MAA209 Numerical Optimization

École Polytechnique

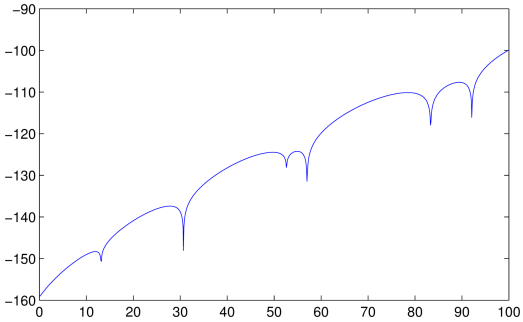
Final Evaluation 2023

Instructions:

- The evaluation starts on May 25th 2023 at 8AM (the beginning of the last practical session). You have 10 days to work on the project. The Deadline is June 4th at midnight. Late submissions will be penalized.
- All course materials and codes from practical sessions can be used. The work is **individual**, collaborations are not allowed. Cheating in any way leads to a failing grade for the course. Similarities between submitted notebooks of different students will be carefully investigated. Be original and do your own work.
- The answers (to questions/problems) should be written in a single Python Notebook and submitted on Moodle. Answers to theoretical questions should be written in Markdown. In addition to **theoretical questions**, choose **one of the problems listed below** and solve it, writing the corresponding theoretical and numerical aspects in the Notebook.
- If you have doubts regarding what problem to choose, you should **work on Problem 1**. If you want something more interesting/challenging, choose one of the other problems.
- The usage of optimization algorithms from `scipy.optimize` is allowed. However, if you use Scipy and the optimization algorithm does not work you cannot not get any points. If you use your own implementation (from the Practical Sessions) then you might get some points even if the results are not perfect.
- All your codes should be thoroughly commented. Results obtained should be interpreted and own comments should be added in markdown.
- You get **7 points** for the theoretical questions and **12 points** for solving the problem of your choice. **One point** is given for the presentation of the Notebook: clarity of the exposition, personal comments, comments in codes.
- The purpose of the Final Evaluation, in addition to evaluating what you learned in this course, is to give a clear classification reflecting the work you invested in this course. The final grades will be tweaked, preserving their order, to satisfy the requirements given by the administration. It is important to **do your best** even though some tasks might seem difficult.
- Before submitting your work **restart the notebook and run all the cells**. If you have cells that give errors in the submitted notebook **one point will be subtracted** from the grade. The corrector role is not to debug your work.
- You may skip questions that you find too difficult. It is not necessary to solve the questions in the given order, but please be precise in your presentation and mention the question you work on.
- An asterisk **(*)** marks questions that may be more difficult.

Theoretical Questions (Mandatory for everyone)

1. **(Course 1)** In a research article you see the following figure. The author says that the downward pointing spikes correspond to solutions $\{x_i\}$ of some problem of interest and that they are found using the Golden Search method.



   A list of the solutions $\{x_i\}$ is given and the author claims they are precise up to 12 significative digits. Knowing that the computations were made using a machine precision which can give 16 significative digits, do you believe this claim? Justify your answer.

2. **(Course 2)** Suppose $f : \mathbb{R} \to \mathbb{R}$ is a strictly convex, unimodal function with global minimum x^*. Does Newton's algorithm
$$x_{i+1} = x_i - f'(x_i)/f''(x_i)$$
converge to x^* regardless of the initialization? What about the Gradient Descent (GD) algorithm with Wolfe line-search? (**no proofs**, just references or examples/counter-examples)

3. **(Course 3)** a) What is the main drawback when using the Gradient Descent algorithm in higher dimensions? Give an example of a strictly convex function for which GD will take a long time to converge.
   b) Suppose you have a function $f : \mathbb{R}^2 \to \mathbb{R}$ which is smooth with a unique global minimum x^* and that the gradient evaluated at the point x_0 is $\nabla f(x_0) = (10^6, 10^{-6})$. Is this function ill-conditioned? Do you expect the Gradient Descent algorithm to work well for this function?
   c) Consider the quadratic function $f(x) = x^T Ax - b^T x$ for a 2×2 matrix A with eigenvalues $\lambda_1 = 0.01, \lambda_2 = 10^6$. What is the condition number of the matrix A? Will the GD algorithm converge towards the unique minimum of f regardless of the initialization? What about the speed of convergence?

4. **(Course 4)** You have to optimize a function $f : \mathbb{R}^{100000} \to \mathbb{R}$. The function is explicit and you are able to compute the Hessian matrix. You observe that the Hessian matrix is full. Can you use Newton's method in order to efficiently minimize the function f? What if the Hessian matrix is tridiagonal? Justify your answers.

5. **(Course 5)** a) You use the Conjugate Gradient (CG) algorithm to solve the system $Ax = b$ where A is a $10^6 \times 10^6$ positive definite tridiagonal matrix (only the main diagonal and its neighbors are non-zero). What is the complexity of the CG algorithm in this case?
   b) Suppose, moreover, that A has exactly 100 distinct eigenvalues. According to the theoretical results shown in the course, what is the required number of iterations for CG to converge?

6. **(Course 6)** Consider the problem
$$\min_{g_i(x,y,z)=1} f(x, y, z)$$
with $f(x, y, z) = x^2 + y^2 + z^2$, $g_1(x, y, z) = 2x + 3y - z$, $g_2(x, y, z) = 5x - y + z$. Does this problem have a solution? Is it possible to write the optimality conditions given by the Lagrange multipliers?

7. **(Course 7)** Consider the problem
$$\min_{g_i(x)\leq 0} x^2 + y^2,$$
with $g_1(x) = x + y - 1$ and $g_2(x) = y - x - 1$. What is the optimal solution to this problem? Can the optimality conditions be written at the optimum? What is the corresponding optimality condition for this problem?

8. **(General)** Let $f : \mathbb{R}^N \to \mathbb{R}$ be a function you want to minimize. Indicate which optimization algorithm presented in the course you would use to search for the minimum and justify your choice.
   (i) $N = 1000$, you know the gradient and the Hessian of f. You observe that the Hessian matrix is full.
   (ii) $N = 10^6$, you know the gradient, the function is ill-conditioned.
   (iii) $N = 3$, you only have access to function values.
   (iv) $N = 1000$, the function f is represented as a sum of squares of C^1 functions.
   (v) You minimize a C^1 function (gradients are available) with some linear constraints on the variables.
   (vi) You minimize a convex C^1 function (gradients are available) with some non-linear convex equality constraints on the variables.

Problem 1 Unconstrained and Constrained optimization

Part A – Global optimization of an oscillating function

The objective is to find the global minimum of the function

$$f(x,y) = e^{\sin(50x)} + \sin(60e^y) + \sin(70\sin x) + \sin(\sin(80y)) - \sin(10(x+y)) + \frac{x^2+y^2}{4},$$

where e^X denotes the usual exponential function $\exp(X)$.

1. Prove that the function f admits global minimizers on $\mathbb{R}^2$.
2. Prove that the global minimizer lies in the region $[-4,4]^2$.
3. Make a plot of the function values in the region indicated above and conjecture a more precise location of the global minimum.
4. What terms in the expression of the objective function generate the oscillatory behavior? Justify your answer.
5. Having restrained the search region to $[-4,4]^2$, define a $N \times N$ uniform grid on this region and evaluate the function f at each of these nodes. In each case find and print the minimal value attained by f on the current grid. See how the results change for $N = 2^k + 1$, $4 \leq k \leq 10$. (use vectorization, not loops, to accelerate the computations)
6. For each $4 \leq k \leq 10$, pick the point x_0 of the previously found grids which gives the lowest value for the function f. Then choose a **gradient-based** algorithm and run it starting from the point x_0, observing what is the corresponding minimal value.
7. Observe how the minimal value found numerically and the point where the minimum is attained change with respect to $k \in \{4, ..., 10\}$. What can you conclude?
8. In view of the previous observation what could be the global minimum of the function f on $\mathbb{R}^2$? What arguments can you give to support your claim?
9. **(*)** Looking at the terms generating oscillatory behavior, give an estimate on the size of the discretization which can capture all the oscillatory behavior. Conclude how one could prove that the numerical algorithm converges to the global minimum.

Part B – Constrained optimization

Consider the function $J : [0,\infty)^2 \to \mathbb{R}$ defined by

$$J(x,y) = \cosh(4x+y).$$

Recall that the hyperbolic cosine function is defined by $\cosh(x) = (\exp(x) + \exp(-x))/2$. Consider the following sets

$$K_1 = \{(x,y) \in [0,\infty)^2 : xy \geq 1\}$$

$$K_2 = \{(x,y) \in [0,\infty)^2 : y \leq 1 + x, y \geq 2 - x\}.$$

10. Prove that the sets K_1 and K_2 are convex. Are the sets K_1 and K_2 closed? Is the function J convex? Finally, prove that J admits minimizers on K_1. What can you say about existence of minimizers on K_2?
11. Use Python to plot the behavior of the function J on the sets K_1 and K_2 (separately). For unbounded sets just choose a bounding region of the type $[0,M]^2$. Conjecture the location of the minimizers and give a more precise answer regarding the existence of minimizers of J on K_2.
12. Compute the projection operator on the set K_1: given X in $\mathbb{R}^2$ find the closest point to X which is in K_1. Approximate the solution of the problem

$$\min_{x \in K_1} J(x)$$

using the Projected Gradient algorithm.

13. Show that the minimizer (x_0, y_0) also minimizes the function $J_2(x,y) = 4x + y$ on K_1 and on K_2. Deduce that

$$\min_{x \in K_2} J(x)$$

is equivalent to a linear programming problem and solve it using a linear programming software (like `scipy.optimize.linprog`, see the examples on Moodle).

14. Are there points for which $\nabla J(x,y) = 0$. If no such points exist then what can we conclude regarding the minimum of J on K_1? Give an exact formula for the minimum of J on K_1 based on the previous facts.

Problem 2 Functional optimization

The goal in this problem is to minimize the following functional:

$$\mathcal{F}_\varepsilon(u) = \varepsilon \int_0^1 (u'(x))^2 dx + \frac{1}{\varepsilon} \int_0^1 u(x)^2 (1-u(x))^2 dx$$

where $u : [0,1] \to \mathbb{R}$ is a **1-periodic** C^1 function such that $\int_0^1 u = m \in (0,1)$ and $\varepsilon > 0$. In what follows, we suppose that there exist functions u_ε minimizing $\mathcal{F}_\varepsilon(u)$ under the previously stated hypotheses and we are interested in finding some theoretical properties of the minimizers as well as numerical approximations.

Theoretical questions. (Any form of correct mathematical reasoning will be taken into account, even if not completely rigorous!)

1. Suppose that u_ε minimizes $\mathcal{F}_\varepsilon$ under the constraints stated above. Show that u_ε should only take values in $[0,1]$. (**Hint**: What happens if you replace the values where $u_\varepsilon(x) > 1$ with $u_\varepsilon(x) = 1$? The same question for the values below 0.)
2. When $\varepsilon \to 0$ what term in the expression of $\mathcal{F}_\varepsilon$ will dominate? Suppose that for a sequence $\varepsilon_n \to 0$ you have that the minimizers u_{ε_n} of $\mathcal{F}_{\varepsilon_n}$ verify $u_{\varepsilon_n} \to u$ pointwise. What can you deduce regarding the values taken by the limit function u?
3. In view of the fact that the functions u are assumed to be periodic, can you expect to find a unique minimizer for $\mathcal{F}_\varepsilon$? Justify your answer.
4. The functional $\mathcal{F}_\varepsilon$ is composed of two terms: one is convex, one is not. Identify them and argue why convexity/non-convexity holds.

Numerical aspects.

In order to find approximation of the minimizers of $\mathcal{F}_\varepsilon$ we should replace u which lies in a functional space (of infinite dimension) by an element of a proper finite dimensional space. One natural way to perform this **discretization** process is to consider a sequence of equidistant points

$$0 = x_0 < x_1 < ... < x_N = 1$$

such that $x_{i+1} - x_i = h = 1/N$ and approximate the values $u_i = u_\varepsilon(x_i)$ at these points. In view of the expression of $\mathcal{F}_\varepsilon$ two natural questions arise:

⋆ how to approximate $u'(x_i)$?

⋆ how to approximate the integral of a function $\int_0^1 u(x)dx$ given the discrete values $u_i = u(x_i)$.

5. **(Finite differences approximation)** The Taylor expansion formula says that for a C^1 function we have $u(x+h) = u(x) + u'(x)h + o(h)$. This allows us to deduce multiple possible approximations of $u'(x)$ from which we retain two options:
   - $u'(x) \approx \dfrac{u(x+h) - u(x)}{h}$ (forward approximation)
   - $u'(x) \approx \dfrac{u(x+h) - u(x-h)}{2h}$ (centered approximation)

   Pick a smooth function u of your choice, a point x_0 in its domain of definition and test the rate of convergence of the above approximations with respect to h. You can achieve this by plotting the difference between the actual expression of $u'(x_0)$ and the corresponding finite difference approximation in terms of h on a logarithmic scale.

   (*) Justify theoretically the numerical observation using the Taylor expansion formulae.
6. The periodicity assumption allows us to identify the points $x_0 = 0$ and $x_N = 1$ and look only at the values of u at grid points $x_0, ..., x_{N-1}$. The approximated values of u at these grid points can be put in a vector $\mathbf{u} = (u_0, .., u_{N-1})$ (we may extend this by periodicity in order to keep notations simple: $u_N = u_0, u_{-1} = u_{N-1}$, etc).

   Find the matrices D_f and D_c such that multiplying these matrices with $\mathbf{u}$ will compute the forward and, respectively, centered approximations of the first derivative:

$$D_f \mathbf{u} = \left(\frac{u_{i+1} - u_i}{h} \right)_{i=0,...,N-1} \qquad D_c \mathbf{u} = \left(\frac{u_{i+1} - u_{i-1}}{2h} \right)_{i=0,...,N-1}.$$

7. **(Quadrature formulas)** Given the values u_i (or approximations) of a function u at points x_i we want to approximate the integral $\int_0^1 u(x)dx$. In this exercise we choose to work with one of the simplest ways of doing this: the **trapezoidal rule**. On each of the small intervals $[x_i, x_{i+1}]$ the function u will be approximated by an affine function and the corresponding integral will be approximated by

$$\int_{x_i}^{x_{i+1}} u(x)dx \approx \frac{h}{2}(u_i + u_{i+1}).$$

Show that in the periodic case the trapezoidal rule gives the simple formula

$$\int_0^1 u(x)dx \approx h(u_0 + ... + u_{N-1}).$$

Choose an arbitrary 1-periodic function $u : [0, 1] \to \mathbb{R}$ and test the order of convergence of this approximation as $h \to 0$.

In the following, the values $u(x_i)$ are approximated by the vector $\mathbf{u} = (u_0, ..., u_{N-1})$, extended by periodicity if needed. We use the **trapezoidal rule** for approximating the integrals and the **finite differences rule** given by one of the options $D \in \{D_f, D_c\}$ found at point 6. above for approximating the first derivative.

8. Show that the discrete counterparts of the two terms in the formula of $\mathcal{F}_\varepsilon(u)$ are given by

$$\varepsilon \int_0^1 (u'(x))^2 dx \approx \varepsilon h (D\mathbf{u})^T (D\mathbf{u})$$

and

$$\frac{1}{\varepsilon} \int_0^1 u(x)^2 (1 - u(x))^2 dx \approx \frac{h}{\varepsilon} \sum_{i=0}^{N} u_i^2 (1 - u_i)^2.$$

As usual, the notation $\mathbf{x}^T\mathbf{y}$ denotes the usual scalar product $\sum_{i=0}^N x_i y_i$.
Show that the constraint $\int_0^1 u(x)dx = m$ is discretized by

$$h(u_0 + ... + u_N) = m.$$

9. In conclusion, for $D \in \{D_f, D_c\}$ the functional $\mathcal{F}_\varepsilon(u)$ can be approximated by the discrete function

$$F_\varepsilon : \mathbf{u} \mapsto \varepsilon h \mathbf{u}^T D^T D \mathbf{u} + \frac{h}{\varepsilon} \sum_{i=0}^{N} u_i^2 (1 - u_i)^2.$$

Compute the gradient $\nabla F(\mathbf{u})$ by simply noting that the first term is a quadratic function, while the second term is explicit in u_i and can be differentiated immediately term by term.
Prove that the discrete problem

$$\min_{\mathbf{u} \in K} F_\varepsilon(\mathbf{u})$$

admits a solution, where the constraint is defined by $K = \{\mathbf{u} : h(u_0 + ... + u_N) = m\}$. Identify the solution in the unconstrained case and observe that without a constraint the problem is trivial.
Hint: Is the set K convex? Can you identify the behavior of $F_\varepsilon(u)$ when $|u| \to \infty$?

10. Compute the matrix $D^T D$ for various small values of N and give a conjecture regarding the structure of this matrix. Prove your conjecture.

11. We are now ready to perform the numerical optimization of the discrete functional $u \mapsto F_\varepsilon(u)$ under the constraint $u \in K$. Recall that this simple affine constraint allows us to use a **projected gradient algorithm**. Note that the constraint defined by the set $K = \{\mathbf{u} : h(u_0 + ... + u_N) = m\}$ is affine. Find the formula for the projection operator $P_K(y) = x^*$ where x^* is the solution of

$$\min_{x \in K} |x - y|$$

You may use directly the results shown in the course.

12. Recall that the projected gradient algorithm has the update iteration given by

$$x_{i+1} = P_K(x_i - \gamma_i \nabla F_\varepsilon(x_i)).$$

Prove that in this case, where we have an affine constraint, the projected gradient algorithm is equivalent to the following:

- Choose an initial point $x_{-1} \in \mathbb{R}^N$ and define $x_0 = P_K(x_{-1})$.
- At each iteration perform the update

$$x_{i+1} = x_i - \gamma_i P_{K_0}(\nabla F_\varepsilon(x_i))$$

where $K_0 = \{\mathbf{u} : u_0 + ... + u_N = 0\}$.

Remark: This simple theoretical trick allows you to solve simple constrained optimization problems using "black-box" optimizers like those in `scipy.optimize.minimize`. When using a personal implementation of the Projected Gradient algorithm, the original formulation may be used without any problem. If you use some other code which does not give you direct access to the update $x_{i+1} = \mathcal{G}(x_i)$ (the iteration step in the optimization algorithm) then the second formulation is useful.

13. Minimize the discrete functional F_ε for $N = 100$, $m = 0.3$ and $\varepsilon = 1/N$.
    As an initialization you may choose $\mathbf{u}_0$ in one of the following ways:

    - Randomly, using `numpy.random.rand`. Note that using this you will probably get different results at every run.
    - Choose $\mathbf{u}_0$ coming from a symmetric function like $1 - (x - 0.5)^2$ and projecting it on the constraint set K.

    Perform the experiment for both choices $D = \{D_c, D_f\}$. Comment on why $D = D_f$ behaves better (looking at the structure of the matrix).
    What is the structure of the solution? Vary the parameter $m \in (0, 1)$ and observe how the solution changes. Vary the parameter N maintaining the relation $\varepsilon = 1/N$ and comment the behavior of the solution. Can you deduce what should be the limit of u_ε as $\varepsilon \to 0$?
14. What happens when N is fixed and ε is too small compared to $1/N$? What happens when ε is too big compared to $1/N$?

Problem 3 Alternate minimization

Consider $p \geq 2$ and let $K = \prod_{i=1}^{p}[a_i, b_i]$ be a p dimensional box, i.e. the product of p non-trivial closed intervals. Consider a function $J : K \to \mathbb{R}$ such that J is of class C^1 and strictly convex.

Consider a mapping $T : K \to K$ where $v = T(u)$ is defined via the relations

$$J(v_1, ..., v_{i-1}, v_i, u_{i+1}, ..., u_p) \leq J(v_1, ..., v_{i-1}, w, u_{i+1}, ..., u_p), \forall w \in [a_i, b_i],\ i = 1, ..., p. \tag{1}$$

More precisely, for each one of the p variables do the following:

- keep the first $i-1$ variables equal to the newly found coordinates $v_1, ..., v_{i-1}$ and the last variables equal to the ones of the previous point $u_{i+1}, ..., u_p$.

- then minimize the resulting one-dimensional function where only the i-th variable is free. This will give the new i-th coordinate v_i.

Consider the following optimization algorithm:

- Choose an initialization $u^0 \in K$.
- Perform the iteration $u^{k+1} = T(u^k)$ for $k = 1, ..., N$.

In the rest of the exercise, multiple theoretical and practical aspects of this algorithm will be investigated.

1. Note that the definition of the application $u \mapsto T(u)$ from (1) implies that for each $i = 1, ...p$ we have

$$v_i \text{ minimizes } w \mapsto J(v_1, ..., v_{i-1}, w, u_{i+1}, ..., u_p) \text{ on } [a_i, b_i]. \tag{2}$$

   Show that v_i, $i = 1, ..., p$ are uniquely defined and therefore the application $T : K \to K$ is well defined.

2. Write the Euler inequalities associated to (2) for $i = 1, ..., p$. Show that in this case the Euler inequalities are necessary and sufficient conditions of optimality. Obtain a characterization for v verifying the equality $v = T(u)$.

3. Consider a sequence $u_n \to u$, $u_n \in K$, $n \geq 1$. Prove the following:
   - $u \in K$ and we may extract a converging subsequence from $T(u_n)$.
   - Write the optimality conditions found at point 2. above for $v_n = T(u_n)$. Show that these conditions pass to the limit for any converging subsequence.
   - Deduce that every limit point v^* of $(T(u_n))$ verifies $T(u) = v^*$. Conclude that $T(u_n) \to T(u)$ and the mapping T is continuous.

4. Show that if $J(u) = J(T(u))$ then $u = T(u)$ and u is the minimizer of J on K. **Hint:** Use the characterization based on optimality conditions found at Question 2.

5. Conclude that the algorithm defined by (1) converges towards the unique minimizer of J on K.

6. Suppose A is a positive definite matrix in $\mathbb{R}^{p\times p}$ and b is a vector in $\mathbb{R}^p$. Consider the usual quadratic function

$$J(u) = \frac{1}{2}x^T Ax - b^T x$$

   and recall that $\nabla J(u) = Au - b$. Denote by $\ell_1, ..., \ell_p$ the lines of the matrix A. Show that if $v = T(u)$ as in (1) then (2) is equivalent to

$$\ell_i \cdot (v_1, ..., v_i, u_{i+1}, ..., u_p) = b_i,$$

   where the dot denotes the usual scalar product. Conclude that the iteration $u^{k+1} = T(u^k)$ is equivalent to a system of the form

$$Mu^{n+1} + Nu^n = b, \tag{3}$$

   where $M + N = A$. Identify M, N and underline why (3) is advantageous from a numerical point of view. **(*)** What is the name of the iterative algorithm given by (3)?

7. Consider A a 2×2 matrix (take on example from the TDs if necessary) and compare the alternate minimization strategy with other algorithms studied in class: GD, GD with optimal step. Comment on the trajectory followed by each one of these algorithms.

8. Consider the function

$$f(x, y) = 2x^2 + 2xy + 2y^2 - 6x.$$

   Show that the function is strictly convex on $\mathbb{R}^2$ and find analytically the unique solution (x^*, y^*) of the minimization problem

$$\min_{(x,y)\in\mathbb{R}^2} f(x, y).$$

9. Using the corresponding optimality conditions, write the alternate minimization algorithm in this case in the form:

$$x^{n+1} = h_x(x^n, y^n), y^{n+1} = h_y(x^{n+1}, y^n),$$

   where h_x, h_y are functions to be determined. Implement the optimization algorithm starting from $(x^0, y^0) = (10, \pm 10)$ and plot the optimization paths. Interpret the result.

10. Denoting the error at step en by $e_n = |(x^*, y^*) - (x^n, y^n)|$, give an estimate of e_{n+1} in terms of e_n. What is the convergence rate of the proposed algorithm in this case? Compare it to the usual Gradient Descent algorithm.
11. **Small Rank approximation.** Consider a matrix $A \in \mathbb{R}^{m\times n}$. We wish to find vectors $b \in \mathbb{R}^m, c \in \mathbb{R}^n$ such that $A \approx bc^T$ (vectors in $\mathbb{R}^k$ are assumed in column form). We are, thus, led to consider a minimization problem of the form

$$\min_{b,c} \|A - bc^T\|, \tag{4}$$

where we use the matrix norm $\|(a_{ij})\|^2 = \sum_{i,j} a_{ij}^2$.
(a) Show that the problem (4) has infinitely many solutions.
(b) Show that (4) is a minimization of a convex function, thus the alternate minimization procedure applies. Instead of minimizing alternately with respect to every coordinate in b or c, we deal with coordinates in b simultaneously (assuming c is fixed), then with coordinates in c (assuming b is fixed). Justify in 2-3 phrases (no detailed proof) why the same reasoning as the one shown in Questions 1.-5. also proves convergence in this case.
(c) Looking carefully at problem (4) we see that variables b, c can be separated. Show that if $c \neq 0$ then the minimization with respect to b is equivalent to

$$b = Ac/|c|,$$

and if $b \neq 0$ the minimization with respect to c is equivalent to

$$c = A^T b/|b|.$$

12. Implement the small rank approximation and test it for an example of your choice (produce a rank 1 matrix and search for a decomposition starting from random initial choices for b and c). You may use the function given below to compute the rank 1 product of two vectors.

```
def tensor(a,b):
    return np.array([a]).T@np.array([b])
```

Give a conjecture regarding the required number of iterations to reach convergence.
13. **(*)** Describe the modifications needed in the algorithm to approximate the matrix A using a rank k matrix of the form BC^T where $B \in \mathbb{R}^{m\times k}$ and $C \in \mathbb{R}^{n\times k}$.