AVT | SVT | RWTH Aachen | 52074 Aachen

Systemverfahrenstechnik

Prof. Alexander Mitsos, Ph.D.

Bonus Point Programming Exercise 2

Deadline: January 20, 2021, 24:00 h

Applied Numerical Optimization

Wintersemester 2020/2021

Rules for bonus point exercises

- Please work on the bonus point exercise in groups of 2, 3 or 4 students. If you cannot find a group, use the forum or send an email to optimierung.svt@avt.rwth-aachen.de
- **One member per group** should submit the solution (typically, one or more MATLAB '.m' files) on Moodle before the deadline. The names and enrollment-numbers (or TIM-number, in case no enrollment-number is available) of the group members should be written as comments at the top of the '.m' file.
- Please take care that your code is well-documented (through comments within the source code) and executes out of the box. The results of the second bonus points exercise will be published by January 31, 2021, on RWTHmoodle.

Background

In Lecture 10, you will learn about methods for deterministic global optimization of non-linear **non-convex** functions. One of these methods is the Branch-and-Bound method where the feasible set is iteratively branched and pruned based on a check against estimated upper and lower bounds on the optimal solution. In this exercise, you will implement the bounding procedure for an example problem.

Exercise 1: Convex underestimation and upper bounds

Problem description. Consider the optimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \tag{1a}$$

$$\text{s.t. } \boldsymbol{x}^T \boldsymbol{Q}_i \boldsymbol{x} + \boldsymbol{a}_i \boldsymbol{x} = b_i \;\forall\, i \in \{1, ..., m\} \tag{1b}$$

$$\boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U \tag{1c}$$

with $n, m \in \mathbb{N}$, $m < n$, a symmetric, positive semidefinite matrix $\boldsymbol{H} \in \mathbb{R}^{n \times n}$, symmetric indefinite matrices $\boldsymbol{Q}_i \in \mathbb{R}^{n \times n}\, \forall i \in \{1, ..., m\}$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with row vectors $\boldsymbol{a}_i$, and $\boldsymbol{b} \in \mathbb{R}^m$. The above problem is a nonconvex optimization problem with the possibility of suboptimal local minima.

Example 1.

$$\min_{\boldsymbol{x} \in \mathbb{R}^3} \; x_1 + x_2 + x_3^2 \tag{2a}$$

$$\text{s.t. } x_1 x_2 + x_3 = 8 \tag{2b}$$

$$x_2 x_3 = 15 \tag{2c}$$

$$0 \leq x_1, x_2, x_3 \leq 10 \tag{2d}$$

Example 2.

$$\min_{\boldsymbol{x} \in \mathbb{R}^4} \; x_1 + x_2 + x_3^2 + x_4^2 \tag{3a}$$

$$\text{s.t. } x_1 x_2 + x_2 x_3 = 2 \tag{3b}$$

$$x_1 x_2 + x_4 = 3 \tag{3c}$$

$$x_1 + x_2 x_3 = 5 \tag{3d}$$

$$0 \leq x_1, x_3, x_4 \leq 10 \tag{3e}$$

$$0 \leq x_2 \leq 4 \tag{3f}$$

The terms $x_1 x_2$ and $x_2 x_3$, which are responsible for the non-convexity of the model, are called *bilinear* terms. In example 1, the matrices and vectors in the general problem definition would be

$$\begin{gathered}
\boldsymbol{c} = [1, 1, 0]^T, \qquad \boldsymbol{b} = [8, 15]^T, \qquad \boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\
\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{Q}_1 = \begin{bmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{Q}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0.5 & 0 \end{bmatrix}
\end{gathered} \tag{4}$$

The Algorithm. In order to get a lower bound on the objective value of the nonconvex optimization problem, we need to construct a *convex relaxation* and solve it to global optimality. One widely used relaxation of the *bilinear* terms $x_i x_j$ are the so-called *McCormick Envelopes*

[1]. The relaxation works as follows: each term $x_i x_j$ is replaced by an auxiliary variable w_{ij}. Then, the following constraints for w_{ij} are defined:

$$w_{ij} \geq x_i^L x_j + x_i x_j^L - x_i^L x_j^L \tag{5a}$$
$$w_{ij} \geq x_i^U x_j + x_i x_j^U - x_i^U x_j^U \tag{5b}$$
$$w_{ij} \leq x_i^U x_j + x_i x_j^L - x_i^U x_j^L \tag{5c}$$
$$w_{ij} \leq x_i^L x_j + x_i x_j^U - x_i^L x_j^U \tag{5d}$$

These constraints are added to the original optimization problem, and the bilinear terms are replaced with the variables w_{ij}. The resulting convex QP (quadratic objective & **linear** constraints) is solved using standard methods, yielding a lower bound on the objective value of (1).

Your Task is to implement the following function in Matlab

```
function [f_lb, f_ub] = convex_bound(n, m, c, H, Q, A, b, lb, ub)
```

Here, $\boldsymbol{Q} \in \mathbb{R}^{m \cdot n \times n}$ are the constraint matrices $\boldsymbol{Q}_i \in \mathbb{R}^{n \times n}$ vertically concatenated, such that $\boldsymbol{Q}_i$ is the i-th submatrix of $\boldsymbol{Q}$. This function must return a lower bound `f_lb` and an upper bound `f_ub` on the optimal value of the optimization problem (1).

To implement your function, follow the steps below:

1. Check the inputs for correct lengths. $\boldsymbol{Q}_i$ should be *symmetric*.
2. Analyze the matrix $\boldsymbol{Q}$ for how many auxiliary variables w need to be generated. It is extremely important that no auxiliary variable is generated twice (why?). Note that $\boldsymbol{H}$ is guaranteed to be positive semidefinite.
3. Generate an additional matrix $\boldsymbol{B}$ that implements the inequality constraints (5).
4. Compose the convex QP and solve it using Matlab's `quadprog` function. (QP solver)
5. Apply the local solver `fmincon` to the **original** problem to compute an **upper bound** on the problem. (general NLP solver)

Implement your function generically so that you can run both **example 1 and 2**.

Hint 1: Solution values

Valid upper and lower bounds for Example 1 are 12.28 and 4.40, respectively.
For example 2, valid upper and lower bounds are 6.4 and 6.2, respectively.

Hint 2: Testing of existence of $\mathbf{w_{jk}}$ in `convex_bound`

The purpose of the following code fragment is the check whether the auxiliary variable w_{jk} already exists. The existence is stored int the Matlab variable w_combinations that is created on the fly late in line 11. Thus in line 3, w_combinations might not exist which is checked by the command if(exist('w_combinations')) in line 1. However the use of the Matlab command is exist is unsafe, since it might also check for a file with the name "w_combinations" in the current directory.

The remedy is to drop the check in line 1 and replace it by the initialization w_combinations = [];.

```
1  if(exist('w_combinations'))
2  for l=1:count-1
3  if((j==w_combinations(l,1)&& k==w_combinations(l,2)))
4  combination_exists=1;
5  save_l=l;
6  end
7  end
8  % ....
9  else
10 % Save combinations
11 w_combinations(count,1)=j;
12 w_combinations(count,2)=k;
13 % write A
14 if(j==k)
15 % Do not double diagonal entries
16 A_help(i,count)=Q((i-1)*n+j,k);
17 else
18 A_help(i,count)=2*Q((i-1)*n+j,k);
19 end
20 end
```

Expected input for problem 1

```
% Problem 1
n = 3;
m = 2;
c = [1; 1; 0];
b = [8;15];
A = zeros(2,3);
A(1,3) = 1;
H = zeros(3,3);
H(3,3) = 1;
Q1 = zeros(3,3);
Q1(1,2) = 0.5;
Q1(2,1) = 0.5;
Q2 = zeros(3,3);
Q2(3,2) = 0.5;
Q2(2,3) = 0.5;
Q = [Q1; Q2];
% bounds
lb = [0;0;0];
ub = [10;10;10];

[f_lb, f_ub] = convex_bound(n, m, c, H, Q, A, b, lb, ub);
```

References

[1] Garth P. McCormick. Computability of global solutions to factorable nonconvex programs. *Mathematical Programming*, 1976.