Exercises

?? **Walker.**[1] [2] (Computation) ④

Reading: Mariano Garcia, Anindya Chatterjee, Andy Ruina, and Michael Coleman, 'The Simplest Walking Model: Stability, Complexity, and Scaling' *Journal of Biomechanical Engineering, Transactions of the ASME* **120**, 281-288 (1998).

In this exercise, we examine the simplest model of human walking. We will solve a mechanical model with two legs that, purely through their swinging through air and striking the ground, exhibit a stable periodic pacing motion down a ramp. The description, derivation, and motivation for the model are described in the reprint above (available on the book web site [8]). We confine ourselves here to guiding you through building this simulation.

The two legs in this problem swing from a common body, with position BodyPos. One of the legs is implanted in the ground at StanceFootPos and that end cannot move. The other leg's free end is at SwingFootPos, and can swing freely, until it strikes the ground ('heelstrike', defined below), at which time the two legs exchange roles, and the former stance foot can swing. We solve Newton's laws for the legs between heelstrikes.

The angle of the stance leg with respect to vertical is θ and the angle between the two legs is ϕ (Fig. 1 in the reprint). The heel-strike condition for both legs to be touching the ground, thus, is $\phi - 2\theta = 0$.

How will we stop Newton's laws at the heelstrike? Our differential equation solvers will integrate forward to a fixed final time, but how do we trick them to stop at a fixed final *value*? The trick is to wait until we are slightly past heel-strike, change variables from time t to $c = \phi - 2\theta$, and integrate the new equations

$$\begin{aligned} \mathrm{d}\theta/\mathrm{d}c &= (\mathrm{d}\theta/\mathrm{d}t)/(\mathrm{d}c/\mathrm{d}t) && (??)\\ \mathrm{d}\phi/\mathrm{d}c &= (\mathrm{d}\phi/\mathrm{d}t)/(\mathrm{d}c/\mathrm{d}t) && (1)\\ \mathrm{d}t/\mathrm{d}c &= 1/(\mathrm{d}c/\mathrm{d}t) \end{aligned}$$

from the current value of c to $c = 0$.[3]

Let us begin by thinking about the case of the pendulum: how would we efficiently measure the period? If we start the pendulum at rest, and measure the time it takes to cross $\theta = 0$, that is one quarter of the period. Let us overshoot and then backtrack to $\theta = 0$.

(a) *As in Exercise 3.12, define dydt(y,t) for the pendulum, where* $y = [\theta, \dot{\theta}]$, *and define dzdc(z,c) for* $c = \theta$, $z = [\theta, \dot{\theta}, t]$. *Start the pendulum at rest at a range of initial angles* θ_0 *between zero and* π; *use the ODE solver provided to integrate forward in time in steps of* $\delta = 0.1$ *until* $\theta < 0$. *Then run* θ *to the zero crossing using* z, *and find the value of* t_c *at the zero crossing. Plot the period* $4t_c$ *versus* θ_0. *What happens at* $\theta_0 = \pi$?

Newton's equations of motion for our walker are given by eqns (1) and (2) in the reprint.

(b) *Set up a file defining a class Walker. For now, it should know its geometry (leg length* L, *ramp slope* γ, *current stance foot position (started at zero), and current* θ, $\dot{\theta}$, ϕ, *and* $\dot{\phi}$). *Start by giving it two member functions, dydt and dzdc. Each member function should unpack the vector into* $\theta, \dot{\theta}, \phi, \dot{\phi}$ *(and* t *for* z), *calculate the derivative for each, and return the appropriate vector. Neither function needs to look at or change the current state of the walker. Finally, write functions GetStateVector which retrieves* $\mathbf{y}$ *from the current state of the walker, and SetStateVector(*$\mathbf{y}$*) which sets the walker to the current state* $\mathbf{y}$.

We will be changing between animations and plots in developing, debugging, and exploring our walker.

[1] New exercise supplementing *Statistical Mechanics: Entropy, Order Parameters, and Complexity* by James P. Sethna, copyright Oxford University Press, 2007, page **??**. A pdf of the text is available at pages.physics.cornell.edu/sethna/StatMech/ (select the picture of the text). Hyperlinks from this exercise into the text will work if the latter PDF is downloaded into the same directory/folder as this PDF.

[2] This exercise and the associated software were developed in collaboration with Christopher Myers.

[3] Another method is to interpolate between existing time-points, but it is important to do so with appropriate accuracy.

Let us set up some of these tools now.

(c) **Plotting trajectories.** *We will use the solver to find* $\mathbf{y}(t)$ *using dydt, as follows. Let us use initial conditions that approximately agree with those used in Fig. 2 of the reprint: start with* $\gamma = 0.009$, $\theta_0 = 0.2$, $\dot{\theta}_0 = -0.2$, $\phi = 0.4001$, *and* $\dot{\phi} = 0$. *Calculate* θ *and* ϕ *at points* t *with intervals* $\mathrm{d}t = 0.1$ *from* $t = 0$ *to* $t = 4$ *(these are just components of* y*). Plot these: they should agree approximately with the plot in Fig. 2 of the reprint. Also plot* $c(t) = \phi - 2\theta$.

Next, check your function dzdc. The initial condition we chose is just past the last collision, so c *starts at* $c_0 = \phi - 2\theta = 0.0001$. *Pick a final value* c_1 *from your plot just below the first maximum. Use your solver to find* $\mathbf{z}(c)$, *and again plot* θ, ϕ, *and* c *versus* t, *if possible on the same graph as the values found from* $y(t)$. The two methods—solving using t and c—should agree.

(d) **Animation.** *Animate your walker, using the tools provided in Exercise 3.12. You will want to add member functions to your Walker GetBodyPos and GetSwingFootPos (that compute the body position and swing foot position using StanceFootPos,* θ, *and* ϕ*), along with GetStanceFootPos (for completeness). You will want a class WalkerDisplay, with*

- *A constructor with a Walker argument* w, *which stores* w *and extracts from it the initial configuration of the walker.*
- *An update function, which re-reads the current state from the walker and displays it. It should draw the stance leg a different color from the swing leg.*

Find $\mathbf{y}(t)$ *for a reasonable length of time* T, *and then run the animation—set the current state vector, update the display, repeat for the next time step. It should swing around in an entertaining way.*

Now, let us figure out how to stop at the end of a step. You can see from your plot of part (c) that the collision condition $c(t) = \phi - 2\theta = 0$ occurs once at mid-step (when the swing foot 'scuffs', briefly going underground), and again at the end of a step (the 'heelstrike').[4]

This means that during the first part of the trajectory the swing foot is beneath the surface of the ramp! In more complicated models, this unphysical behavior is avoided through the introduction of knees, ankles, or side-to-side rocking (see the reprint).

(e) **Step.** *Add a member function Step(*t_{initial}, t_{final}, $\mathrm{d}t$*) to Walker. Step will move the current swing leg forward until either* t_{final} *or heelstrike, whichever comes first, and returns the time taken. As in the pendulum, Step will creep forward in time increments* $\mathrm{d}t$, *and when it overshoots it will change variables to* c *and integrate backward to the heelstrike condition. In particular,*

- *Integrate forward in time in tiny steps* $\mathrm{d}t$, *to a time* t_f *which is the smaller of* $t + \mathrm{d}t$ *and* t_{final}, *and update the state of the walker*
- *If* $c(t - \mathrm{d}t) < 0$ *and* $c(t) > 0$, *use the change of variables in eqn* **??** *to integrate backward in time until* $c = 0$ *(the heelstrike), update the state of the walker and return the final time* $t(c = 0)$.
- *If no heelstrike occurs before* t_{final}, *return nothing (implying final time is* t_f*).*

We now need to include the special feature of this subject area: the exchange of dynamics at the point of impact. At the heelstrike, the impulse as the swing foot abruptly hits the ground changes the values of the velocities of the two legs. This impulse, plus the exchange of roles of the swing and stance legs, are given in eqn (4) of the reprint.

(f) **Heelstrike.**

(1) *Add to the class Walker a member function ExecuteHeelstrike(), which changes the four state variables in the walker according to eqn (4) of the reprint.*

(2) *Just after the heelstrike,* c *is near zero, but likely will be a small positive or negative value due to rounding errors. If it is positive there is no problem, but if it is negative you will immediately trigger another heelstrike! You can set a flag to keep track of whether you are just before or after a heelstrike, but we recommend instead just increasing* ϕ *by* $-2c$ *after ExecuteHeelstrike to ensure that* $\phi - 2\theta$ *is positive. (Print an error message if* $-c > 10^{-10}$, *so you*

[4] The height of the swing foot above the ground is $L(\cos(\theta) - \cos(\phi - \theta))$, which is zero if $\phi = 2\theta$. It is also zero for $\phi = 0$: the scuff occurs in the small region between the zero crossings of ϕ and c. To avoid this unphysical subterranean scuffing, more elaborate models use knees and ankles...

will know that this rounding-up is not causing significant accuracy problems.)

(3) *Write a function Walk, that successively takes steps and executes heel strikes. Animate your walker, and debug it using the parameters from part (c).*

The *attractor* of a dynamical system represents its behavior after a long time has passed. For our system, a steady walking stride is represented by a periodic cycle through the space of θ, ϕ, $\dot{\theta}$ and $\dot{\phi}$. It is natural to focus our attention once per step, after each heelstrike: a steady stride then becomes a fixed-point under the mapping which moves forward one step. We cannot directly visualize this four-dimensional space, but we can plot one variable θ and look to see whether it settles down to a fixed-point, at various values of the slope γ. Figure 6 of the reprint shows precisely this plot: the behavior of $\theta(\gamma)$ just after the heelstrike after many steps have been taken.

(g) **Bifurcation diagram and period-doubling.** *Duplicate this period-doubling bifurcation diagram. The initial conditions are tricky (for lots of possible starting states, the walker goes nuts): we have found by trial and error that the initial condition $\theta_0 = 0.235$, $\dot{\theta}_0 = -0.23$, $\phi = 0.475$, and $\dot{\phi} = -0.026$ gets it walking for $0.01 < \gamma < 0.019$. Run for $n_{\mathrm{transient}}$ steps, then record for n_{cycles} steps, plotting θ just after each heelstrike. You will need to throw away a few tens of steps to get a good plot.*

The walker has a stable walking stride only for $\gamma < \gamma_1 \approx 0.15$: it then starts limping (short-long, short-long, repeating after a period of two steps). Qualitative changes like this in the behavior of dynamical systems as parameters are tuned are called *bifurcations*: notice the attractor forks open or bifurcates at γ_1. This period-two cycle continues until a larger slope γ_2, where it then goes into a period four cycle. This *period doubling* repeats in a geometrical series, until at γ_∞ the walker goes into a chaotic motion. This is known as the period-doubling *route to chaos*: many different systems exhibit precisely this same sequence of bifurcations: see Exercise 12.9.

(h) **Animating Chaos.** *Run your walker in the chaotic region. Is the irregularity in the steps striking?*

In most cases, the chaos in dynamical systems is rather subtle: not a tornado, but more of a gentle, irregular eddy.