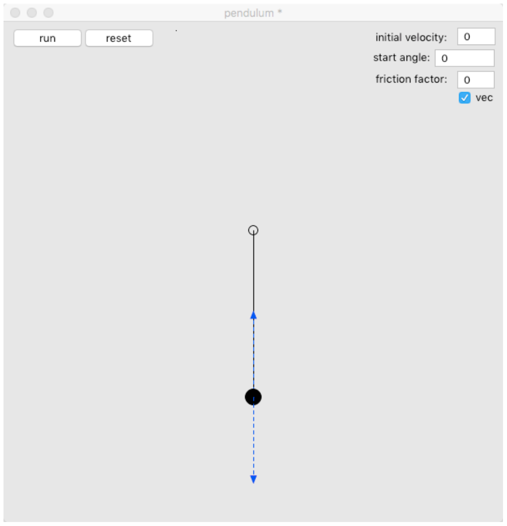
Pendulum as analogy used in helping to explain oscillating chemical reactors

R. K. Herz, rich@reactorlab.net, 2017

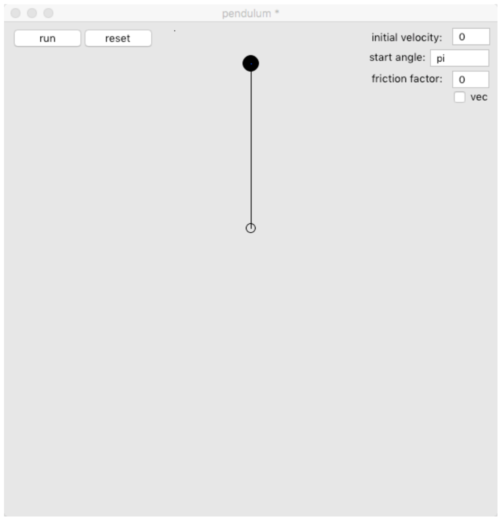
Consider a point mass "bob" connected to a central "swivel" point through a massless, rigid "rod." The system is subject to our usual gravitational acceleration but there is negligible air resistance to motion. We can set the friction at the swivel point to zero or to some value proportional to angular velocity. In class, we saw a simulation of such a pendulum, which was written using LiveCode.

There are two steady states. One steady state is stable, the other is unstable. At a stable steady state, the system returns to that state after a small perturbation. At an unstable state, the system moves away from that state after a small perturbation. Here are the two steady states of the pendulum.



initial velocity: 0
start angle: 0
friction factor: 0
✓ vec

Stable steady state

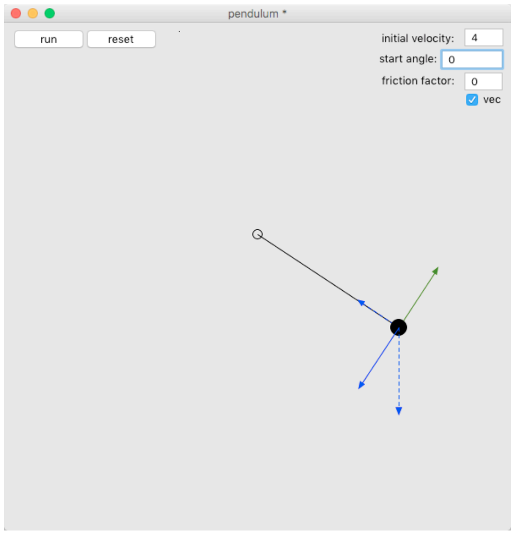


initial velocity: 0
start angle: pi
friction factor: 0
☐ vec

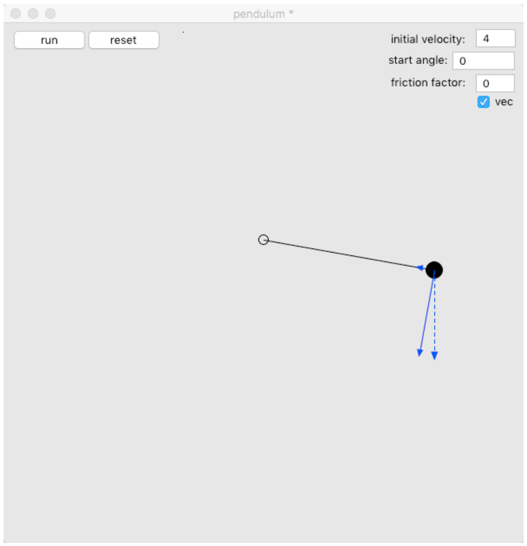
Unstable steady state

The dashed blue vectors shown on the left are the components of acceleration acting on the bob. The vector pointing down is gravitational acceleration, which always points down with a constant magnitude. The vector pointing up is tension in the rod that exactly offsets gravity on the bob. On the right, the vectors are not shown but the downward pointing gravity vector is counteracted by an upward pointing vector due to compression in the rod which offsets gravity on the bob.

From the stable steady state, we set the pendulum swinging. The system is frictionless.



initial velocity: 4
start angle: 0
friction factor: 0
✓ vec



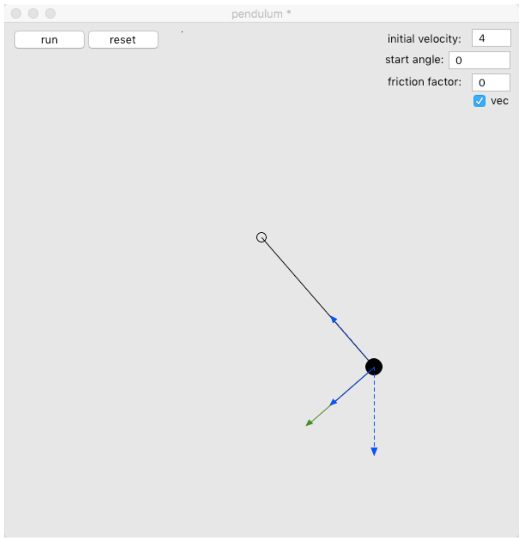
initial velocity: 4
start angle: 0
friction factor: 0
✓ vec

The green vector is the tangential velocity of the bob. The solid blue vector is the tangential component of acceleration acting on the bob. On the left, the bob is swinging upwards.

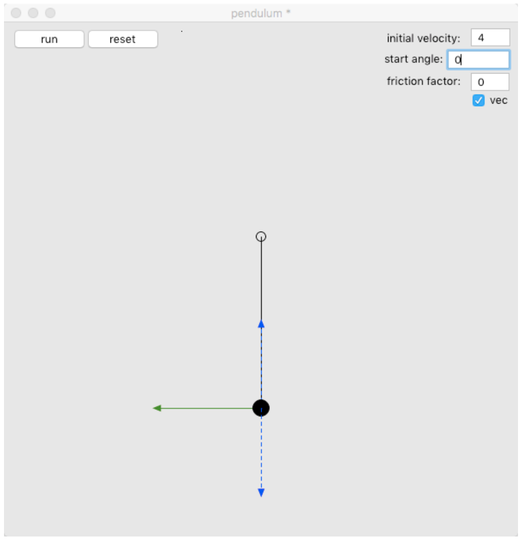
The upwards velocity will not continue to move the bob indefinitely because it is opposed by the tangential gravitation acceleration. As the bob moves upward, its motion slows.

This acceleration on the bob can be called the "restoring force" in this oscillating system.

On the right, the bob has reached its highest point such that the tangential velocity is zero. Since the bob is subject to gravity, it starts its downward motion, as seen on the left below.



initial velocity: 4
start angle: 0
friction factor: 0
✓ vec

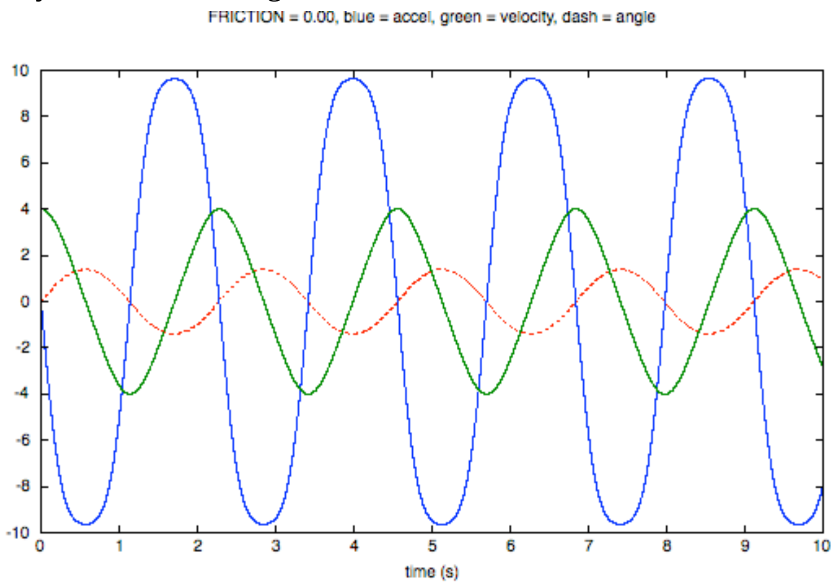


initial velocity: 4
start angle: 0
friction factor: 0
✓ vec

On the right, the bob is moving through the stable steady state position. The tangential acceleration is zero, as it is at the steady state. However, the tangential velocity is not zero. Thus, the bob moves through the steady state and continues to swing to the left.

In order for the system to stay at the steady state, both the tangential acceleration and the tangential velocity must equal zero.

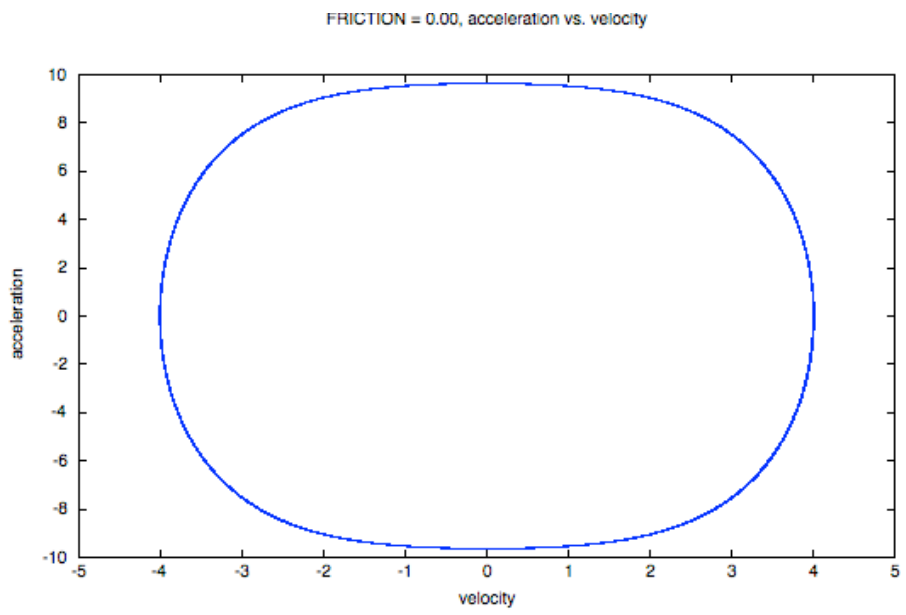
Below is a timeline showing the tangential velocity (green), tangential acceleration (blue), and the angle (dashed) vs. time. The system is oscillating.



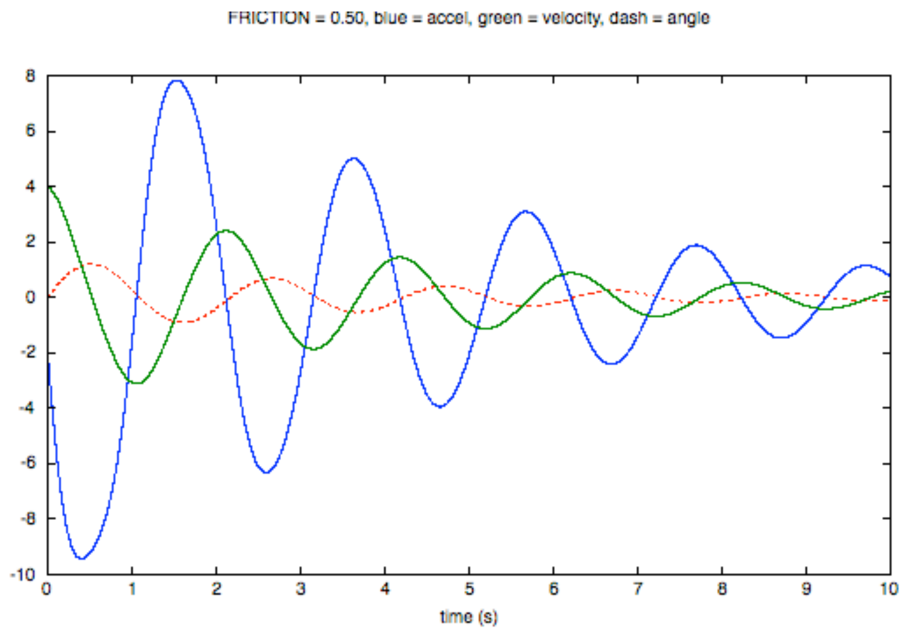
Another way to display the behavior of the system is to plot the acceleration vs. the velocity.

In this "phase curve" plot, the state of the system circulates clockwise about the stable steady state, which is at 0,0 at the center of the plot. This phase curve exhibits a "limit cycle" for these sustained oscillations.

With no friction, the system will continue this behavior forever.

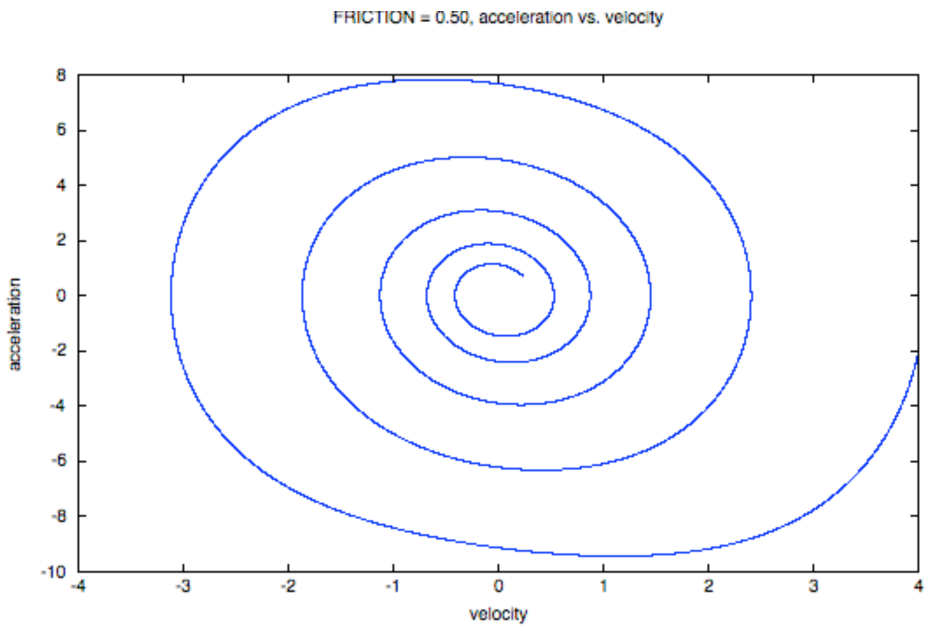


Now we add some friction at the swivel. You can see that the oscillations are damped and decay with time.



Continued next page...

The acceleration vs. velocity phase curve for the system with friction now shows the system spiraling inwards until it will eventually reach, and stay at, the stable steady state at 0,0 at the center of the plot.



Vocabulary

steady state
- stable
- unstable
restoring force
phase curve
limit cycle
damped oscillation
sustained oscillation

Listing of the Matlab program used to generate the plots above (actually done using FreeMat).

Note for students who learned other computer languages: Matlab array indices (subscripts) start at 1, not 0. Array elements are addressed as, e.g., `a(j)` and `a(j,k)` vs. `a[j]` and `a[j][k]` as in C. % starts comments.

```
% pendulum simulation, Richard K. Herz, rich@reactorlab.net, 2017
% Point mass bob connected by massless, rigid rod to central swivel, no air resistance.
% Can add friction at swivel that is proportional to angular velocity.
% For oscillation about small angles there is an approximate analytical solution given
%   at https://en.wikipedia.org/wiki/Pendulum_(mathematics)#Small-angle_approximation
% Here we allow full revolution and use Euler's method to solve the differential equations.
% Because of numerical approximation errors inherent in Euler's method,
% we need to add a small amount of friction to closely approach frictionless behavior.

% R = radius (m) = length of pendulum
% ---- R is a constant (scalar), the following five variables are arrays -----
% B (beta) = angle (radians) from steady-state position, positive to right, neg to left
% t = time (s)
% dBdt = dB/dt = tangential angular velocity (radian/s), pos to right, neg to left
% V = tangential velocity (m/s) = R (circumferential meters / radian) * dB(radian)/dt(s)
% dVdt = dV/dt = tangential acceleration (m/s2), neg to left, pos to right = R * d(dB/dt)/dt
% NOTE: "/" not allowed in variable names, thus, dBdt and dVdt used as variable names
```

```
clear all

g = 9.8; % (m/s2), gravitation acceleraton in vertical direction
R = 1; % (m), radius = length of pendulum

ff = 0.0; % friction factor, larger = more friction
ff = ff + 0.0016; % need some to offset Euler errors, 0.0016-0.003

B(1) = 0; % (radians), initial angle beta, NOTE: Matlab array indexes start at 1
V(1) = 4; % (m/s2), initial tangential velocity

t(1) = 0; % (s), initial time
dt = 0.0005; % (s), time step

% use Euler method to expose logic

% repeat the following for values of array subscript k = 2, 3, 4, ... 20000
for k = 2:20000
  % compute tangential component of acceleration due to gravity
  % given angle at "current" time step k-1
  dVdt(k-1) = - g * sin(B(k-1));
  % compute friction proportional to tangential velocity
  f = ff * V(k-1);
  % adjust tangential acceleration by friction
  dVdt(k-1) = dVdt(k-1) - f;
  % knowing tangential acceleration at "current" time step k-1,
  % compute tangential velocity (m/s) at "new" time step k
  V(k) = V(k-1) + dVdt(k-1) * dt;
  % knowing tangential velocity (m/s), compute angular velocity (radian/s)
  dBdt = V(k-1) / R; % (radian/s) = (m/s) / (circumferential meters/radian)
  % knowing angular velocity (radian/s) at "current" time step k-1,
  % compute angle (radian) at "new" time step k
  B(k) = B(k-1) + dBdt * dt;
  % correct angle if pendulum goes past top in counter-clockwise direction
  if B(k) > pi
    B(k) = -pi + mod(B(k),pi);
  end
  % correct angle if pendulum goes past top in clockwise direction
  if B(k) < -pi
    B(k) = pi + mod(B(k),-pi);
  end
  % update time
  t(k) = t(k-1) + dt;
  % update tangential accel at new time step, k, so array has same length as others for plottng
  dVdt(k) = - g * sin(B(k));
  f = ff * V(k);
  dVdt(k) = dVdt(k) - f;
end

plot(t,B,'r--',t,dVdt,'b',t,V,'g')
tt = sprintf('FRICTION = %0.2f, blue = accel, green = velocity, dash = angle',ff);
title(tt)
xlabel('time (s)')

figure
plot(V, dVdt)
tt = sprintf('FRICTION = %0.2f, acceleration vs. velocity',ff);
title(tt)
ylabel('acceleration')
xlabel('velocity')
```