ME46060 “Engineering Optimization – Concepts and Applications” Exercise 7

Subject: “Valve spring”

Objectives

- Linearization of constrained optimization problem.
- Solve such problems by sequential linear programming.
- Introduce move limits.

Course material

- Section 5.8 of Chapter 5, Papalambros and Wilde [1]
- Section 6.3 of Chapter 6, Haftka and Gürdal [2].

Introduction

We consider the same problem as in exercise 6, except some enlargement of the set constraint of design variables x(1) and x(2):

Minimize f(**x**) = smass (spring mass)
Subject to:
L2 ≥ Lmin
F1 ≥ F1min
F2 ≥ F2min
Tau2 ≤ Tau12max (= 600e+6 N/m^2) (1)
freq1 ≥ freq1lb
Set constraint:
0.01 ≤ x(1) ≤ 0.04 (m)
0.002 ≤ x(2) ≤ 0.006 (m)

Row vector **x** represents the design variables, being the coil diameter D (= x(1)) and the wire diameter d (= x(2)) respectively. The other parameters are given in the file springparams2.m. This week we will use linear programming to solve the problem. Therefore the problem is linearized as follows around the initial design point x0:

$$flin(x) = f(x0) + \sum_{i=1}^{2} \frac{df}{dx_i}(x_i - x0_1) \qquad (2)$$

$$glin_j(x) = g_j(x0) + \sum_{i=1}^{2} \frac{dg_j}{dx_i}(x_i - x0_i)\ , \qquad j = 1...5 \qquad (3)$$

flin and *glin* are programmed in the files flinw7.m and glinw7.m respectively. The derivatives *df/dx* and *dg/dx* are programmed in the files dfw7ex1.m and dgw7ex1.m respectively. The program springw7ex1.m visualizes both the original (solid lines; files springobj2.m, springcon5.m and springparams2.m are used) and the linearized problem (dotted lines). The point x0 is indicated with a marker.
Have short a look at the programs flinw7.m, glinw7.m, dfw7ex1.m and dgw7ex1.m; do not just use them as black boxes.

Exercise 7.1: Sequential solution of linearized spring optimization problem by inspection.

Start Matlab and choose x0 = [0.035 0.0045] (before running springw7ex1.m). Next run springw7ex1.m and compare the original and the linearized problem. Note that some linearized constraints move into the infeasible region of the original problem, while others move into the feasible region. Can you explain that?

Estimate from the figure the solution of the linearized problem. Change x0 to this point (tip: use x0=ginput(1);) and rerun the program. Repeat that cycle until (visual) convergence. How many cycles are necessary?

Exercise 7.2: Solution by means of sequential linear programming.

Extend the file springw7ex1.m to springw7ex2.m in order to solve the linearized problem using the LP algorithm linprog of Matlab and the given files dfw7ex1.m and dgw7ex1. First, carefully derive the vectors and matrices which have to be given as input parameters of linprog. Use the set constraint given above as lower and upper bounds of the design variables. At the end of the optimization cycle update the starting point, x0, to the current solution point.

Run a number of cycles using the program, starting from the initial point x0 = [0.035 0.0045]. Compare the results with those of exercise 1.

Exercise 7.3: Implementation of move limits and constraint relaxation.

Move limits build a (usually small) area around the current design point in which the design variables are allowed to move. Extend your program springw7ex2.m to springw7ex3.m by implementing the following move limits (modify the lower and upper bounds input parameters of linprog):

x0(1) – ml(1) ≤ `x(1) ≤ x0(1) + ml(1)`

x0(2) – ml(2) ≤ `x(2) ≤ x0(2) + ml(2)`

where ml represents the move limits. Use ml = [0.003 0.0003]. Assure (in your program) that the original set constraint cannot be violated by applying the move limits.

Use x0 = [0.030 0.0045] as initial point, and run a number of cycles until convergence. See if you can spot the effect of the move limit.
Now try x0 = [0.035 0.0045] as initial point. What happens? Modify your program springw7ex3.m to relax the constraints to prevent this issue, by introducing an extra variable β. Constraint relaxation gives the following problem:

$$\min_{\mathbf{x},\beta} \quad f(\mathbf{x}_k) + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_k (\mathbf{x}-\mathbf{x}_k) + k\beta$$

$$\text{s. t.} \quad \mathbf{g}(\mathbf{x}_k) + \left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right|_k (\mathbf{x}-\mathbf{x}_k) - \beta\mathbf{1} \le \mathbf{0}$$

Note that β is a variable just like **x**, so the input for linprog must be adjusted accordingly. Use k=10 and $0 \leq \beta \leq$inf (i.e. also include lower and upper bounds for the new variable). Rerun the program from initial point x0 = [0.035 0.0045], and check and describe the improvement.

References

[1] Papalambros, P.Y. and Wilde, D.J., *Principles of optimal design: modeling and computation,* 3rd edition, Cambridge University Press, 2017.
[2] Haftka, R.T. and Gürdal, Z., Elements of Structural Optimization, 3rd edition, Kluwer Academic Publishers, 1992.