Umeå universitet
Institutionen för matematik
och matematisk statistik
P. H. Lundow

Continuous optimization
Fall 2023
Laboration 1
Pear-shaped

1 "... and then it all went pear-shaped."

The pear curve C is defined as the points (x_1, x_2) satisfying

$$g(x) = \left(x_1^2 + x_2^2\right)\left(1 + 2x_1 + 5x_1^2 + 6x_1^3 + 6x_1^4 + 4x_1^5 + x_1^6 - 3x_2^2 + 2x_2^4 + x_2^6 \right.$$
$$\left. -2x_1x_2^2 + 4x_1x_2^4 + 8x_1^2x_2^2 + 3x_1^2x_2^4 + 8x_1^3x_2^2 + 3x_1^4x_2^2\right) - 2 = 0$$

which is the boundary of the sublevel set

$$X = \{x \in \mathbb{R}^2 : g(x) \leq 0\}$$

shown in Figure 1. We define the functions $d, D : \mathbb{R}^2 \to \mathbb{R}$ as

$$d(p) = \min_{x \in X} \|p - x\|, \quad D(p) = \max_{x \in X} \|p - x\|,$$

so that $d(p)$ is the distance from a point $p \in \mathbb{R}^2$ to the *nearest* point in X and $D(p)$ is the distance from p to the *farthest* point in X. This laboration is about computing $d(p)$ and $D(p)$ for a range of points p.

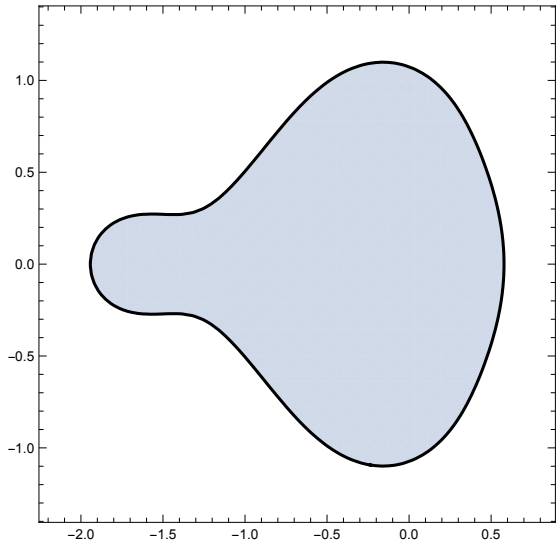

Figure 1: X is indeed pear-shaped.

2 Before the heavy lifting starts, some warmups

Before we get our teeth into the pear region, let us first have a quick look at the functions $d(p)$ and $D(p)$ for some other set X. Let $p \in \mathbb{R}^2$ and $X \subseteq \mathbb{R}^2$ be any set. Here are the most important properties of the functions above:

- $d(p)$ is a convex function if $X \subseteq \mathbb{R}^2$ is a convex set
- $D(p)$ is a convex function for any set $X \subseteq \mathbb{R}^2$

This can, of course, be generalized to p and X in n-dimensional Euclidean space. We can also replace the norm function with other convex functions but that takes a little more explaining. It is quite interesting that the max-function has this amazing property.

Try the following short Matlab experiments: Let X be, say, 10 random points in $[0,1] \times [0,1]$ and define $D(p)$ as the maximum distance from p to a point in X. Plot D over some range in the plane:

```
X = rand(2,10);
D = @(p1,p2) max(vecnorm([p1;p2] - X));
fsurf(D, [0,1,0,1])
```

Try the same with the minimum distance. The difference is striking: D is nice and convex whereas d is (probably) anything but. For d the problem is that a set X of scattered points is not convex. If we replaced X with, say, a disk or a rectangle, then d would be convex too. Remember, d is convex *if* X is convex. What happens when X is not convex depends on other properties of X.

Another thing to keep in mind: when we say a function is convex, we mean that it is convex over a convex set. Otherwise the definition of a convex function does not make sense, since any line segment between two points in the function's domain must remain in the domain. For example, if we plot D over a nonconvex set, like a circle, the function does not have to look convex at all:

```
fplot(@(t)D(cos(t),sin(t)),[0,2*pi])
```

However, plotting over any convex set, like a line, preserves convexity:

```
fplot(@(t)D(t,t),[0,1])
```

This of course works for any line in the plane.

Now we are ready for the Laboration.

3 Let's go to work

Henceforth we let X denote the pear-shaped region in Figure 1. Consider now the plots of Figure 2. The left plot shows an example of drawing a line between $p = (0, 1.5)$ and its nearest point in X giving $d(p) \approx 0.4144$. The right plot shows a line to the farthest point in X giving $D(p) \approx 2.6053$.

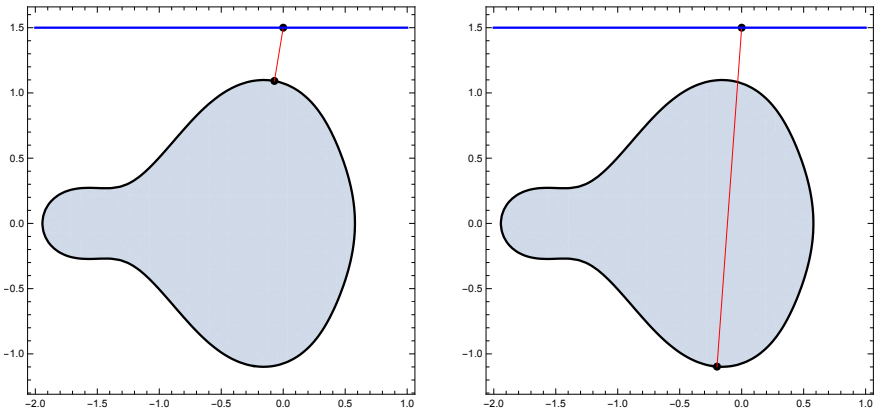

Figure 2: Left: Red line between point $(0, 1.5)$ and the nearest point $x \in X$. Right: Red line between $(0, 1.5)$ and the farthest point $x \in X$.

The first part of your mission, should you choose to accept it, is to complete both pictures for points $p = (t, 1.5)$ with $t = -2.0, -1.9, -1.8, \ldots, 1.0$ (31 points along the horisontal line $(t, 1.5)$). When you have found the nearest and farthest points, plot $d(p)$ and $D(p)$ versus t in separate figures. Of course, D must be convex but will d be convex?

For $t \leq t_1$ the farthest point is at the bottom of the fat part of X and for $t > t_1$ we expect the farthest point to be at the left-most narrow part of X. Locate t_1 as precisely as possible.

The second part of your mission, is to compute $d(p)$ and $D(p)$ (finding the nearest and farthest points) for $p = (1, t)$, with $t = -4.0, -3.8, -3.6, \ldots, 4.0$ (41 points along the vertical line $(1, t)$). As in the first part, draw lines between p and the corresponding nearest and farthest points in separate figures. Plot D and d versus t in separate figures. Is d convex in t?

When increasing t (here we focus only on $t > 0$ due to symmetry), near some $t = t_2$, we expect the farthest point to jump from the left-most narrow part to the bottom of the fat part of X. Locate this t_2 as precisely as possible.

4 Technical stuff

We here describe the computation of $D(p)$ and the corresponding farthest point $x^* \in X$. You need to make the necessary adjustments for computing $d(p)$ and the corresponding nearest point.

Let $p = (p_1, p_2)$ be some fixed point. We want to minimize the negative square distance

$$f(x) = -\|p - x\|^2 = -(x_1 - p_1)^2 - (x_2 - p_2)^2$$

over the points $x = (x_1, x_2)$ in X, so that

$$-D(p)^2 = \min_{x \in X} f(x)$$

The Lagrange function is

$$\mathcal{L}(x, v) = f(x) + v\, g(x)$$

and the dual function is

$$h(v) = \min_{x \in \mathbb{R}^2} \mathcal{L}(x, v), \quad v \geq 0$$

where the sub-problem is to find the optimum $x(v)$ so that

$$x(v) = \arg\min_x \mathcal{L}(x, v)$$
$$h(v) = \mathcal{L}(x(v), v)$$

The real problem lies in finding $x(v)$ since $\mathcal{L}(x, v)$ is not convex in x. Algorithms like Steepest Descent or Newton's Method can only search for stationary points where $\nabla_x \mathcal{L}(x, v) = 0$ and there will probably be several of them, depending on v. Also, when changing the parameter v the stationary points move. To find the correct stationary point we need to use several starting points for our algorithm and then choose the best stationary point found, that is, the one that gave the least value of $\mathcal{L}(x, v)$, thus giving us $h(v)$.

Then we need to repeat this until we find the v giving the maximum $h(v)$, that is, $w^* = h(v^*)$. If strong duality holds then we can be (fairly) sure of having found the correct value of $-D(p)^2$ so that $D(p) = \sqrt{-w^*}$ and $x^* = x(v^*)$. Then we can at least be sure of having found a KKT-point (within numerical tolerance).

You need to write code for the following:

- Finding a stationary point with Newton's method (step length 1)
- Computing $h(v) = \min_x \mathcal{L}(x, v)$ for a given v by running the stationary point search from many different starting points and then select the minimum value as $h(v)$
- Computing $w^* = \max_{v \geq 0} h(v)$ using interval halving
- Checking for strong duality

Make the code flexible and readable so that you can adjust for numerical tolerance and use different choices of starting points. Do not try to solve everything with one big main program but break it down into manageable functions with a good choice of arguments. Develop the code from the ground up, start with a function running Newton's method and make sure that it works properly. Then move on to the next level and so on.

5 The Report

The report should be kept as short and succinct as possible, like a technical report that your reviewer can read and appreciate, written in Proper English. Give the answers to the questions and illustrate with nice figures with clear captions. Please comment on any difficulties or unforeseen complexities you encountered during the laboration. If you also want to investigate a variant of the problems that you found interesting, feel free to do so but place this in separate chapters. Using LaTeX almost guarantees an approved report and is thus very much recommended but is of course not absolutely necessary[1].

The code should be written in Matlab. Put the relevant code (no need to include the plot commands) at the end of the report as an appendix. Split the code into convenient functions instead of using the monolithic style.

You may work alone or in teams up to three, but two is probably best. One person from each group must send me an email with the names of the members (including one-member teams). To get good reports we will use peer-review. We will pair you with another group well before the deadline, saying which group reviews which. The review is a written one-page document containing critique, suggestions for improvement and comments on any errors in the text or code.

As reviewer your job is to check that all tasks have been completed, all questions answered and all figures included, all presented in good style. Remember that nitpicking is always appreciated, especially by the nitpicked. The reviews are then returned at least a few days before the deadline so that each group can take the critique into account and update the report, unless they think the critique is misguided. Write a response to the review as a separate one-page document. The final version of the report is then submitted to Canvas before the deadline. What you upload to Canvas is **one (1) pdf-file** (all else is ignored) containing the following:

1. a cover-sheet with the names of the team members
2. the review by the other team
3. your response to their critique
4. the updated and now (probably) perfect report with code in the appendix

Only *one* person from each group submits the file. The deadline is 17.00 on October 17. Only very minor corrections may be handed in after we have evaluated the report but everything should be completed no later than October 30. After that you will be assigned a *new* laboration and its report should be handed in no later than November 30.

[1] I'm joking, of course it is. Try overleaf.com.