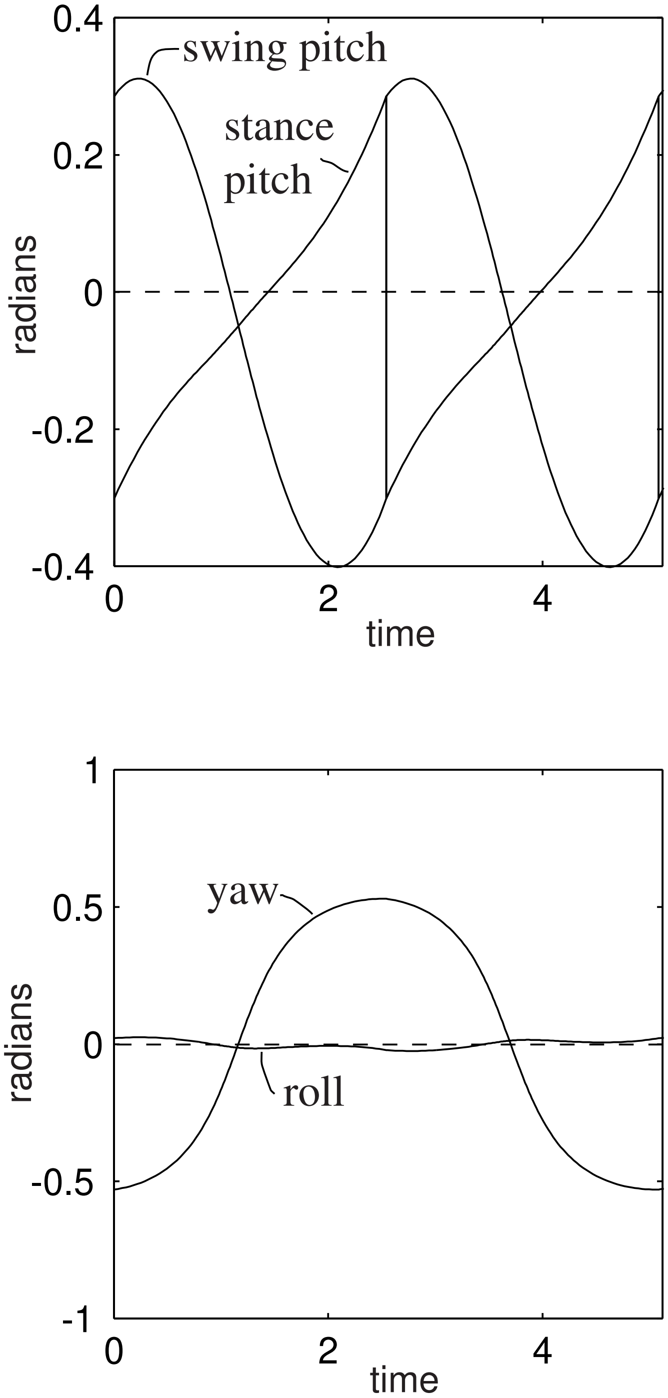

Figure 5.4: Leg angles as functions of time for our version of McGeer's walker. The angles plotted are meant to match those in figure 5.3. Parameters are the same as those of McGeer's walker.

results do not completely match those of McGeer (1991) (since the state variables used are different, the eigenvectors do not match), but *do* match those found independently by Coleman (1998a). Some of this discrepancy may be attributed to differences in numerical tolerance (in this work, numerical integration tolerances of $1e-8$ or $1e-10$ are used throughout, while McGeer (1993a) uses fixed time steps of 0.05, giving numerical tolerances of about $1e-4$ or $1e-5$ with his Runge-Kutta 2nd/3rd order method). Although the largest eigenvalues agree to within about 10 %, McGeer still has two other eigenvalues which are significantly greater than one in magnitude, while here there are no others (ignoring the steering instability). This discrepancy calls for further investigation.

The method used here to calculate eigenvalues involves perturbing all eight state variables (allowing the configuration to stray off the section), and then expecting to see an extra zero eigenvalue. In reality, of course, this (expected zero) eigenvalue is not quite zero; this gives an estimate of the numerical accuracy of our eigenvalue computation. Typically, this eigenvalue is zero to four or five decimal places when numerical tolerances of $1e-8$ or $1e-10$ are used.

McGeer (1991) varied hip spacing and ramp slope simultaneously while keeping the stance angle fixed at 0.3. A plot of his results is shown in Figure 5.5. Unlike McGeer, in the searches presented here, one parameter is varied at a time, regardless of stance angle, and the ramp slope is fixed. Turning points (points of infinite slope on ramp slope vs. parameter plots) are dealt with separately.

Figure 5.6 shows how the eigenvalue moduli and stability ratio vary as the slope, hip spacing, and foot radius are varied. The slope does not appear to have a strong effect on the eigenvalue.

We vary hip spacing from the beginning value of 0.15 to 0.33 (nondimensional

Table 5.1: Some eigenvalues and corresponding eigenvectors from McGeer (1991). This data is for the gait of Figure 5.3. θ_C, ϕ, and ψ are the stance pitch, lean, and yaw (steering) angles respectively (ϕ and ψ have their order *and* meaning switched as compared with our variables in Table 5.2). Although it is unclear in the text, ω_x, ω_y, and Ω_F appear to be the angular rates of the walker about Mcgeer's global x, y, and z axes, while Ω_C appears to be the angular rate of the swing leg relative to the the stance leg (about the hip axis z_c). These eigenvalues do not fully agree with our calculations.

McGeer e-value	-10.34	4.341	2.224	-1.066
θ_C	-0.319	0.547	0.471	-0.009
ϕ	0.592	0.166	-0.056	0.032
ψ	-0.074	-0.688	-0.364	-0.999
ω_x	0.380	0.145	-0.176	-0.010
ω_y	0.608	0.223	-0.179	0.002
Ω_C	0.061	0.136	0.675	0.008
Ω_F	-0.157	0.333	0.351	0.010

Table 5.2: Eigenvalues and corresponding eigenvectors for our simulation of the walker of McGeer (1991). The eigenvalues do not completely agree with McGeer's values (see above text), but are in agreement with independent simulations by Coleman (1998a). The eigenvalue near one indicates quasi-neutral stability in heading. The eigenvalue of zero is expected because the Poincaré section reduces the dimension of the phase space by one at the section.

our e-value	11.145	1.088	0.642	-0.259±0.428i	0.064±0.067i	0.000
ϕ	-0.205	0.996	0.994	0.777∓0.164i	-0.391∓0.648i	0.0638
ψ	0.550	-0.020	-0.020	0.038±0.017i	-0.011±0.116i	0.0357
θ_{st}	-0.329	-0.022	0.024	-0.196±0.146i	0.110±0.052i	-0.396
θ_{sw}	0.257	0.060	-0.035	0.374∓0.313i	-0.217∓0.195i	0.138
$\dot{\phi}$	-0.316	0.023	-0.090	-0.118±0.085i	0.239±0.075i	0.350
$\dot{\psi}$	0.599	-0.007	0.023	-0.074±0.052i	-0.054∓0.125i	-0.176
$\dot{\theta}_{st}$	-0.153	0.045	-0.017	0.157∓0.052i	-0.056∓0.021i	0.3977
$\dot{\theta}_{sw}$	-0.025	-0.036	0.025	0.036±0.112i	0.473±0.077i	0.7119

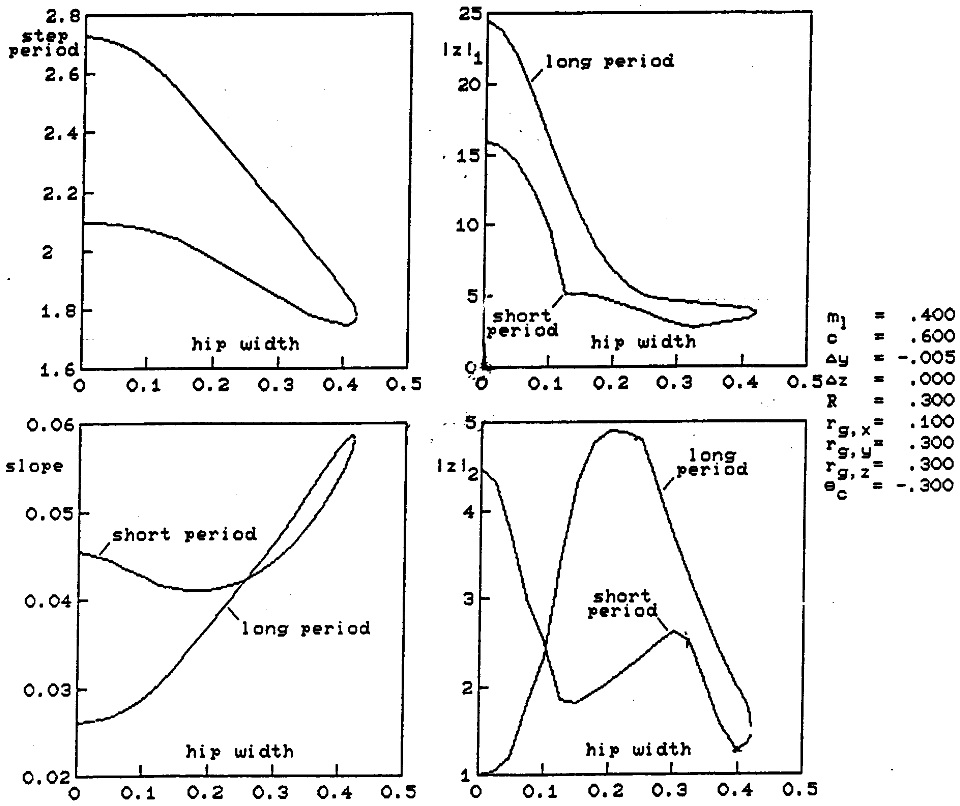

Figure 5.5: McGeer's data: step period, slope, and eigenvalue moduli at each fixed point, shown as a function of hip spacing for his walker parameters. $|z|_1$ and $|z|_2$ are the largest and second-largest eigenvalue moduli, respectively. Along the curves, the stance angle is constant. (Reprinted with permission from McGeer (1991). Also reproduced by Coleman (1998b)).

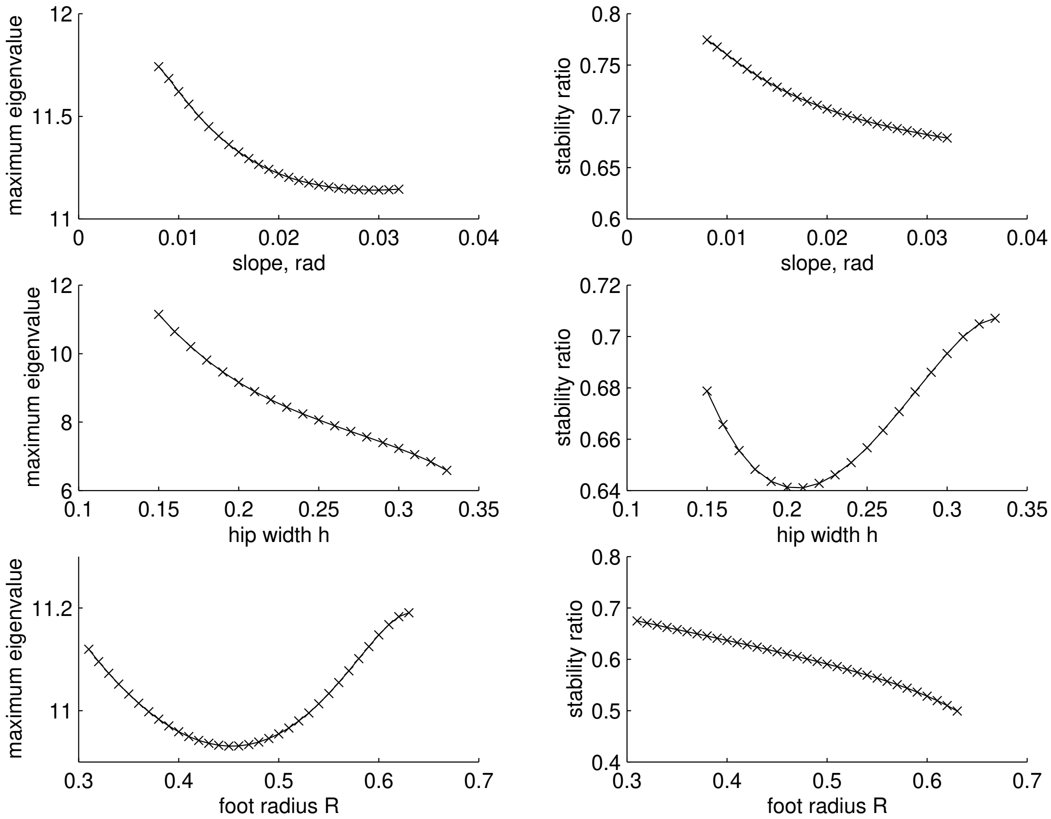

Figure 5.6: Eigenvalue modulus and stability ratio at each fixed point shown as a function of slope γ, hip spacing h, and foot radius R for McGeer's walker. In each plot, except for the parameter varied, the parameter values are the same as those in Figure 5.3.

units of leg length). As hip spacing increases, the maximum eigenvalue drops somewhat until the solutions cease to exist. Figure 5.6 also shows maximum eigenvalue modulus and stability ratio as functions of foot radius, with the center of mass position and total leg length held constant.

Figure 5.7 shows maximum eigenvalue modulus and stability ratio as a function of the leg center-of-mass position. In the x case, the maximum eigenvalue decreases until solutions disappear. In the z case, the maximum eigenvalue asymptotes to one from above due to a balance-bar effect as shown by Coleman (1998b)). In the y case, there is a transition to a short-period gait and other eigenvalues which were previously less than one in magnitude grow larger than one.

We postpone calculations involving the moment of inertia matrix until later in the chapter.

A torsional hip spring and damper and a steering damper are our new additions to the model. The hip damper is proportional to the relative angular velocity between the stance and swing legs, while the steering damper is proportional to the heading rate. Figure 5.8 shows maximum eigenvalue modulus and stability ratio as functions of the hip spring stiffness and torsional and steering damper coefficients. As might be expected, the spring reduces the step period and so improves lateral stability by the eigenvalue measure, but not much by the stability ratio measure. At the end of all three curves, the solutions disappear.

5.2.2 A Gradient Search Algorithm

Since stable walking appears to be more elusive in 3-D than in 2-D, a more automated method of searching for stable gaits is developed: namely, a *gradient search* algorithm. This idea has never before been used to find stable gaits in passive

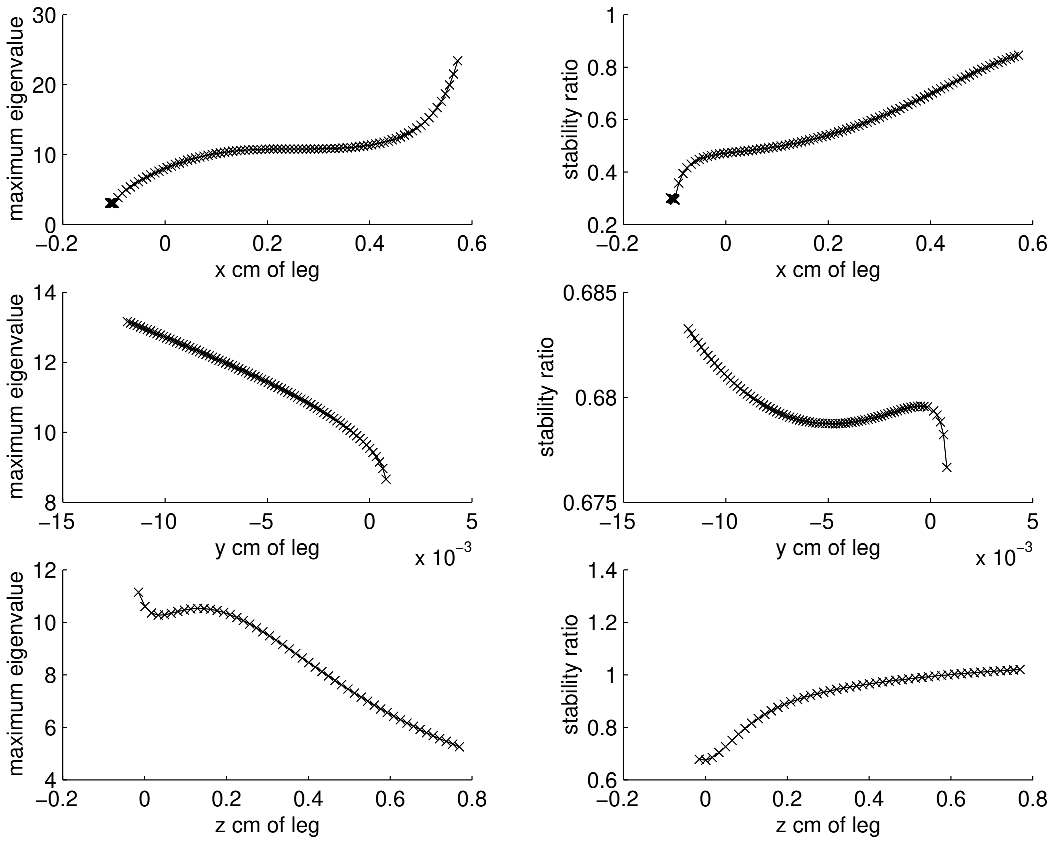

Figure 5.7: Eigenvalue modulus and stability ratio, at each fixed point shown as a function of x-position of the center of mass, y-position of the center of mass, and z-position of the center of mass of the leg. In each plot, except for the parameter varied, the parameter values are the same as those shown in Figure 5.3.

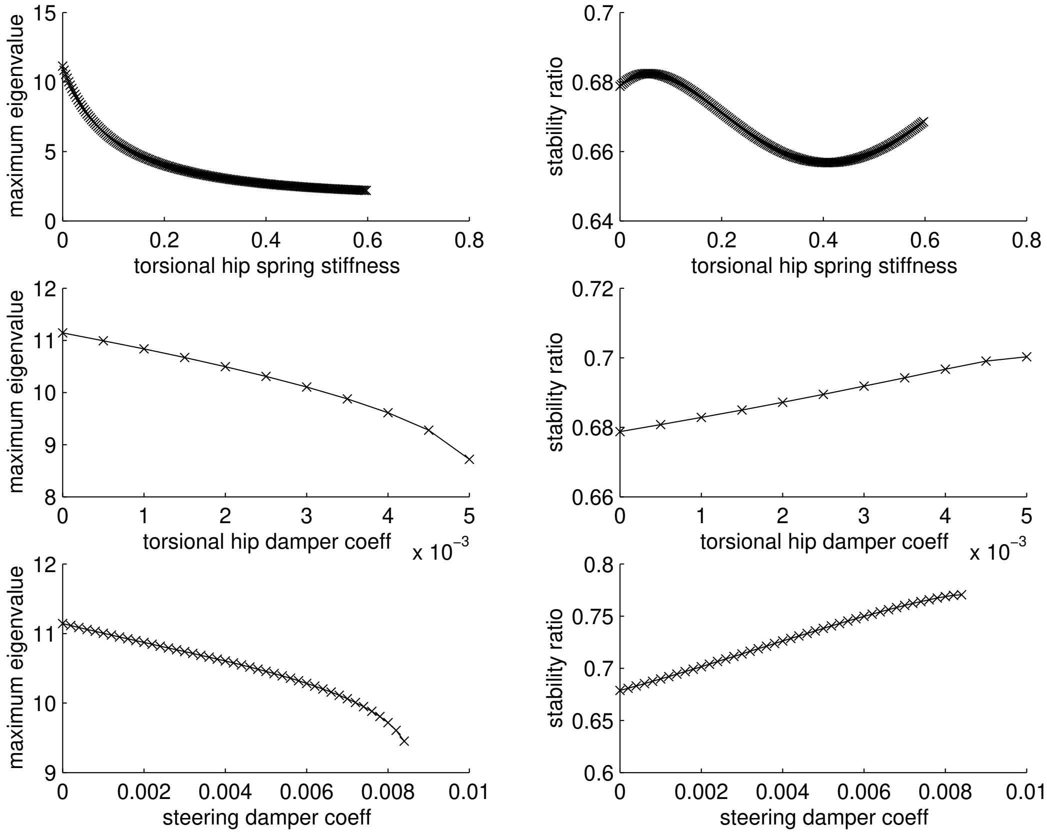

Figure 5.8: Eigenvalue modulus and stability ratio at each fixed point shown as a function of the torsional hip spring stiffness, and hip and steering damper coefficients. In each plot, except for the parameter varied, the parameter values are the same as those shown in Figure 5.3.

models. The idea behind the gradient search algorithm is to follow a path in parameter space whose direction is along the (in our case, negative of the) gradient of the objective function (in our case, maximum eigenvalue modulus) with respect to parameters in order to minimize the objective function. This technique is also called the *method of steepest descent*; it gets its name from the oft-quoted analogy of descending a hill where the height of the hill h is a function of location on the surface (see Marsden and Tromba (1976), for example).

Some Details Of The Algorithm

In general, gradient-search algorithms are well known. However, several complications arise when adapting a generic gradient-search to the task of minimizing the maximum eigenvalue for a straight-legged walker in 3-D.

The most obvious complication is that not all parameter combinations will result in a gait. If the solution path is nearing a region in parameter space where solutions are about to terminate, the condition number of the difference function Jacobian $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ gets very large and/or the lowest non-zero eigenvalue begins to approach zero. To avoid losing solutions, the condition number of $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ should be kept below some ceiling. At times, this may provide an extra constraint. A condition number which is too large indicates that the walker is close to a parameter region where solutions are about to be lost. If the condition number becomes too large, then parameter adjustments which cause the condition number to increase are not allowed. Instead, the parameter adjustment is calculated by

$$\Delta \mathbf{P}_{used} = \Delta \mathbf{P} - \hat{\lambda}(\hat{\lambda} \cdot \Delta \mathbf{P}) \tag{5.8}$$

where $\Delta \mathbf{P}$ is the parameter adjustment suggested by the steepest descent and $\hat{\lambda}$ is a unit vector in the direction of the gradient of the condition number (this gradient must be calculated). Equation 5.8 is illustrated in Figure 5.9.

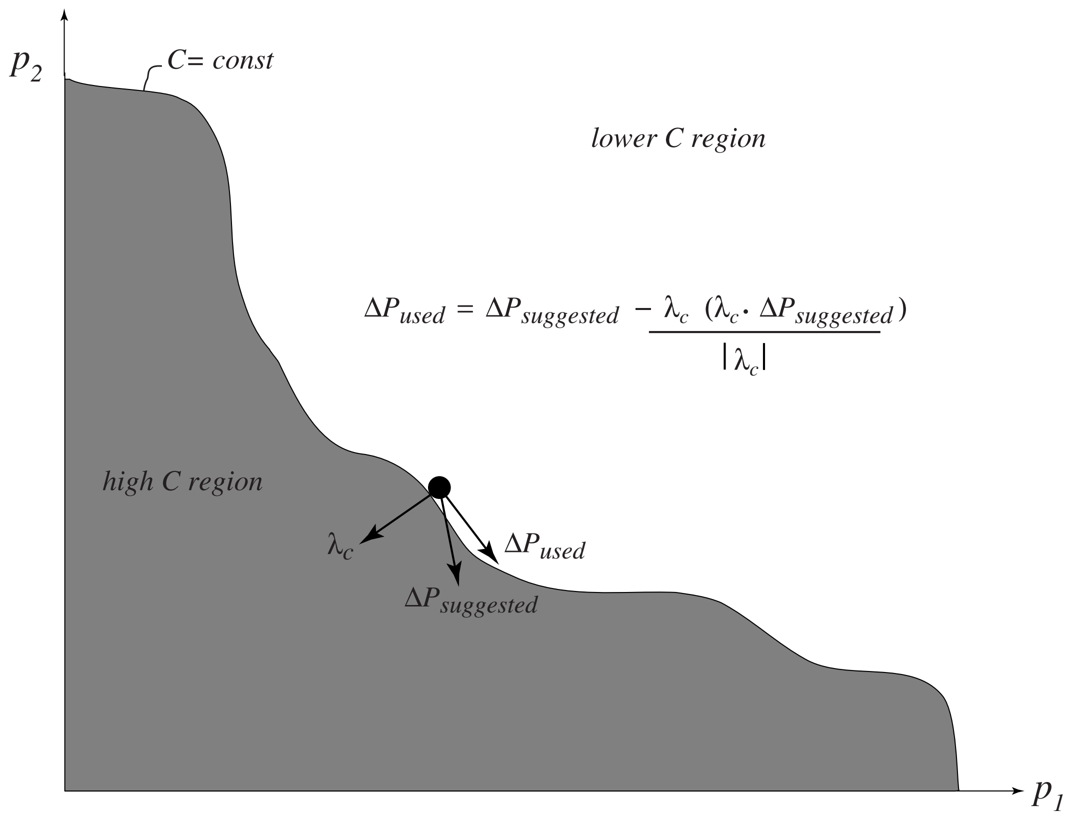

Figure 5.9: Graphical description of constraining the search algorithm to avoid areas of high condition number.

Another complication is the possibility that the algorithm will call for parameter adjustments which wander around near a local objective function minimum but do not get close enough to the minimum. This is one drawback to this approach, namely that there are no good step size adjustment criteria. To avoid this somewhat, the following damping criteria is applied to the parameter adjustment for each step in parameter space as follows:

- If the maximum eigenvalue at the current fixed point is higher than the maximum eigenvalue at the previous fixed point, then the parameter step size decreases by about 30%. This would occur, for example, if the old parameters were on one side of a valley and the new parameter adjustment resulted in crossing to the other side of the valley and raising the elevation.

- If the dot product of the unit parameter adjustment with the previous unit parameter adjustment is less than -0.25, the parameter step size is decreased by another 30%. This would occur if the new parameter adjustment doubled back too much along the previous path, for instance if the local parameter surface was like a bowl.

- If the gradient of the objective function with respect to parameters is less than one in magnitude (indicating proximity to a local minimum) then the gradient instead of the unit gradient is used in any calculations of parameter adjustments, since it decreases as the minimum is approached. Normally, the gradient is greater than one in modulus.

- If none of the above two conditions are met, we assume that the step size can be increased a little, so the parameter step size is incremented by about 30%.

Gradient Search Results For McGeer's Parameters

The second search begins with parameters approximating those of McGeer's walker, as before, and the six parameters which make up the moment of inertia matrix (see Section 5.1.2) are varied. A torsional hip spring is also added to the model before the search is begun.

Using the results from our single-parameter search with a torsional hip spring, a search is begun with a hip spring in place, varying only the moment of inertia

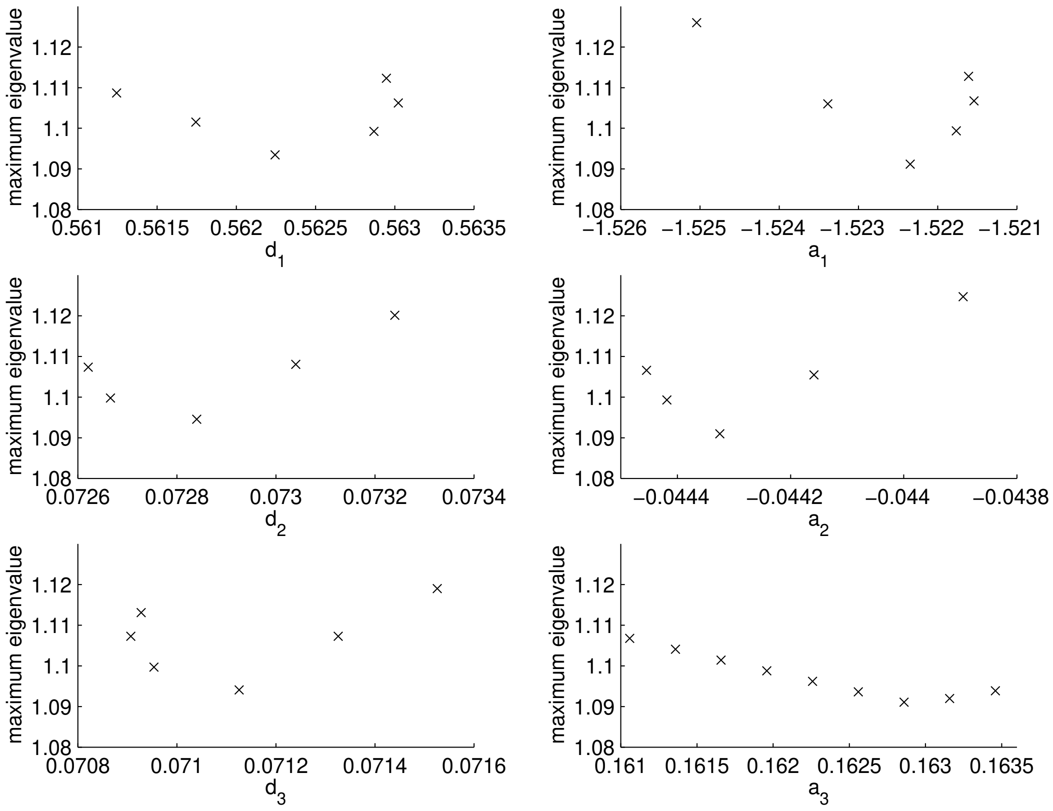

Figure 5.10: A local minimum of maximum eigenvalue with respect to six parameters. This was found by application of a gradient search routine beginning with McGeer's original parameters plus a torsional hip spring. Parameters for this case are as follows: $R = 0.3$, $\gamma = 0.032$, $h = 0.15$, ${}^{3}\mathbf{p}_3^c = [0.38; \ -0.004 \ \ -0.015]$, $d_1 = 0.562244$, $d_2 = 0.072705$, $d_3 = 0.070994$, $\alpha_x = -1.521920$, $\alpha_y = -0.044393$, $\alpha_z = 0.159859$, and the torsional hip spring coefficient is 0.54.

parameters. This leads eventually to a maximum eigenvalue modulus of about 1.095, which is the lowest eigenvalue found in simulations thus far for any walker which fits the form of the model above. Parameters for this case are as follows: $R = 0.3$, $\gamma = 0.032$, $h = 0.15$, ${}^3\mathbf{p}_3^c = [0.38;\ -0.004\ \ -0.015]$, $d_1 = 0.562244$, $d_2 = 0.072705$, $d_3 = 0.070994$, $\alpha_x = -1.521920$, $\alpha_y = -0.044393$, $\alpha_z = 0.159859$, and the torsional hip spring coefficient is 0.54. This eigenvalue appears to be a local minimum with regard to parameterizations of $\mathbf{I}_{cm}$, as shown in figure 5.10. This is the first multi-parameter local minimum discovered for a 3-D passive walker.

A second search was begun where all possible parameters were varied. However, the algorithm called for parameter adjustments which led to regions in parameter space where no fixed points existed, despite our condition-number correction technique (described above). This work is being continued by Eric Phipps in the Cornell Center For Applied Math.

5.3 Conclusions And Future Work

To date, no stable walking motions for any parameters have been found by any researchers, using the 3-D models described above. Previously, the most stable case was that of Coleman (1998b), with a maximum eigenvalue modulus of 1.145. By adding a hip spring and using our gradient search with six parameters, a parameter set was found with a maximum eigenvalue modulus of about 1.095.

Some suggestions for further searches are:

- Change the objective function from the maximum eigenvalue modulus to some function of all the eigenvalues that are greater than one. This would balance the preference between long-period gaits and short-period gaits, and help to eliminate discontinuties in the objective function.

- Use more sophisticated path-following techniques, such as parameterization by arclength, to track solutions. This avoids some numerical problems associated with turning points.

- Include models with ellipsoid or toroid feet. Physical models that are known to be stable (including the Tinkertoy® walker) all have some kind of toroidal geometry (or limiting cases) at their contact patches.

- As suggested by McGeer (1991), include ball-and-socket hips with torsional springs for stability. This might minimize the excessive steering motions of the models above.

- Include freely swinging arms in the models.

5.4 Appendix Of Straight-Legged 3-D Equations

5.4.1 Initialization File

The parameters used below are those reported by by McGeer (1991). This file is run first, to initialize parameters and state. The notation follows from Figures 5.1 and 2.3.

```
clear
global g gam m R p01 p23 p34 pc3 pc4 p4f
global Icm3 Icm4 h hspr hdamp sdamp sspr
format long

g=1;  % gravitational constant
h=0.15;  % hip width
R=0.3;   % foot radius
ml=0.4; mt=0.2; % mass of each leg; mass of point mass on hip
yoff=-0.005; c=0.6; z=0; % McGeer's cm location parameters
rgx=0.1; rgy=0.3; rgz=0.3; % radii of gyration about cm
gam=0.032; % ramp slope
m=[0 0 ml+mt/2 ml+mt/2]'; % put half of hip mass on each leg

p23=[R 0 0]'; % from contact point to center of foot
p34=[1-R 0 -h]'; % from center of foot to swing leg-hip xing
                 % (this is origin of frame 4)
pc3nohm=[c-R yoff -z]'; % cm of leg with no hip mass

% Icm of leg with no extra mass
Icm3nohm=ml.*[rgx^2 0 0;0 rgy^2 0;0 0 rgz^2];

% compute new leg cm with hip mass
pc3= (ml.*pc3nohm + (mt/2).*[p34(1) p34(2) p34(3)/2]')./m(3);

% compute new Icm3 with hip mass
diffl=-(pc3-pc3nohm);
difft=-(pc3-[p34(1) p34(2) p34(3)/2]');
Icm3 = [ml*(diffl(2)^2+diffl(3)^2)+ ...
       mt/2*(difft(2)^2+difft(3)^2)+Icm3nohm(1,1) ...
       -ml*diffl(1)*diffl(2)-mt/2*difft(1)*difft(2)+ ...
       Icm3nohm(1,2) ...
```

```
        -ml*diffl(1)*diffl(3)-mt/2*difft(1)*difft(3)+ ...
        Icm3nohm(1,3);
        -ml*diffl(2)*diffl(1)-mt/2*difft(2)*difft(1)+ ...
        Icm3nohm(2,1) ...
        ml*(diffl(1)^2+diffl(3)^2)+mt/2*(difft(1)^2+ ...
        difft(3)^2)+Icm3nohm(2,2) ...
        -ml*diffl(2)*diffl(3)-mt/2*difft(2)*difft(3)+ ...
        Icm3nohm(2,3);
        -ml*diffl(3)*diffl(1)-mt/2*difft(3)*difft(1)+ ...
        Icm3nohm(3,1) ...
        -ml*diffl(3)*diffl(2)-mt/2*difft(3)*difft(2)+ ...
        Icm3nohm(3,2) ...
        ml*(diffl(1)^2+diffl(2)^2)+mt/2*(difft(1)^2+ ...
        difft(2)^2)+Icm3nohm(3,3)];
Icm4 = Icm3; % x and y dirs are reversed by geometry. z dir is
             % reversed by leg symmetry. Off diag terms are
             % same sign.

% from  frame 4 to cm of swing leg
pc4=[1-R-pc3(1) -pc3(2) -pc3(3) ]';

%from frame 4 to center of swing foot
p4f=[p34(1) 0 0]';

hspr = 0;  % torsional hip spring stiffness
hdamp = 0.0000;  % torsional hip damper
sspr=0;  % steering spring (I never use this)
sdamp=0; % steering damping


% fixed point for McGeer's params to 1e-10
% see chapter 2 of thesis for angle definitions
% [theta1(steer) theta2(lean) theta3(pitch) theta4(pitch) ...
% theta1dot theta2dot theta3dot theta4dot];
y0=[0.5298614882 1.54919263047 -0.30129411451 3.7281651863 ...
-0.0558477940 -0.0311301295 0.3458136041 -0.1201100774 ];

t0=0; % initial time
tfinal=10; % final time
tol=1e-10; % numerical tolerance
```

5.4.2 Cross-Product Function

The functions here is for taking cross products in 3-D. When placed in the same directory as the integration code, it is faster than the `cross` function in MATLAB®.

```
function s=scross(v,b)
% cross product a x b is scross(a)*b

S=zeros(3);
S(1,2)=-v(3);S(1,3)=v(2);S(2,3)=-v(1);
S=S-S';
s=S*b;
```

5.4.3 On-Line ODEs

The integration code calls the derivative file below. This more or less follows the development of equations 2.2 through 2.13 in Section 2.3.

```
function ydot = yderivs_3D(t,y);
global g gam m R p23 p34 pc3 pc4 p4f Icm3 Icm4 h ...
        hspr hdamp sspr sdamp
% 3D file for derivatives of 2-link walker,
% general mass distribution
% and with point-feet.
% Modified for disc feet 3/18/98 and
% checked against Greenwood
% using init_params.m
% Disk feet do not destroy symmetry of M!

% hfoot1 and hdotfoot1 are the z1-height and z1-speed
% of the [bottom of the] swing foot in frame 1.

% define sines and cosines
c01 = cos(y(1));
s01 = sin(y(1));

c02 = cos(y(2));
s02 = sin(y(2));
```

```
c03 = cos(y(3));
s03 = sin(y(3));

s04 = sin(y(4));
c04 = cos(y(4));

% define rotation matrices
R01=[c01 -s01 0 ;s01 c01 0; 0 0 1];
R12=[c02 -s02 0 ;0 0 -1;s02 c02 0];
R23=[c03 -s03 0 ;0 0 -1;s03 c03 0];
R34=[c04 -s04 0 ;s04 c04 0; 0 0 1];

R10=R01';
R21=R12';
R32=R23';
R43=R34';

% first construct [V+G] and tau with only thetadots
% and g terms
vg=zeros(4,1);

% accel of base with only gam and thetadot terms
vdot0 = [0 g*sin(gam) g*cos(gam)]';
vdot1 = R10*vdot0;

% angular velocities and accels of frames 1, 2, and 3
omega1=[0 0 y(5)]';
omega2=R21*omega1+[0 0 y(6)]';
omega2dot=scross(R21*omega1,[0 0 y(6)]');
omega3=R32*omega2+[0 0 y(7)]';
omega3dot=R32*omega2dot+scross(R32*omega2,[0 0 y(7)]');

p233=R32*p23; % from frame 2 to frame 3 in frame 3 coord

v3 = scross(omega3,p233); %vel of stance foot center

                % kludge to make rolling work
vdot3an= [s02*y(6)*y(5)*s03*R+(-c02*y(5)+y(7))*c03*y(7)*R
        s02*y(6)*y(5)*c03*R-(-c02*y(5)+y(7))*s03*y(7)*R 0 ];
         % from omegadot x r but with thetaddots =0

% accel of frame 3
vdot3 = vdot3an + scross(omega3,v3)+ R32*R21*vdot1;
```

```
% accel of cm of stance leg
vcdot3 = scross(omega3dot,pc3)+ ...
        scross(omega3,scross(omega3,pc3))+vdot3;

% m a and Hdot of stance leg
F3 = m(3)*vcdot3;
N3 = Icm3*omega3dot+scross(omega3,Icm3*omega3);

% angular velocity and accel of swing leg
omega4=R43*omega3+[0 0 y(8)]';
omega4dot=R43*omega3dot+scross(R43*omega3,[0 0 y(8)]');

% accel of frame 4
vdot4 = R43*(scross(omega3dot,p34)+ ...
        scross(omega3,scross(omega3,p34))+vdot3);

% accel of cm of swing leg
vcdot4 = scross(omega4dot,pc4)+ ...
        scross(omega4,scross(omega4,pc4))+vdot4;

% velocity of bottom of swing foot in frame 3
vf3 =v3+scross(omega3,p34)+ ...
     R34*(scross(omega4,(p4f-R43*p233)));

% this is for collision detection
% hfoot1 and hdotfoot1 are the z1-height and z1-speed
% of the [bottom of the] swing foot in frame 1.
hfdot1=R12*(R23*vf3);
hfdot1=hfdot1(3);
hf1=R12*(R23*(p233+p34+R34*(p4f-R43*p233)));
hf1=hf1(3);

% m a and Hdot of swing leg
F4 = m(4)*vcdot4;
N4 = Icm4*omega4dot+scross(omega4,Icm4*omega4);

% Sum(F) = m a and Sum(T) = Hdot for swing (about hip
% axis) and stance
f4 = F4;
n4 = N4+scross(pc4,F4);
vg(4) = n4(3);
```

```
f3 = R34*f4+F3;
n3 = N3+R34*n4+scross(pc3,F3)+ ...
     scross(p34,R34*f4)-scross(-p233,f3);
vg(3) = n3(3);

n2 = R23*n3;%+scross(p23,R23*f3);
vg(2) = n2(3);

n1 = R12*n2;
vg(1) = n1(3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Now get mass matrix column by column         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% construct column 1 by taking O1dotdot=1, all others =0
M=zeros(4);

% angular accels of frames 1-3
omega1dot = [0 0 1]';

omega2dot=R21*omega1dot;

omega3dot=R32*omega2dot;

% accels of frame 3 and cm of stance leg
vdot3 = scross(omega3dot,p233);
vcdot3 = scross(omega3dot,pc3)+vdot3;

% m a and I w of stance leg
F3 = m(3)*vcdot3;
N3 = Icm3*omega3dot;

% angular accel frame 4
omega4dot=R43*omega3dot;

% accel of frame 4 and cm of swing leg
vdot4 = R43*(scross(omega3dot,p34)+vdot3);
vcdot4 = scross(omega4dot,pc4)+vdot4;

% m a and I w of swing leg
F4 = m(4)*vcdot4;
```

```
N4 = Icm4*omega4dot;

% Sum(F) = m a and Sum(T) = Hdot for swing and stance
f4 = F4;
n4 = N4+scross(pc4,F4);
M(4,1) = n4(3);

f3 = R34*f4+F3;
n3 = N3+R34*n4+scross(pc3,F3)+ ...
     scross(p34,R34*f4)-scross(-p233,f3);
M(3,1) = n3(3);

f2 = R23*f3;
n2 = R23*n3;%+scross(p23,R23*f3);
M(2,1) = n2(3);

n1 = R12*n2;
M(1,1) = n1(3);

% Repeat Procedure for Column 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% construct column 2
% theta2dot = 1 all other thetadots = 0

%angular accel of frame 2 and 3
omega2dot=[0 0 1]';
omega3dot=R32*omega2dot;

% accels of frame 3 and cm of stance leg
vdot3 = scross(omega3dot, p233);
vcdot3 = scross(omega3dot,pc3)+vdot3;

% m a and I w of stance leg
F3 = m(3)*vcdot3;
N3 = Icm3*omega3dot;

% angular accel frame 4
omega4dot=R43*omega3dot;

% accel of frame 4 and cm of swing leg
vdot4 = R43*(scross(omega3dot,p34)+vdot3);
vcdot4 = scross(omega4dot,pc4)+vdot4;
```

```
% m a and I w of swing leg
F4 = m(4)*vcdot4;
N4 = Icm4*omega4dot;

% Sum(F) = m a and Sum(T) = Hdot for swing and stance
f4 = F4;
n4 = N4+scross(pc4,F4);
M(4,2) = n4(3);

f3 = R34*f4+F3;
n3 = N3+R34*n4+scross(pc3,F3)+ ...
     scross(p34,R34*f4)-scross(-p233,f3);
M(3,2) = n3(3);

n2 = R23*n3;%+scross(p23,R23*f3);
M(2,2) = n2(3);

M(1,2) = M(2,1); % get this for free from symmetry


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% construct column 3
% theta3dot = 1 all other thetadots = 0

% angular accel of frame 3
omega3dot=[0 0 1]';

% accels of frame 3 and cm of stance leg
vdot3 = scross(omega3dot,p233);
vcdot3 = scross(omega3dot,pc3)+vdot3;

% m a and I w of stance leg
F3 = m(3)*vcdot3;
N3 = Icm3*omega3dot;

% angular accel frame 4
omega4dot=R43*omega3dot;

% accel of frame 4 and cm of swing leg
vdot4 = R43*(scross(omega3dot,p34)+vdot3);
vcdot4 = scross(omega4dot,pc4)+vdot4;

% m a and I w of swing leg
```

```
F4 = m(4)*vcdot4;
N4 = Icm4*omega4dot;

% Sum(F) = m a and Sum(T) = Hdot for swing and stance
f4 = F4;
n4 = N4+scross(pc4,F4);
M(4,3) = n4(3);

f3 = R34*f4+F3;
n3 = N3+R34*n4+scross(pc3,F3)+ ...
     scross(p34,R34*f4)-scross(-p233,f3);
M(3,3) = n3(3);

M(2,3) = M(3,2); % symmetry gets these for free
M(1,3) = M(3,1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% construct column 4
% careful w/ copy/pasting non-zero F3 and N3 here
% theta4dot = 1 all other thetadots = 0

% angular accel frame 4
omega4dot=[0 0 1]';

% angular accel of cm of swing leg
vcdot4 = scross(omega4dot,pc4);

% m a and I w of swing leg
F4 = m(4)*vcdot4;
N4 = Icm4*omega4dot;

% Sum(F) = m a and Sum(T) = Hdot for swing
f4 = F4;
n4 = N4+scross(pc4,F4);
M(4,4) = n4(3);

M(3,4) = M(4,3);  % symmetry gets these for free
M(2,4) = M(4,2);
M(1,4) =M(4,1);


tau=[-sspr*y(1)-sdamp*y(5) 0 0 ...
```

```
    -hspr*(y(4)-pi)-hdamp*y(8)]';%torques from springs/dampers

X=chol(M);  % solve for thetadotdot with M symmetric
z = X'\(tau-vg);
Oddot=X \ z;

% feed back extra things in the state vector
% depending on its length

vp = R01*[0 -y(7)*R 0]'; % velocity of contact point

if length(y) == 8
ydot=[y(5) y(6) y(7) y(8) ...
    Oddot(1) Oddot(2) Oddot(3) Oddot(4) hf1 hfdot1]';
end;

if length(y) == 10
ydot=[y(5) y(6) y(7) y(8) ...
    Oddot(1) Oddot(2) Oddot(3)  Oddot(4) ...
     vp(1) vp(2) hf1 hfdot1 ]'; % longer one for graphics
end;
```

5.4.4 On-Line Heelstrike Equations

At the instant that the swing leg touches the ground, the following algorithm solves for new angles and angular velocities. The procedure is explained in Section 2.5.

```
function ynew = heelstrike_try(y);
global g gam m R p23 p34 pc3 pc4 p4f Icm3 Icm4 p4f

% define sines and cosines
c01 = cos(y(1));
s01 = sin(y(1));

c02 = cos(y(2));
s02 = sin(y(2));

c03 = cos(y(3));
s03 = sin(y(3));
```

```
s04 = sin(y(4));
c04 = cos(y(4));

% define rotation matrices
R01=[c01 -s01 0 ;s01 c01 0; 0 0 1];
R12=[c02 -s02 0 ;0 0 -1;s02 c02 0];
R23=[c03 -s03 0 ;0 0 -1;s03 c03 0];
R34=[c04 -s04 0 ;s04 c04 0; 0 0 1];

R10=R01';
R21=R12';
R32=R23';
R43=R34';

% first get angular momentum of whole walker
% about new contact point (cp)

% angular velocities of frames 1-4
omega1=[0 0 y(5)]';
omega2=R21*omega1+[0 0 y(6)]';
omega3=R32*omega2+[0 0 y(7)]';
omega4=R43*omega3+[0 0 y(8)]';

% from contact point to center of foot
p233 = R32*p23;

% velocity of frame 3 and cm of stance leg
v3 = scross(omega3,p233);
vc3=scross(omega3,pc3)+v3;

% velocity of frame 4 and cm of swing leg
v4 = R43*(scross(omega3,p34)+v3);
vc4 = scross(omega4,pc4)+v4;

% from swing foot contact to cm 4 in frame 4
rcpc4 = R43*p233-p4f+pc4;

% from swing foot contact to cm 3 in frame 3
rcpc3 = p233+R34*(-p4f)-p34+pc3;

% H of whole walker about new contact pt
Hcp3 = R34*(cross(rcpc4,m(4)*vc4)+Icm4*omega4)...
```

```
+cross(rcpc3,m(3)*vc3)+Icm3*omega3;

% get components in different frames
Hcp2 = R23*Hcp3;
Hcp1 = R12*Hcp2;

% now get angular momentum of old stance leg
% about hip in z4-dir

% from hip to cm of stance leg
rhc3=-p34+pc3;
Hh3 = cross(rhc3,m(3)*vc3)+Icm3*omega3;

% swap angles and legs
% swing becomes stance, stance becomes swing
% heading and bank angles stay constant

theta3minus=y(3);
y(3)=-pi+y(4)+theta3minus;
y(4)=pi-y(3)+theta3minus;

% some parameters change sign since z3 and z4 change sign
pc3(3)=-pc3(3);
p34(3)=-p34(3);
pc4(3)=-pc4(3);

Icm3(1,3)=-Icm3(1,3);
Icm3(2,3)=-Icm3(2,3);
Icm3(3,1)=-Icm3(3,1);
Icm3(3,2)=-Icm3(3,2);
Icm4=Icm3;

Hpost=getmassmtrx(y);
% this creates a mass matrix identical to the one
% created in the deriv file
% It is a nice shortcut for heelstrike
% (see section 2.5.5 in thesis)

% solve for new angular velocities
ydot=Hpost\ [Hcp1(3) Hcp2(3) Hcp3(3) Hh3(3)]';

% make ynew different things depending on the
% original length of state vector.
```

```
% if graphics are involved, add [0 0] onto the state
% which is the new x,y position of the contact point

if length(y) == 10
ynew=[y(1) y(2) y(3) y(4) ...
     ydot(1) ydot(2) ydot(3) ydot(4) 0 0];
end;

if length(y) == 8
ynew=[y(1) y(2) y(3) y(4) ...
 ydot(1) ydot(2) ydot(3) ydot(4)];
end;
```