Department of Mathematics and Computer Science
University of Southern Denmark, Odense

February 11, 2025

Marco Chiarandini

AI505 – Optimization

Sheet 01, Spring 2025

Solution:

Included.

Exercises with the symbol $^+$ are to be done at home before the class. Exercises with the symbol * will be tackled in class. The remaining exercises are left for self training after the exercise class. Some exercises are from the text book and the number is reported. They have the solution at the end of the book.

Exercises on

Exercise 1$^+$ Python

Show that the function $f(x) = 8x_1 + 12x_2 + x_1^2 - 2x_2^2$ has only one stationary point, and that it is neither a maximum or minimum, but a saddle point (or inflection point). Plot the contour lines of f in Python (see slides 17, 18 of the tutorial material Part 3).

Exercise 2$^+$

Write the second-order Taylor expansion for the function $cos(1/x)$ around a nonzero point x, and the third-order Taylor expansion of $cos(x)$ around any point x. Evaluate the second expansion for the specific case of $x = 1$.

Exercise 3

Suppose that $f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x}$, where Q is an $n \times n$ *symmetric positive semidefinite* matrix. Show using the definition of convex functions, that $f(\mathbf{x})$ is convex on the domain $\mathbb{R}^n$. Hint: It may be convenient to prove the following equivalent inequality: $f(\mathbf{y} + \alpha(\mathbf{x} - \mathbf{y})) - \alpha f(\mathbf{x}) - (1 - \alpha)f(\mathbf{y}) \leq 0$ for all $\alpha \in [0, 1]$ and all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Exercise 4

Suppose that f is a convex function. Show that the set of global minimizers of f is a convex set.

Exercise 5*

Consider the function $f(x_1, x_2) = (x_1 + x_2^2)^2$. At the point $\mathbf{x}_0 = [1, 0]$ we consider the search direction $\mathbf{p} = [-1, 1]$. Show that $\mathbf{p}$ is a descent direction and find all minimizers of the problem $\min_\alpha f(\mathbf{x}_0 + \alpha\mathbf{p})$.

Exercise 6$^+$

Consider the case of a vector function $f : \mathbb{R}^n \to \mathbb{R}^m$. The matrix $J(x)$ of first derivatives for this function is defined as follows:

$$J(\mathbf{x}) = \left[\frac{\partial}{\partial x_i} f_j\right]_{\substack{j=1..m \\ i=1..n}}$$

write the forward-difference calculations needed to compute $J(\mathbf{x})$ at a given point $\mathbf{x}$.

Exercise 7$^+$ (2.1)

Adopt the forward difference method to approximate the Hessian of $f(x)$ using its gradient, $\nabla f(x)$.

Exercise 8 (2.6)

Combine the forward and backward difference methods to obtain a difference method for estimating the second-order derivative of a function f at x using three function evaluations.

Exercise 9 Python (2.3)

Implement in Python a finite difference method and the complex step method and compute the gradient of $f(x) = \ln x + e^x + 1/x$ for a point x close to zero. What term dominates in the expression?

Exercise 10* (2.5)

Draw the computational graph for $f(x, y) = sin(x + y^2)$. Use the computational graph with forward accumulation to compute $\frac{\partial f}{\partial y}$ at $(x, y) = (1, 1)$. Label the intermediate values and partial derivatives as they are propagated through the graph.

Exercise 11* Python

Implement dual numbers in Python overriding the operators `+,-,*,/`. Test the implementation on the following operations:

- $\epsilon * \epsilon$
- $1/(1 + \epsilon)$
- $(1 + 2\epsilon)*(3 - 4\epsilon)$

Calculate the forward accumulation of the dual numbers $a = 3 + 1\epsilon$ and $b = 2$ on the computational graph of $\log(a * b + max(a, 2))$.

Solution:
Dual numbers $D(a, b)$ can be writen as $a + b\epsilon$, where ϵ satisfies $\epsilon^2 = 0$, so we can drop all $O(\epsilon^2)$ terms. The four rules are:

- $(a + b\epsilon) \pm (c + d\epsilon) = (a \pm c) + (b \pm d)\epsilon$
- $(a + b\epsilon) \times (c + d\epsilon) = (ac) + (ad + bc)\epsilon$
- $(a + b\epsilon)/(c + d\epsilon) = (a/c) + (ad - bc)/c^2\epsilon$

As the other rules, the last one is *designed* such that the mulitpliers of ϵ implement the quotient rules of derivatives: Let $h(x) = \dfrac{f(x)}{g(x)}$, where both f and g are differentiable and $g(x) \neq 0$, the quotient rule states that the derivative of $h(x)$ is

$$h'(x) = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}.$$

```
import math

class Dual:
    """
    A Dualiable which holds two numbers a value and a gradient.
    """

    def __init__(self, val: float |int, grad: float | int):
        assert type(val) in {float, int}
        assert type(grad) in {float, int}
        self.v = val
        self.g = grad

    def __add__(self: 'Dual', other: 'Dual') -> 'Dual':
        return Dual(self.v + other.v, self.g+other.g)

    def __mul__(self: 'Dual', other: 'Dual') -> 'Dual':
        return Dual(self.v * other.v, self.v*other.g + self.g*other.v)
```

```
    def __sub__(self: 'Dual', other: 'Dual') -> 'Dual':
        return Dual(self.v + other.v, self.g+other.g)

    def __truediv__(self: 'Dual', other: 'Dual') -> 'Dual':
        return Dual(self.v/other.v, (self.a*other.g-self.g*other.v)/(other.c^2))

    def __repr__(self):
        return "Dual(v=%.4f, g=%.4f)" % (self.v, self.g)
```

We also need to implement how elementary function would treat dual numbers, in particular the derivative that they apply:

$$\frac{\partial \ln(x)}{\partial x} = \frac{x'}{x}$$

$$\frac{\partial \max(x, p)}{\partial x} = \begin{cases} 0 & \text{if } p > x \\ x' & \text{if } p < x \end{cases}$$

```
def log(a: 'Dual') -> 'Dual':
    return Dual(math.log(a.v), a.g/a.v)

def max(a: 'Dual', b: int) -> 'Dual':
    return Dual(a.v if a.v > b else b, 0 if b>=a.v else a.g)


a = Dual(3,1)
b = Dual(2,0)
print(log(a*b + max(a,2)))
```

We get the same result as the text book:

```
Dual(v=2.1972, g=0.3333)
```

Exercise 12* Python

Read about nanograd and use it to compute by reverse accumulation the gradient of

$$f(x_1, x_2, x_3) = \max\left\{0, \frac{x_1 + (-x_2 x_3)^2}{x_2 x_3}\right\}.$$

Exercise 13* (3.6)

Suppose we have a unimodal function defined on the interval $[1, 32]$. After three function evaluations of our choice, will we be able to narrow the optimum to an interval of at most length 10? Why or why not?