Differential equations

Equations of motion with constant acceleration are based on the differential equations: $v(t) = \frac{dx(t)}{dt}$, $a(t) = \frac{dv(t)}{dt}$. *Integrating for* $a(t) = \text{constant}$*:*

$$a(t) = \text{constant} \tag{1}$$

$$v(t) = \int a\,dt = \boxed{at + v_0} \tag{2}$$

$$x(t) = \int v\,dt = \boxed{\frac{1}{2}at^2 + v_0 t + x_0} \tag{3}$$

For any interval $[t_1, t_2]$ and for $\Delta t = (t_2 - t_1)$...

$$x(t)\Big|_{t_1}^{t_2} = \int_{t_1}^{t_2} v\,dt = \left(\frac{1}{2}a{t_2}^2 + v_0 t_2 + x_0\right) \tag{4}$$

$$-\left(\frac{1}{2}a{t_1}^2 + v_0 t_1 + x_0\right) \tag{5}$$

$$= \frac{1}{2}a\left({t_2}^2 - {t_1}^2\right) + v_0(t_2 - t_1) \tag{6}$$

$$= \frac{1}{2}a(t_2 - t_1)(t_2 + t_1) + v_0(t_2 - t_1) \tag{7}$$

$$= \boxed{\frac{1}{2}a(\Delta t)(t_2 + t_1) + v_0(\Delta t)} \tag{8}$$

Difference equations

To implement a difference equation version, calculate the initial versions of x_i, v_i, *and* t_i *based on* x_0, v_0, *and* t_0. *Then calculate the next versions* x_{i+1}, v_{i+1}, *and* t_{i+1} *based on the current versions* x_i, v_i, *and* t_i *and previous time* t_{i-1}.

$$x_i = x_0 + v_0 t_0 + \frac{1}{2}a{t_0}^2 \tag{9}$$

$$v_i = v_0 + a t_0 \tag{10}$$

$$t_i = t_0 \tag{11}$$

For $\Delta t = (t_i - t_{i-1})$...

$$x_{i+1} = x_i + v_0(\Delta t) + \frac{1}{2}a(\Delta t)(t_i + t_{i-1}) \tag{12}$$

$$v_{i+1} = v_i + a(\Delta t) \tag{13}$$

$$t_{i+1} = t_i + (\Delta t) \tag{14}$$

Code

The following Python code (`accel.py`*) demonstrates the instantaneous (continuous, differential) and recursive (discrete, difference) versions. They yield the same results.*

```
#!/usr/bin/env python3
#
# accel.py
#

# initial values
x0, v0, t0, dt, a, tot = 3, 5, 7, 2, 10, 20

# instantaneous
for t in range(t0, tot + t0 + dt, dt):
    # print('*', t, x0, v0 * t, a * t * t / 2)            # debug print
    x = x0 + v0 * t + a * t * t / 2
    v = v0 + a * t
    print(f"{x:6.1f} {v:6.1f} {t:6.1f}")

print()

# recursive
xi, vi, ti = x0 + v0 * t0 + a * t0 * t0 / 2, v0 + a * t0, t0
print(f"{xi:6.1f} {vi:6.1f} {ti:6.1f}")
for t in range(t0 + dt, tot + t0 + dt, dt):
    # print('*', ti, xi, v0 * dt, a * dt * (t + ti) / 2)  # debug print
    # xi = xi + v0 * dt + a * dt * dt / 2                 # *not* dt ** 2
    # xi = xi + v0 * dt + a * (t * t - ti * ti) / 2       # alternate version
    xi = xi + v0 * dt + a * dt * (t + ti) / 2
    vi = vi + a * dt
    ti = ti + dt
    print(f"{xi:6.1f} {vi:6.1f} {ti:6.1f}")
```

Which results (with these example initial values) in:

```
 283.0   75.0    7.0
 453.0   95.0    9.0
 663.0  115.0   11.0
 913.0  135.0   13.0
1203.0  155.0   15.0
1533.0  175.0   17.0
1903.0  195.0   19.0
2313.0  215.0   21.0
2763.0  235.0   23.0
3253.0  255.0   25.0
3783.0  275.0   27.0

 283.0   75.0    7.0
 453.0   95.0    9.0
 663.0  115.0   11.0
 913.0  135.0   13.0
1203.0  155.0   15.0
1533.0  175.0   17.0
1903.0  195.0   19.0
2313.0  215.0   21.0
2763.0  235.0   23.0
3253.0  255.0   25.0
3783.0  275.0   27.0
>>>
```

The code `xi = xi + v0 * dt + a * dt * dt / 2` (that uses $(\Delta t)^2$ and is commented out) is not correct. The discrete calculation must include the last term of equation (12) to match the continuous calculation.

If $a(t) \neq \text{constant}$, then this analysis does not apply .

David C. Petty • <david_petty@psbma.org> • +1.772.444.2710