UNIVERSITÄT BERN

# Applied Optimization
# Exercise 5 - Line Search Methods

October 22, 2021

## Hand-in instructions:

Please hand-in **only one** compressed file named after the following convention: `Exercise`$n$`-GroupMemberNames.zip`, where $n$ is the number of the current exercise sheet. This file should contain:

- The complete code folder except for the `build` subfolder. Make sure that your code submission compiles on your machine. Zip the code folder.
- A `readme.txt` file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Other files that are required by your `readme.txt` file. For example, if you mention some screenshot images in `readme.txt`, these images need to be submitted too.
- Submit your solutions to ILIAS before the submission deadline.

## Exact line search for the convex quadratic function (2 pts)

Consider the convex quadratic function

$$f(x) = \frac{1}{2}x^T Q x + q^T x + c,$$

where $x \in \mathbb{R}^n$, and constant parameters $Q$ is a symmetric positive definite matrix $\in \mathbb{R}^{n \times n}$ , $q$ is a vector in $\mathbb{R}^n$, and $c \in \mathbb{R}$.

Given a search direction $\Delta x$, compute the exact line search parameter $t$ for an arbitrary point $x$ in the domain of $f$.

$$t := \arg\min_{s \geq 0} f(x + s\Delta x)$$

## Gradient descent with exact line search (2 pts)

Consider the unconstrained minimization problem:

$$\text{minimize } \frac{1}{4}x_1^2 + x_2^2$$

starting from point $x^{(0)} = (2,1)$, perform one iteration of the gradient descent algorithm with exact line search. Sketch the function, the line and the update. [1]

What are the values of $\|\nabla f(x)\|^2$ before and after the update $x^{(1)} := x^{(0)} + t^{(0)}\Delta x^{(0)}$ ?

## Programming Exercise: Modified Mass Spring System (6 pts)

In Exercise 3, we setup the mass spring system problem with two different element functions:

$$E_{a,b} = \frac{1}{2}k_{a,b}||\mathbf{x_a} - \mathbf{x_b}||^2$$

and

$$\hat{E}_{a,b} = \frac{1}{2}k_{a,b}(||\mathbf{x_a} - \mathbf{x_b}||^2 - \mathbf{l}_{\mathbf{a,b}}^2)^2.$$

Now let's introduce extra force to the system by fixing certain nodes in the spring graph. Let $C$ be the set of nodes equipped with extra force and $p_n$ be the target coordinates of node $n$. The force drags the nodes in the spring graph to the target coordinates. Hence, the energy function of the modified mass spring system are:

$$E = \sum_{e(a,b)} E_{a,b} + \sum_{n \in C} \frac{1}{2} w_n ||(\mathbf{x_n} - \mathbf{p_n})||^2$$

and

$$\hat{E} = \sum_{e(a,b)} \hat{E}_{a,b} + \sum_{n \in C} \frac{1}{2} w_n ||(\mathbf{x_n} - \mathbf{p_n})||^2,$$

where $w_n$ is a constant that controls the effect of the force. Here we set $w_n = 100000$. You need to implement the gradient descent method with backtracking line search and find the (local) optimum solutions for the following two different scenarios:

1. Attach the four corner nodes in a $m \times n$ spring graph to target coordinates (0,0), (m, 0), (0, n) and (m, n) respectively.
2. Attach the nodes on the side with index $N_{(i,0)}$ to the point with coordinates (i, 0) and nodes $N_{(i,n)}$ to coordinates $(i,n)$, where $i \in [0,m]$.

Regarding the code that is missing, you are requested to fill in:

- Fill the `add_constrained_spring_elements(...)` function in `MassSpring-SystemT_impl.hh` to add those penalty terms to the objective function.

[1] here the link to our interactive plotting system
https://slides.cgg.unibe.ch/aopt20/plots/plot2d_create.html

- Complete `eval_f(...)` and `eval_gradient(...)` in the `ConstrainedSpringElement.hh` file.
- Update `eval_f(...)` and `eval_gradient(...)` in the `MassSpringProblem2DSparse.hh` file to take the additional term into account when computing the energy.
- Implement the `solve(...)` function in `Algorithms/GradientDescent.hh` and `Algorithms/backtracking_line_search(...)` in `LineSearch.hh`.

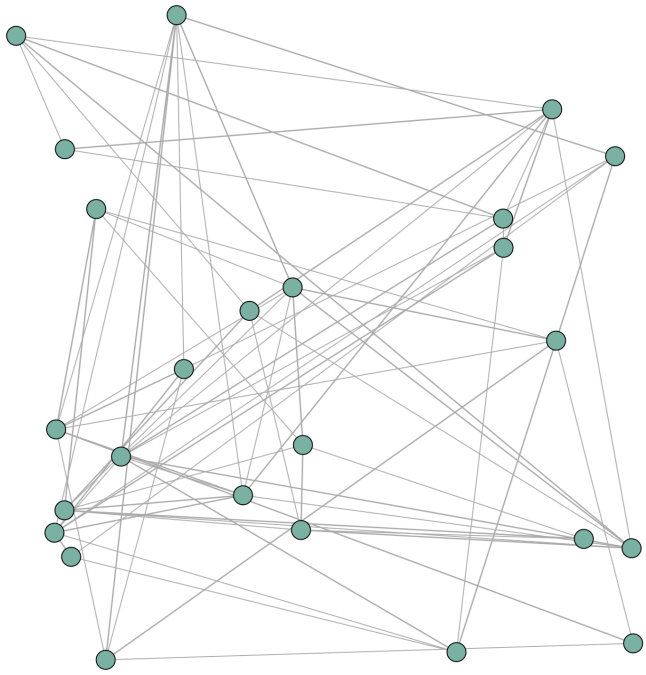

Figure 1: random $4 \times 4$ spring graph

For the usage of the executable, please see the `main.cc` in `GradientDescent` folder. You should provide the filename as a parameter so that the program can save the optimized spring graph to the specified files which you can then visualize on this website. Figure 1 is a $4 \times 4$ random spring graph.

In the `main.cc`, the gradient descent algorithm runs twice from two different start points. Test the algorithm on mass spring systems of dimension $8 \times 8$ of both the spring element without length and with length types. Submit screenshots (8 in total) of the spring graphs (initial and optimized). Record the initial energy value and final energy value at convergence of each test case. Give some comments on how the start points affect the gradient descent method on different function types.

We also provide the unit test for you to check each part of the implementation that needs to be done. Run it by executing `GradientDescent-test`.