AAE 550 Homework Assignment #2

Alex Hagen

November 4, 2016

1 Constrained Minimization in N Variables - Direct Methods

1.1 Provision of optimization problem statement

The problem asks us to minimize the mass of a rectangular beam under a uniform load. The mass of a rectangular beam is defined as

$$f(l, w, h) = \rho l w h$$

where ρ is the density of the material ($7800\,{}^{kg}/_{\mathrm{m}^3}$ for A36 steel), and l, b, and h are the length, width and height of the beam, respectively. The length of the beam is set at 20 m. The other two dimensional variables (b and h) are the design variables.

We also have constraints, the simplest of which are the design variable contraints. For the width, b, we have

$$0.05 \leq b \leq 5.0$$

which can be tranlsated to

$$\begin{aligned} g_1(x) &\leq 0 \\ g_1(x) = 0.05 - b &\leq 0 \end{aligned}$$

and

$$\begin{aligned} g_2(x) &\leq 0 \\ g_2(x) = b - 5.0 &\leq 0 \end{aligned}$$

For the height constraint, we have

$$0.05 \leq h \leq 5.0$$

which can be tranlsated to

$$\begin{aligned} g_3(x) &\leq 0 \\ g_3(x) = 0.05 - h &\leq 0 \end{aligned}$$

and

$$\begin{aligned} g_4(x) &\leq 0 \\ g_4(x) = h - 5.0 &\leq 0 \end{aligned}$$

We also have constraints for excessive deflection, torsional twisting, and flexural bending. We must constrain the system against excessive deflection

$$D \leq D_a = 0.025\,\mathrm{m}$$

which can be translated to

$$\begin{aligned} g_5(x) &\leq 0 \\ g_5(x) = D - D_a &\leq 0 \\ g_5(x) = \frac{L^3}{48 E I_{zz}} \left(P + \frac{5}{8} q L \right) - D_a &\leq 0 \end{aligned}$$

where L is the length of the beam, E is the modulus of elasticity (200 GPa for A36 steel), I_{zz} is the moment of inertia along the z direction ($I_{zz} = \frac{bh^3}{12}$), q is the uniform load applied ($25\,{}^{kN}/_{\mathrm{m}}$), and P is the concentrated load (100 kN). When expanding this, we get

$$g_5(x) = \frac{L^3}{4Ebh^3}\left(P + \frac{5}{8}qL\right) - D_a \leq 0$$

which has both design variables in the denominator. So, in order to get the design variables in the numerator, we can instead translate the constraint as

$$\begin{aligned} g_5(x) &\leq 0 \\ g_5(x) = 1 - \frac{D_a}{D} &\leq 0 \\ g_5(x) = 1 - \frac{D_a}{\frac{L^3}{4Ebh^3}\left(P + \frac{5}{8}qL\right)} &\leq 0 \\ g_5(x) = 1 - \frac{4D_aEbh^3}{L^3\left(P + \frac{5}{8}qL\right)} &\leq 0 \end{aligned}$$

The constraint for the maximum bending is given by

$$\sigma_{max} = \frac{M}{I_{zz}}\frac{h}{2} \leq \sigma_a = 200\,\mathrm{MPa}$$

where M is the moment, given by $M = \frac{L}{8}(2P + Lq)$, I_{zz} is defined as before, and h is the height of the beam. Expanding this and translating to $g_i(x)$, we get

$$\begin{aligned} g_6(x) &\leq 0 \\ g_6(x) = \frac{3L}{2bh^3}(2P + Lq)\frac{h}{2} - \sigma_a &\leq 0 \end{aligned}$$

which again, has both design variables in the denominator. We can perform the same kind of translation to get the constraints in the numerator

$$\begin{aligned} g_6(x) &\leq 0 \\ g_6(x) = 1 - \frac{\sigma_a}{\sigma} &\leq 0 \\ g_6(x) = 1 - \frac{\sigma_a}{\frac{3L}{2bh^3}(2P + Lq)\frac{h}{2}} &\leq 0 \\ g_6(x) = 1 - \frac{4\sigma_a bh^2}{3L(2P + Lq)} &\leq 0 \end{aligned}$$

To constrain the critical load, we have the constraint

$$P + qL \leq P_{cr} = \frac{4.013\sqrt{0.312hb^3GEI_{least}}}{L^2}$$

where G is the modulus of rigidity ($G = 75$ GPa) for A36 Steel, and I_{least} is the least of the two moments of inertia, I_{zz} as defined earlier, and $I_{yy} = \frac{b^3h}{12}$. To formally translate this into $g_i(x)$, we must split the relation into two constraints. Thus, we have

$$\begin{aligned} g_7(x) &\leq 0 \\ g_7(x) = P + qL - \frac{4.013\sqrt{0.312hb^3GEbh^3}}{L^2\sqrt{12}} &\leq 0 \\ g_7(x) = P + qL - \frac{4.013\sqrt{0.312h^4b^4GE}}{L^2\sqrt{12}} &\leq 0 \end{aligned}$$

and

$$g_8(x) \leq 0$$
$$g_8(x) = P + qL - \frac{4.013\sqrt{0.312hb^3GEb^3h}}{L^2\sqrt{12}} \leq 0$$
$$g_8(x) = P + qL - \frac{4.013\sqrt{0.312h^2b^6GE}}{L^2\sqrt{12}} \leq 0$$

Finally, there is a verbal constraint, that the beam height must not be any longer than eight times the beam width, which can be written as

$$h \leq 8b$$

and translated to the formal constraint

$$g_9(x) \leq 0$$
$$g_9(x) = h - 8b \leq 0$$

Finally, we can write the entire constraint problem:
We want to minimize

$$f(b,h) = \rho Lbh$$

given constraints

$$\vec{g}(b,h) = \left\{ \begin{array}{c} 0.05 - b \\ b - 5.0 \\ 0.05 - h \\ h - 5.0 \\ 1 - \frac{4D_aEbh^3}{L^3\left(P + \frac{5}{8}qL\right)} \\ 1 - \frac{4\sigma_abh^2}{3L(2P+Lq)} \\ P + qL - \frac{4.013\sqrt{0.312h^4b^4GE}}{L^2\sqrt{12}} \\ P + qL - \frac{4.013\sqrt{0.312h^2b^6GE}}{L^2\sqrt{12}} \\ h - 8b \end{array} \right\} \leq 0$$

Then, we must perform scaling on this $\vec{g}$ to put the constraints on the order of 1 at the constraints. The program below shows an easy way to do this by hand, which gives us the final

$$\vec{g}(b,h) = \left\{ \begin{array}{c} 0.05 - b \\ b - 5.0 \\ 0.05 - h \\ h - 5.0 \\ 1 \times 10^{-2}\left(1 - \frac{4D_aEbh^3}{L^3\left(P + \frac{5}{8}qL\right)}\right) \\ 1 \times 10^{-2}\left(1 - \frac{4\sigma_abh^2}{3L(2P+Lq)}\right) \\ 1 \times 10^{-9}\left(P + qL - \frac{4.013\sqrt{0.312h^4b^4GE}}{L^2\sqrt{12}}\right) \\ 1 \times 10^{-9}\left(P + qL - \frac{4.013\sqrt{0.312h^2b^6GE}}{L^2\sqrt{12}}\right) \\ h - 8b \end{array} \right\} \leq 0$$

```
clear; clc;
% use a script to call all the variables called out by the problem
% statement
parameters

% iterate through all of the constraints and print their values, so that we
% can add a scaling factor for them to get them on the order of 1
for b=[0.05, 5.0]
    for h=[0.05, 5.0]
        g(1) = 0.05 - b;
        g(2) = b - 5.0;
```

```
        g(3) = 0.05 - h;
        g(4) = h - 5.0;
        g(5) = 1E-2 * (1 - (4*D_a*E*b*h^3)/(L^3 * (P+(5/8)*q*L)));
        g(6) = 1E-2 * (1 - (4*sigma_a*b*h^2)/(3*L*(2*P+L*q)));
        g(7) = 1E-9 * ...
            (P + q*L - 4.013*sqrt(0.312*h^4*b^4*G*E)/(L^2*sqrt(12)));
        g(8) = 1E-9 * ...
            (P + q*L - 4.013*sqrt(0.312*h^2*b^6*G*E)/(L^2*sqrt(12)));
        g(9) = h - 8*b;
        b
        h
        g
    end
end
```

Listing 1: Unscaled constraints visual program for finding scaling factors

1.2 Optimization of Beam Mass Minimization using Sequential Linear Programming

Sequential Linear programming was used following the code listed in Listing 4 on the following page, with the constraints and objective function listed in Listing 2 and 3, respectively. The solutions for several different cases (including initial guesses across the whole constraint space) are shown in table 1.

```
function [g, h_i, grad_g, grad_h] = constraints(x)
    % First convert the input vector to variables in the problem
    b = x(1);
    h = x(2);

    % Then use a script to read in all parameters
    parameters;

    % Now, construct the constraints under g
    g(1) = 0.05 - b;
    g(2) = b - 5.0;
    g(3) = 0.05 - h;
    g(4) = h - 5.0;
    g(5) = 1E-2 * (1 - (4*D_a*E*b*h^3)/(L^3 * (P+(5/8)*q*L)));
    g(6) = 1E-2 * (1 - (4*sigma_a*b*h^2)/(3*L*(2*P+L*q)));
    g(7) = 1E-9 * ...
        (P + q*L - 4.013*sqrt(0.312*h^4*b^4*G*E)/(L^2*sqrt(12)));
    g(8) = 1E-9 * ...
        (P + q*L - 4.013*sqrt(0.312*h^2*b^6*G*E)/(L^2*sqrt(12)));
    g(9) = h - 8*b;

    % now, return the equality constraints h_i
    h_i = [];

    % For later, we take the gradient of g (and h_i)
    grad_g = ...
        [-1, 0; ...
         1, 0; ...
         0, -1; ...
         0, 1; ...
         -1E-2*(4*D_a*E*h^3)/(L^3*(P+(5/8)*q*L)), ...
            -1E-2*(12*D_a*E*b*h^2)/(L^3*(P+(5/8)*q*L)); ...
         -1E-2*(4*sigma_a*h^2)/(L^3*(P+(5/8)*q*L)), ...
            -1E-2*(8*sigma_a*b*h)/(L^3*(P+(5/8)*q*L)); ...
         -1E-9*(2*4.013*b*sqrt(0.312*h^4*G*E))/(L^2*sqrt(12)), ...
            -1E-9*(2*4.013*h*sqrt(0.312*b^4*G*E))/(L^2*sqrt(12)); ...
         -1E-9*(3*4.013*b^2*sqrt(0.312*h^2*G*E))/(L^2*sqrt(12)), ...
            -1E-9*(4.013*sqrt(0.312*b^6*G*E))/(L^2*sqrt(12)); ...
         8, 1];
    grad_g = grad_g';
    grad_h = [];
end
```

Listing 2: Function to return constraints and analytical gradient

```
function [f, grad_f] = objective_function(x)
```

```
    % First convert the input vector to variables in the problem
    b = x(1);
    h = x(2);

    % Then use a script to read in all parameters
    parameters;

    % Then we define our objective function
    f = rho * L * b * h;

    if nargout > 1  % fun called with two output arguments
        % Matlab naming convention will use grad_f(1) as df/dx(1);
        % grad_f(2) as df/dx(2)
        grad_f = ...
            [rho*L*h; ...
             rho*L*b];
    end
end
```

Listing 3: Objective function and gradient function

```
clear all;
parameters;
x0 = [2.5; 2.5];
delta_x = [5.0, 5.0];

epsilon_f = 1e-04;
epsilon_g = 1e-04;
epsilon_h = 1e-04;
max_ii = 100;

options = optimoptions('linprog','Algorithm','simplex','Display','iter');
lb = [-Inf;-Inf];
ub = [Inf;Inf];

f_last = 1e+5;
ii = 0;

x = x0;
[f, grad_f] = objective_function(x);
[g, ~, grad_g, ~] = constraints(x);

while (((abs(f_last - f) >= epsilon_f) || (max(g) >= epsilon_g)) ...
        && (ii < max_ii))
    % increment counter
    ii = ii + 1;

    % store 'f_last' value
    f_last = f;

    % linprog uses vector of coefficients as input; does not need constant
    % term
    fhat = grad_f;

    A = grad_g';
    b = grad_g' * x0 - g';

    Aeq = [];
    beq = [];

    lb_LP = max(x0' - delta_x, lb');
    ub_LP = min(x0' + delta_x, ub');

    [x,fval,exitflag,output] = linprog(fhat,A,b,Aeq,beq,lb_LP,ub_LP,x0,...
        options);

    [f, grad_f] = objective_function(x);
    [g, ~, grad_g, ~] = constraints(x);

    x0 = x;
end
```

Listing 4: Sequential Linear Programming script

Table 1: Optimization summary for Sequential Linear Programming

	Run 1	Run 2	Run 3	Run 4
$\vec{x}^0$	$\begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix}$	$\begin{bmatrix} 0.5 \\ 4.5 \end{bmatrix}$	$\begin{bmatrix} 4.5 \\ 0.5 \end{bmatrix}$	$\begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$
$\vec{x}^*$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	
$f(\vec{x}^*)$	2.3324×10^4	2.3324×10^4	2.3324×10^4	*
$g(\vec{x}^*)$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	
Number of Iterations	12	10	12	
Optimality Condition	1	1	1	

[a] Matlab provided the error `Exiting: The constraints are overly stringent; no feasible point exists.` for all $\vec{x}^0 < 0.5$, which shows that we are outside of constraints to begin with.

1.3 Optimization of Beam Mass Minimization using Generalized Reduced Gradient Method

Excel's GRG solver was used with the cell setup described in table 2, and the results of that method across the whole constraint space (and even outside of it) are shown in table 3.

Table 3: Optimization summary for Generalized Reduced Gradient Method

	Run 1	Run 2	Run 3	Run 4	Run 5
$\vec{x}^0$	$\begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix}$	$\begin{bmatrix} 0.5 \\ 4.5 \end{bmatrix}$	$\begin{bmatrix} 4.5 \\ 0.5 \end{bmatrix}$	$\begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$	$\begin{bmatrix} 6.0 \\ 0.01 \end{bmatrix}^*$
$\vec{x}^*$	$\begin{bmatrix} 0.1423 \\ 1.0506 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0506 \end{bmatrix}$
$f(\vec{x}^*)$	23323.34	23323.63	23323.60	23323.56	23323.54
$g(\vec{x}^*)$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9494 \\ \sim 0 \\ -.0199 \\ -0.0038 \\ \sim 0 \\ -0.0879 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0006 \\ -3.9494 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$
Number of Iterations	7	8	9	5	12

[a] Note that the starting point $\vec{x}^0 = \begin{bmatrix} 6.0 & 0.01 \end{bmatrix}^T$ is infeasible.

design variables	
x1	0.142315055956681
x2	1.0505554058966
rho	7800
L	20
sigma_a	200000000
D_a	0.025
E	200000000000
G	75000000000
P	100000
q	25000

Table 2: Excel Table Setup for Generalized Reduced Gradient Method

	objective function
f(x)	=B5*B6*B2*B3
	constraints
g1(x)	=0.05 - B2
g2(x)	=B2-5
g3(x)	=0.05-B3
g4(x)	=B3-5
g5(x)	=0.01 * (1-(4*B8*B9*B2*B3*B3*B3)/(B6*B6*B6 * (B11+5*B12*B6/8)))
g6(x)	=0.01 * (1 - (4*B7*B2*B3*B3)/(3*B6*(2*B11+B6*B12)))
g7(x)	=0.000000001 * (B11 + B12*B6 - 4.013*SQRT(0.312*B3*B3*B3*B3*B2*B2*B2*B2*B10*B9)/(B6*B6*SQRT(12)))
g8(x)	=0.000000001 * (B11 + B12*B6 - 4.013*SQRT(0.312*B3*B3*B2*B2*B2*B2*B2*B2*B10*B9)/(B6*B6*SQRT(12)))
g9(x)	=B3-8*B2

1.4 Optimization of Beam Mass Minimization using Sequential Quadratic Programming

Sequential Quadratic Programming was used following the code listed in Listing 6, with the constraints listed in Listing 5 and objective function as before. The solutions for several different cases (including initial guesses across the whole constraint space and one outside) are shown in table 4.

```
function [g, h_i, grad_g, grad_h] = nonlinear_constraints(x)
    % First convert the input vector to variables in the problem
    b = x(1);
    h = x(2);

    % Then use a script to read in all parameters
    parameters;

    % Now, construct the constraints under g
    g(1) = 1E-2 * (1 - (4*D_a*E*b*h^3)/(L^3 * (P+(5/8)*q*L)));
    g(2) = 1E-2 * (1 - (4*sigma_a*b*h^2)/(3*L*(2*P+L*q)));
    g(3) = 1E-9 * ...
        (P + q*L - 4.013*sqrt(0.312*h^4*b^4*G*E)/(L^2*sqrt(12)));
    g(4) = 1E-9 * ...
        (P + q*L - 4.013*sqrt(0.312*h^2*b^6*G*E)/(L^2*sqrt(12)));

    % now, return the equality constraints h_i
    h_i = [];

    % For later, we take the gradient of g (and h_i)
    grad_g = ...
        [-1E-2*(4*D_a*E*h^3)/(L^3*(P+(5/8)*q*L)), ...
            -1E-2*(12*D_a*E*b*h^2)/(L^3*(P+(5/8)*q*L)); ...
         -1E-2*(4*sigma_a*h^2)/(3*L*(P+(5/8)*q*L)), ...
            -1E-2*(8*sigma_a*b*h)/(3*L*(P+(5/8)*q*L)); ...
         -1E-9*(2*4.013*b*sqrt(0.312*h^4*G*E))/(L^2*sqrt(12)), ...
            -1E-9*(2*4.013*h*sqrt(0.312*b^4*G*E))/(L^2*sqrt(12)); ...
         -1E-9*(3*4.013*b^2*sqrt(0.312*h^2*G*E))/(L^2*sqrt(12)), ...
            -1E-9*(4.013*sqrt(0.312*b^6*G*E))/(L^2*sqrt(12))];
    grad_g = grad_g';
    grad_h = [];
end
```

Listing 5: Nonlinear Constraints Function for SQP

```
clear all;
% linear inequality constraints
A = [-1, 0; 1, 0; 0, -1; 0, 1; -8, 1];
b = [-0.05; 5; -0.05; 5; 0];
% no linear equality constraints
Aeq = [];
beq = [];
% lower bounds
lb = [];
% upper bounds
ub = [];
% set options for medium scale algorithm with active set (SQP as described
% in class; these options do not include user-defined gradients
options = optimoptions('fmincon', 'Algorithm', 'sqp', 'Display', 'iter');
% initial guess  - note that this is infeasible
x0 = [6; 0.01];
[x,fval,exitflag,output] = fmincon('objective_function', x0, A, b, Aeq, ...
    beq, lb, ub, 'nonlinear_constraints', options)
```

Listing 6: Script for calling Sequential Quadratic Programming with Numerical Gradients

Table 4: Optimization summary for Sequential Linear Programming

	Run 1	Run 2	Run 3	Run 4	Run 5
$\vec{x}^0$	$\begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix}$	$\begin{bmatrix} 0.5 \\ 4.5 \end{bmatrix}$	$\begin{bmatrix} 4.5 \\ 0.5 \end{bmatrix}$	$\begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$	$\begin{bmatrix} 6.0 \\ 0.01 \end{bmatrix}^*$
$\vec{x}^*$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$
$f(\vec{x}^*)$	2.3324×10^4	2.3324×10^4	2.3324×10^4	390	2.3324×10^4
$g(\vec{x}^*)^{**}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} \sim 0 \\ -4.95 \\ \sim 0 \\ -4.95 \\ 0.01 \\ 0.01 \\ 0.0006 \\ 0.0006 \\ -0.35 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$
Number of Iterations	12	9	12	2	12
Optimality Condition	1	1	1	-2	1

* Note that the starting point $\vec{x}^0 = \begin{bmatrix} 6.0 & 0.01 \end{bmatrix}^T$ is infeasible.

** Note that I've combined both the nonlinear and linear constraints into the original constraint vector $\vec{g}(\vec{x}^*)$ in this table.

1.5 Optimization of Beam Mass Minimization using Analytic Gradients and Sequential Linear Programming

To first perform this, we must determine the gradient of the constraints, which can be calculated in two columns:

$$\frac{\partial \vec{g}(b,h)}{\partial b} = \begin{Bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ -1 \times 10^{-2}\left(\frac{4D_a E h^3}{L^3\left(P+\frac{5}{8}qL\right)}\right) \\ -1 \times 10^{-2}\left(\frac{4\sigma_a h^2}{3L(2P+Lq)}\right) \\ -1 \times 10^{-9}\left(\frac{2\times 4.013b\sqrt{0.312h^4GE}}{L^2\sqrt{12}}\right) \\ -1 \times 10^{-9}\left(\frac{3\times 4.013b^2\sqrt{0.312h^2GE}}{L^2\sqrt{12}}\right) \\ 8 \end{Bmatrix} \quad \text{and} \quad \frac{\partial \vec{g}(b,h)}{\partial h} = \begin{Bmatrix} 0 \\ 0 \\ -1 \\ 1 \\ -1 \times 10^{-2}\left(\frac{12D_a E b h^2}{L^3\left(P+\frac{5}{8}qL\right)}\right) \\ -1 \times 10^{-2}\left(\frac{8\sigma_a bh}{3L(2P+Lq)}\right) \\ -1 \times 10^{-9}\left(\frac{2\times 4.013h\sqrt{0.312b^4GE}}{L^2\sqrt{12}}\right) \\ -1 \times 10^{-9}\left(\frac{4.013\sqrt{0.312b^6GE}}{L^2\sqrt{12}}\right) \\ 1 \end{Bmatrix}$$

The return of these calculated vectors as one matrix can be seen as the return for grad_g in the function listed in Listing 2 on page 4. The results for initial guesses across the entire space are given in table 5.

```
clear all;
% linear inequality constraints
A = [-1, 0; 1, 0; 0, -1; 0, 1; -8, 1];
b = [-0.05; 5; -0.05; 5; 0];
% no linear equality constraints
Aeq = [];
beq = [];
% lower bounds
lb = [];
% upper bounds
ub = [];
% set options for medium scale algorithm with active set (SQP as described
% in class; these options do not include user-defined gradients
options = optimoptions('fmincon', 'Algorithm', 'sqp', ...
    'GradObj', 'on', 'GradConstr', 'on', 'Display', 'iter');
% initial guess  - note that this is infeasible
x0 = [6; 0.01];
```

```
[x,fval,exitflag,output] = fmincon('objective_function', x0, A, b, Aeq, ...
    beq, lb, ub, 'nonlinear_constraints', options)
```

Listing 7: Script for calling Sequential Quadratic Programming with Analytic Gradients

Table 5: Optimization summary for Sequential Quadratic Programming with Analytic Gradients

	Run 1	Run 2	Run 3	Run 4	Run 5
$\vec{x}^0$	$\begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix}$	$\begin{bmatrix} 0.5 \\ 4.5 \end{bmatrix}$	$\begin{bmatrix} 4.5 \\ 0.5 \end{bmatrix}$	$\begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$	$\begin{bmatrix} 6.0 \\ 0.01 \end{bmatrix}^*$
$\vec{x}^*$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$	$\begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$
$f(\vec{x}^*)$	2.3324×10^4	2.3324×10^4	2.3324×10^4	390	2.3324×10^4
$g(\vec{x}^*)^{**}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$	$\begin{bmatrix} \sim 0 \\ -4.95 \\ \sim 0 \\ -4.95 \\ 0.01 \\ 0.01 \\ 0.0006 \\ 0.0006 \\ -0.35 \end{bmatrix}$	$\begin{bmatrix} -0.0923 \\ -4.8577 \\ -1.0005 \\ -3.9495 \\ \sim 0 \\ -0.0199 \\ -0.0038 \\ \sim 0 \\ -0.0880 \end{bmatrix}$
Number of Iterations	12	11	12	2	6
Optimality Condition	1	1	1	-2	1

* Note that the starting point $\vec{x}^0 = \begin{bmatrix} 6.0 & 0.01 \end{bmatrix}^T$ is infeasible.

** Note that I've combined both the nonlinear and linear constraints into the original constraint vector $\vec{g}(\vec{x}^*)$ in this table.

1.6 Results

There seemed to be no difference in the solution for the minimization solved by any of the four methods. The answer

$$\vec{x}^* = \begin{bmatrix} 0.1423 \\ 1.0505 \end{bmatrix}$$

with a final mass of $m = f(\vec{x}^*) = 23324\,\text{kg}$ is universal amongst the solutions, and constraints $g_5(\vec{x})$ and $g_8(\vec{x})$ become active in every case. However, there are some differences between the methods. Sequential Linear Programming is unable to use infeasible starting guesses, and also converges to an incorrect value (with violated constraints) for even some feasible starting guesses. The Generalized Gradient Reduced Gradient Method is less computationally expensive as the SLP method (i.e. Run 1 took 7 iterations instead of 12), and is able to handle the infeasible starting guess and seems to converge to the right answer across the whole space. The Sequential Quadratic Programming methods both are similarly computationally expensive, even though it was predicted that the analytic gradients would make for faster convergence. Perhaps a comparison in function iterations would show the difference in these methods computationally. Both of the SQP methods handled the infeasible constraint, but displayed the same incorrect value at $\vec{x}^0 = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}^T$ as SLP did. In general, the GRG seems to be the most valuable method here, requiring less iterations and handling initial guesses across the entire space (and outside).