Thomas Satterly
AAE 550, HW 2

I. Constrained Minimization in *N* Variables – Direct Methods

Design a hollow circular beam-column with minimum weight for two conditions:

a) When P = 55 kN, the axial stress must not exceed the allowable stress
b) When P = 0, the deflection due to self weight should be less that 0.001 of the beam's length

The manufacturing process limits the radius and thickness of the beam and ratio of the radius and thickness of the beam as follows:

$$0.02 \text{ m} <= R <= 0.2 \text{ m}$$
$$0.001 \text{ m} <= t <= 0.01 \text{ m}$$
$$R / t <= 18$$

The beam is fabricated with the following material properties:

Density: 7850 kg/m^3
Allowable Axial Stress: 405 MPa
Modulus of Elasticity: 250 GPa

Find:

1) Provide an optimization problem statement. Clearly present the variables intended to use as the design variables. Formulate the constraints into $g_i <= 0$ and clearly present them. Present how the variable bounds will be treated. Consider variable scaling and note any conditioning used.
2) Solve the beam design problem with one of the two available constrained optimization methods that make use of linear programming (SLP of Method of Centers). Be sure to indicate which method will be used. Also indicate the value of move limits used. Use at lease two different x^0 points, if not more. Tabulate the complete results.
3) Solve the problem using the GRG method in the Excel solver. Use at least two different x^0 and tabulate the results as before. Also include the number of iterations needed to reach the solution. Include a cut and paste of the important cells in the workbook.
4) Solve the problem using the SQP method available in Matlab's *fmincon* function. Use numerical gradients. Again, use at least two different x^0 starting points and tabulate the results. Include the number of iterations needed to reach a solution.
5) Repeat question (4) using analytic gradients.
6) Describe the results. Do different choices of x^0 result in different values of x^*? How do different x^0 values impact the chosen LP method? Does one method appear to use fewer iterations on average? Are there any features of this problem that make one of these methods more suitable? In this problem, were there any transformations or scaling that were helpful?

Solution:

1) See attached for the problem statement, objective function and constraint definitions, and gradient calculations

AAE550 10/29/17 HW #2 Thomas Sitterly 1/2

1) The goal is to minimize the weight of the beam while staying within constraints:

Minimize : $f(x) = m = \rho AL = \rho \pi t(2R-t)L$, $x = \begin{Bmatrix} R \\ t \end{Bmatrix}$

Constraints:

1. $0.02 \leq R \leq 0.2$ (m)
2. $0.001 \leq t \leq 0.01$ (m)
3. $R/t \leq 18$
4. $\delta \leq 0.001L \Rightarrow \frac{5wL^4}{384EI} = \frac{5 \cdot \frac{mg}{L} \cdot L^4}{384 E\pi R^3 t} = \frac{5 \cdot \frac{\rho\pi t(2R-t)Lg}{L} \cdot L^4}{384 E \pi R^3 t}$

$$= \frac{5}{384} \cdot \frac{\rho(2R-t)gL^3}{ER^3} \leq 0.001$$

5. $\sigma \leq \sigma_{all} \Rightarrow \frac{P}{\pi t(2R-t)} \leq \sigma_{all}$

Translating constraint equations:

$g_1(x) = 1 - \frac{R}{0.02} \leq 0$

$g_2(x) = \frac{R}{0.2} - 1 \leq 0$

$g_3(x) = 1 - \frac{t}{0.001} \leq 0$

$g_4(x) = \frac{t}{0.01} - 1 \leq 0$

$g_5(x) = \frac{R}{18t} - 1 \leq 0$ or: $g_5(x) = R - 18t \leq 0$

$g_6(x) = \frac{5}{0.384} \cdot \frac{\rho(2R-t)gL^3}{ER^3} - 1 \leq 0$ or: $g_6(x) = 5\rho(2R-t)gL^3 - 0.384ER^3 \leq 0$

$g_7(x) = \frac{P}{\pi t(2R-t)\sigma_{all}} - 1 \leq 0$ or: $g_7(x) = 1 - \frac{\pi t(2R-t)\sigma_{all}}{P} \leq 0$

Minimize : $f(x) = \rho \pi t(2R-t)L$, $x = \begin{Bmatrix} R \\ L \end{Bmatrix}$

$L = 3.5$ m
$\sigma_{all} = 405 \cdot 10^6$ Pa
$\rho = 7{,}850$ kg/m^3
$P = 55 \cdot 10^3$ N
$E = 250 \cdot 10^9$ Pa

Alternate equations for g_5, g_6, and g_7 are presented in case design variables in the denominator give ill-conditioned effects. In this case, constraints will also be scaled to better the problem conditioning.

Moving forward, the alternate expressions without design variables in the denominator will be used

The analytic gradients are found to ease computation.

$$\left.\begin{aligned}\frac{df}{dR} &= 2\rho\pi tL \\ \frac{df}{dt} &= -\rho\pi tL + \rho\pi L(2R-t) = 2\rho\pi L(R-t)\end{aligned}\right\}\nabla f$$

$$\left.\begin{aligned}\frac{dg_1}{dR} &= -\frac{1}{0.02} \\ \frac{dg_1}{dt} &= 0\end{aligned}\right\}\nabla g_1$$

$$\left.\begin{aligned}\frac{dg_2}{dR} &= \frac{1}{0.2} \\ \frac{dg_2}{dt} &= 0\end{aligned}\right\}\nabla g_2$$

$$\left.\begin{aligned}\frac{dg_3}{dR} &= 0 \\ \frac{dg_3}{dt} &= -\frac{1}{0.001}\end{aligned}\right\}\nabla g_3$$

$$\left.\begin{aligned}\frac{dg_4}{dR} &= 0 \\ \frac{dg_4}{dt} &= \frac{1}{0.01}\end{aligned}\right\}\nabla g_4$$

$$\left.\begin{aligned}\frac{dg_5}{dR} &= 1 \\ \frac{dg_5}{dt} &= -18\end{aligned}\right\}\nabla g_5$$

$$\left.\begin{aligned}\frac{dg_6}{dR} &= 10\rho gL^3 - 1.152ER^2 \\ \frac{dg_6}{dt} &= -5\rho gL^3\end{aligned}\right\}\nabla g_6$$

$$\left.\begin{aligned}\frac{dg_7}{dR} &= -2\pi t\sigma_{all}/P \\ \frac{dg_7}{dt} &= \frac{(-\pi t\sigma_{all}(-1) - \pi\sigma_{all}(2R-t))}{P} = \frac{-2\pi\sigma_{all}(t-R)}{P}\end{aligned}\right\}\nabla g_7$$

2) For this problem, the Sequential Linear Programming method is used. See attached for Matlab code. The move limits were set constant at [0.1, 0.01], although the method never utilized the full limits during any part of the optimization.

SLP	Run 1	Run 2	Run 3
x_0	[0.14, 0.008]	[0.2, 0.01]	[0.02, 0.0012]
x*	[0.020017, 0.00111205]	[0.020017, 0.00111206]	[0.020017, 0.00111207]
f(x*)	3.736019787	3.7360464	3.7360931
g1(x*)	-8.50669E-05	-8.54236E-04	-8.60498E-04
g2(x*)	-9.94440E-01	-9.94440E-01	-9.94440E-01
g3(x*)	-1.12056E-01	-1.12060E-01	-1.12067E-01
g4(x*)	-8.88794E-01	-8.88794E-01	-8.88793E-01
g5(x*)	3.46945E-18	-3.46944E-18	3.46945E-18
g6(x*)	-1.27411E+05	-1.27417E+05	-1.27427E+05
g7(x*)	-1.29904E-03	-1.30618E-03	-1.31871E-03
# of Iterations	786	944	53
Optimality condition	exitflag = 1 (Optimal solution found)	exitflag = 1 (Optimal solution found)	exitflag = 1 (Optimal solution found)

```
function [f, grad_f] = fx(x, L, sigma, rho, P, E, grav)
%Thomas Satterly. f(x) for AAE 550, HW 2


R = x(1);
t = x(2);

f = rho * pi * t * (2 * R - t) * L;

if nargout == 2
    grad_f(1, 1) = 2 * rho * pi * t * L;
    grad_f(2, 1) = 2 * rho * pi * L * (R - t);
end


end
```

Published with MATLAB® R2016a

```
function [g, h, grad_g, grad_h] = gx(x, L, sigma, rho, P, E, grav)
%Thomas Satterly. g(x) for AAE 550, HW 2

R = x(1);
t = x(2);

g(1, 1) = 1 - (R / 0.02);
g(1, 2) = (t / 0.2) - 1;
g(1, 3) = 1 - (t / 0.001);
g(1, 4) = (t / 0.01) - 1;
g(1, 5) = R - 18 * t;
g(1, 6) = (5 * rho * (2 * R - t) * grav * L^3 - 0.384 * E * R^3);
g(1, 7) = (P - pi * t * (2 * R - t) * sigma) / P;

if nargout >= 2
    h = []; % No equality constraints
end

if nargout >= 3
    grad_g(1, 1) = -1 / 0.02;
    grad_g(2, 1) = 0;

    grad_g(1, 2) = 1 / 0.2;
    grad_g(2, 2) = 0;

    grad_g(1, 3) = 0;
    grad_g(2, 3) = -1 / 0.001;

    grad_g(1, 4) = 0;
    grad_g(2, 4) = 1 / 0.01;

    grad_g(1, 5) = 1;
    grad_g(2, 5) = -18;

    grad_g(1, 6) = (10 * rho * grav * L^3 - 1.152 * E * R^2);
    grad_g(2, 6) = (-5 * rho * grav * L^3);

    grad_g(1, 7) = -2 * pi * t * sigma / P;
    grad_g(2, 7) = -2 * pi * sigma * (t - R) / P;
end

if nargout >= 4
    grad_h = []; % No equality constraints
end


end
```

Published with MATLAB® R2016a

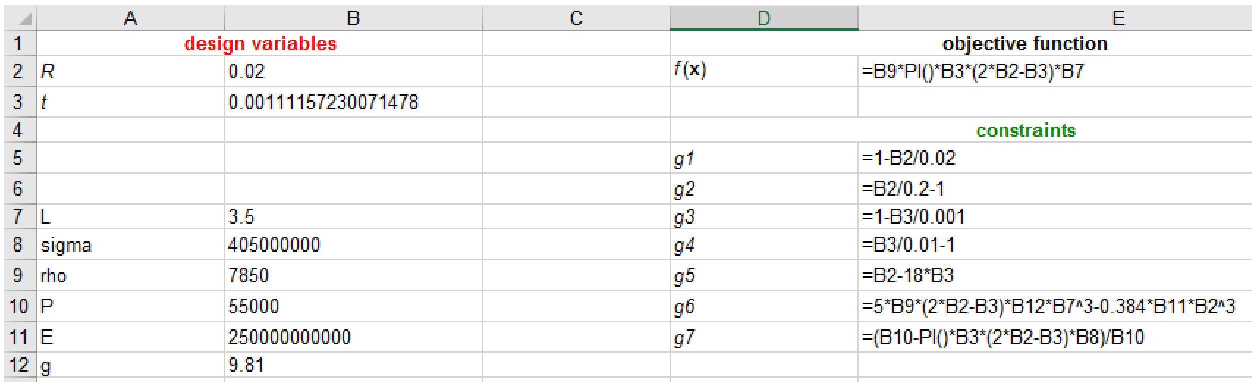
```
%  Thomas Satterly
% AAE 550, HW 2
% SLP example, adapted

close all;
clear;
%
L = 3.5; % m
sigma = 405e6; % Pa
rho = 7850; % kg/m^3
P = 55e3; % N
E = 250e9;  %Pa
grav = 9.81; % m/s^2

% convergence tolerance for change in function value between
 minimizations
epsilon_f = 1e-04;
% convergence tolerance for maximum inequality constraint value
epsilon_g = 1e-04;
% convergence tolerance for maximum equality constraint violation
epsilon_h = 1e-04;
% stopping criterion for maximum number of sequential minimizations
max_ii = 1000;

% set options for linprog to use medium-scale algorithm
% also suppress display during loops
% Use dual-simplex algorithm if using Matlab R2016b or newer
options = optimoptions('linprog','Algorithm','dual-
simplex','Display','iter');

% design variables:
x0 = [0.14; 0.008];   % initial design point
% delta x values for move limits
delta_x = [0.1; 0.01];
% lower bounds from original problem - must enter values, use -Inf if
 none
lb = [-Inf;-Inf];
% upper bounds from original problem - must enter values, use Inf if
 none
ub = [Inf;Inf];

% initial objective function and gradients
[f,gradf] = aae550.hw2.fx(x0, L, sigma, rho, P, E, grav);
% initial constraints and gradients; here, these have been computed
% analytically and are available from example_con
[g, h, gradg, gradh] = aae550.hw2.gx(x0, L, sigma, rho, P, E, grav);

f_last = 1e+5;       % set first value of f_last to large number
ii = 0;              % set counter to zero

while (((abs(f_last - f) >= epsilon_f) | (max(g) >= epsilon_g) | ...
        (max(abs(h)) >= epsilon_h)) & (ii < max_ii))
```

```
   % increment counter
   ii = ii + 1   % no semi-colon to obtain output

   % store 'f_last' value
   f_last = f;

   % linearized objective function follows this format:
   % fhat = gradf(1) * x(1) + gradf(2) * x(2)
   % linprog uses vector of coefficients as input; does not need
constant
   % term
   fhat = gradf;

   % linearized constraints follow this format:
   % ghat_i = gradg_i(1) * x(1) + gradg_i(2) * x(2) ...
   %   ... + (g_i(x0) - gradg_i(1) * x0(1) - gradg_i(2) * x0(2))
   % for linprog, these linear constraints are entered using A * x <=
b
   % format
   % note that Matlab will treat g and h as row vectors, so g' and h'
here
   % makes these column vectors to match class convention
   A = gradg';
   b = gradg' * x0 - g';

   % the example problem has no equality constraints
   Aeq = [];
   beq = [];

   % move limits on LP problem (see slide 23-26 from class 17)
   % combines original problem bounds on x with move limits
   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);

   % This will only provide the solution to the current
approximation.
   % At the new x, evaluate the original objective function, the
original
   % constraints and the gradients of these functions to build the
next
   % approximation.  Compute the real function values at the current
point,
   % set x0 for next approximation to the current x.
   x   % no semi-colon to obtain output
   [f,gradf] = aae550.hw2.fx(x, L, sigma, rho, P, E, grav);
   f = f;   % no semi-colon to obtain output
   [g, h, gradg, gradh] = aae550.hw2.gx(x, L, sigma, rho, P, E,
grav);
   g   % no semi-colon to obtain output
   h   % no semi-colon to obtain output
```

```
    % Scale gradients
    for i = 1:size(gradg, 2)
        cj(i) = norm(gradf) / norm(gradg(:, i));
        newgradg(:, i) = cj(i) * gradg(:, i);
    end
    gradg = newgradg;

    x0 = x;
end
```

Published with MATLAB® R2016a

3) The Excel workbook was set up as seen below:



	A	B	C	D	E
1		design variables			objective function
2	*R*	0.02		*f*(**x**)	=B9*PI()*B3*(2*B2-B3)*B7
3	*t*	0.0011157230071478			
4					constraints
5				*g1*	=1-B2/0.02
6				*g2*	=B2/0.2-1
7	L	3.5		*g3*	=1-B3/0.001
8	sigma	405000000		*g4*	=B3/0.01-1
9	rho	7850		*g5*	=B2-18*B3
10	P	55000		*g6*	=5*B9*(2*B2-B3)*B12*B7^3-0.384*B11*B2^3
11	E	250000000000		*g7*	=(B10-PI()*B3*(2*B2-B3)*B8)/B10
12	g	9.81			

GRG, Excel	Run 1	Run 2	Run 3
x_0	[0.14, 0.008]	[0.2, 0.01]	[0.02, 0.0012]
x*	[0.020003, 0.0011114]	[0.020000, 0.0011116]	[0.020000, 0.0011116]
f(x*)	3.731170775	3.731172268	3.731175477
g1(x*)	-1.50636E-04	0.00000E+00	0.00000E+00
g2(x*)	-8.99985E-01	-9.00000E-01	-9.00000E-01
g3(x*)	-1.11394E-01	-1.11571E-01	-1.11572E-01
g4(x*)	-8.88861E-01	-8.88843E-01	-8.88843E-01
g5(x*)	-2.07190E-06	-8.28370E-06	-8.30141E-06
g6(x*)	-1.26247E+05	-1.26003E+05	-1.26003E+05
g7(x*)	5.53384E-07	1.53232E-07	-7.06840E-07
# of Iterations	5	6	2
Optimality condition	Local minimum found	Local minimum found	Local minimum found

4) See attached for Matlab code

SQP, numeric gradients	Run 1	Run 2	Run 3
x_0	[0.14, 0.008]	[0.2, 0.01]	[0.02, 0.0012]
x*	[0.020000, 0.0011117]	[0.0200000, 0.00111166]	[0.0200000, 0.00111157
f(x*)	3.731652533	3.731483177	3.731172836
g1(x*)	-2.26981E-05	0.00000E+00	0.00000E+00
g2(x*)	-9.94442E-01	-9.94442E-01	-9.94442E-01
g3(x*)	-1.11692E-01	-1.11667E-01	-1.11571E-01
g4(x*)	-8.88831E-01	-8.88833E-01	-8.88843E-01
g5(x*)	-1.00000E-05	-1.00000E-05	-8.28683E-06
g6(x*)	-1.26042E+05	-1.26004E+05	-1.26003E+05
g7(x*)	-1.28564E-04	-8.31743E-05	9.29169E-10
# of Iterations	7	8	2
Optimality condition	exitflag = 1 (Optimal solution found)	exitflag = 1 (Optimal solution found)	exitflag = 1 (Optimal solution found)

```
%  Thomas Satterly
% AAE 550, HW 2
% SQP example, adapted for numeric gradients
clear all

L = 3.5; % m
sigma = 405e6; % Pa
rho = 7850; % kg/m^3
P = 55e3; % N
E = 250e9;  %Pa
grav = 9.81; % m/s^2

f_x = @(x) aae550.hw2.fx(x, L, sigma, rho, P, E, grav);
g_x = @(x) aae550.hw2.gx(x, L, sigma, rho, P, E, grav);



% no linear inequality constraints
A = [];
b = [];
% no linear equality constraints
Aeq = [];
beq = [];
% lower bounds (no explicit bounds in example)
lb = [ ];
% upper bounds (no explicit bounds in example)
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 = [0.02; 0.0012];
[x,fval,exitflag,output] = fmincon(f_x,x0,A,b,Aeq,beq,lb,ub, ...
    g_x,options)

% NOTES:  since this is a direct constrained minimization method, you
% should try several initial design points to be sure that you have
 not
% located a local minimum.
```

Published with MATLAB® R2016a

5) See attached for Matlab code

SQP, analytic gradients	Run 1	Run 2	Run 3
x_0	[0.14, 0.008]	[0.2, 0.01]	[0.02, 0.0012]
x*	[0.0200000, 0.0011117]	[0.0200000, 0.0011117]	[0.0200000, 0.00111167]
f(x*)	3.731483177	3.731483177	3.731483177
g1(x*)	0.00000E+00	0.00000E+00	0.00000E+00
g2(x*)	-9.94442E-01	-9.94442E-01	-9.94442E-01
g3(x*)	-1.11667E-01	-1.11667E-01	-1.11667E-01
g4(x*)	-8.88333E-01	-8.88833E-01	-8.88833E-01
g5(x*)	-1.00000E-05	-1.00000E-05	-1.00000E-05
g6(x*)	-1.26004E+05	-1.26004E+05	-1.26004E+05
g7(x*)	-8.31742E-05	-8.31742E-05	-8.31742E-05
# of Iterations	9	10	4
Optimality condition	exitflag = 1 (Optimal solution found)	exitflag = 1 (Optimal solution found)	exitflag = 1 (Optimal solution found)

```
%  Thomas Satterly
% AAE 550, HW 2
% SQP example, adapted for analytic gradients
clear all

L = 3.5; % m
sigma = 405e6; % Pa
rho = 7850; % kg/m^3
P = 55e3; % N
E = 250e9;  %Pa
grav = 9.81; % m/s^2

f_x = @(x) aae550.hw2.fx(x, L, sigma, rho, P, E, grav);
g_x = @(x) aae550.hw2.gx(x, L, sigma, rho, P, E, grav);



% no linear inequality constraints
A = [];
b = [];
% no linear equality constraints
Aeq = [];
beq = [];
% lower bounds (no explicit bounds in example)
lb = [ ];
% upper bounds (no explicit bounds in example)
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', ...
    'SpecifyObjectiveGradient', true, ...
    'SpecifyConstraintGradient', true);
% initial guess  - note that this is infeasible
x0 = [0.02; 0.0012];
[x,fval,exitflag,output] = fmincon(f_x,x0,A,b,Aeq,beq,lb,ub, ...
    g_x,options)

% NOTES:  since this is a direct constrained minimization method, you
% should try several initial design points to be sure that you have
 not
% located a local minimum.
```

Published with MATLAB® R2016a

6) While choosing a different x^0 does produce different optimization results, the final values for x^* typically vary by less than 0.003%, making the solution difference negligible. However, different x^0 values can significantly impact how quickly an optimization method is able to find an optimal solution. For example, the number of iterations required for the SLP method to find an optimal solution varied between 786 and 53 iterations depending on how close the initial point was to the optimal solution. Comparatively, the SQP method only took between 4 and 10 iterations for the same set of initial points. This highlights the weakness of sequential linear programming optimization, in that finding the optimal solution can be slow if the objective and constraint functions are near linear.

   Based on the different starting points used, the GRG method consistently used the fewest number of iterations to reach an optimal solution. However, it should be noted that the SQP method, using either numerical or analytic gradients, still had very few iterations on the same order of the GRG method. For a balance between the most consistent results regardless of starting point and the fewest iterations, the SQP method with analytic gradients would seem like the most desirable choice.

   For this problem, the gradients of the constraint function required scaling to be on the same order of the objective function in order for the SLP method to converge. Otherwise, the method would continuously hop between two points which surround the optimal point. Other means of eliminating this behavior were attempted, including reducing the move limits if the algorithm was hopping between two points and the current point was valid, but this would run into errors of still hopping out of bounds and failing to find a path back into feasible space. Constraint and objective function scaling was also attempted, but in the end, constraint gradient scaling proved the most reliable. All other methods needed no such additional treatment.