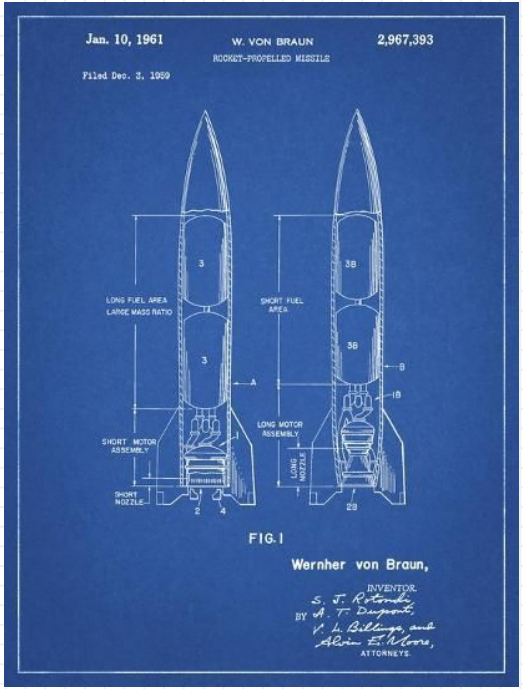
ROCKET OPTIMIZATION: PHYS 100

Ibrahim Al-Akash

MISSION: Determine an Optimal Rocket Design

Optimization Parameters:

- Cost
- Distance
- Flight Time



Methodology

PHASE 1: TEST PRESET DESIGNS

I will simulate previous rocket models and determine the optimal rocket meeting my criteria based on simulation results. This is analogous to the "discretization" we do on homework problems.

PHASE 2: GENERATE DESIGN

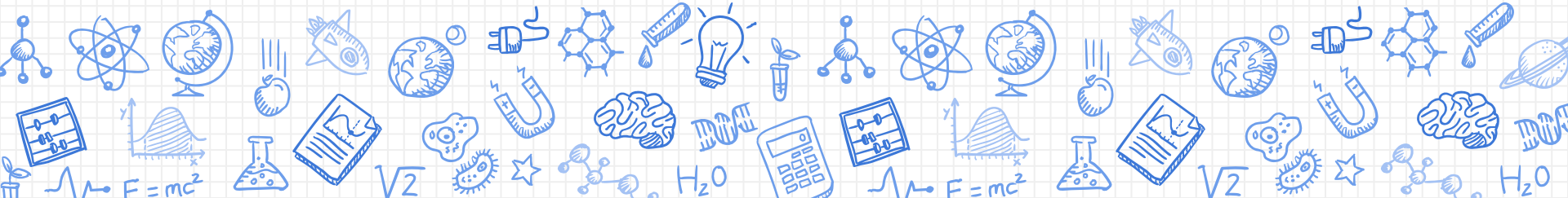
I will generate multiple designs with differing composition and rocket shapes. Iterative testing will produce the optimal rocket. This is analogous to the "analytical approach" we do on homework problems.

Considerations:

- Shape effects on drag (fins and nose cone)
- Fuel burn rate
- Thrust and lift
- Rocket mass and propellant mass
- Center of gravity and center of pressure
- Fin aerodynamics
- Development cost
- Atmospheric variation with altitude*
- Moon's gravity*

Phase 1

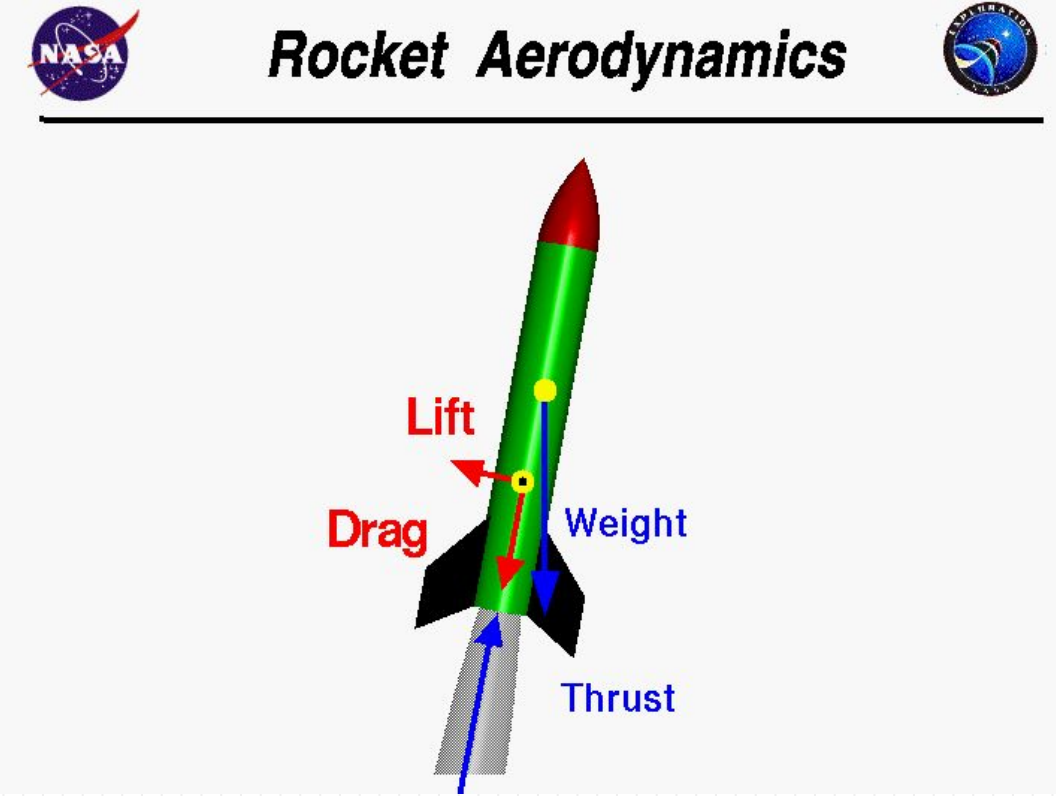
Preset Design Testing



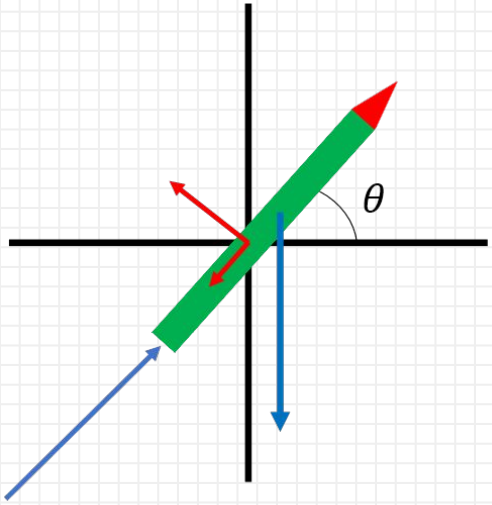
Assumptions

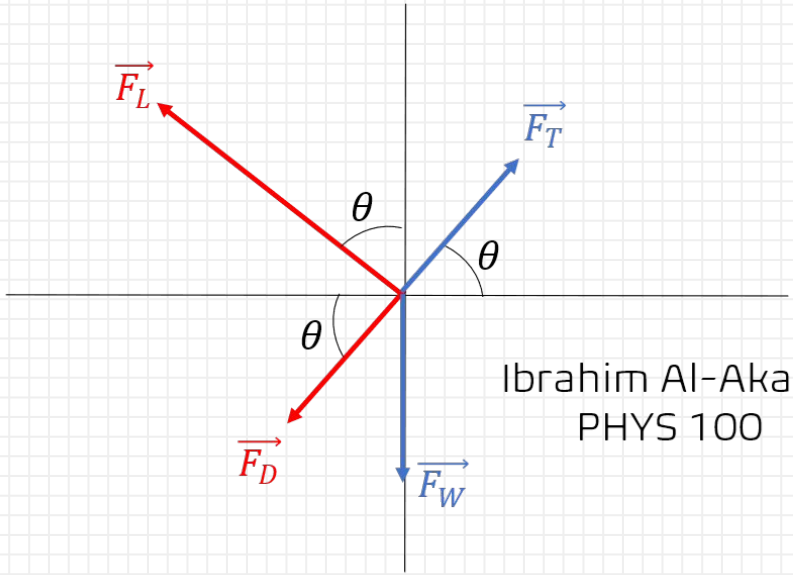
- ✘ Constant burn
- ✘ Drag coefficients from NASA website will be used for simulation
- ✘ Only nose cone and body tube shape will be used to determine drag
- ✘ Carry a 3000 kg payload
- ✘ Launched in identical environments and locations
- ✘ Same launch angle, all launch at 35 degrees - no guidance systems are used
- ✘ Uncontrolled landings to maximize flight speed
- ✘ Simulations are in 2D
- ✘ Rotational motion is neglected
- ✘ Simplified costs
- ✘ Cylindrical shape for body
- ✘ Invariable thrust
- ✘ All forces act on center of gravity (center of pressure neglected)
- ✘ Bullet-shaped nose cone is used on all rockets
- ✘ Same engine

Rocket Aerodynamics Physics



Rocket Free Body Diagram



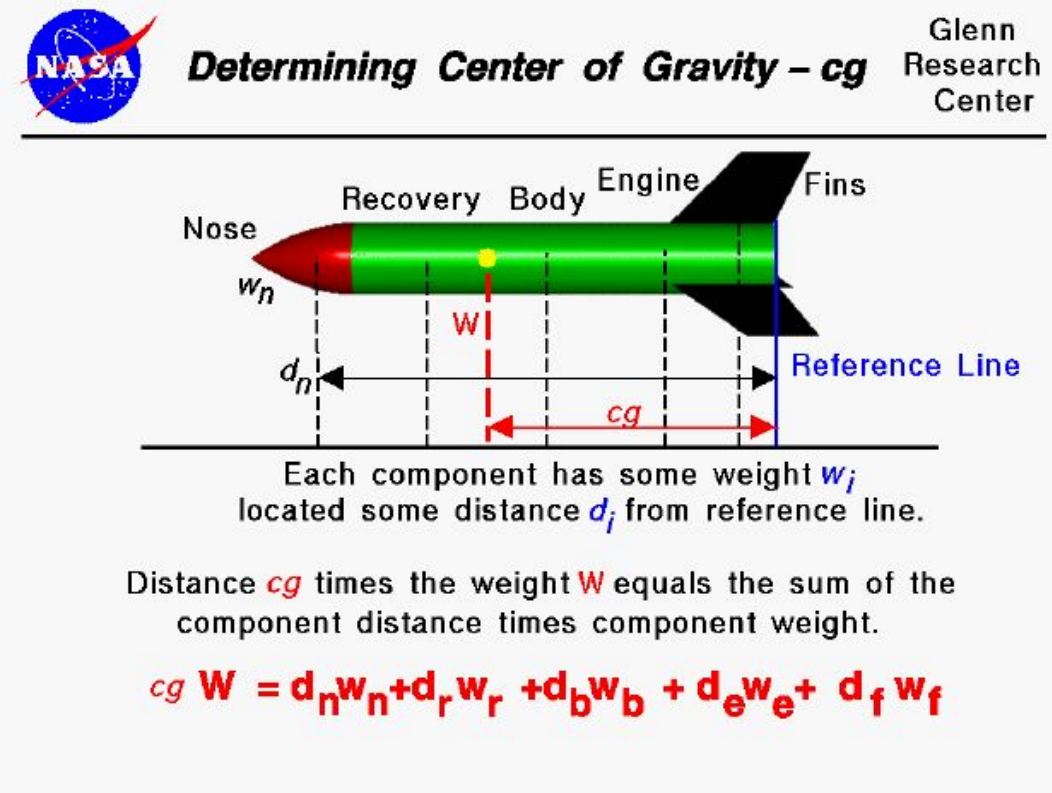


Ibrahim Al-Akash
PHYS 100

$$\sum F_x = F_T \cos(\theta) - F_D \cos(\theta) - F_L \sin(\theta)$$

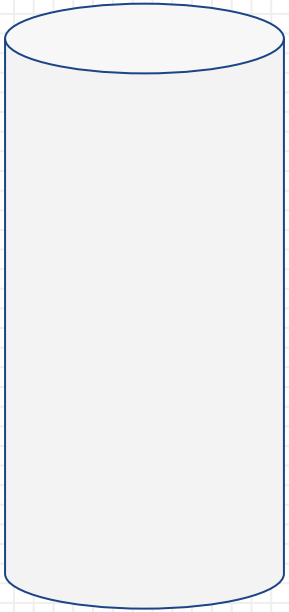
$$\sum F_y = F_T \sin(\theta) - F_D \sin(\theta) + F_L \cos(\theta) - mg$$

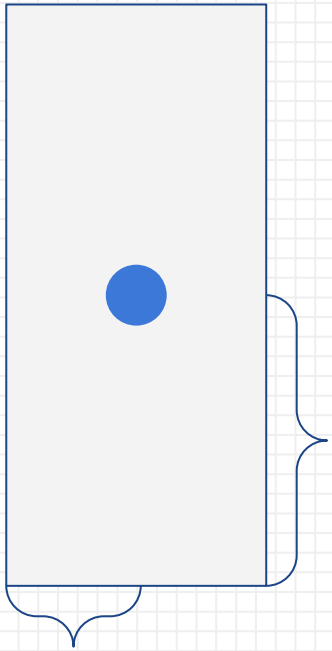
Center of Gravity



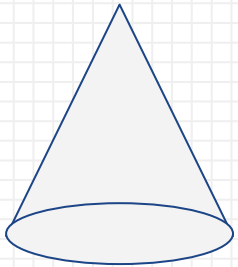
Center of Gravity Approximations

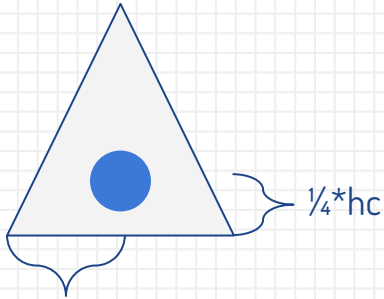
Body



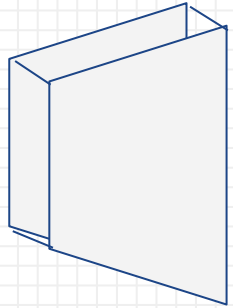


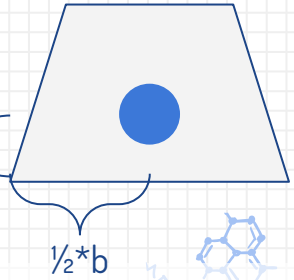
Nose Cone





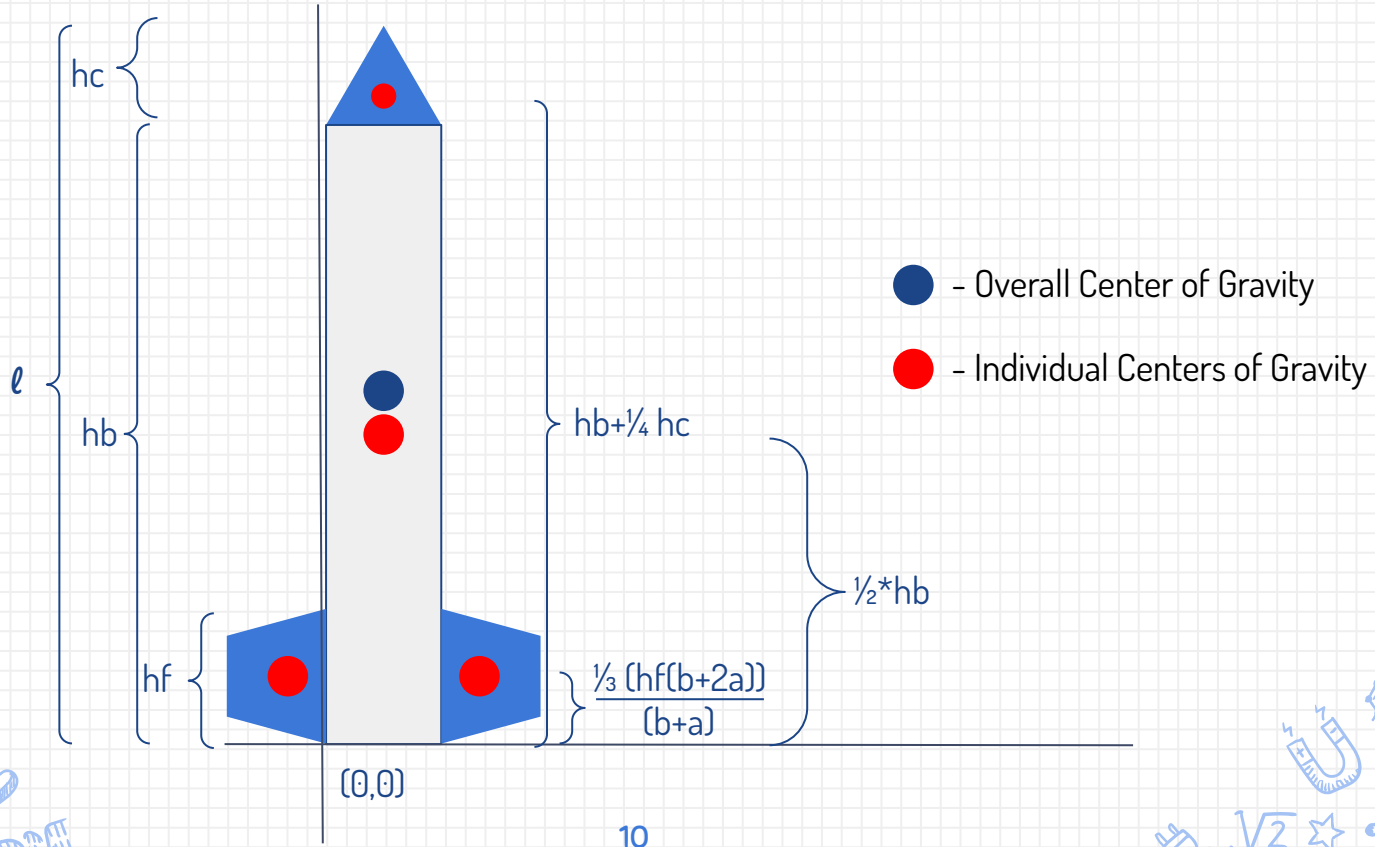
Fin





*Centers of mass of fins cancel out

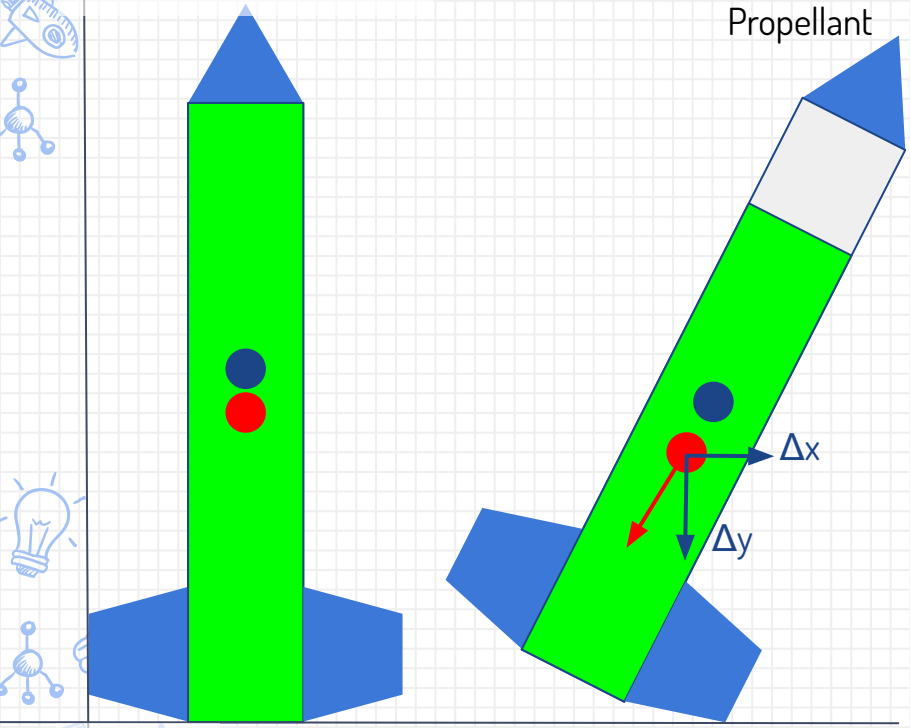
Center of Gravity Approximations



Center of Gravity Considerations

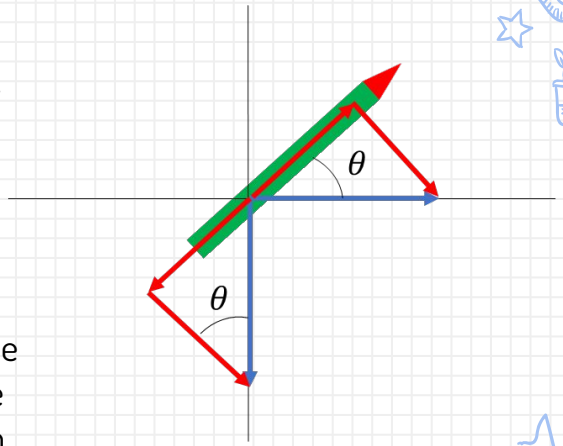
Initial Orientation/ Full Propellant

New Orientation/ Used Propellant



As propellant is used, center of gravity is lowered.

Due to rocket's new angle, the center of gravity vector must be transformed to move only in the y direction with respect to the rocket.

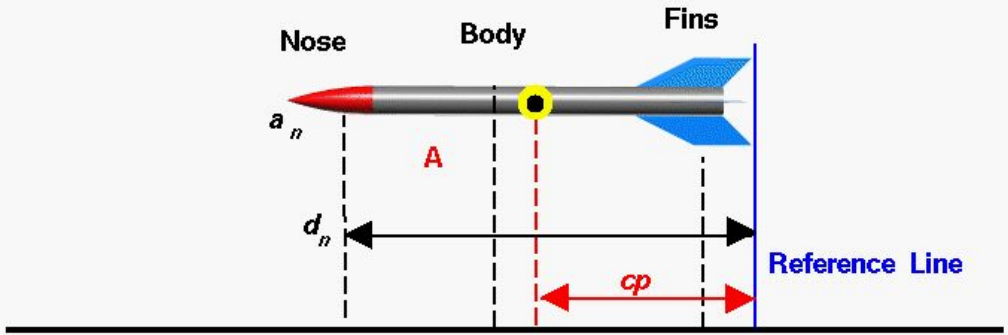


$$\Delta x_{adj} = \Delta x \sin(\theta) + \Delta y \cos(\theta)$$
$$\Delta y_{adj} = \Delta x \cos(\theta) - \Delta y \sin(\theta)$$

- Amount of Propellant
- Overall Center of Gravity
- Body Center of Gravity

Center of Pressure

Determining Center of Pressure – cp
(simplified)

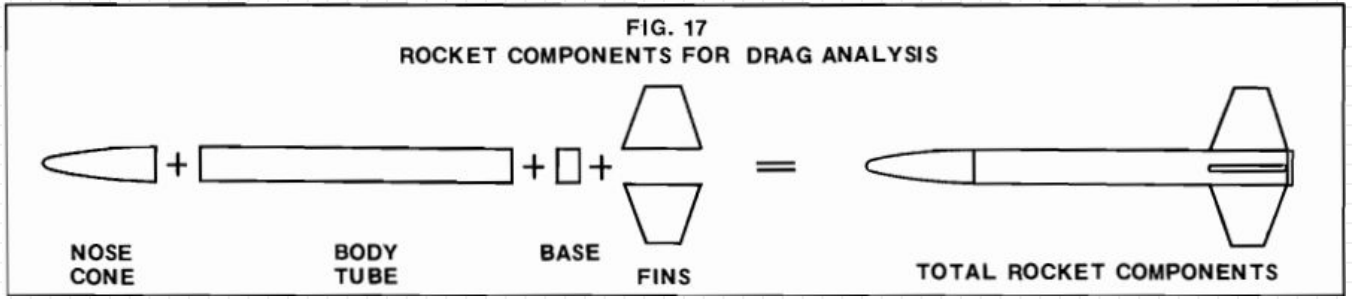


Each component has some area a_i
located some distance d_i from reference line.

Distance *cp* times the area A equals the sum of the component distance times area.

$$cp\ A = d_n a_n + d_b a_b + d_f a_f$$

Drag Equation



$$R = \frac{1}{2} D \rho A v^2$$

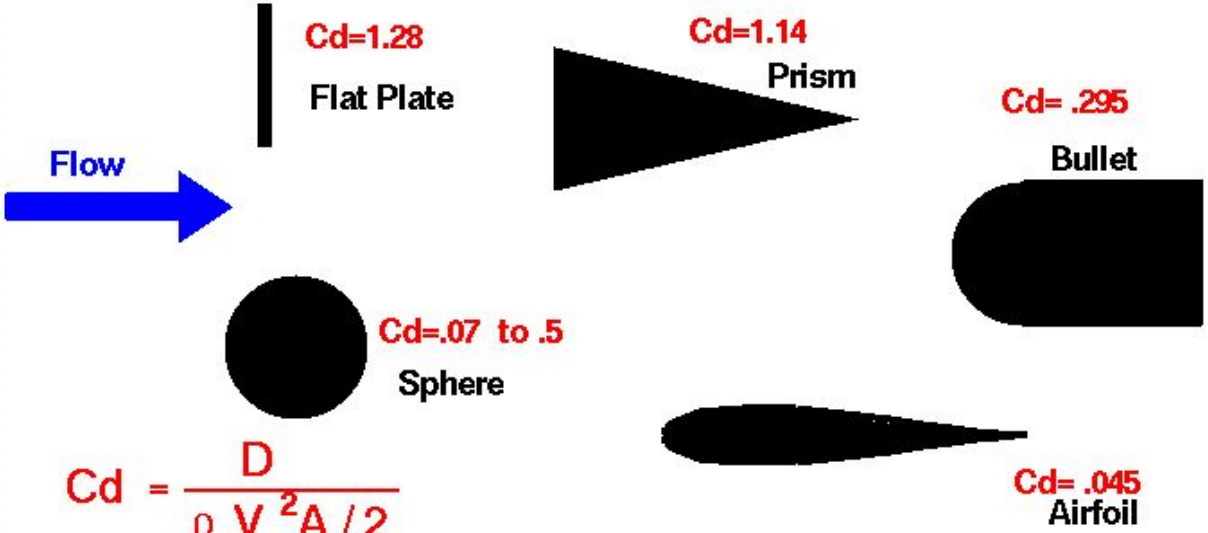
$$C_{D_O} = C_{D_N} + C_{D_{BT}} + C_{D_B} + C_{D_F} + C_{D_{int}} + C_{D_{LL}}$$

Nose Cone Drag + Body Tube Drag + Base Drag + Fin Drag Equal Total Component Drag

Shape Effects on Drag

Glenn Research Center

The shape of an object has a very great effect on the amount of drag.



$$Cd = \frac{D}{\rho V^2 A / 2}$$

A = frontal area

All objects have the same frontal area.

Lift Equation

The Lift Equation

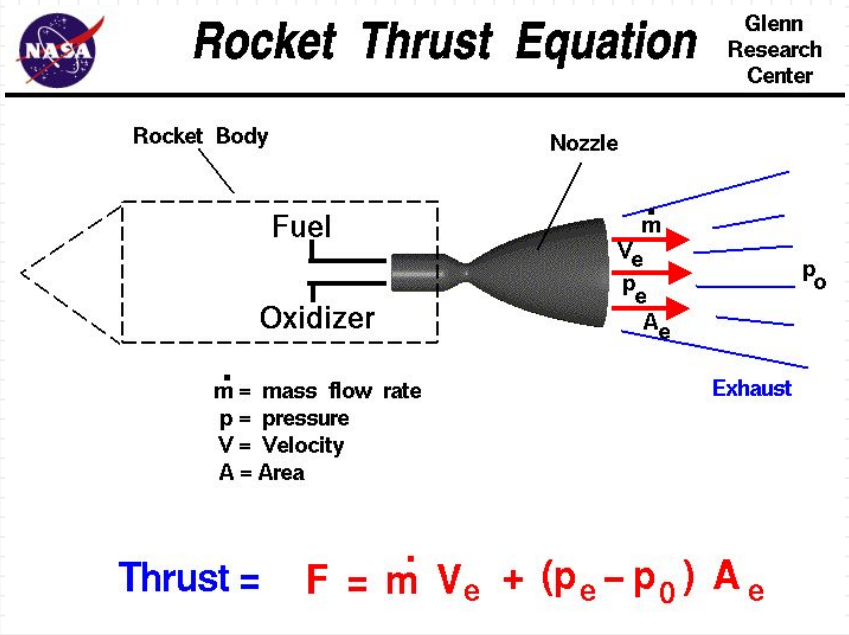
$$L = Cl \frac{\rho V^2 A}{2}$$

Lift = coefficient x $\frac{\text{density x velocity squared}}{\text{two}}$ x wing area

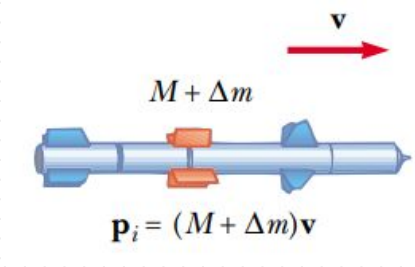
Coefficient Cl contains all the complex dependencies and is usually determined experimentally.

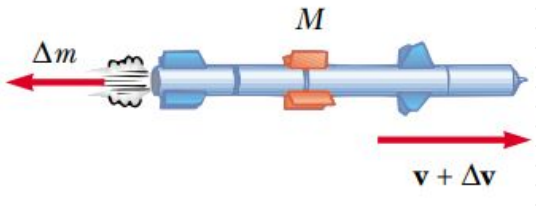
Thrust Equation

$$\text{Thrust} = M\frac{dv}{dt} = \left| v_e \frac{dM}{dt} \right|$$



Rocket Equation





$$(M + \Delta m)v = M(v + \Delta v) + \Delta m(v - v_e)$$

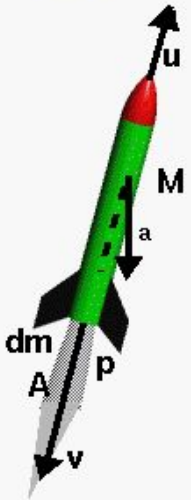
$$M\,\Delta v = v_e\,\Delta m$$

$$M\,dv = v_e\,dm = -v_e\,dM$$

Tsiolkovsky Equation

$$v_f - v_i = v_e \ln\left(\frac{M_i}{M_f}\right)$$

Ideal Rocket Equation



M = instantaneous mass of rocket
u = velocity of rocket
v = exhaust velocity
A = exhaust area
p = exhaust pressure
p_0 = atmospheric pressure

In time increment dt, exhausted mass = dm $\quad dm = \dot{m}\, dt$

Change in momentum of system = $M\, du - dm\, v$

Force on system = $(p - p_0)\, A - M g \cos a$ (neglect drag)

Change in momentum = Impulse = Force dt

$$M\, du - dm\, v = [(p - p_0)\, A - M g \cos a]\, dt$$

$$M\, du = [(p - p_0)\, A + \dot{m} v]\, dt \quad \text{(neglect weight)}$$

V_{eq} = equivalent exhaust velocity = $\dfrac{(p - p_0)\, A}{\dot{m}} + v$

$$M\, du = V_{eq} \dot{m}\, dt = -V_{eq}\, dM$$

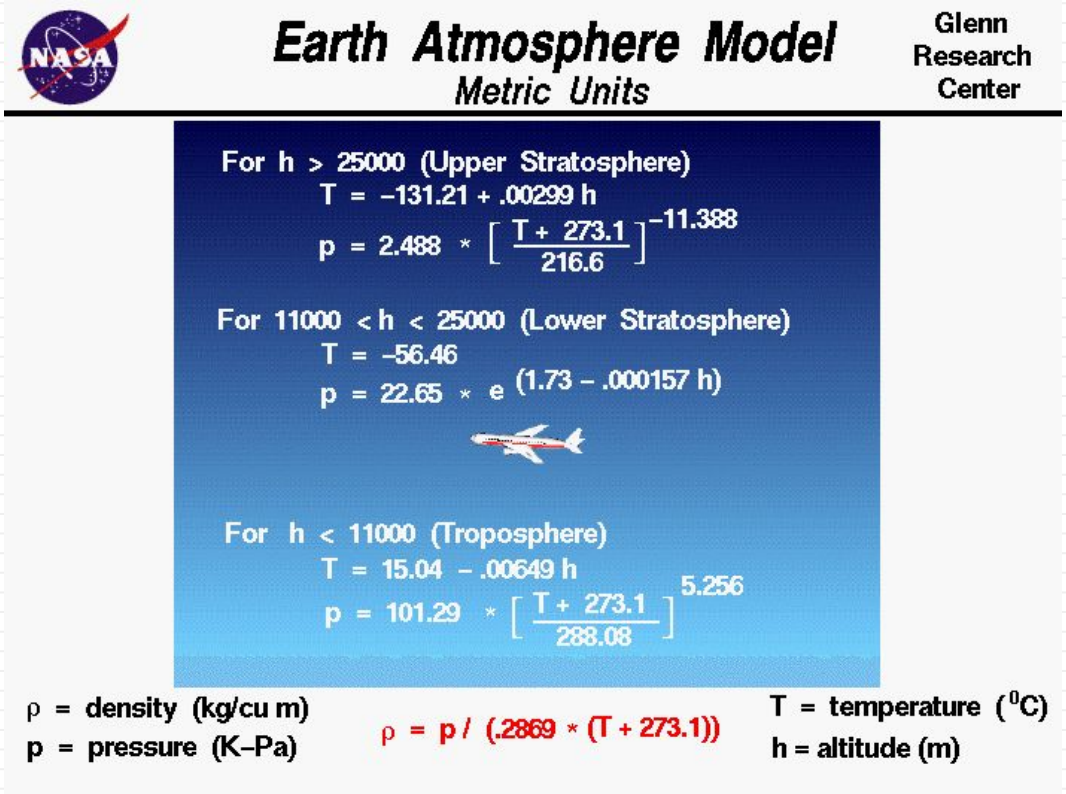
$$du = -V_{eq} \frac{dM}{M}$$

$$\Delta u = -V_{eq} \ln (M) \Big|_{mf}^{me}$$

MR = propellant mass ratio = $\dfrac{mf}{me}$

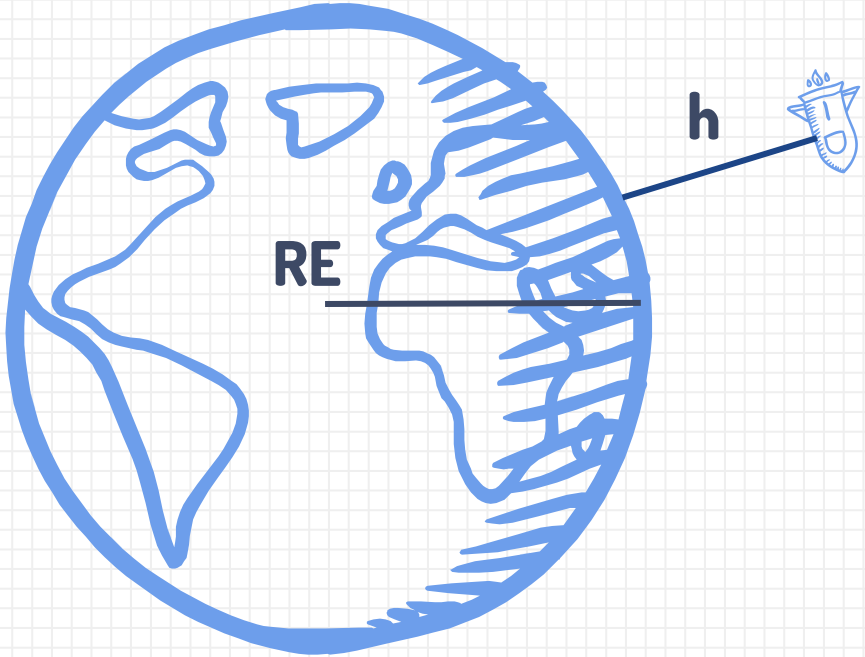
$$\Delta u = V_{eq} \ln \left(\frac{mf}{me}\right) = V_{eq} \ln MR = Isp\, g_o \ln MR$$

Atmospheric Density Equation



Gravitational Constant Equation

$$g = \frac{GM_E}{r^2} = \frac{GM_E}{(R_E + h)^2}$$



Flight Sim Algorithm Overview

1. Start with initial position at sea level with initial angle input
2. Calculate physical properties of rocket:
    a. Burn time = propellant mass (kg) / burn rate (kg/s)
    b. Propellant mass ratio = propellant mass (kg) / empty mass (kg)
    c. Center of gravity
3. Launch rocket
    a. Calculate atmospheric density and gravitational constant at instant
    b. Calculate thrust, drag, lift, and weight forces at instant
    c. Find acceleration
    d. Find velocity with integration
    e. Find position with integration
    f. Update mass of rocket and propellant mass
    g. Loop until burn time is reached, then set thrust to zero
4. Model impact
    a. Redo the previous loop until the rocket crashes into ground

Previous Rocket Designs

Specifications – CSS3 (DF-4) [China]

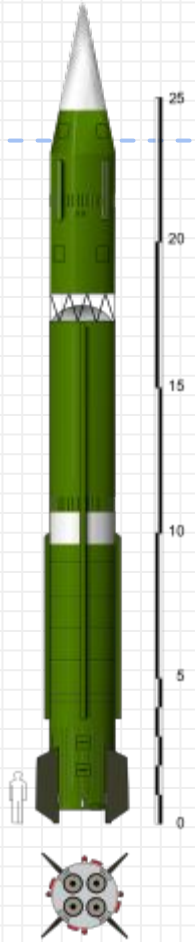
Rocket Length: 29 m

Rocket Weight: 82 t

Propellant Weight: 80 t

Fuel: red fuming nitric acid [RFNA] and unsymmetrical dimethylhydrazine [UDMH]

Diameter: 2.25 m



Specifications – Hwasong 14 [North Korea]

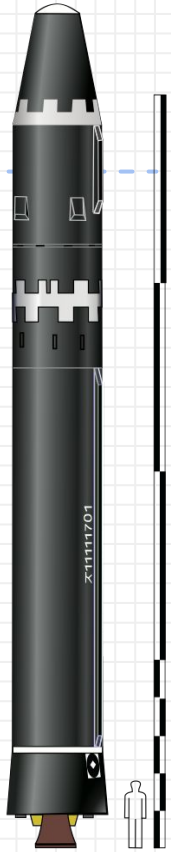
Rocket Length: 19.5 m

Rocket Weight: 33.8 t

Propellant Weight: 30.0 t

Fuel: Unsymmetrical dimethylhydrazine [UDMH]

Diameter: 1.70 m



Specifications – SS-18 (SATAN) [Russia]

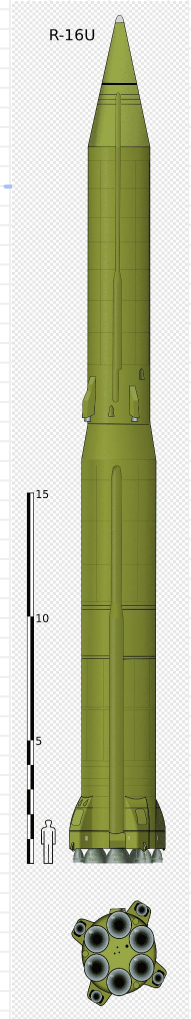
Rocket Length: 37.25 m

Rocket Weight: 211.1 t

Propellant Weight: 188 t

Fuel: Unsymmetrical dimethylhydrazine [UDMH]

Diameter: 3 m



Specifications – SS-19 (UR-100N) [Russia]

Rocket Length: 24.0 m

Rocket Weight: 105.6 t

Propellant Weight: 93.1 t

Fuel: Unsymmetrical dimethylhydrazine [UDMH]

Diameter: 2.50 m

UR-

Specifications – SS-27 (SARMAT) [Russia]

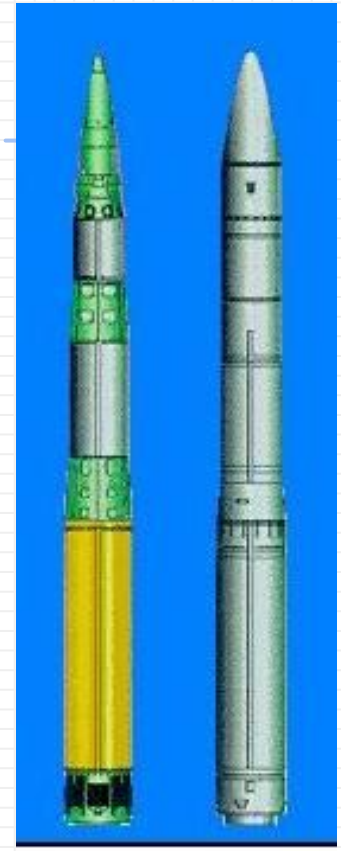
Rocket Length: 22.7 m

Rocket Weight: 47.2 t

Propellant Weight: 28.6 t

Fuel: Unsymmetrical dimethylhydrazine [UDMH]

Diameter: 1.95 m



Specifications – Minuteman III [United States]

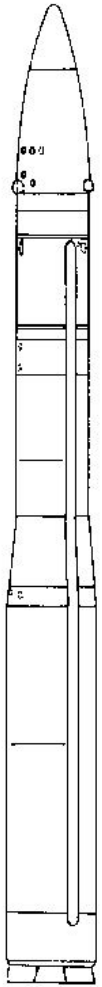
Rocket Length: 18.23 m

Rocket Weight: 36.03 t

Propellant Weight: 30.0 t

Fuel: Ammonium perchlorate composite propellant [APCP] (Isp = 200s, Density: 2g/cc, Cost: $11.30/kg)

Diameter: 1.67 m



Liftoff!

Code

```python
def atmospheric_density(altitude):
    '''
    This function finds the atmospheric density in kg/m^3 at a certain altitude.
    Parameters:
        - Altitude = Altitude in m
    '''
    if altitude < 11000:
        temperature = 15.04 - 0.00649*altitude
        pressure = 101.29 * ((temperature+273.15)/288.08)**5.256
    elif altitude < 25000:
        temperature = -56.46
        pressure = 22.65 * math.exp(1.73 - 0.000157*altitude)
    else:
        temperature = -131.21 + 0.00299 * altitude
        pressure = 2.488 * ((temperature + 273.15)/216.6)**(-11.388)
    rho = pressure / (0.2869 * (temperature + 273.15))
    return rho
```

```python
def gravitational_constant(altitude):
    '''
    This function finds the gravitational constant g0 in m/s^2 at a certain altitude.
    Parameters:
        - Altitude = Altitude in m
    '''
    G = 6.673*10**(-11)
    RE = 6.37*10**6
    ME = 5.98*10**24
    g0 = G*ME/((altitude+RE)**2)
    return g0
```

Code

```
class Propellants:
    def __init__(self, name, isp, density, cost, exhaust_velocity):
        self.name = name
        self.isp = isp # s
        self.density = density # g/cc
        self.cost = cost # $/kg
        self.velocity = exhaust_velocity # m/s

    def set_burn_rate(self, diameter):
        nozzle_area = math.pi * (1/300*diameter*100)**2 # m^2
        self.burn_rate = nozzle_area*self.velocity*100*self.density/1000 # kg/s


class Materials:
    def __init__(self, name, density, cost):
        self.name = name
        self.density = density # g/cc
        self.cost = cost # $/kg
```

Code

```
class Rocket:
    def __init__(self, name, rocket_length, diameter, material, propellant, empty_rocket_mass = 0, propellant_mass=0):
        self.name = name
        self.propellant = propellant
        self.material = material
        self.rocket_length = rocket_length
        self.diameter = diameter
        self.fin_height = 1.25*self.diameter
        fin_a = 1/6*rocket_length
        fin_b = 1/6*rocket_length*0.45
        self.fin_volume = 0.01*self.diameter*(fin_a+fin_b)/2*1.05*self.diameter
        self.body_volume = (math.pi*(1/2*self.diameter)**2*0.85*self.rocket_length) - (math.pi*(1/2*self.diameter)**2*0.85*sel
        self.nose_cone_volume = (1/3*math.pi*(1/2*self.diameter)**2*0.15*self.rocket_length) - (1/3*math.pi*(1/2*self.diameter
        self.fin_mass = self.fin_volume * 1000 * material.density
        self.body_mass = self.body_volume * 1000 * material.density
        self.nose_cone_mass = self.nose_cone_volume * 1000 * material.density + 3000
        if propellant_mass != 0:
            self.propellant_mass = propellant_mass
        else:
            self.propellant_mass = (math.pi*(1/2*self.diameter)**2*0.85*self.rocket_length*0.99)*self.propellant.density/1000
        self.total_mass = self.fin_mass*2 + self.body_mass + self.nose_cone_mass + self.propellant_mass
        if empty_rocket_mass != 0:
```

Code

Sample setup output

```
--- ROCKET SPECS ---
Name: LOX-RP1, Aluminum
Rocket Length: 22.7 m
Rocket Diameter: 1.95 m
Empty Rocket Mass: 18600 kg
Rocket Cost: $11406603.25
Propellant Used: UDMH
Propellant Mass: 28600 kg
Propellant Burn Rate: 269.3934550009169 kg/s
Burn Time: 106.16442036389732 s
Center of Gravity: (0.9749999999999999 m, 10.459365954120667 m)
--------------------

--- LAUNCH PREP ---
Launch Angle: 0.5235987755982988 rad
Atmospheric Density at Launch: 1.0088950807649093 kg/m^3
Gravitational Constant at Launch: 9.828127333606815 m/s^2
--------------------
```

Code

```
def thrust(propellant, time):
    '''
    This function returns the magnitude of the thrust force for the rocket in N.
    Parameters:
        - Propellant = Type of propellant (str of 'UDMH', 'LOX_RP1', or 'LOX_LH2')
        - Time = Time from takeoff in s
    '''
    if time < propellant.burn_time:
        thrust = propellant.burn_rate*propellant.velocity
    else:
        thrust = 0
    return thrust
```

```
def lift(rocket, velocity, rho):
    '''
    This function returns the magnitude of the lift force for the rocket in N.
    Parameters:
        - Rocket = Rocket being simulated
        - Velocity = Velocity vector of rocket at instant in m/s [vx, vy]
        - Rho = Atmospheric density at instant in kg/m^3
    '''
    area = math.pi * (1/2*rocket.diameter)**2
    cl = 0.5
    lift_x = 1/2*cl*rho*area*velocity[0][0]**2
    lift_y = 1/2*cl*rho*area*velocity[0][1]**2
    lift = np.array([[lift_x, lift_y]])
    lift = np.linalg.norm(lift)
    return lift
```

```
def weight(rocket, g0):
    '''
    This function returns the magnitude of the thrust force for the rocket in N.
    Parameters:
        - Rocket = Rocket being simulated
        - G0 = Gravitational constant at instant in m/s^2
    '''
    weight = rocket.total_mass * g0
    return weight
```

```
def drag(rocket, velocity, rho):
    '''
    This function returns the magnitude of the drag force for the rocket in N.
    Parameters:
        - Rocket = Rocket being simulated
        - Velocity = Velocity vector of rocket at instant in m/s [vx, vy]
        - Rho = Atmospheric density at instant in kg/m^3
    '''
    area = math.pi * (1/2*rocket.diameter)**2
    cd = 0.295
    drag_x = -1/2*cd*rho*area*velocity[0][0]**2
    drag_y = -1/2*cd*rho*area*velocity[0][1]**2
    drag = np.array([[drag_x, drag_y]])
    return drag
```

Code

```
def force(rocket, propellant, altitude, time):
    '''
    This function finds the summation of the x and y components of force and returns force vector.
    Parameters:
        - Rocket = Rocket being simulated
        - Propellant = Type of propellant (str of 'UDMH', 'LOX_RP1', or 'LOX_LH2')
        - Altitude = Altitude at current instant in m
    '''
    rho = atmospheric_density(altitude)
    g = gravitational_constant(altitude)
    thrust_magnitude = thrust(propellant, time)
    drag_magnitude = drag(rocket, rocket.velocity, rho)
    lift_magnitude = lift(rocket, rocket.velocity, rho)
    weight_magnitude = weight(rocket, g)
    return thrust_magnitude, drag_magnitude, lift_magnitude, weight_magnitude
```

Code

```
def integrateGraph(time, array):
    resArray = [0]
    for n in range(0, len(time)-1):
        resArray.append(
            resArray[-1] + 0.5*(array[n+1] + array[n])*(time[n+1] -
            time[n])
        )
    return np.array(resArray)

velocity = integrateGraph(time, acceleration)
```

Rocket Efficiency Score

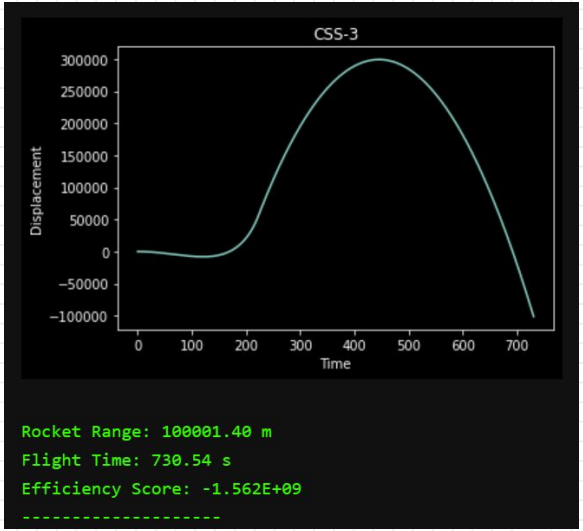
$$\varepsilon = \frac{\gamma}{\tau}(1 - \kappa)$$

ε = Efficiency Score, γ = Flight Range (m), τ = Flight Time (s), κ = Rocket Cost ($)

The objective is to maximize this function.

Results

CSS-3




```
Rocket Range: 100001.40 m
Flight Time: 730.54 s
Efficiency Score: -1.562E+09
--------------------
```

Score: −1.562E9

Hwasong 14


```
Rocket Range: 100001.64 m
Flight Time: 639.20 s
Efficiency Score: -1.784E+09
--------------------
```

Score: −1.784E9

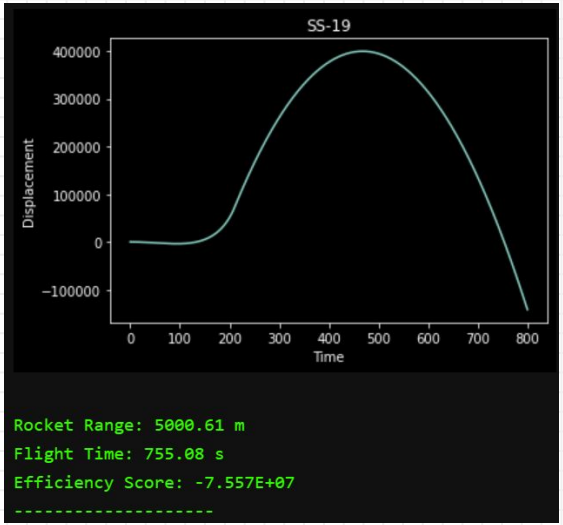
SS-18 SATAN


```
Rocket Range: 50000.11 m
Flight Time: 196.85 s
Efficiency Score: -2.902E+09
--------------------
```

Score: −2.902E9

Results

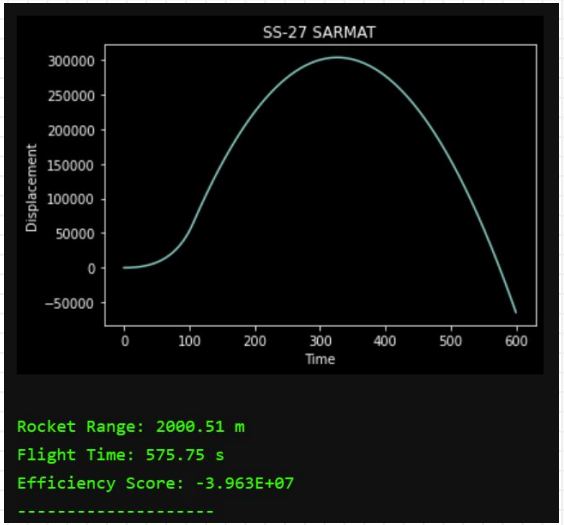
SS-19




```
Rocket Range: 5000.61 m
Flight Time: 755.08 s
Efficiency Score: -7.557E+07
--------------------
```

Score: −7.5572E7

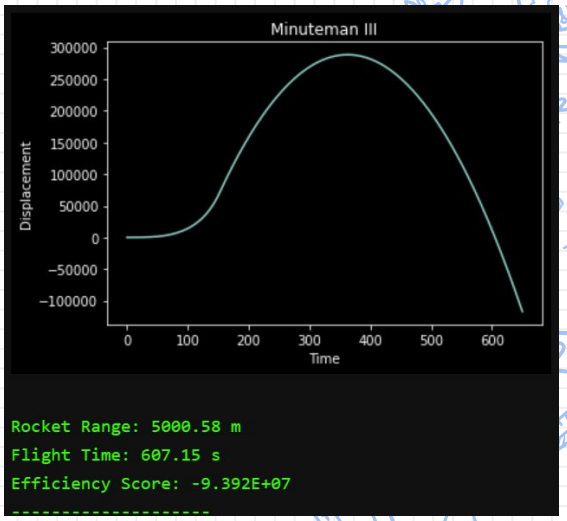
SS-27 SARMAT




```
Rocket Range: 2000.51 m
Flight Time: 575.75 s
Efficiency Score: -3.963E+07
--------------------
```

Score: −3.963E7

MINUTEMAN III

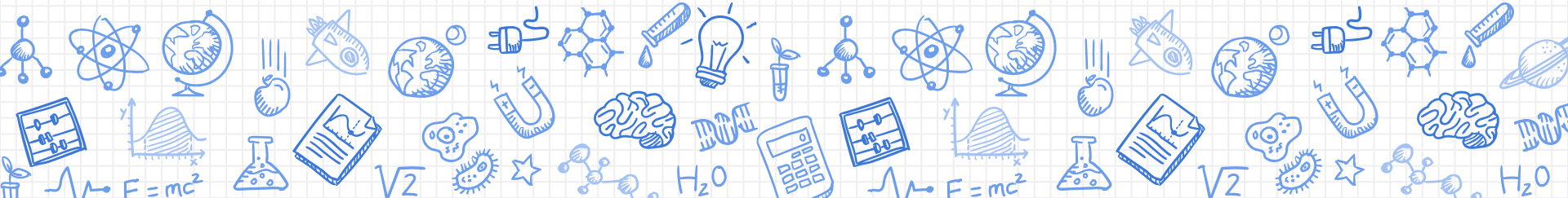



```
Rocket Range: 5000.58 m
Flight Time: 607.15 s
Efficiency Score: -9.392E+07
--------------------
```

Score: −9.392E7

Phase 2

Generative Design Testing



Testing

Independent Variables

- ✘ Propellant Type
- ✘ Rocket Material

Dependent Variables

- ✘ Rocket Cost (based on rocket size, propellant type, and amount of propellant)
- ✘ Flight range
- ✘ Flight time (from launch to impact)

Rocket Propellants

Name	Physical Properties	Cost
UDMH	Specific impulse: 333 s. Density: 1.18 g/cc. Characteristic velocity c: 1,720 m/s (5,640 ft/sec). Oxidizer Density: 1.450 g/cc. Fuel Density: 0.793 g/cc. Optimum Oxidizer to Fuel Ratio: 2.61.	$ 1.00 per kg
LOX + RP-1	Specific impulse: 353 s. Density: 1.02 g/cc. Characteristic velocity c: 1,805 m/s (5,921 ft/sec). Oxidizer Density: 1.140 g/cc. Fuel Density: 0.806 g/cc. Optimum Oxidizer to Fuel Ratio: 2.56.	$ 1.17 per kg
LOX + LH2	Specific impulse: 451 s. Characteristic velocity c: 2,435 m/s (7,988 ft/sec). Density: 0.28 g/cc. Oxidizer Density: 1.140 g/cc. Fuel Density: 0.071 g/cc. Optimum Oxidizer to Fuel Ratio: 6.	$ 1.09 per kg

Rocket Materials

Name	Density	Cost
Titanium	4.5 g/cc.	$ 14.90 per kg
Aluminum	2.7 g/cc.	$ 3.05 per kg

Rocket Cost

$$\kappa = \psi_\kappa * \psi_\alpha + \varrho_\kappa * \varrho_\sigma + 10^6$$
$$\varrho_\sigma = \eta_\sigma + \beta_\sigma + \Omega_\sigma$$

$$\beta_\sigma = \left(\pi\left(\frac{1}{2}d\right)^2 * 0.85\lambda\right) - \left(\pi\left(\frac{1}{2} * 0.98d\right)^2 * 0.85\lambda * 0.98\right)$$

$$\eta_\sigma = \left(\frac{1}{3}\pi\left(\frac{1}{2}d\right)^2 * 0.15\lambda\right) - \left(\frac{1}{3}\pi\left(\frac{1}{2} * 0.98d\right)^2 * 0.15\lambda * 0.98\right)$$

$$\Omega_\sigma = 0.02 * d * \frac{\left(\frac{1}{6} * \lambda + \frac{1}{6} * \lambda * 0.45\right)}{2} * 1.25d$$

$$\psi_\alpha = \left(\pi\left(\frac{1}{2} * 0.98d\right)^2 * 0.85\lambda * 0.98\right)$$

κ = Rocket Cost (\$), ψ_κ = Propellant Cost $\left(\frac{\$}{\text{cc}}\right)$, ψ_α = Amount of Propellant (cc),
ϱ_κ = Rocket Cost (\$), ϱ_σ = Rocket Size (m^3), η_σ = Nose Cone Size (m^3), β_σ = Body Size (m^3),
d = Diameter (m), λ = Rocket Length (m)

Rocket Weight

$$\varpi = \eta_{\varpi} + \beta_{\varpi} + \psi_{\varpi}$$
$$\eta_{\varpi} = \eta_{\sigma} * \sigma_{\mu} + 3000$$
$$\beta_{\varpi} = \beta_{\sigma} * \sigma_{\mu} + \psi_{\varpi}$$
$$\psi_{\varpi} = \psi_{\alpha} * \psi_{\sigma}$$

ϖ = Total Weight (kg), η_{ϖ} = Nose Cone Weight (kg), η_{σ} = Nose Cone Size (m^3), σ_{μ} = Material Density $\left(\frac{kg}{m^3}\right)$

β_{ϖ} = Body Weight (kg), β_{σ} = Body Size (m^3), ψ_{ϖ} = Propellant Weight (kg), ψ_{α} = Propellant Amount (m^3)

Testing Matrix – ε Scores

Propellant / Material	UDMH	LOX + RP-1	LOX + LH2
Aluminum	-3.066E+08	-9.051E+07	-2.303E+09
Titanium	-1.037E+08	-1.003E+08	-2.322E+09

$11,403,746.18

Rocket Cost

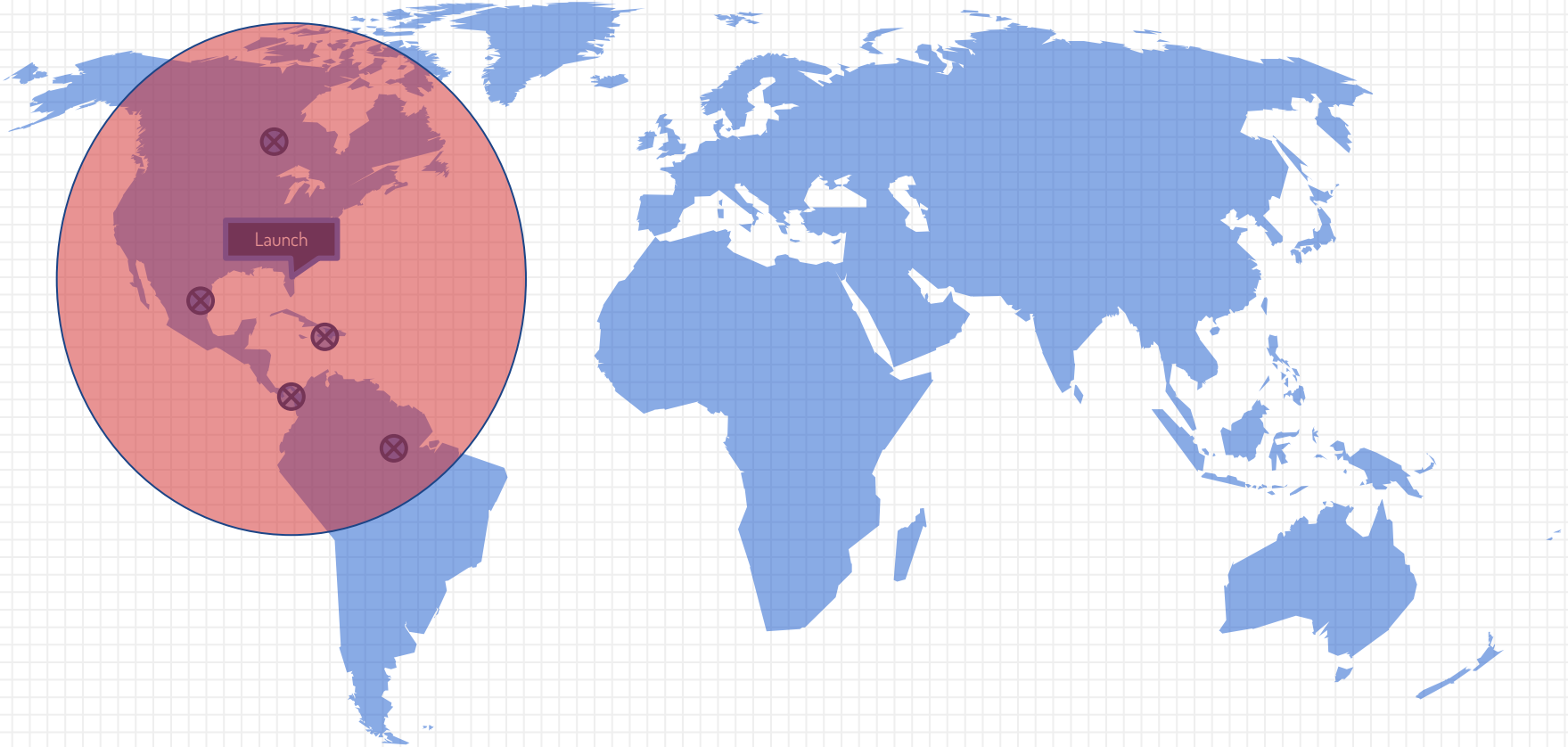
4,391.29 km

Flight Range

553.27 s

Flight Time

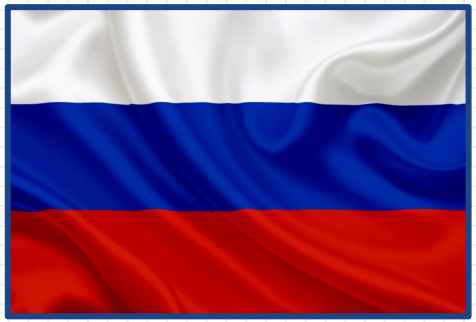
Rocket Range



Conclusion

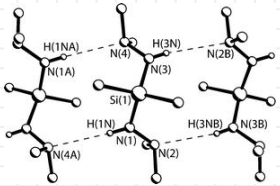
Phase 1

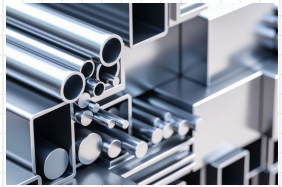
The best performing rocket was the SS-27 SARMAT rocket produced by Russia.



Phase 2

The best performing rocket used the LOX + RP-1 propellant and was built out of aluminum.





Summary

UDMH and aluminum seems to be the materials that are used most often by militaries around the world. LOX + RP-1 and aluminum are the most cost effective, with the biggest bang for the buck.

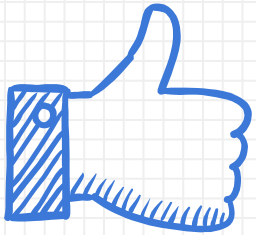
Future Steps

Realistic Forces

Simulate the aerodynamic forces acting on the center of pressure and weight and thrust acting on the center of gravity. This means torque will need to be introduced to calculations. Use vector fields for modeling force of wind and include Reynolds calculations and angle of attack. Conduct further research on effect of Moon's gravity. Include 3D motion.

Improve Data

Find more data regarding the rocket components such as engine specifications and thrust profiles. Incorporate real-life recorded atmospheric density data and weather data.



THANKS!

Any questions?

References

- https://www.globalsecurity.org/wmd/world/russia/icbm.htm
- https://nuke.fas.org/guide/russia/icbm/index.html
- http://www.astronautix.com/n/n2o4udmh.html
- http://ftp.demec.ufpr.br/foguete/bibliografia/TR-11%20AERODYNAMIC%20DRAG%20OF%20MODEL%20ROCKET.pdf
- https://www.grc.nasa.gov/www/k-12/rocket/rktcp.html