Velocity from Acceleration — Sinusoidal Force

Done in class, January 24, 2025

This is another notebook for you to finish in-class. I have tried to gather the essentials of all the numerical analysis you have done so far into this one place, so you don't have to jump around opening other documents.

Position From Velocity — Theory — Recap

Using the midpoint approximation, the position from velocity procedure boiled down to:

$t_{i+1} = t_i + \Delta t$

$x(t_{i+1}) \approx x(t_i) + v\left(t_i + \frac{\Delta t}{2}\right) \cdot \Delta t$

Constant Acceleration — Problem Description — Recap

In the last class, you created a pleasantly compact and elegant implementation of the solution to this problem: with $\Delta t = 0.4$, $v(t) = 6 \cdot t$, $t_{\text{initial}} = 0.0$, and $t_{\text{final}} = 6.0$ we wanted to apply the above formulas 15 times.

The code that described the problem looked like:

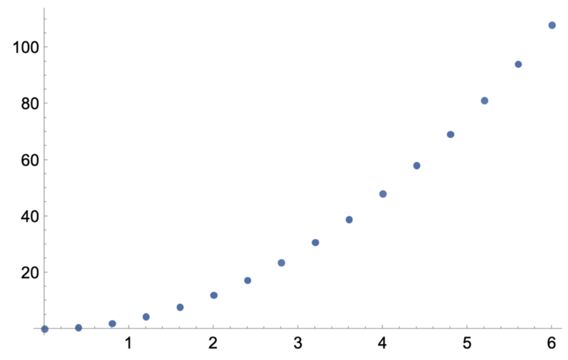
```
v[t_] := 6t
steps = 15;
tInitial = 0.0;
tFinal = 6.0;
deltaT = (tFinal - tInitial)/steps;
```

Constant Acceleration — Implementation — Recap

Because **`NestList`** only takes functions that have one argument, we needed to hide two variables (time and velocity) in a list. Using that idea, a reasonably elegant solution to this problem was simply:

```
procedure[timeAndPosition_] := {
    timeAndPosition[[1]] + deltaT,
    timeAndPosition[[2]] + v[timeAndPosition[[1]] + deltaT/2]deltaT
}
results = NestList[procedure, {0.0, 0.0}, steps];
```

The list plot of results looked like:



Sinusoidal Acceleration — Problem Description

Let’s do sinusoidal acceleration with $F = -10 \sin t$ and $m = 5$. Let’s do 1 1/2 periods of the sine function, with $t_{\text{initial}} = 0.0$ and $t_{\text{final}} = 3\,\pi$. Let’s do 72 steps in the 1 1/2 periods.

```
In[15]:= force[t_] := -10 Sin[t]
         m = 5;
         a[t_] := force[t] / m;
         tInitial = 0.0;
         tFinal = 3 Pi;
         steps = 72;
         deltaT = (tFinal - tInitial) / steps;
```

We have to choose the initial x and v at $t = 0$. I’ll choose $x_{\text{initial}} = 0$ and $v_{\text{initial}} = 2.0$.

```
In[22]:= xInitial = 0.0;
         vInitial = 2.0;
         initialConditions = {tInitial, xInitial, vInitial};
```

You see how I have hidden three variables in a list, using the same strategy as we used for hiding two variables in a list in the last notebook?

Velocity From Acceleration — Theory — Summary

The theory you will implement updates all three of time, position, and velocity:

$$t_{i+1} = t_i + \Delta t$$

$$x(t_{i+1}) \approx x(t_i) + \frac{v(t_i) + v(t_{i+1})}{2} \cdot \Delta t$$

$$v(t_{i+1}) \approx v(t_i) + a\left(t_i + \frac{\Delta t}{2}\right) \cdot \Delta t$$

We are using midpoint in the velocity-from-acceleration formula and we are using trapezoid in the position-from-velocity formula. Since $v(t_{i+1})$ is calculated in the last equation and used in the middle equation, we could rewrite the middle equation using the last equation:

$$x(t_{i+1}) \approx x(t_i) + \frac{v(t_i)+v(t_{i+1})}{2} \cdot \Delta t = x(t_i) + \frac{2\,v(t_i)+a\left(t_i+\frac{\Delta t}{2}\right)\cdot\Delta t}{2} \cdot \Delta t = x(t_i) + \left(v(t_i) + \frac{a\left(t_i+\frac{\Delta t}{2}\right)\cdot\Delta t}{2}\right)\cdot \Delta t$$

Sinusoidal Acceleration — Your Implementation

Everything you need is now summarized above. What remains is to implement the core of the notebook, which is the procedure that will be applied to the initial conditions 72 times. Here is a dummy placeholder for the procedure:

```
In[25]:= procedure[cc_] := {
           cat,
           dog,
           rabbit
          }
         lotsaConditions = NestList[procedure, initialConditions, steps];
```

Once you get a proper version of **`procedure`** defined, the output of **`NestList`** will need a little massaging in order to get it into a form that **`ListPlot`** will accept, and the last couple of lines in this notebook do that and make the plot:

```
In[27]:= lotsaPoints = Transpose[Take[Transpose[lotsaConditions], 2]];
         ListPlot[lotsaPoints]
```