

```
npoints = 5;
```

```
Δt tmax / (npoints - 1);
```

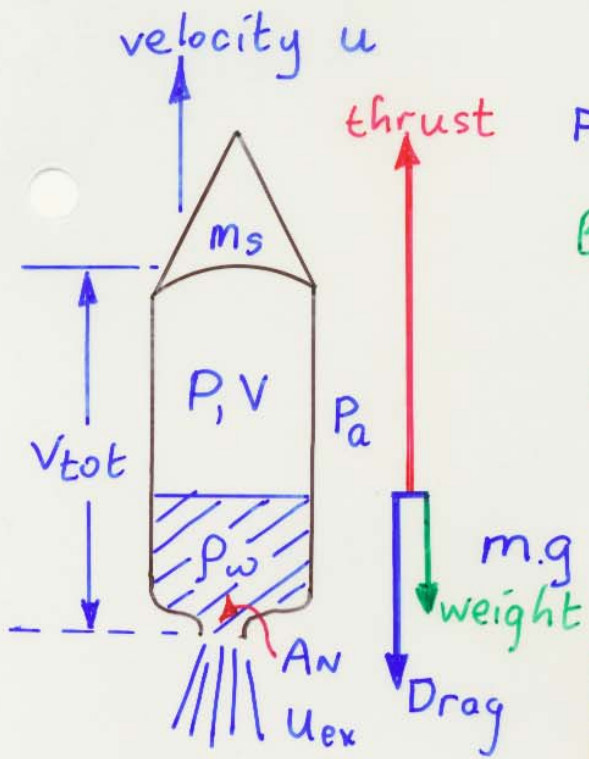
```
for (int i = 0; i < npoints; i++) {
```

```
    t = i * Δt;
```

```
    u = myRocket.find_u(t);
```

```
    cout << t << u;
```

```
}
```



$$F_{\text{thrust}} = \dot{m} u_{\text{ex}} = \rho_w \cdot A_n \cdot u_{\text{ex}}^2$$

$$\text{Bernoulli: } \frac{P - P_a}{\rho_w} = \frac{u_{\text{ex}}^2}{2}$$

thus:

$$F_{\text{thrust}} = 2(P - P_a) A_n$$

$$F_{\text{drag}} = \frac{1}{2} \rho_{\text{air}} C_d A_{\text{bottle}} u^2$$

Newton's Second Law

$$m \frac{du}{dt} = F_{\text{thrust}} - F_{\text{drag}} - mg$$

↑
upward acceleration
a

$$m = m_s + \rho_w (V_{\text{tot}} - V)$$

$$\text{acceleration } a = \frac{du}{dt} = \frac{F_{\text{thrust}} - F_{\text{drag}}}{m} - g$$

however, F_{drag} contains u

$$u = u + \Delta t \cdot a$$

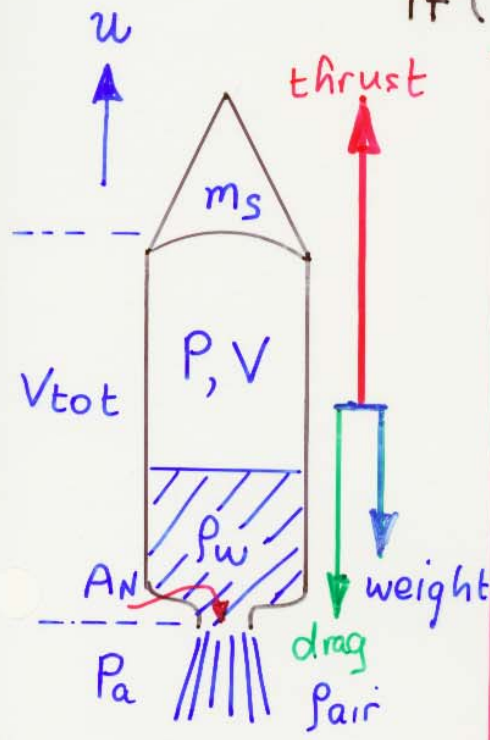
find_u (t_max)

n = ? $\Delta t = \frac{t_{max}}{n}$

$u = \emptyset; V = V_0;$

for (int i = 1; i <= n; i++) {

if (V < V_tot) { // thrust phase



$$m = m_s + \rho_w (V_{tot} - V)$$

$$P = P_0 \left(\frac{V_0}{V} \right)^k$$

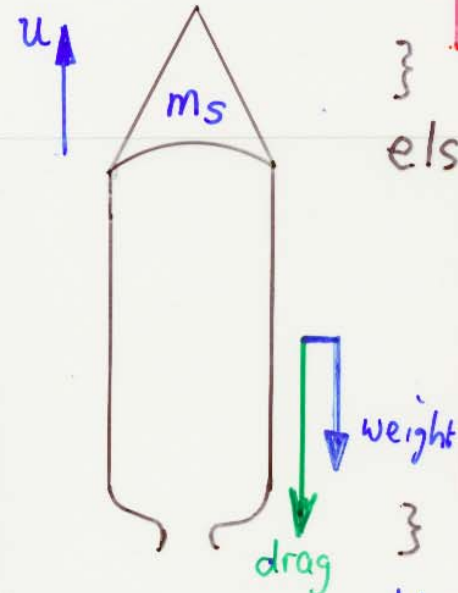
$$V = V + \Delta t \cdot A_n \sqrt{\frac{2(P - P_a)}{\rho_w}}$$

$$F_{thrust} = 2(P - P_a) A_n$$

$$F_{drag} = \frac{1}{2} \rho_{air} C_d A_{bottle} u |u|$$

$$\frac{du}{dt} = a = \frac{(F_{thrust} - F_{drag})}{m} - g$$

} else { // projectile phase



$$F_{drag} = \frac{1}{2} \rho_{air} C_d A_{bottle} u |u|$$

$$\frac{du}{dt} = a = \frac{(-F_{drag})}{m_s} - g$$

$u = u + \Delta t \cdot a$

} return u →