FYS-MEK1110 - Mandatory assignment 1

William Dugan

February 6, 2022

1 100m sprint

a Free-body diagram

Figure 1: Plot of driving force F, air resistance F_D and friction force F_f.

b Finding $x(t)$

With $F = 400\text{N}$ and $m = 80\text{kg}$ we can find the acceleration; $a_{net} = \frac{\sum F}{m} = 5\text{ms}^{-2}$. We use the equation for motion in a straight line:

$$x(t) = x_0 + v_0 t + \int_0^t \int_0^{t''} a(t')dt'dt'' \tag{1}$$
$$= 0 + 0 * t + \frac{1}{2}at^2 \tag{2}$$
$$= \frac{5}{2}t^2 \tag{3}$$

c Calculating running time analytically

$$x(t) = 100 = \frac{5}{2}t^2$$
$$t = \sqrt{\frac{2 * 100}{5}}$$
$$t \approx 6.32s$$

d Finding an expression for acceleration

We are given the following equation as a model of the force due to air resistance:

$$D = \frac{1}{2}\rho C_D A_0 (v - w)^2 \tag{4}$$

where ρ is the density of air, C_D is a coefficient of resistance, A_0 is the surface area of the sprinter, v is the runners velocity and w is the velocity of the air. We are given the values $\rho = 1.293\text{kgm}^{-3}$, $A_0 = 0.45\text{m}^2$, $C_D = 1.2$ and $w = 0\text{ms}^{-1}$. To find the acceleration of the sprinter we use Newton's second law.

$$\sum F = ma = F - D \tag{5}$$
$$\implies a = \frac{2F - \rho C_D A_0 (v - w)^2}{2m} \tag{6}$$

e Finding $x(t)$ and $v(t)$ numerically

```
import numpy as np
import matplotlib.pyplot as plt

class Sprint:
    def __init__(self, rho, cd, a0, w, m, F, dt, x0, v0, acc0):
        self.rho = rho
        self.cd = cd
        self.a0 = a0
        self.w = w
        self.m = m
        self.F = F
        self.dt = dt
        self.n = int(np.ceil(10/dt))
        self.x0 = x0
        self.v0 = v0
        self.acc0 = acc0

    def calculate_100m(self):
        t = np.array([0])
        x = np.zeros(self.n); x[0] = self.x0
        v = np.zeros(self.n); v[0] = self.v0
        a = np.zeros(self.n); a[0] = self.acc0

        i_max = 0
        for i in range(self.n-1):
            if x[i-1] > 100:
                break
```

```python
            a[i+1] = (2*self.F - self.rho*self.cd*self.a0*(v[i]**2))/(2*self.m)
            v[i+1] = v[i] + a[i] * self.dt
            x[i+1] = x[i] + v[i+1] * self.dt
            t = np.append(t, (i+1)*self.dt)
            i_max += 1

        self.i_max = i_max
        self.t_max = t[i_max]
        return t[:i_max], x[:i_max], v[:i_max], a[:i_max]

bolt = Sprint(rho, cd, a0, w, m , F, 0.01, x0, v0, acc0)
t, x, v, a = bolt.calculate_100m()
bolt.plot(t, x, v, a)
plt.show()
```

I chose Δt to be 0.01s as i figured 100 points per second would be sufficient given the inaccuracy of our model.

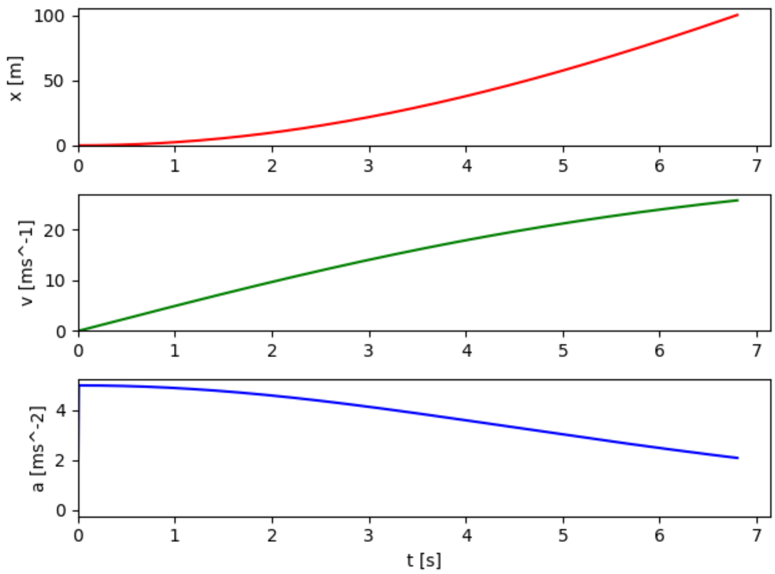

Figure 2: Plot of $x(t)$, $v(t)$ and $a(t)$ for $t \in [0, 6.8]$.

f Calculating running time numerically

```
>>> print(f'Time to run {x[-1]:.2f}m is {t[-1]:.2f}s')
Time to run 100.15m is 6.80s
```

g Theoretical terminal velocity

The terminal velocity is reached when acceleration is 0.

$$
\begin{aligned}
a = 0 &= \frac{2F - \rho C_D A_0 (v - w)^2}{2m} && (7) \\
2F &= \rho C_D A_0 v^2 && (8) \\
v^2 &= \frac{2F}{\rho C_D A_0} && (9) \\
v &= \sqrt{\frac{2F}{\rho C_D A_0}} && (10)
\end{aligned}
$$

h Numerical terminal velocity

I add the following method to the previously defined class **Sprinter**:

```
def terminal_velocity(self):
    t, x, v, a = self.calculate_100m()
    self.dt = 1  # Scaling dt to reduce computing time
    i = self.i_max-1
    while a[i] > 0.01:
        a = np.append(a, (2*self.F - self.rho*self.cd*self.a0*(v[i]**2)) \
                        / (2*self.m))
        v = np.append(v, v[i] + a[i] * self.dt)
        x = np.append(x, x[i] + v[i+1] * self.dt)
        t = np.append(t, (i+1)*self.dt)
        i += 1

    return v[-1]

vt = bolt.terminal_velocity()
print(f'Terminal velocity: {vt:.2f}m/s')
```

```
Terminal velocity: 33.82m/s
```

This value is far above what is achievable by any human being.

i Improved terminal velocity

We are given

$$F_D = F - f_v v \tag{11}$$

with values $F = 400\text{N}$ and $f_v = 25.8\text{Nsm}^{-1}$. Since we can neglect air resistance in this task we achieve the terminal velocity when $F_D = 0$.

$$
\begin{aligned}
F_D = 0 &= F - f_v v \\
F &= f_v v \\
v &= \frac{F}{f_v} = \frac{400}{25.8} \approx 15.5 ms^{-1}
\end{aligned}
$$

j Improved force model

```python
class Sprint2(Sprint):
    def __init__(self, rho, cd, a0, w, m, F, dt, x0, v0, acc0, fc, tc, fv):
        super().__init__(rho, cd, a0, w, m, F, dt, x0, v0, acc0)
        self.fc = fc
        self.tc = tc
        self.fv = fv

    def calculate_100m(self):
        t = np.array([0])
        x = np.zeros(self.n); x[0] = self.x0
        v = np.zeros(self.n); v[0] = self.v0
        a = np.zeros(self.n); a[0] = self.acc0

        i_max = 0
        for i in range(self.n-1):
            if x[i-1] > 100:
                break

            D = -0.5*self.rho*self.cd*self.a0 \
                *(1-0.25*np.exp(-(t[i]/self.tc)**2))*(v[i]-self.w)**2
            F_net = self.F + self.fc*np.exp(-(t[i]/self.tc)**2) \
                    - self.fv*v[i] + D

            a[i+1] = (F_net/self.m)
            v[i+1] = v[i] + a[i] * self.dt
            x[i+1] = x[i] + v[i+1] * self.dt
            t = np.append(t, (i+1)*self.dt)
            i_max += 1

        self.i_max = i_max
        self.t_max = t[i_max]
        return t[:i_max], x[:i_max], v[:i_max], a[:i_max]

fc = 488        # N
tc = 0.67       # s
fv = 25.8       # Ns/m

bolt2 = Sprint2(rho, cd, a0, w, m , F, 0.01, x0, v0, acc0, fc, tc, fv)
t, x, v, a = bolt2.calculate_100m()
bolt2.plot(t, x, v, a)
plt.show()
print(f'Time to run {x[-1]:.2f}m is {t[-1]:.2f}s')
```

```
Time to run 100.03m is 9.31s
```

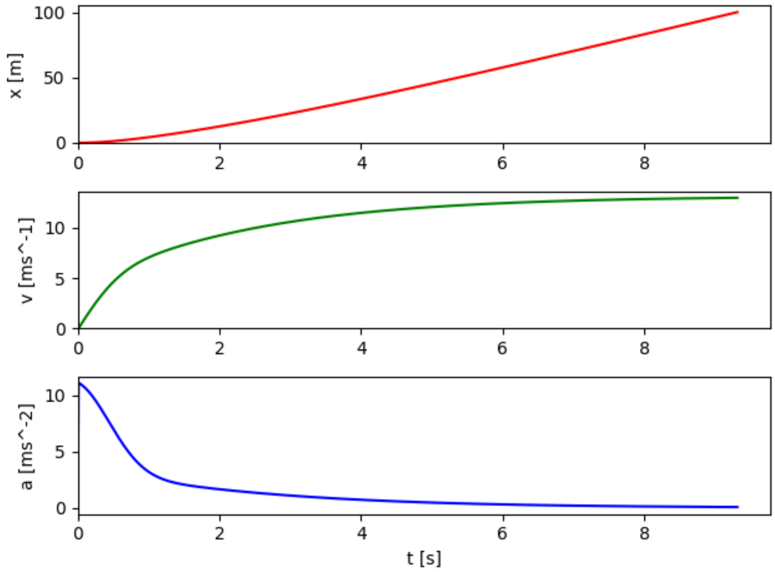

Figure 3: Plot of $x(t)$, $v(t)$ and $a(t)$ for $t \in [0, 9.3]$ with improved force model.

k Comparison of forces

```
class Forces(Sprint2):
    def forces(self):
        t = np.array([0])
        x = np.zeros(self.n); x[0] = self.x0
        v = np.zeros(self.n); v[0] = self.v0
        a = np.zeros(self.n); a[0] = self.acc0

        F = np.full(self.n, self.F)
        Fc = np.zeros(self.n)
        Fv = np.zeros(self.n)
        D = np.zeros(self.n)

        i_max = 0
        for i in range(self.n-1):
            if x[i-1] > 100:
                break

            Fc[i+1] = self.fc*np.exp(-(t[i]/self.tc)**2)
            Fv[i+1] = -self.fv*v[i]
            D[i+1] = -0.5*self.rho*self.cd*self.a0 \
                     *(1-0.25*np.exp(-(t[i]/self.tc)**2))*(v[i]-self.w)**2

            F_net = self.F + Fc[i] + Fv[i] + D[i]

            a[i+1] = (F_net/self.m)
            v[i+1] = v[i] + a[i] * self.dt
            x[i+1] = x[i] + v[i+1] * self.dt
            t = np.append(t, (i+1)*self.dt)
            i_max += 1

        self.i_max = i_max
```

```
        self.t_max = t[i_max]
        return t[:i_max], F[:i_max], Fc[:i_max], Fv[:i_max], D[:i_max]

bolt3 = Forces(rho, cd, a0, w, m , F, 0.01, x0, v0, acc0, fc, tc, fv)
t, F_, Fc, Fv, D = bolt3.forces()
plt.plot(t, F_, t, Fc, t, Fv, t, D)
plt.legend(['F - konstant kraft', 'Fc - sammenkrypning', \
            'Fv - fysisk begrensning', 'D - luftmotstand'], \
            loc='best', bbox_to_anchor=(0.5, 0.3, 0.5, 0.5))
plt.xlim(0)
plt.ylabel('Force [N]')
plt.xlabel('Time [s]')
plt.show()
```

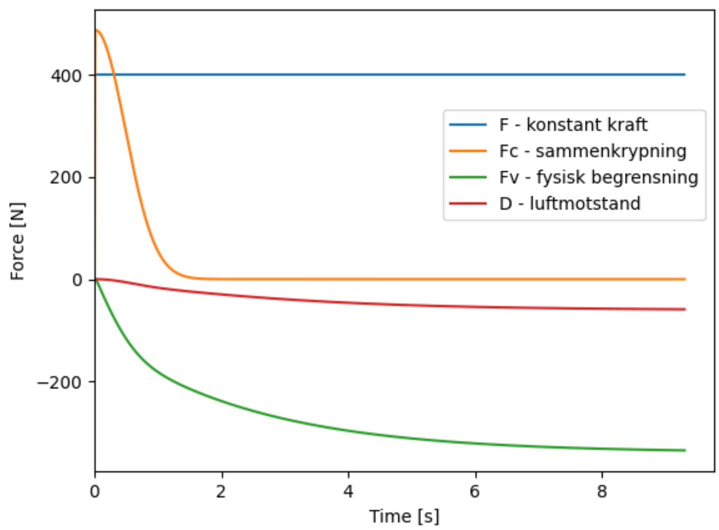

Figure 4: Plot of F, F_c, F_v and D for $t \in [0, 9.3]$ with improved force model.

The force due to crouching is rapidly decreasing at the start, and the physical limitations plays a larger role at the latter half of the race.

1 Running times with $\pm 1.0\text{ms}^{-1}$ wind

```
for wind in (-1.0, 1.0):
    runner = Sprint2(rho, cd, a0, wind, m , F, 0.01, x0, v0, acc0, fc, tc, fv)
    t, x, v, a = runner.calculate_100m()
    print(f'Time to run {x[-1]:.2f}m with {wind}m/s wind is {t[-1]:.2f}s')
```

```
Time to run 100.06m with -1.0m/s wind is 9.43s
Time to run 100.11m with 1.0m/s wind is 9.21s
```