Exercise 5 - Line Search Methods

Swiss Joint Master of Science in Computer Science - Applied Optimization

Vincent Carrel, Jonas Fontana, Alain Schaller

```
%matplotlib inline
%load_ext autoreload
%autoreload 2

import numpy as np
from numpy import linalg as LA
import matplotlib.pyplot as plt
from sympy import symbols, Derivative, latex, list2numpy
from IPython.display import display, Math
```

Exact line search for the convex quadratic function

Consider the convex quadratic function

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

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

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

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

```
#TODO complete exercise
```

Gradient descent with exact line search

Consider the unconstrained minimization problem:

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

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.

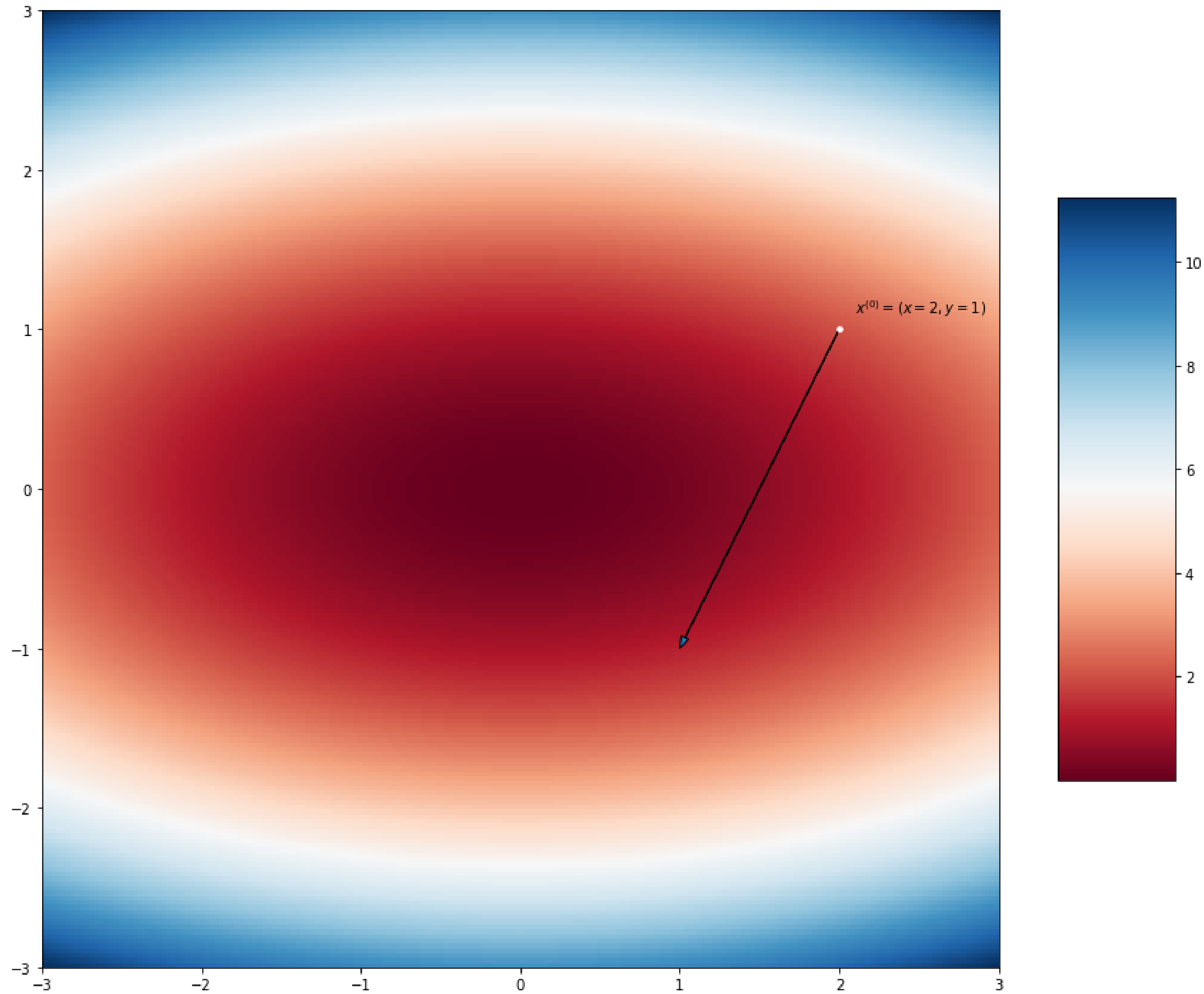
```
x1, x2 = symbols("x_1 x_2")

dx1 = Derivative(1/4 * x1**2 + x2**2, x1, evaluate=True)
dx2 = Derivative(1/4 * x1**2 + x2**2, x2, evaluate=True)

display(Math(f"\\nabla f(x) = \\left({latex(dx1)}, {latex(dx2)}\\right)^T"))

x_0 = {x1: 2, x2: 1}
g_x1 = dx1.subs(x_0).round()
g_x2 = dx2.subs(x_0)
display(Math(f"\\nabla f(x^{{(0)}}) = \\left({g_x1}, {g_x2}\\right)^T"))
display(Math(f"\\text{{Grandient descent : }} \\Delta x = - \\nabla f(x) = \\left({-g_x1}, {-g_x2}\\right)
```

$\nabla f(x) = (0.5x_1, 2x_2)^T$

$\nabla f(x^{(0)}) = (1, 2)^T$

Grandient descent : $\Delta x = -\nabla f(x) = (-1, -2)^T$

```
fig, ax = plt.subplots(figsize=(15, 15))

x = np.linspace(-3.0, 3.0, 200)
y = np.linspace(-3.0, 3.0, 200)

X, Y = np.meshgrid(x, y)

f = 1./4 * X**2 + Y**2

# Plot the function to optimize, we want the min so a Red value
im = plt.imshow(f, cmap=plt.cm.RdBu, extent=[-3, 3, -3, 3])

fig.colorbar(im, shrink=0.5, aspect=5)

ax.plot(2, 1, 'w.', markersize=7)
ax.text(2.1, 1.1, "$x^{(0)} = (x=2, y=1)$", ha="left", fontsize=10)
ax.arrow(2, 1, float(-g_x1), float(-g_x2), head_width=0.05, length_includes_head=True)

plt.show()

# TODO MISSING distance t^{(k=0)} (entire purpose of exercise 1 but not sure how to do it...)
```



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

```
norm_g_0_square = LA.norm(np.array([float(g_x1), float(g_x2)])) ** 2
display(Math(f"\\| \\nabla f(x^{{(0)}}) \\|^2 = {norm_g_0_square}"))


# TODO change x_1
# x_1 = {x1: X, x2: X}x
# g_1_x1 = dx1.subs(x_1).round()
# g_1_x2 = dx2.subs(x_1)
#
# norm_g_1_square = LA.norm(np.array([float(g_1_x1), float(g_1_x2)])) ** 2
# display(Math(f"\\| \\nabla f(x^{{(1)}}) \\|^2 = {norm_g_1_square}"))
```

$\|\nabla f(x^{(0)})\|^2 = 5.000000000000001$