MAA209 Numerical Optimization

Beniamin Bogosel, École Polytechnique

Practical Session #1

Instructions: The codes should be written in Python. The use of notebooks is encouraged and you may start from the given examples. Collaboration and discussions are encouraged, if they are related to the subjects concerning the practical sessions.

Important: Show your work to the instructor **before the end of the practical session.**

Students which in addition to the main subjects solve some **supplementary** or **Challenge** questions will get additional bonus points. Recall however, that the **Challenge** and **supplementary** exercises are not mandatory.

Exercise 1. Simple line-search methods for unimodal functions

1. Write codes implementing the following zero order methods shown in the course:

- the Golden search method (you may start from the Notebook `Golden_search_Start.ipynb`)
- the Fibonacci search (you may start from the Notebook `Fibonacci_Start.ipynb`)
- the bisection method (you may start from the Notebook `Bisection_Start.ipynb`)

and run them for several unimodal functions of your choice. The trisection method is shown as an example in the Notebook `Trisection.ipynb`. The stopping criterion of a bracketing algorithm is the **length of the bracketing interval**.

Important: Pay attention to the number of function evaluations you perform in each of the given algorithms. You can **define a counter** in order to keep track.

Test numerically their convergence rates and compare your findings with the theoretical results. It is recommended to structure your code such that the function f to be optimized as well as the initial search interval $[a, b]$ can be easily changed. For example you may use Python functions to define your optimization algorithm and the function f.

Some test cases are given below. In each case the analytic solution can be found. Verify the precision of the optimization algorithms.

(a) $f(x) = (x-\alpha)^2 + \beta$, $[a, b] = [-2, 1]$, $\alpha \in (-0.5, 0.5)$, $\beta \in \mathbb{R}$.

(b) $f(x) = \sqrt{|x-1|}$ with starting guess $[a, b] = [-2.1, 2]$. Note that in this case f is not C^1 at the optimum. Does the bisection method work in this case?

(c) $f : [0, 10] \to \mathbb{R}$, $f(x) = \begin{cases} \sin(1/x) & \text{if } x > 0 \\ f(0) = 0 \end{cases}$. Run the algorithms with $[a, b] = [0, 1]$,$[0, 4]$, $[0, 10]$. Why do you get different results?

2. Compare the methods in the following aspects:

- The Fibonacci search is run when the number of function evaluations N is known in advance. For a chosen unimodal function, run the Fibonacci search with $N = 25$ function evaluations and stop the trisection and golden search methods when the number $N = 25$ of function evaluations is reached. Compare the precision of the numerical results with respect to the analytical solution.
- Run the Golden search (or Fibonacci) and bisection methods for the function

$$f(x) = -\cos x \text{ on } [a, b] = [-2, 1].$$

Try to be as precise as possible. When using NumPy you should note that Golden search does not do a very good job and you will not obtain a better approximation than $\sqrt{\varepsilon}$ where ε is the machine precision (typically $1e-16$). Try to find the minimum with the bisection algorithm. Does this algorithm do a better job? Explain the phenomenon by looking at the arguments given in the course.

Hint: Around a minimum x^* the second order Taylor expansion tells us that $f(x) \approx f(x^*) + \frac{1}{2}f''(x^*)(x-x^*)^2$. If the second term is ε times the first, where ε is the machine precision (typically around 10^{-16}) then $f(x)$ will have the same value as $f(x^*)$. Check that this is the case for the function considered here.

Exercise 2. Build a bridge

Two cities A and B are separated by a river of width w. The two communities want to build a road which links the two cities and a bridge which will cross the river. The bridge should be orthogonal to the two banks of the river. There are no other constraints for the construction of the roads on land.

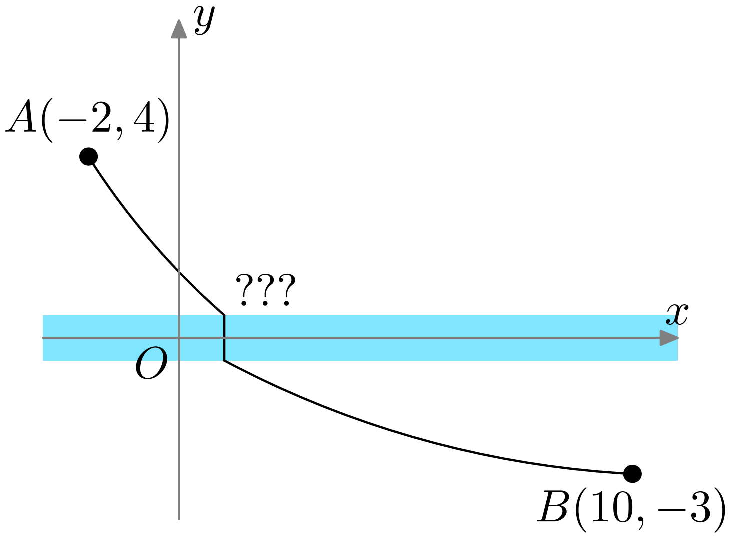
Due to obvious budget issues the cost of this project, and therefore, the length of this road, should be minimal. You should formulate an optimization problem which solves the problem and finds the position of the bridge which minimizes the construction cost.

In order to structure your ideas you may suppose that A and B are two points in the plane with coordinates $A(-2,4), B(10,-3)$. The river has width $w = 1$ and is centered along the Ox axis. (see the Figure below)

1. It is possible to solve analytically the problem using various methods. You may try one of the following two approaches:

(a) Denote by x the x-coordinate of the bridge and find the length of the road $\ell(x)$ in terms of b. Prove that $\ell(x)$ is a regular convex function and find its minimum.

(b) You may also reason geometrically: the bridge is orthogonal to the banks of the river, so it has a fixed length. You may translate one half of the picture so that the two banks **coincide**. What is then the optimal path between A and B? Use this to find the minimal length of the road.

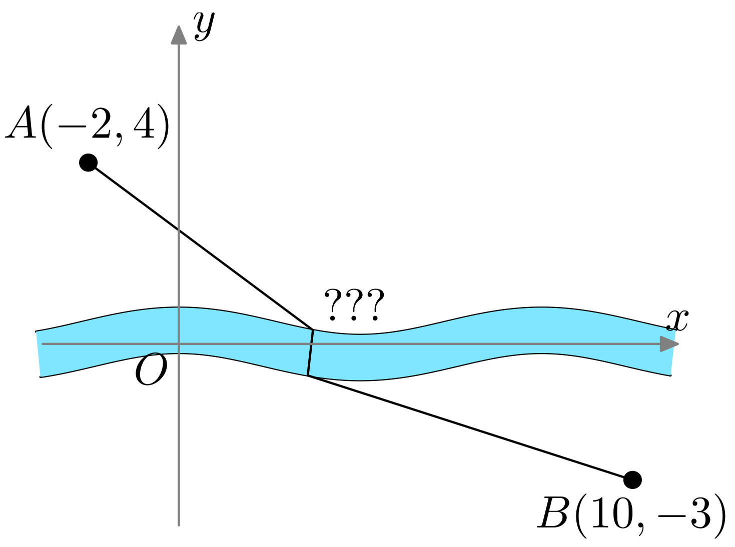


2. Formulate a numerical algorithm which solves the problem using one of the line-search methods coded in the previous exercise. Write the corresponding objective function and choose a proper initialization for the search interval $[a,b]$. Compare your result with the analytical solution. Draw a picture in Python showing the two cities A, B, the river and the optimal road.

3. **(Challenge)** What happens when the river is not straight? Suppose that the river is of constant width 1 and its center follows a path $(x, \gamma(x))$ in the plane. The width is computed always in the normal direction to the graph of $\gamma(x)$. You may take an ondulating path like $\gamma(x) = 0.3\sin(0.25(x+1))$. As above, find a formula for the length of the road in terms of x and try to optimize the length of the road in terms of x.

The use of an algorithm without derivatives should facilitate your task. In the end make a plot in Python of the configuration and the optimal result.

Hints: The width is always computed in the normal direction to the curve defined by γ. Recall that a tangent vector to the graph of γ is given by $(1, \gamma'(x))$. This can help you obtain a unit normal vector and find parametrization of the upper and lower banks in terms of γ and γ'.



Hints:

1.

(a) The road of shortest length on each side of the river is a straight segment. Therefore, if x is the coordinate of the bridge then the optimal road is given by the points

$$(-2,4),(x,0.5),(x,-0.5),(10,-3).$$

Now it is straightforward to compute the length of the road as the sum of lengths of the segments between the points given above. This is an explicit function of x and minimizing it is straightforward.

(b) Suppose that the region above the river is translated by the vector $(0,-1)$ so that the banks coincide. Then the road of minimal length may be obtained by joining the points $A'(-2,3)$ and $B(10,-3)$. The x coordinate is found by intersecting the segment $A'B$ and the line $y=-0.5$. You should get the same solution as for part (a).

2. Convince yourself that the length of the road is unimodal in terms of the x-coordinate. Then implement this function in Python and use one of the algorithms developed previously.

Compare the numerical value with the analytical result obtained previously.

3. When the river is not straight. The center is on the path $(x,\gamma(x))$. A normal vector is $(-\gamma'(x),1)$ and normalized unit vectors on the two sides of the curve are given by

$$n_{1,2}(x)=\pm\frac{1}{\sqrt{1+(\gamma'(x))^2}}(-\gamma'(x),1)$$

Thus, at coordinate x the extremities of the bridge have coordinates

$$(x,\gamma(x))+0.5n_1(x) \text{ and } (x,\gamma(x))+0.5n_2(x).$$

As in the part 1. (a) it is possible to write explicitly the length of the road in this case, in terms of x.

Exercise 3. Approximate eigenvalues of a matrix (Challenge)

It is possible to use some of the algorithms implemented previously in order to compute the eigenvalues of some symmetric matrix. Consider for $N\geq 2$ the matrix

$$A=\begin{pmatrix} 2 & -1 & 0 & \dots & 0 & 0\\ -1 & 2 & -1 & \dots & 0 & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & 0 & \dots & -1 & 2\end{pmatrix}$$

which comes from the discretization of the Laplace operator in dimension one. This matrix is symmetric and therefore has real eigenvalues. The goal of this exercise is to be able to approximate some of them.

1. (Optional) Use the Gershgorin circle theorem to prove that the eigenvalues of the matrix A belong to the interval $[0,4]$.

2. Use Numpy functions in order to approximate the eigenvalues of A for reasonably large $N\leq 100$.

3. Choose an interval around one of the observed eigenvalues and perform a search for the minimizer of the functions

$$f(t) = |\det(A - tI)| \text{ and } g(t) = \log f(t).$$

4. What is the maximal precision that you can achieve using the minimization of f? What about g? Interpret your results in terms of the observations shown in the course regarding the precision of the zero-order bracketing algorithms.