Linnæus University
Optimization 2MA404/2MA918
Björn Lindenberg
Jonas Nordqvist
bjorn.lindenberg@lnu.se

Computer Laboration II: 2MA404/2MA918

Instructions for hand-in: Submit a **SINGLE** pdf-file. (use LaTeXif you are used to it).

Start with a heading and your name. Continue with subheadings for the exercises. For each exercise you need to hand in all data produced and your code. Be sure to answer to all questions. Please present your results with coherent sentences. Save figures in a suitable format, so that they can be inserted in your Word- or TeX-document. Please use a different type face or color for code, so that it is easy to separate text and copy-pasted code.

Exercise 1: Minimum Cost Flow Problems

In this exercise we will solve minimum cost network flow problems. A company in the Kronoberg region needs your help with logistic planning. The company currently owns two production facilities in Almhult and Markaryd with production units in need of weekly transportation to retail stores in Ljungby, Alvesta and Vaxjo. Specifically, assuming that the supply and demands of units are met exactly we obtain the following table:

City	Supply	Demand
Almhult	2500	
Markaryd	1000	
Ljungby		1000
Alvesta		500
Vaxjo		2000

In addition the company also has access to two terminals for storage, which are located in Liatorp and Osby. Moreover, using the information given by their logistics provider we obtain the following table of connections, costs and capacity constraints:

From	To	Cost	Capacity
Almhult	Liatorp	18	1000
Almhult	Osby	24	1000
Almhult	Vaxjo	62	1000
Osby	Markaryd	29	1000
Liatorp	Ljungby	31	500
Liatorp	Vaxjo	46	1000
Markaryd	Ljungby	51	2000
Ljungby	Alvesta	42	500
Alvesta	Vaxjo	20	2000

1. Formulate the problem as minimum cost network flow problem. Solve for the optimal flow and the minimum cost.
   Hint: If you aim to use LP with `scipy linprog` you only need equality constraints via `A_eq` with capacity constraints via the `bounds` function argument instead.

2. The company is also planning to introduce a new production facility in either Varnamo or Vislanda in order to shift some of their weekly production and reduce costs. Given the data below, which site should be chosen with respect to the minimum flow cost? State the optimal cost and flow.

   If Varnamo is chosen then the new production schedule and connections are as follows:

City	Units
Almhult	2000
Markaryd	750
Varnamo	750

From	To	Cost	Capacity
Varnamo	Ljungby	43	2000
Varnamo	Alvesta	50	500

   If Vislanda is chosen we instead get:

City	Units
Almhult	2120
Markaryd	1000
Vislanda	380

From	To	Cost	Capacity
Vislanda	Alvesta	15	500
Vislanda	Vaxjo	29	500

Exercise 2: Unconstrained Optimization I

In this exercise you are going to implement your own versions of the steepest descent method and Newton's method. The function to minimize is

$$f(x, y) = (x+1)^2 - xy + 3(y-5)^2.$$

Tasks (Analysis):

1. Find the optimal point of f by analyzing its gradient and Hessian matrix. We will use this point to measure the effectiveness of our algorithms.

Tasks (Steepest Descent):

1. In the steepest descent method our aim is to iterate to new points by

   $$\mathbf{x}_{k+1} = \mathbf{x}_k - t_k \nabla f(\mathbf{x}_k),$$

   where t_k is a suitably chosen step length. Start by implementing f and ∇f as Python functions. A suggestion here is for our functions to take a single numpy-array as argument (our current point $\mathbf{x}_k$) and where ∇f returns a numpy-array to serve as a direction vector.

2. To choose our step length t_k in each iteration we'll use a simple approach called *Armijo's method*. In our case this method can be boiled down to the following set of instructions.

   (a) Start with $t = 1$. Let $\mathbf{w}(t) := \mathbf{x}_k - t\nabla f(\mathbf{x}_k)$.

   (b) We are now given a choice of two different paths:

   - If $f(\mathbf{w}(t)) \leq f(\mathbf{x}_k) - 0.2 \cdot t \cdot ||\nabla f(\mathbf{x}_k)||^2$ then iteratively assign $t \leftarrow 2t$ and update $\mathbf{w}(t)$ until the condition breaks. Backtrack and put t_k as the last value of t for which the condition holds.
   - Otherwise we have $f(\mathbf{w}(t)) > f(\mathbf{x}_k) - 0.2 \cdot t \cdot ||\nabla f(\mathbf{x}_k)||^2$. Iteratively assign $t \leftarrow t/2$ and update $\mathbf{w}(t)$ until $f(\mathbf{w}(t)) \leq f(\mathbf{x}_k) - 0.2 \cdot t \cdot ||\nabla f(\mathbf{x}_k)||^2$. Take t_k as the first t for when the condition ($\leq$) begins to hold.

   Hint: If your function returns $\nabla f(\mathbf{x}_k)$ as an array g then $||\nabla f(\mathbf{x}_k)||^2$ can be computed as `g@g`.

   (c) Let $x_{k+1} = \mathbf{w}(t_k)$.

3. Starting with $\mathbf{x}_0 = (1, 1)$, how many iterations does it take until $||\nabla f(\mathbf{x}_k)|| < 10^{-3}$? What is the absolute error made with respect to the optimal function value?

Tasks (Newton's Method):

1. In Newton's method our aim is to iterate to new points by

   $$\mathbf{x}_{k+1} = \mathbf{x}_k - H_f^{-1}(\mathbf{x}_k)\nabla f(\mathbf{x}_k).$$

   Derive and implement $H_f^{-1}(\mathbf{x}_k)$ in Python. Implement the algorithm.

2. Starting with $\mathbf{x}_0 = (1, 1)$, how many iterations does it take until $||\nabla f(\mathbf{x}_k)|| < 10^{-3}$? What is the absolute error made with respect to the optimal function value? Can you explain its effectiveness for this problem?

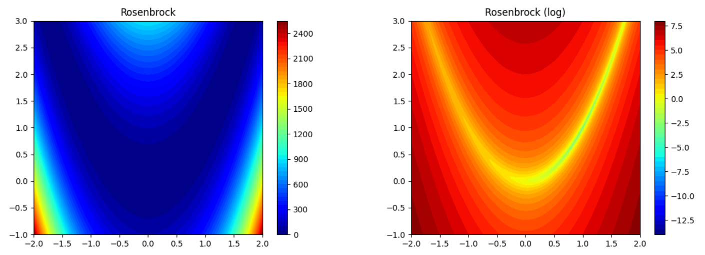

Figure 1: Contour plots of the Rosenbrock function.

Exercise 3: Unconstrained Optimization II

In this exercise we build upon the work done in exercise 2, but switch our focus to optimizing the Rosenbrock function

$$f(x, y) = (a - x)^2 + b(y - x^2)^2$$

with $a = 1$ and $b = 100$.

1. Consider the code template

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

x = np.linspace(-2,2,1000)
y = np.linspace(-1,3,1000)
[X,Y] = np.meshgrid(x,y)
Z = rosenbrock([X,Y]) # To be implemented

plt.figure(1)
plt.contourf(X,Y,Z, 50, cmap=cm.jet)
plt.title('Rosenbrock')
plt.colorbar()

plt.figure(2)
plt.contourf(X,Y,np.log(Z), 50, cmap=cm.jet)
plt.title('Rosenbrock (log)')
plt.colorbar()
```

   Your first task is to complete the code by implementing the Rosenbrock function and produce the plots. If using a script remember to call `plt.show()` when you're ready to show the plots.

2. With the starting point $\mathrm{x}_0 = (-1, 1.5)$, solve for the minimum of f by using the steepest descent algorithm from exercise 2. Store every visited point, including x_0, and stop when $||\nabla f|| < 10^{-4}$. Overlay the trajectory on a log-plot. Overlay template:

```
plt.figure(3)
plt.contourf(X,Y,np.log(Z), 50, cmap=cm.jet)
plt.title('Steepest Descent')
plt.colorbar()
ps = # List of points with steepest descent. To be implemented
px = np.array([p[0] for p in ps])
py = np.array([p[1] for p in ps])
plt.plot(px, py)
plt.plot(px[0],py[0], 'o', color='black') # Starting point
plt.plot(px[-1], py[-1], 'o', color='white') # End point
plt.xlim(-2 , 2)
plt.ylim(-1, 3)
```

Complete the code. State the optimal point and the length of your steepest descent trajectory.

3. Solve the same problem but now with Newton's method. Overlay the trajectory in a new plot, e.g., `plt.figure(4)`. What is the trajectory length with Newton's method?

4. Finally solve for the minimum using `scipy` with starting point $\mathbf{x}_0 = (-1, 1.5)$. Template:

```python
from scipy.optimize import minimize

res = minimize(rosenbrock, [-1, 1.5])
```

   Interpret the returned `OptimizeResult` stored in `res`. What is the final value of $||\nabla f||$? State the performed number of iterations.

Exercise 4: Constrained Optimization

In this exercise we want to tackle the problem

$$\begin{array}{llll} \min & f(x,y) & = (2x+y)^2 + y^4 - 4x - 4y & \\ \text{s.t.} & & x^2 + y^2 & \leq 4 \\ & & 4x + 5y & \geq 5. \end{array}$$

Given the reformulation

$$\begin{aligned} &\min f(\mathbf{x}) \\ \text{s.t.} \quad &g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \end{aligned}$$

you should optimize with the penalty function $\alpha(\mathbf{x}) := \sum_{i=1}^{2} \left(\max(0, g_i(\mathbf{x}))\right)^2$ and where $\mu_0 = 0.1$ and $\beta = 10$. To solve any unconstrained subproblem you may use `minimize` from `scipy`.

Tasks:

1. Find a feasible point $\mathbf{x}_0$ to use as an initial guess.

2. Perform successive improvements by iteratively solving the unconstrained problem

$$F_{\mu_i}(\mathbf{x}) := f(\mathbf{x}) + \mu_i \alpha(\mathbf{x}),$$

   $\mu_i = \beta\mu_{i-1}$. Extract optimal points $\mathbf{x}_k$ and continue the iteration until $\mu_k \alpha(\mathbf{x}_k) < 10^{-4}$. Estimate the optimal point and function value. State the number of iterations needed.

Hint: You can supply arguments to the callable function used in `minimize` by the `args` keyword. Example:

```python
def F(x, mu):
    return f(x) + mu*alpha(x)

res = minimize(F, x0, args=mu)
```