Simulating a Chain Sliding off a Desktop

Jan Vrbik

The main purpose of this article is to develop an algorithm for simulating a chain sliding off a desktop and to design and demonstrate the corresponding program.

Equations of Motion

Consider a chain consisting of $n+1$ point-like particles of the same mass (equal to 1, by a choice of units), connected by n massless, perfectly flexible, inelastic links of equal length (also equal to 1). The chain is laid on a table top, straight and perpendicular to the edge. Then the first particle is pulled (together with the rest of the chain) gently over the edge of the table. This causes the chain to start sliding down, due to gravity (also of unit magnitude), in a *frictionless* manner [1].

Let us assume now that k particles have already left the table, and that their positions are defined by k angles $\varphi_1, \varphi_2, \ldots, \varphi_k$ by which the first k links deviate from the vertical, and by s, the distance of the last particle to have left the table edge (φ_k is thus the angle of the hanging part of the corresponding link; the rest of it still lies flat on the table). Collectively, these $k+1$ variables are known as *generalized coordinates* [2], as they fully specify the position of every particle.

Now, using rectangular coordinates with the origin at the table's edge, the x axis oriented vertically downward, and the y axis pointing horizontally, away from the table, we can compute the corresponding x and y coordinates of each particle by

$$x_i = \begin{cases} s\cos\varphi_k + \sum_{j=i}^{k-1}\cos\varphi_j & 1 \le i \le k \\ 0 & k+1 \le i \le n+1 \end{cases} \tag{1}$$

and

$$y_i = \begin{cases} s\sin\varphi_k + \sum_{j=i}^{k-1}\sin\varphi_j & 1 \le i \le k \\ s+k-i & k+1 \le i \le n+1 \end{cases} \tag{2}$$

The following program does exactly that.

```
n = 10;
coord[φ_List, s_] := Module[{k = Length[φ]},
Table[
   If[i ≤ k, Sum[If[j == k, s, 1] {Cos[φ[[j]]], Sin[φ[[j]]]},
     {j, i, k}],
 {0, s + k - i}], {i, n + 1}]]
```

The ensuing Lagrangian is the sum of the kinetic energies (that is, $\frac{\dot{x}_i^2+\dot{y}_i^2}{2}$, where a dot means a time derivative) of all $n+1$ particles, minus the sum of their potential energies (which, individually, equal $-x_i$), namely

$$L = \sum_{i=1}^{n+1}\left(\frac{\dot{x}_i^2+\dot{y}_i^2}{2} + x_i\right). \tag{3}$$

This is easily converted to a *Mathematica* function.

```
Lagr[k_] := Module[{xy = coord[Table[φi[t], {i, k}], s[t]]},
Total[Flatten[D[xy, t]^2]] / 2 + Sum[xy[[i, 1]], {i, k}] //
   TrigReduce]
```

The resulting equations of motion are obtained from

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = \frac{\partial L}{\partial q}, \tag{4}$$

where q is, one by one, each of the generalized coordinates. Here is the corresponding program.

```
eqs[k_] := Module[{L = Lagr[k]},
Append[Table[D[D[L, φi'[t]], t] == D[L, φi[t]], {i, k}] //
    TrigReduce,
 D[D[L, s'[t]], t] == D[L, s[t]]]]
```

For any particular set of initial values, namely a list of k angles φ_i (the first argument), their time derivatives (the second argument), distance s (for our purpose set to 0 or, to avoid a technical problem, to a negligibly small value), and the derivative of s (the third argument), we can solve these equations by the following program.

```
step[p_List, pd_List, sd_] := Module[{k = Length[p], sol, T},
sol =
   NDSolve[{eqs[k], Table[{φi[0] == p[[i]], φi'[0] == pd[[i]]},
        {i, k}], s[0] == 10^-5,
 s'[0] == sd}, Append[Table[φi, {i, k}], s], {t, 0, 1/sd},
      SolveDelayed → True][[1]];
T = t /. FindRoot[s[t] == 1 /.sol, {t, 1/sd}];
{Table[φi[T], {i, k}],
     Append[Table[φi'[T], {i, k - 1}],
      φk'[T] - Cos[φk[T]] s'[T]],
 s'[T], (1 - Sin[φk[T]]) s'[T]} /. sol]
```

The program advances the solution until s reaches the value of 1, that is, when the $(k+1)^{\text{st}}$ particle has just left the desktop (technically, it is easier to let it go a bit further and then backtrack to time T defined by $s(T) = 1$). The program returns the new values of the φ_i angles, their derivatives, the horizontal velocity of the $(k+1)^{\text{st}}$ particle (equal to $\dot{s}(T)$; its vertical velocity is zero), and the rate of increase of the length of the k^{th} link, given by $(1 - \sin\varphi_k(T))\,\dot{s}(T)$. Note that the final value of $\dot{\varphi}_k(T)$ has to be modified to $\dot{\varphi}_k(T) - \dot{s}(T)\cos\varphi_k(T)$, since as soon as the $(k+1)^{\text{st}}$ particle leaves the table (even though, at this point, only by an infinitesimal amount), φ_k is no longer measured from the edge of the table; the k^{th} link now begins at this newly freed particle (this does not change the value of φ_k itself, but it does change its derivative; the reader should be able to clearly visualize this). Furthermore, this also means that now the k^{th} and $(k+1)^{\text{st}}$ particles no longer keep their distance fixed (as the last item of the program's output clearly indicates). Before we can continue with the next such step (to pull yet another particle off the desktop), we need to apply some new mechanism to quickly (preferably instantaneously) modify the velocities of these two particles, to make them maintain their fixed distance of 1 unit.

■ Impulsive Solution

To achieve this, we replace the link connecting the last two "free" particles by a *spring* with a large spring constant of λ^2 and an equilibrium length of 1. Furthermore, we have to introduce two extra generalized coordinates X and Y (the rectangular coordinates of the $(k+1)^{\text{st}}$ point). This requires only the following minor modifications of the `coord` and `Lagr` routines (now combined into a single program).

```
comb[k_] := Module[{xy = Table[If[i ≤ k + 1, {X[t], Y[t]} +
Sum[If[j == k, s[t], 1] {Cos[φj[t]], Sin[φj[t]]}, {j, i, k}],
{0, √(X[t]^2 + Y[t]^2) + k - i}], {i, n + 1}]},
(Total[Flatten[D[xy, t]^2]] - λ^2 (s[t] - 1)^2) / 2 +
    Sum[xy[[i, 1]], {i, k + 1}] // TrigReduce]
```

The resulting Lagrangian is then easily converted into the corresponding equations of motion; unfortunately, the resulting solution exhibits undamped oscillations of s (the length of the kth link). To be able to continue with our simulation, we must first dampen these by adding to the corresponding differential equation a large frictional term of $-15\,\lambda\,\dot{s}$ (the coefficient of 15 was found empirically, to approximate critical damping). Adding friction will ultimately result in a small (and inevitable, for all models of this kind) loss of energy.

Then, we magnify the time scale by using $\tau \equiv \lambda\, t$ as a new independent variable, and similarly replace each of the generalized coordinates q by $q_0 + \frac{\hat{q}}{\lambda}$, where q_0 is the corresponding initial value, and $\hat{q}$ thus becomes the new dependent variable on a similarly magnified scale (its initial value is always 0). Finally, we let $\lambda \to \infty$ and keep only λ-proportional terms of each equation (all other terms are, in this limit, discarded). The following program does all this, building the corresponding set of differential equations (yet to be solved).

```
damp[p_List] := Module[{k = Length[p], ode, L}, L = comb[k];
ode = {Table[D[D[L, φi'[t]], t] - D[L, φi[t]], {i, k}],
    D[D[L, s'[t]], t] - D[L, s[t]]
 + 15 λ s'[t], D[D[L, X'[t]], t] - D[L, X[t]],
    D[D[L, Y'[t]], t] - D[L, Y[t]]};
ode =
   ode/λ /. {Derivative[i_][q_][t] → λ^(i-1) Derivative[i][q̂][τ],
φi_[t] :> p[[i]] + φ̂i[τ]/λ, s[t] → 1 + ŝ[τ]/λ, q_[t] -> q̂[τ]/λ} //
    Cancel;
ode = Series[ode, {λ, Infinity, 0}] // Normal // Chop]
```

All we need to do then is to integrate the resulting equations until a fully damped solution is attained (this means that $\hat{s}$ and $\dot{s}$ have reached the value of 0, and the other velocities no longer change). In *real time*, this happens *instantaneously* (a finite change of τ represents only an infinitesimal increase in t, due to the $\lambda \to \infty$ limit); consequently, the values of our generalized coordinates remain fixed (again, a finite change in $\hat{q}$ does not modify the value of q). This is no longer true for the corresponding derivatives—note that

$$\frac{d\,\hat{q}}{d\,\tau} \equiv \frac{d\,q}{d\,t}. \tag{5}$$

The procedure thus changes, instantaneously, only the values of all generalized velocities. The actual solution is carried out by the program called `impulse`.

```
impulse[p_List, pd_List, yd_, sd_] :=
 Module[{k = Length[p], ode, sol},
ode = {damp[p], Table[φ̂ᵢ[0], {i, k}],
     Table[φ̂ᵢ'[0] - pd[[i]], {i, k}],
ŝ[0], ŝ'[0] - sd, X̂[0], X̂'[0], Ŷ[0] - 10^-5, Ŷ'[0] - yd};
ode = Thread[Equal[ode // Flatten, 0]];
sol = NDSolve[ode, {Table[φ̂ᵢ, {i, k}], X̂, Ŷ, ŝ} // Flatten,
      {τ, 0, 100},
 SolveDelayed → True][[1]];
  {Append[p, ArcTan[X̂[τ], Ŷ[τ]]],

                                       X̂[τ] Ŷ'[τ] - X̂'[τ] Ŷ[τ]
      Append[Table[φ̂ᵢ'[τ], {i, k}], ─────────────────────────],
                                           X̂[τ]^2 + Ŷ[τ]^2

 X̂[τ] X̂'[τ] + Ŷ[τ] Ŷ'[τ]
 ─────────────────────────} /. τ → 100 /. sol]
    Sqrt[X̂[τ]^2 + Ŷ[τ]^2]
```

The program takes the output of `step` as its arguments, initializes all variables to 0 (except for $\hat{Y}$, which needs to be made negligibly small) then, based on (5), initializes their derivatives (the program's input), and advances the solution for 100 units of the τ scale (this appears sufficient to reach the desired equilibrium). If needed, the routine could be made more efficient by eliminating the $\hat{\varphi}_i$ as dependent variables (each $\hat{\varphi}_i'$ can be expressed as a linear combination of $\hat{X}'$, $\hat{Y}'$, and s').

■ Simulation

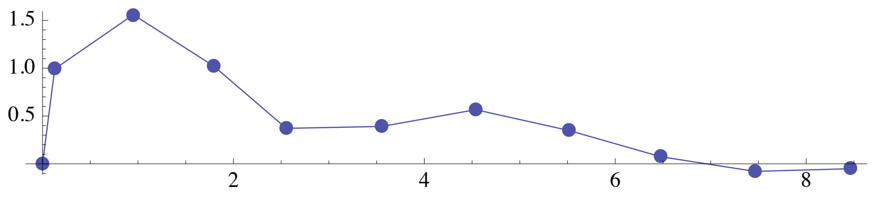
Now we want to simulate a complete motion of such a chain until it loses touch with the table. The first step (until the *second* particle is about to lose contact with the desktop) is very simple; it results in the first particle being suspended vertically (i.e., $\varphi_1 = \dot{\varphi}_1 = 0$) from the desktop's edge, sliding down (and pulling the rest of the chain) at the speed $\sqrt{\frac{2}{n+1}}$ (the chain's kinetic energy must equal the loss of potential energy). After that, we have to alternate applying `impulse` and `step` to the current solution.

$$n = 10;\ \text{chain} = \left\{\{0\},\ \{0\},\ \sqrt{\frac{2}{n+1}},\ \sqrt{\frac{2}{n+1}}\right\};$$

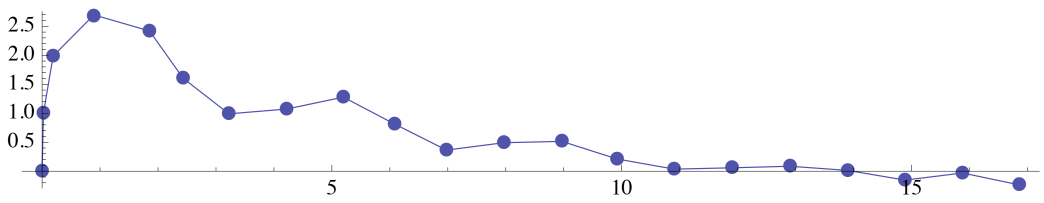
```
Do[chain = Apply[step, Apply[impulse, chain]], {n - 1}]
```

The final shape of the chain (rotated by 90 degrees due to our coordinate system) is then displayed.

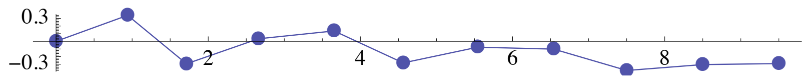
```
ListPlot[Take[coord[chain[[1]], 1], 1 + Length[chain[[1]]]],
 Joined → True, AspectRatio → Automatic,
  PlotMarkers → Automatic]
```



To get a more realistic approximation to a real chain (or rope), one would have to substantially increase the value of *n*. This would require rewriting our programs in a more efficient, task-dedicated manner. Nevertheless, even with the existing program, we can still produce (in several minutes) the following final profile of a 20-link chain.



One can prove that, in the limit as $n \to \infty$, the chain will maintain an inverted L shape until exactly half of it has left the desktop, and only then start building the characteristic bulge of the final solution [3, 4]. This is to a good approximation true even with our 21 points, as we can see by displaying the chain's shape after 10 particles have left the tabletop.



We will leave it up to the reader to explore how (in)sensitive the solution is to the value of the damping constant, how much energy is lost in the impulse part of the solution, and other such interesting issues.

■ References

[1] J. Vrbik, “Chain Sliding off a Table,” *American Journal of Physics*, **61**(3), 1993 pp. 258–261. doi:10.1119/1.17442.

[2] H. Lamb, *Dynamics*, 2nd ed., Cambridge: Cambridge University Press, 1961.

[3] D. Prato and R. J. Gleiser, “Another Look at the Uniform Rope Sliding over the Edge of a Smooth Table,” *American Journal of Physics*, **50**(6), 1982 pp. 536–539. doi:10.1119/1.12817.

[4] J. R. Sanmartin and M. A. Vallejo, “Comment on ‘Another Look at the Uniform Rope Sliding over the Edge of a Smooth Table’,” *American Journal of Physics*, **51**(7), 1983 p. 585. doi:10.1119/1.13170.

Jan Vrbik, “Simulating a Chain Sliding off a Desktop,” *The Mathematica Journal*, 2011.
dx.doi.org/doi:10.3888/tmj.13–3.

About the Author

Jan Vrbik
Department of Mathematics, Brock University
500 Glenridge Ave., St. Catharines
Ontario, Canada, L2S 3A1
jvrbik@brocku.ca