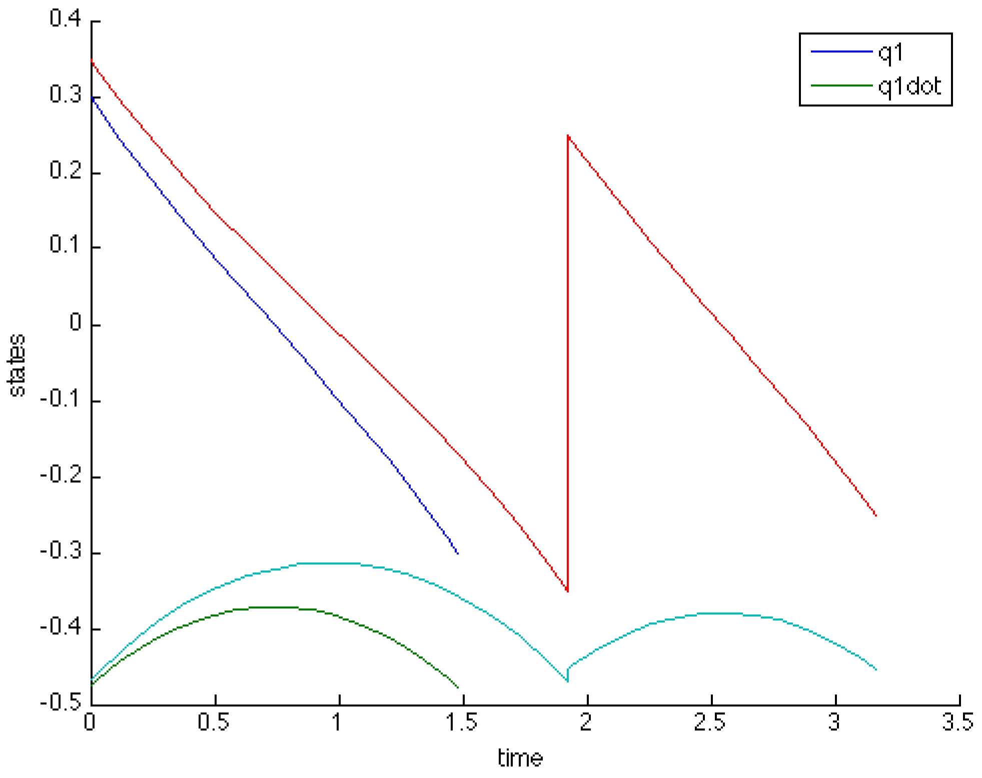
Rimless wheel walking on uneven terrain

Table of Contents

Simultion study of walking with up-down steps, plus two kinds of random bumpiness: uniform and normal distributions of bumps. Full simulations with nonlinear pendulum ODE

Set up rimless wheel normally and with bumps

```
wrw2 = walkrw2;      % original rimless wheel
wu = walkrw2uneven();  % uneven terrain
wub = gradsearch(set(wu, 'bumps', [-0.05 0.05]),[],'info',0); % a limit cycle with

% Plot the states for the two models
clf; hold all
onestep(wu); legend('q1', 'q1dot');
onestep(wub); hold off
```



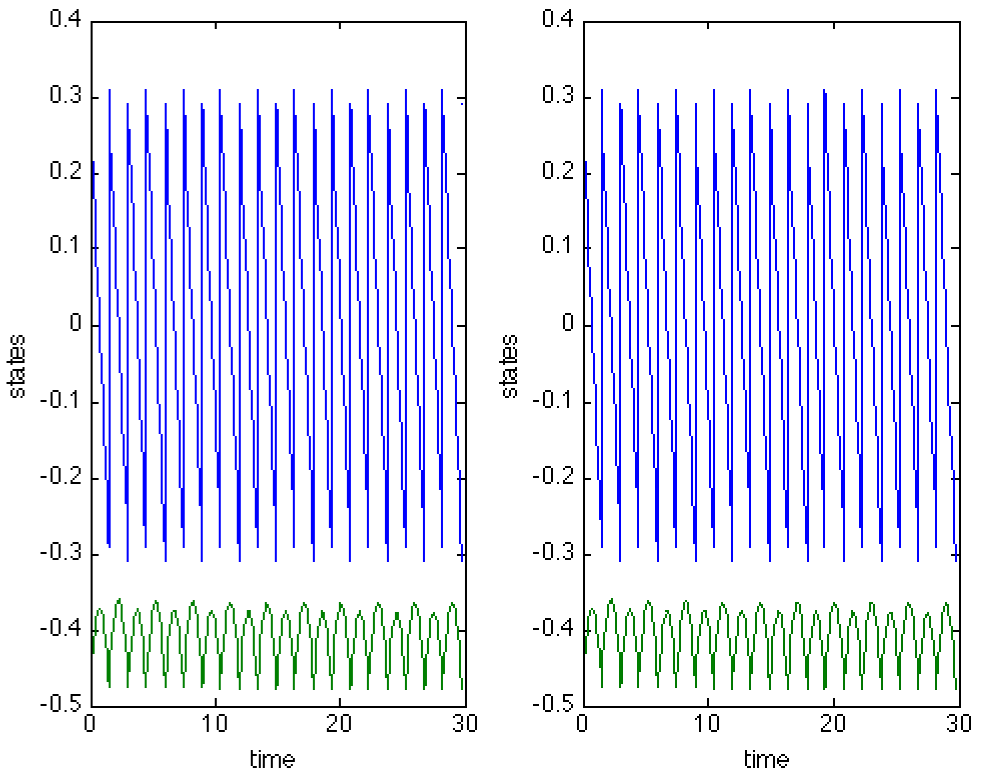
Demo a few operations

The uneven walker can be simulated using two functions: manybumps, starting on the level gnd fixed pt; and onestep where bumps can be incorporated into the parameters. Demonstrate that step times are slower on bumps

```
[xc,normaltime,xs,ts,normalenergies] = onestep(wrw2);

[xc,tc] = onestep(wu);
% manybumps: demonstrate several normal steps, using the automatic plot
manybumps(wu, [], [0 0 0 0 0 0]); % where the list is a bunch of bump heights
fprintf(1,'average step times normally: %f\n', tc) % each step time

clf; subplot(121)
% and now do the same for some bumps
upanddownbumps = repmat([0.01 -0.01],1,10);
[xd,td,xs,ts] = manybumps(wu, [], upanddownbumps, 'plotstep', 1);
fprintf(1,'average step times with bumps: %f\n', mean(diff([0; td]))) % each step

% onestep: here is another way to run several steps, by incorporating the bumps in
% the class itself. Here onestep is overloaded to provide this function,
% and when bumps is a vector, it takes as many steps as there are bumps
subplot(122)
wub2 = set(wu, 'bumps', upanddownbumps);
onestep(wub2);
```

```
wub4 = findgaitspeed1(wub, 'speed', 0.4, 'parmvary', 'P','info',0);
[xc,tc,xs,ts,energies,indices] = onestep(wub4);
speedinfo = walkspeed(wub4);
costoftransport = 0.5*(get(wub4,'P')^2)/(speedinfo.steplength)
```

```
        average step times normally: 1.477426

        average step times with bumps: 1.487063

        costoftransport =

            0.0201
```



Step time goes up with bumpiness squared

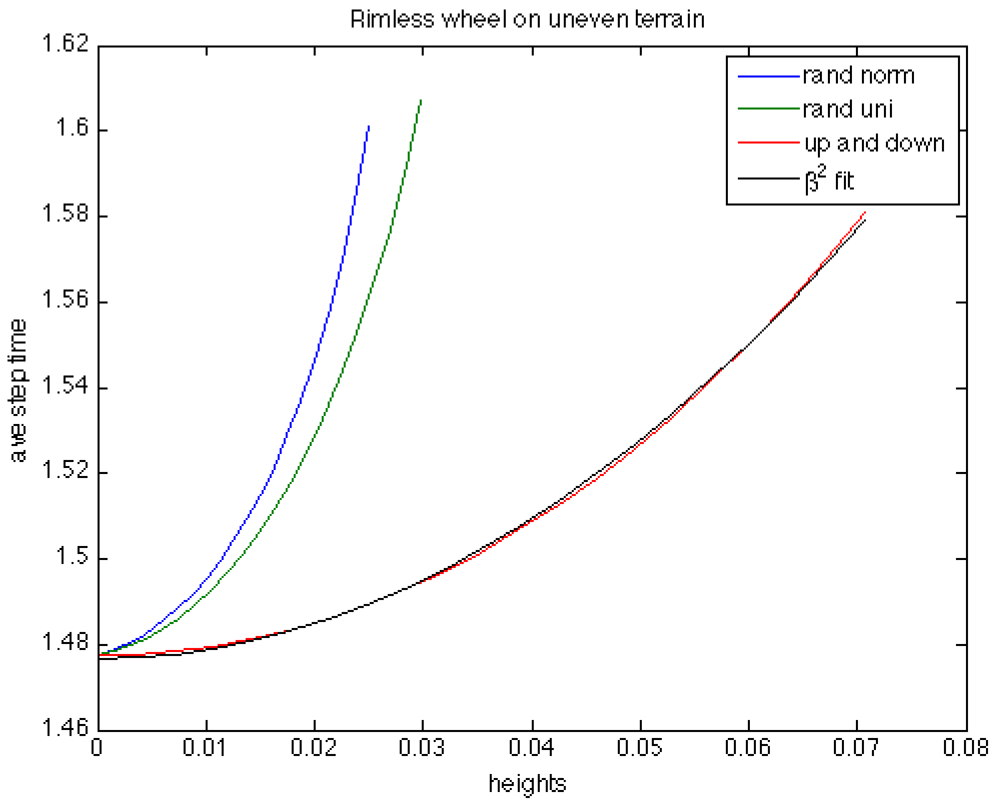
For rimless wheel with up-and-down bumps plus randomly distributed. Show that the average step times increase quadratically with bumpiness and that random bumps are more costly than up-and-down bumps, mainly because the distributions have tails, and it is the tails that cause very uneven step times.

```
clear fupdown
fupdown.heights = linspace(0, 0.05, 21); fupdown.steptimes = 0*fupdown.heights;
fupdown.bumps = repmat([1 -1],1,10);
for i = 1:length(fupdown.heights)
  [xd,td,xs,ts] = manybumps(wu, [], fupdown.bumps*fupdown.heights(i));
  fupdown.steptimes(i) = mean(diff([0; td]));
end
```

```
clf; plot(fupdown.heights, fupdown.steptimes)
% indeed, time goes up quadratically

% a smarter way is to just use the limit cycle for up and down
lupdown.heights = linspace(0, 0.05, 21); lupdown.steptimes = 0*lupdown.heights;
lupdown.bumps = [1 -1];
w = wu;
for i = 1:length(lupdown.heights)
  w = gradsearch(set(w,'bumps',lupdown.heights(i)*lupdown.bumps),[],'info',0);
  [xc,tc,xs,ts,energies,indices] = onestep(w);
  lupdown.steptimes(i) = mean(diff([0; ts(indices)]));
end
plot(lupdown.heights, lupdown.steptimes, fupdown.heights, fupdown.steptimes)
bumps = lupdown.heights*std(lupdown.bumps); % quadratic fit for bumps
X = [bumps'.^2 1+0*bumps'];
lupdown.pfit = regress(lupdown.steptimes', X);

% uniformly distributed bumps
rng('default')
numsteps = 100;
clear frand
frand.bumps = 2*rand(1, numsteps); frand.bumps = frand.bumps - mean(frand.bumps);
frand.heights = linspace(0, 0.05, 21); frand.steptimes = 0*frand.heights;
for i = 1:length(frand.heights)
  [xd,td,xs,ts,sis,es] = manybumps(wu, [], frand.bumps*frand.heights(i));
  steptimes = diff([0; td]);
  frand.steptimes(i) = mean(steptimes);
end
% with a plot of average step times vs bump height
plot(frand.heights*std(frand.bumps), frand.steptimes,...
  lupdown.heights*std(lupdown.bumps), lupdown.steptimes)
xlabel('heights'); ylabel('times'); legend('random','up and down')

% normally distributed bumps
rng('default')
numsteps = 100;
clear frandn
frandn.bumps = 1*randn(1, numsteps); frandn.bumps = frandn.bumps - mean(frandn.bum
frandn.bumps = frandn.bumps / std(frandn.bumps);
frandn.heights = linspace(0, 0.025, 21); frandn.steptimes = 0*frandn.heights;
for i = 1:length(frandn.heights)
  [xd,td,xs,ts,sis,es] = manybumps(wu, [], frandn.bumps*frandn.heights(i));
  steptimes = diff([0; td]);
  frandn.steptimes(i) = mean(steptimes);
end
% with a plot of average step times vs bump height
clf; plot(frandn.heights*std(frandn.bumps), frandn.steptimes,...,
  frand.heights*std(frand.bumps), frand.steptimes,...
  lupdown.heights*std(lupdown.bumps), lupdown.steptimes)
hold on; plot(bumps, lupdown.pfit(1)*bumps.^2+lupdown.pfit(2), 'k');
xlabel('heights'); ylabel('ave step time');
legend('rand norm','rand uni','up and down', '\beta^2 fit')
title('Rimless wheel on uneven terrain')
```

```
% this is for these standard deviations
fprintf(1,'Statistical info about the bumps:\n');
fprintf(1,'standard dev of up and down = %g\n', std(lupdown.bumps,1));
fprintf(1,'standard dev of random uni  = %g\n', std(frand.bumps,1));
% which is supposed to be (b-a)/sqrt(12)
fprintf(1,'standard dev of randomnorm  = %g\n', std(frandn.bumps,1));
% of course, the long tail is costly
```

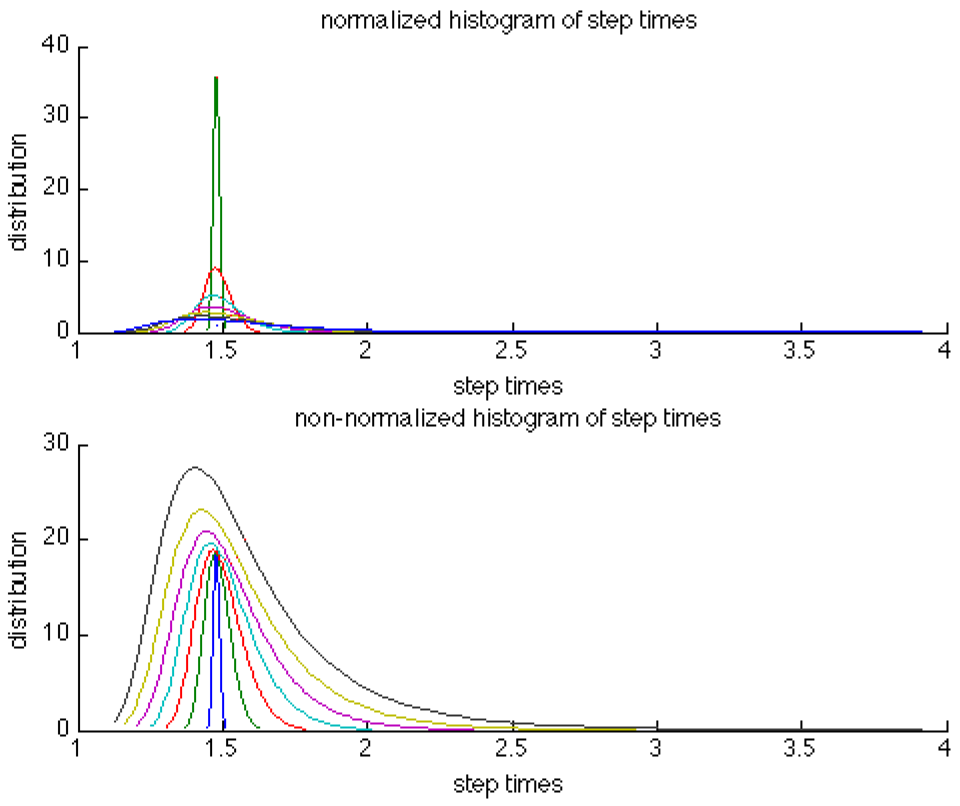
```
Statistical info about the bumps:

standard dev of up and down = 1
standard dev of random uni  = 0.591056
standard dev of randomnorm  = 0.994987
```



Distribution of step times skews with bumpiness

Make a plot of the distributions of step times and try this for a bunch of heights, for normally and uniformly distributed bumps.

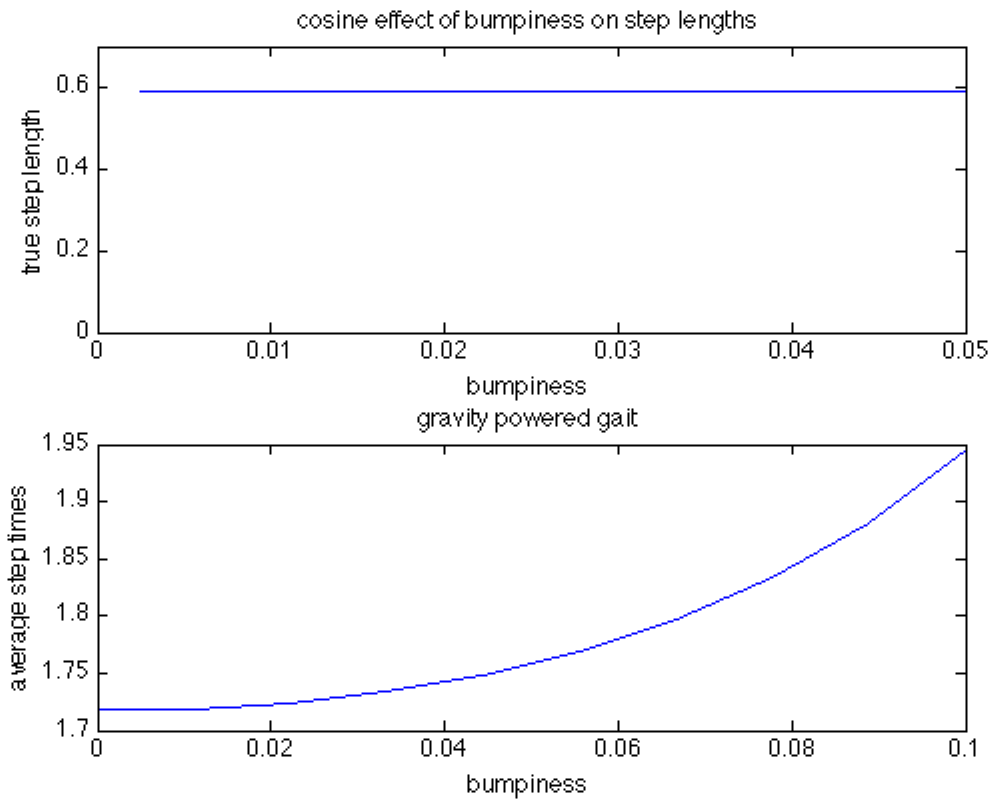
```
rng('default')
numsteps = 100;
clear distp
distp.bumps = 2*rand(1, numsteps);
distp.bumps = distp.bumps - mean(distp.bumps); % uniform -1 to 1
distp.heights = linspace(0, 0.05, 21);
distp.heights = distp.heights(2:end); distp.bumptimes = 0*distp.heights;
clf
for i = 1:length(distp.heights)
  [xd,td,xs,ts,sis,es] = manybumps(wu, [], distp.bumps*distp.heights(i));
  steptimes = diff([0; td]);
  distp.bumptimes(i) = mean(steptimes);
  distp.collisionworks(i) = mean([es.heelstrikework]);
  % here's the distributino of collision work
  h = histfit(-[es.heelstrikework],[],'gev'); hold on; %plot([normaltime normaltim
  distp.workx{i} = get(h(2),'xdata');
  distp.worky{i} = get(h(2),'ydata');

  h = histfit(steptimes,[],'gev'); hold on; plot([normaltime normaltime],get(gca,'
  distp.timex{i} = get(h(2),'xdata');
  distp.timey{i} = get(h(2),'ydata');
  plot(mean(steptimes)*[1 1], get(gca,'ylim'),'b--'); hold off; %pause
end

% Normalized plot tries to get areas right, but looks a bit too spikey
clf; subplot(211); hold all;
plot(normaltime*[1 1],get(gca,'ylim'),'--');
for i = 1:3:length(distp.heights)
  curvearea = trapz(distp.timex{i},distp.timey{i});
  plot(distp.timex{i},distp.timey{i}/curvearea);    % normalized plot
  [ymin,imin] = min((distp.timex{i}-distp.bumptimes(i)).^2);
  plot(distp.timex{i}(imin),distp.timey{i}(imin)/curvearea,'r.');
end
xlabel('step times'); ylabel('distribution');
title('normalized histogram of step times')

% The non-normalized timing distribution plot
% has wrong areas but makes it easier to see what's going on
subplot(212); hold all; for i=1:3:length(distp.heights);
  plot(distp.timex{i},distp.timey{i});
  [ymin,imin] = min((distp.timex{i}-distp.bumptimes(i)).^2);
  plot(distp.timex{i}(imin),distp.timey{i}(imin),'r.');
end
xlabel('step times'); ylabel('distribution');
title('non-normalized histogram of step times')

% The following verifies what we already know: (commented out)
%clf;
%plot(distp.heights, distp.bumptimes)
% obviously step times go up with square of bumpiness as we already know
```



A couple other minor findings

Step length doesn't vary much with bumpiness, and a gravity powered gait functions just like the push-off gait

```
% Show that step length doesn't vary much with bumpiness
% and I don't think step length matters:
clf; subplot(211); plot(distp.heights, mean(2*sin(0.3)*cos(distp.bumps'*distp.heig
xlabel('bumpiness'); ylabel('true step length'); set(gca,'ylim',[0 0.7])
title('cosine effect of bumpiness on step lengths');

% Do it again with a gravity gait
subplot(212);
wrwg = findgait(wu,set(wu,'gamma',0.06,'P',0),'info',0);
rng('default')
numsteps = 25;
bumps = 2*rand(1, numsteps); bumps = bumps - mean(bumps); % uniform -1 to 1
heights = linspace(0, 0.1, 10); bumptimes = 0*heights;
for i = 1:length(heights)
  [xd,td,xs,ts] = manybumps(wrwg, [], bumps*heights(i));
  bumptimes(i) = mean(diff([0; td]));
end
plot(heights, bumptimes)
xlabel('bumpiness'); ylabel('average step times'); title('gravity powered gait');
```

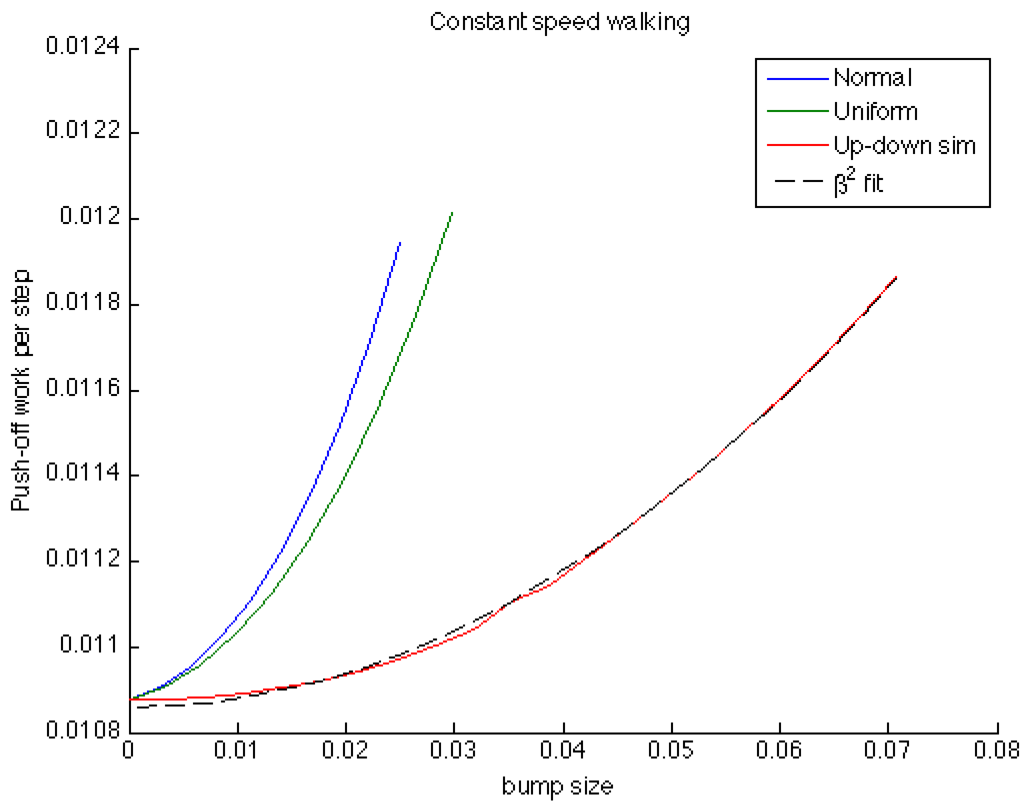


Cost goes up with square of bumpiness

For walking at constant speed, for up-down bumps and for uniformly distributed bumpiness

```
% using a two-step limit cycle
clear ludcv
ludcv.heights = linspace(0, 0.05, 21); ludcv.steptimes = 0*ludcv.heights;
ludcv.bumps = [1 -1];
% This is how to find a gait where P is adjusted to get constant speed
w = findgaitspeed1(wu, 'speed', 0.4, 'parmvary', 'P','info',0);
ludcv.steptimes(1) = normaltime; % no bumps for first element
ludcv.Ps(1) = get(w,'P');
for i = 2:length(ludcv.heights)
  w = gradsearch(set(w,'bumps',ludcv.heights(i)*ludcv.bumps),[],'info',0);
  w = findgaitspeed1(w, 'speed', 0.4, 'parmvary', 'P','info',0);
  [xc,tc,xs,ts,energies,indices] = onestep(w);
  ludcv.steptimes(i) = mean(diff([0; ts(indices)]));
  ludcv.Ps(i) = get(w,'P');
end
actualbumps = ludcv.heights*std(ludcv.bumps);
% First verify that step times are about constant, as needed for speed
clf; plot(ludcv.heights*std(ludcv.bumps), ludcv.steptimes)
% Then plot
clf; plot(ludcv.heights*std(ludcv.bumps), 0.5*ludcv.Ps.^2)
X = [(actualbumps)'.^2 ones(length(ludcv.heights),1)];
pfit = regress(0.5*ludcv.Ps'.^2, X);
```

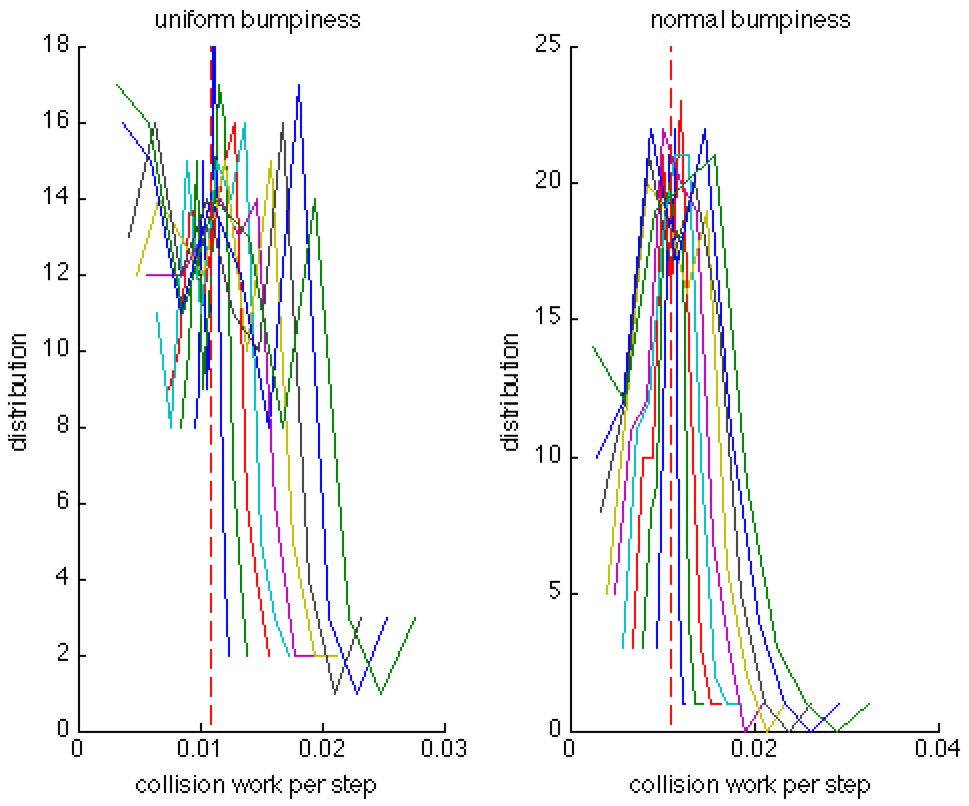
```
hold on; plot(actualbumps,pfit(1)*actualbumps.^2+pfit(2),'r');
xlabel('bump size'); ylabel('Push-off work');
legend('Simulation', '\beta^2 fit');

% and uniformly distributed bumps
clear consvu
rng('default'); numsteps = 100; % same bumps as frand above
consvu.bumps = 2*rand(1, numsteps); consvu.bumps = consvu.bumps - mean(consvu.bump
consvu.heights = linspace(0, 0.05, 10); %consvu.steptimes = 0*consvu.heights;
w = wu;
consvu.Ps(1) = get(w, 'P'); P = consv.Ps(1);
consvu.speeds(1) = gaitspeed(w);
opts = optimoptions(@fsolve, 'Display', 'off');
[xc,tc,xs,ts,energies,indices] = onestep(w);
consvu.energies{1} = energies;

for i = 2:length(consvu.heights)
  w = set(w, 'bumps', consvu.bumps*consvu.heights(i));
  consvu.Ps(i) = fsolve(@(pee) getspeedp(w, pee, 0.4), P, opts);
  P = consvu.Ps(i);
  [xc,tc,xs,ts,energies,indices] = onestep(set(w, 'P', P));
  consvu.energies{i} = energies;
  td = ts(indices);
  steptimes = diff([0; td]);
  consvu.steptimes{i} = steptimes;
end
clf; hold all
plot(std(consvu.bumps)*consvu.heights, 0.5*consvu.Ps.^2); % this is work per step
xlabel('bump size'); ylabel('Push-off work per step'); title('Constant speed walki
plot(ludcv.heights*std(ludcv.bumps), 0.5*ludcv.Ps.^2)
X = [(actualbumps)'.^2 ones(length(ludcv.heights),1)];
pfit = regress(0.5*ludcv.Ps'.^2, X);
plot(actualbumps,pfit(1)*actualbumps.^2+pfit(2),'r');
legend('Uniform', 'Up-down sim', '\beta^2 fit');

% and normally distributed bumps
clear consvn
rng('default'); numsteps = 100; % same bumps as frand above
consvn.bumps = 1*randn(1, numsteps); consvn.bumps = consvn.bumps - mean(consvn.bum
consvn.bumps = consvn.bumps / std(consvn.bumps);
consvn.heights = linspace(0, 0.025, 10); %consvn.steptimes = 0*consvn.heights;
w = wu;
consvn.Ps(1) = get(w, 'P'); P = consvn.Ps(1);
consvn.speeds(1) = gaitspeed(w);
opts = optimoptions(@fsolve, 'Display', 'off');
[xc,tc,xs,ts,energies,indices] = onestep(w);
consvn.energies{1} = energies;

for i = 2:length(consv.heights)
  w = set(w, 'bumps', consvn.bumps*consvn.heights(i));
  consvn.Ps(i) = fsolve(@(pee) getspeedp(w, pee, 0.4), P, opts);
  P = consvn.Ps(i);
  [xc,tc,xs,ts,energies,indices] = onestep(set(w, 'P', P));
  consvn.energies{i} = energies;
  td = ts(indices);
  steptimes = diff([0; td]);
  consvn.steptimes{i} = steptimes;
end
clf; hold all
plot(std(consvn.bumps)*consvn.heights, 0.5*consvn.Ps.^2); % this is work per step
plot(std(consvu.bumps)*consvu.heights, 0.5*consvu.Ps.^2); % this is work per step
xlabel('bump size'); ylabel('Push-off work per step'); title('Constant speed walki
plot(ludcv.heights*std(ludcv.bumps), 0.5*ludcv.Ps.^2)
X = [(actualbumps)'.^2 ones(length(ludcv.heights),1)];
pfit = regress(0.5*ludcv.Ps'.^2, X);
plot(actualbumps,pfit(1)*actualbumps.^2+pfit(2),'k--');
legend('Normal', 'Uniform', 'Up-down sim', '\beta^2 fit');
```



Distribution of work per step

As bumpiness increases, the distribution of collision work changes. For minor bumpiness, the collisions are roughly normally distributed. But for larer bumps, the distribution becomes skewed. It becomes a negatively sloped line, meaning that there are more frequent small collisions, and relatively few large (costly) collisions.

```
clf; subplot(121); hold all
for i = 2:length(consvu.heights)
  %hist(-[consvu.energies{i}.heelstrikework]); % use this to plot regular histogra
  [nelements,xcenters] = hist(-[consvu.energies{i}.heelstrikework]);
  plot(xcenters, nelements);
  xlabel('collision work per step'); ylabel('distribution'); title('uniform bumpin
  %plot(mean(-[consvu.energies{i}.heelstrikework])*[1 1],get(gca,'ylim'),'--')
end
plot(normalenergies.pushoffwork*[1 1], get(gca,'ylim'), '--'); % smooth ground

subplot(122); hold all
for i = 2:length(consvn.heights)
  %hist(-[consvn.energies{i}.heelstrikework]); % use this to plot regular histogra
  [nelements,xcenters] = hist(-[consvn.energies{i}.heelstrikework]);
  plot(xcenters, nelements);
  xlabel('collision work per step'); ylabel('distribution'); title('normal bumpine
  %plot(mean(-[consvn.energies{i}.heelstrikework])*[1 1],get(gca,'ylim'),'--')
end
plot(normalenergies.pushoffwork*[1 1], get(gca,'ylim'), '--'); % smooth ground
```

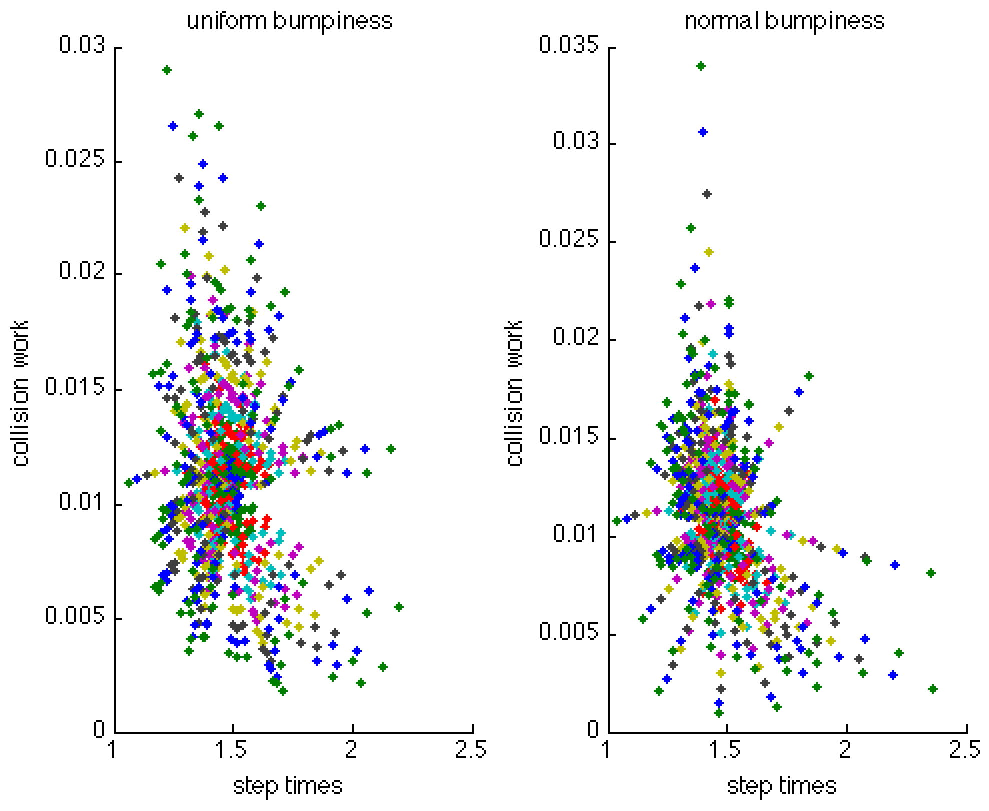


Work vs time

Observation: Long step times seem to go with relatively lower collisions. Highest collisions seem to go with the nominal step times.

```
clf; subplot(121); hold all
for i = 2:length(consvu.heights)
  plot([consvu.steptimes{i}],-[consvu.energies{i}.heelstrikework],'.',...
    'MarkerSize', 10)
  xlabel('step times'); ylabel('collision work'); title('uniform bumpiness');
end
```

```
subplot(122); hold all
for i = 2:length(consvn.heights)
  plot([consvn.steptimes{i}],-[consvn.energies{i}.heelstrikework],'.',...)
    'MarkerSize', 10)
  xlabel('step times'); ylabel('collision work'); title('normal bumpiness');
end
```



Published with MATLAB® R2013a