```matlab
%  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