Jesse Wynn
ME EN 575 Optimization
Homework 2
Limestone Mill
Jan 26, 2018

# Report

## Objective

Minimize the total cost of starting and running a Limestone mill operation over the course of seven years by determining optimal pipe diameter, limestone slurry velocity, and average lump size of limestone particles after grinding (D, V and d respectively)

## I. Main Optimization Results

```
D = 0.1819 ft
V = 7.2357 ft/s
d = 0.0005 ft

total_cost = $399,193.86
P_g = 174.3648 HP
P_f = 223.2327 HP
P_total = 397.5975 HP
c_slurry = 0.3999
Q_w = 0.1127
rho = 1.0484e+02
```

Constraints (c <= 0)

```
c(1) = D - 0.5
c(2) = -V + 1.1*V_c                      - binding
c(3) = c_slurry - 0.4                    - binding
```

## II. Procedure

Part a and b:

Equation Sequence

Get values of design vars. from Optimization

$D, V, d$

Compute Slurry flow rate

$$Q = \frac{1}{4}\pi D^2 V$$

Compute Flow Rate of Limestone

$$Q_\ell = W/\gamma$$

Compute Flow Rate of water

$$Q_w = Q - Q_\ell$$

Compute Slurry Consentration

$$c = Q_\ell/Q$$

Compute Slurry density

$$\rho = \rho_w + c(\gamma - \rho_w)$$

Compute Power for Grinding

$$P_g = 218W\left(\frac{1}{\sqrt{d}} - \frac{1}{\sqrt{a}}\right)$$

Compute $R_w$

$$R_w = \frac{\rho_w V D}{\mu}$$

Compute $f_w$

$$f_w = 0.3164/R_w^{0.25} \quad \text{if } R_w \le 10e^5$$

$$f_w = 0.0032 + 0.221 R_w^{-0.237} \quad \text{if } R_w \ge 10e^5$$

Compute $C_d R_p^2$

$$C_d R_p^2 = \frac{4 g \rho_w d^3 (\gamma - \rho_w)}{3\mu^2}$$

Compute $f$

$$f = f_w\left(\frac{\rho_w}{\rho} + 150 c \frac{\rho_w}{\rho}\left(\frac{g D (s-1)}{V^2\sqrt{C_d}}\right)^{1.5}\right)$$

Compute $\Delta P$

$$\Delta P = \frac{f \rho L V^2}{2 D g_c}$$

Compute Power for Pumping

$$P_f = \Delta P Q$$

Compute Initial Cost

$$\text{cost\_initial} = 300(P_g/550) + 200(P_f/550)$$

Compute Yearly Operating Cost

$$\text{cost\_yearly} = 0.07\left(\frac{P_g}{550}\right)8*300 + 0.05\left(\frac{P_f}{550}\right)8*300$$

Compute Net Present Value for Operating Costs

$$P_{cost} = \text{cost-yearly}\ \frac{(1+i)^n - 1}{i(1+i)^n}$$

Compute Total Cost

$$\text{cost\_total} = \text{cost\_initial} + P_{cost}$$

Compute $V_c$ (for constraint)

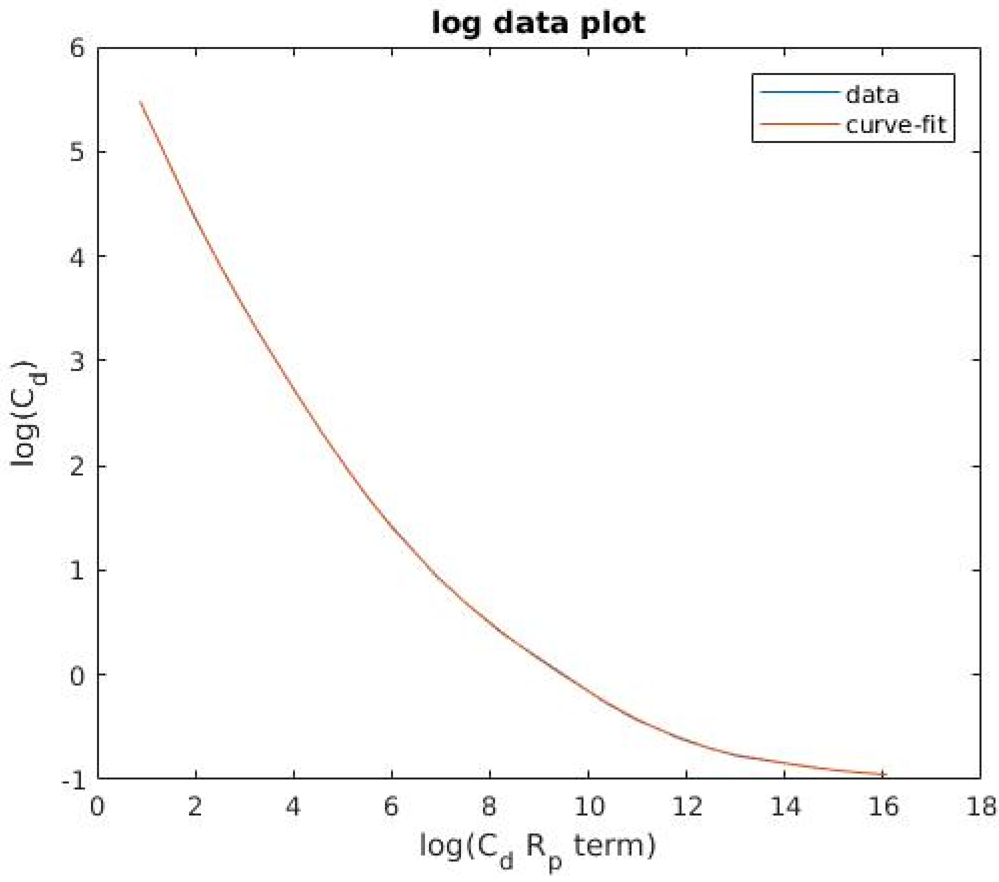
$$V_c = \left(\frac{40 g c (s-1) D}{\sqrt{C_d}}\right)^{1/2}$$

Loop until optimum found

Part c:

Model validation was done by first carefully checking units in the supplied equations, and then by checking the results of each equation for feasibility. This was done by carefully stepping through the code using breakpoints. Ultimately the model was verified when it produced a correct result. In the real world where there is not a correct result to compare against, further verification would be necessary in the form of verification of the validity of the equations used, and a thorough model and design review by a team of engineers.

For determining the drag coefficient C_d, I fit a curve to the log of the data supplied. Curve fitting was accomplished by fitting a $10^{th}$ order polynomial to the log data using a least-squares approach I implemented in MatLab. Once I had the polynomial coefficients, I wrote a simple look-up function that used the coefficients to find the log value of C_d. This value was then exponentiated and the result passed out of the function as C_d. The accuracy of my curve fit was assessed first by visually inspecting the result, and then by computing the Mean Square Error (MSE). I found the fit to be exceptionally good with an MSE of `2.07e-05`.



## III. Results and Discussion

Part a.

Optimum Values of Variables and Functions:

```
D = 0.1819 ft
V = 7.2357 ft/s
d = 0.0005 ft
c_slurry = 0.3999
Q_w = 0.1127
rho = 1.0484e+02

total_cost = $399,193.86
P_g = 174.3648 HP
P_f = 223.2327 HP
P_total = 397.5975 HP

**Highlight Indicates variable at constraints or bounds
```

Constraints (c <= 0)

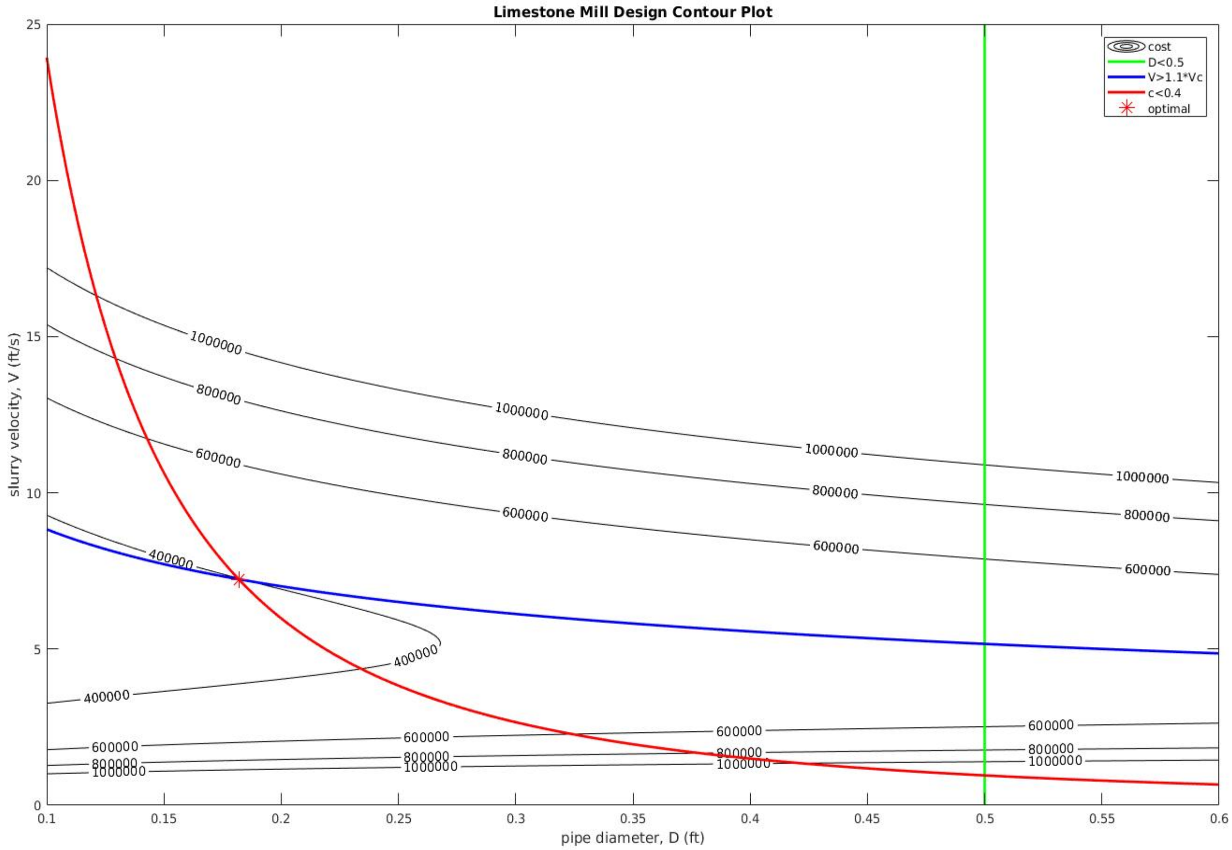
```
c(1) = D - 0.5
c(2) = -V + 1.1*V_c                      - binding
c(3) = c_slurry - 0.4                    - binding
```

Part b.

The optimum lies at the intersection of the slurry concentration constraint and the slurry velocity constraint. This makes these constraints binding constraints and the diameter constraint a non-binding constraint. As can be seen below, this location in the contour plot also corresponds to the lowest cost contour that is inside the feasible design space. This optimum is a local min and there is good evidence that it is also a global minimum because the result converges to this location even when the optimization is started outside the design space.

Part c.

**Contour Plot of Design Space**




Part d.

The primary observation I made was that this problem is very sensitive to small changes of the design variables as far as how much the total cost ends up being. This high sensitivity further highlights the importance of carefully validating the model.

IV. Appendix

Matlab Script:

```
function [xopt, fopt, exitflag, output] = mill_optimization()

    % ------------Starting point and bounds------------
    % design variables: x0 = [D, V, d]
    x0 = [0.3, 10.0, 0.001];   % starting point
    ub = [1.0, 1000, 0.01];  % upper bound
    lb = [0.1, 0.5, 0.0005];  % lower bound

    % ------------Linear constraints------------
    A = [];
    b = [];
    Aeq = [];
    beq = [];

    % ------------Objective and Non-linear Constraints------------
    function [f, c, ceq] = objcon(x)

        % set objective/constraints here

        % design variables (things we'll adjust to find optimum)
        D = x(1);  % internal pipe dia. (ft)
        V = x(2);  % avg flow velocity (ft/sec)
        d = x(3);  % avg limestone particle size after grinding (ft)

        % other analysis variables (constants that the optimization won't touch)
        % L = 15;  % pipe length (miles)
        L = 15*5280;  % pipe length (ft)
        W = 12.67;  % flowrate of limestone (lbm/sec)
        als = 0.01;  % avg lump size of limestone before grinding (ft)

        gamma = 168.5;  % limestone density (lb_m/ft^3)
        rho_w = 62.4;  % water density (lb_m/ft^3)
        g = 32.17;  % gravity (ft/s^2)
        g_c = 32.17;  % conversion factor
        mu = 7.392e-4;  % water viscosity (lb_m/ft-sec)
        S = gamma/rho_w;  % specific gravity of the limestone

        % analysis functions
        Q = 0.25*pi*(D^2)*V;  % flow rate of the slurry (volumetric)
        Q_l = W/gamma;  % flow rate of limestone (volumetric)
        Q_w = Q - Q_l
        c_slur = Q_l/Q
        % c_w = (c_slur*gamma)/((1-c_slur)*rho_w + c_slur*gamma);  % consentration of solid by
weight in the slurry
        % rho = 1/(c_w/gamma + (1 - c_w)/rho_w);  % density of the slurry (lb_m/ft^3)
        rho = rho_w + c_slur*(gamma - rho_w)

        P_g = 218*W*((1/sqrt(d)) - (1/sqrt(als)));  % power for grinding (ft-lbf/sec)

        R_w = rho_w*V*D/mu;
        if R_w <= 10e5
            f_w = 0.3164/(R_w^0.25);
        else
            f_w = 0.0032 + 0.221*(R_w^-0.237);
```

```matlab
        end

        Cd_Rp_term = 4*g*rho_w*(d^3)*((gamma - rho_w)/(3*(mu^2)));
        C_d = cd_lookup(Cd_Rp_term);

        fric = f_w*((rho_w/rho)+150*c_slur*(rho_w/rho)*(g*D*(S-1)/((V^2)*sqrt(C_d)))^1.5)

        delta_p = (fric*rho*L*(V^2))/(D*2*g_c);

        P_f = delta_p*Q;

        total_power = (P_g + P_f)/550

        grinder_hp = P_g/550
        pump_hp = P_f/550

        V_c = ((40*g*c_slur*(S-1)*D)/(sqrt(C_d)))^0.5

        % COST STUFF
        initial_cost = 300*(P_g/550) + 200*(P_f/550);
        hrs_per_year = 8*300;  % 8 hrs per day, 300 days per year
        yearly_operating_cost = 0.07*(P_g/550)*hrs_per_year + 0.05*(P_f/550)*hrs_per_year;
        ir = 0.07;
        n = 7;  % number of years

        P = yearly_operating_cost*(((1 + ir)^n - 1)/(ir*(1 + ir)^n));

        total_cost = initial_cost + P

        % objective function (what we're trying to optimize)
        f = total_cost;  % minimize total cost

        % inequality constraints (c<=0)
        c = zeros(3,1);
        c(1) = D - 0.5;       % D <= 0.5
        c(2) = -V + 1.1*V_c;  % V >= 1.1*V_c
        c(3) = c_slur - 0.4;  % c_slur <= 0.4

        % equality constraints (ceq=0)
        ceq = [];  % empty when we have none

    end

    % ------------Call fmincon------------
    options = optimoptions(@fmincon, 'display', 'iter-detailed');
    [xopt, fopt, exitflag, output] = fmincon(@obj, x0, A, b, Aeq, beq, lb, ub, @con, options);

    % ------------Separate obj/con (do not change)------------
    function [f] = obj(x)
            [f, ~, ~] = objcon(x);
    end
    function [c, ceq] = con(x)
            [~, c, ceq] = objcon(x);
    end
end
```