Prof. Dr. Karsten Urban
M. Sc. Lewin Ernst

Winter term 2024/2025
Sheet 13, 26 Points

Numerical Optimization
Solution to exercise sheet

review on 29.01.2024 during the exercise class

1. *(Interior Point Method for Linear Problems I)*
Let $B \in \mathbb{R}^{m\times n}$, $m \leq n$, be a matrix with full rank and $(x, \lambda, s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ a vector with $x > 0$ and $s > 0$. Prove that the (Jacobian) matrix

$$\nabla F_0(x, \lambda, s) = \begin{pmatrix} 0 & B^T & I \\ B & 0 & 0 \\ S & 0 & X \end{pmatrix},$$

with $X = \operatorname{diag}(x_1, \ldots, x_n)$ and $S = \operatorname{diag}(s_1, \ldots, s_n)$, is regular.

(6 Points)

Solution: To prove that $\nabla F_0(x, \lambda, s)$ is regular, it is sufficient to show that the system

$$\begin{pmatrix} 0 & B^T & I \\ B & 0 & 0 \\ S & 0 & X \end{pmatrix} y = 0, \tag{1}$$

only has the trivial solution $y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

We split (1) in three equations:

$$B^T y_2 + y_3 = 0, \tag{2}$$
$$B y_1 = 0, \tag{3}$$
$$S y_1 + X y_3 = 0. \tag{4}$$

First, we multiply (2) with $y_1^\top$ and use (3) to get

$$0 = y_1^\top B^\top y_2 + y_1^\top y_3 = y_2^\top B y_1 + y_1^\top y_3 = y_1^\top y_3. \tag{5}$$

Since X is a diagonal matrix with non-zero entries, it is invertible, and therefore we get with (4):

$$y_3 = -X^{-1} S y_1. \tag{6}$$

By multiplying (6) with $y_1^\top$ gives

$$y_1^\top y_3 = -y_1^\top X^{-1} S y_1. \tag{7}$$

With (5) and (7), we finally get

$$0 = y_1^T X^{-1} S y_1. \tag{8}$$

Since $x > 0$ and $s > 0$, $X^{-1}S = \operatorname{diag}(\frac{s_1}{x_1}, ..., \frac{s_n}{x_n})$ is a diagonal matrix with just positive entries, therefore $X^{-1}S$ is in particular positive definite. With that, (8) can just be fulfilled if $y_1 = 0$. With (6), this directly implies $y_3 = 0$. Since $m \leq n$ and B has rank m, the null space of $B^\top$ is just $\{0\}$. Therefore, by inserting $y_3 = 0$ in (2), we get as well that $y_2 = 0$.

2. *(Interior Point Method for Linear Problems II,* MATLAB*)*
In this task, we will work with the Interior point method for linear problems. The linear equation system in step (6) of Algorithm 4.5.1. in the lecture notes can be solved efficiently due to the sparse structure of the matrix ∇F_0. For this, let

$$r := XSe - \sigma\tau e, \qquad D^2 := XS^{-1},$$

and consider the linear equation system

$$\nabla F_0 \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 & B^\top & I \\ B & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ XSe - \sigma\tau e \end{pmatrix}.$$

Solving the first equation for Δs, the third equation for Δx and insert this in the second equation yields:

$$\begin{aligned} \Delta\lambda &= -\left(BD^2B^\top\right)^{-1} BS^{-1}r = -\left(BD^2B^\top\right)^{-1} B(Xe - \sigma\tau S^{-1}e), \\ \Delta s &= -B^\top\Delta\lambda, \\ \Delta x &= S^{-1}r - S^{-1}X\Delta s = (Xe - \sigma\tau S^{-1}e) - S^{-1}X\Delta s. \end{aligned}$$

a) Implement a MATLAB routine `InteriorPointMethod.m`, which performs the interior point method for linear problems.

b) Test your routine by using the file `test_InteriorPointMethod.m`.

(6 + 2 = 8 Points)

Solution:

a) See the routine `InteriorPointMethod.m`:

```
function [x, lambda, s, iter] = InteriorPointMethod(f, B, b, x, lambda, s, sigma, tol)

% initial values
eta    = 0.99;
[~,n] = size(B);
tau    = (x'*s)/n;
iter   = 0;

while tau > tol

   % solve DF_0 * (Delta x, Delta lambda, Delta s) = (0, 0, XSe-sigma*tau*e)
   % Cholesy-Zerlegung:
   L = chol(B*(diag(x./s))*B');

   Deltalambda = -L\(L'\(B*(x-sigma*tau./s)));
   Deltas      = -B'*Deltalambda;
   Deltax      = (x-sigma*tau./s) - x./s.*Deltas;

   % calculate alpha:
   alpha =  eta / max([eta; Deltax./x; Deltas./s]);

   % Update:
   x        = x      - alpha * Deltax;
   lambda   = lambda - alpha * Deltalambda;
   s        = s      - alpha * Deltas;
```

```
    % throw an error
    if min([x;s])<=0
        error('Bedingung x,s>0 verletzt!');
    end

    % calculate tau:
    tau = (x'*s)/n;

    % count iterations steps:
    iter = iter+1;

end
```

One may wonder why the vectors b, f are not needed inside the routine. b and f are both related to equality constraints regarding the optimality system. It is already assumed that the initial value $(x^{(0)}, y^{(0)}, s^{(0)})$ fulfils these equality constraints. For the direction $(\Delta x, \Delta y, \Delta s)$ in the next step, it is required that they don't change the equations, which is guaranteed in Algorithm 3.6.1 in step 6. And with that, the equality constraints automatically hold true for $(x^{(1)}, y^{(1)}, s^{(1)})$ and for all following $(x^{(k)}, y^{(k)}, s^{(k)})$.

Precisely for the first equation $Bx = b$:
It holds $Bx^{(0)} = b$. For $\Delta x = x^{(1)} - x^{(0)}$ it holds that $B(\Delta x) = 0$. And with that, the equality is still fulfilled in the next step: $Bx^{(1)} = B\Delta x + Bx^{(0)} = b$.

b) See the script `test_InteriorPointMethod.m`:

```
clear, close all,
clc

% problem definition
f = [-1;-3;0;0];
B = [1 1 1 0 ; 2 1 0 1];
b = [3;2];

% initial values
x0      = [1/2;1/2;2;1/2];
lambda  = [-2;-2];
s       = [5;1;2;2];

% matlabs reference solution
x_matlab = linprog(f,-eye(4),zeros(4,1),B,b);

% your own code here
[x_IPM, y_IPM, s_IPM, iter_IPM] = InteriorPointMethod(f,B,b,x0,lambda,s,0.8,1e-6);

% Check the solution by compairing with matlab internal routine
fprintf('The error ||x_IPM - x_matlab||_2 = %4.3e \n', norm(x_IPM-x_matlab))
```

3. *(Interior Point Method for Linear Problems III,* MATLAB*)*
A toy store owner sells four types of Lego boxes in his store, we name them A, B, C and D. The store owner pays for each box A \$11, for a box B \$14, \$20 for C and \$25 for D. One unit of toys A yields a profit of \$3, a unit of B yields a profit of \$4, one unit of Lego C yields a profit of \$5, while one unit of D yields a profit of \$6. The store owner estimates that no more than 1000 lego boxes will be sold every month and he can't order more than 400 boxes of one kind in a month. Further, since he also sells different toys in his store, he does not plan to invest more than \$30 000 in inventory of Lego boxes. How many units of each type should be stocked in order to maximize his monthly total profit?

a) Formulate the task above as a linear maximization problem and determine the dual problem (a minimization problem).

b) Apply your interior point method from above or an internal routine of MATLAB to the dual problem. How many units of each type should be stocked in order to maximize his monthly total profit?

(6 + 6 = 12 Points)

Solution:

First, we collect the details from the text in a table

box type	A	B	C	D
cost	11	14	20	25
profit	3	4	5	6

We get the following constraints:

- Monthly order for each type of boxes: $y = (y_1, y_2, y_3, y_4)^\top \in \mathbb{R}^4$.
- Monthly inventory constraint: $11 \cdot y_1 + 14 \cdot y_2 + 20 \cdot y_3 + 25 \cdot y_4 \leq 30\,000$.
- Upper limit for monthly total order: $y_1 + y_2 + y_3 + y_4 \leq 1000$.
- Limit for monthly order of each type: $0 \leq y_i \leq 400$, for $i \in \{1, 2, 3, 4\}$.
- The goal is to find the maximizer of monthly profit.

Formulation of the maximization problem:

$$(D) \begin{cases} b^\top y \to \max!, \quad b := (3,4,5,6)^\top, \\ \\ s.t.\ B^\top y \leq f, \quad B := \begin{pmatrix} 1 & 11 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 14 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 20 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 25 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}, \quad f := \begin{pmatrix} 1000 \\ 30000 \\ 400 \\ 400 \\ 400 \\ 400 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{cases}$$

The expanded maximization problem is

$$(D_e) \begin{cases} b^\top y \to \max! \\ s.t.\ B^\top y + s = f, \quad s \geq 0. \end{cases}$$

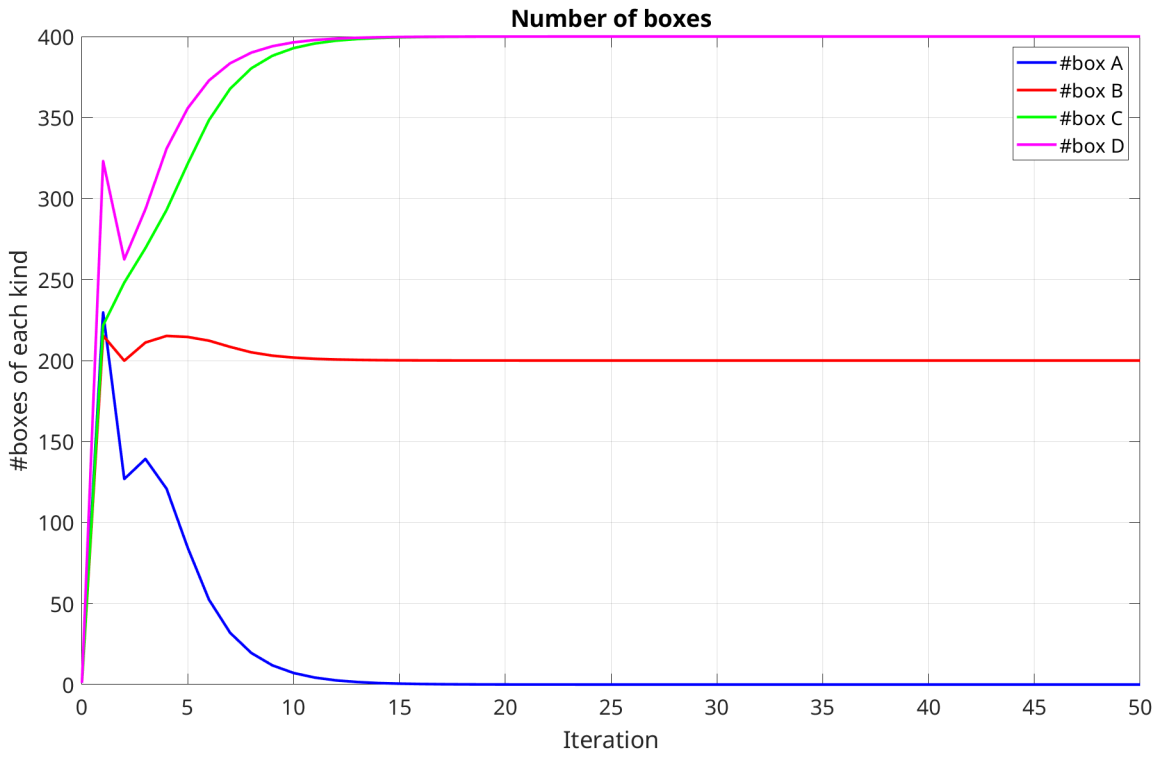
The dual minimization problem of the above maximization problem is

$$(P) \begin{cases} f^\top x \to \min! \\ s.t.\ Bx = b, \quad x \geq 0. \end{cases}$$

These problems have the desired form for our linear interior point method we worked with on this sheet. For the MATLAB script, see **main_Legoboxes.m**, where the problem is solved with the code from above. The optimal solution is $y^* = (0, 200, 400, 400)^\top$.

```
clear, close all
clc

% Write
f = [1000;30000;400;400;400;400; 0; 0; 0; 0];
B = [1 11 1 0 0 0 -1  0  0  0;
     1 14 0 1 0 0  0 -1  0  0;
     1 20 0 0 1 0  0  0 -1  0;
     1 25 0 0 0 1  0  0  0 -1];
b = [3; 4; 5; 6];

x=B\b;
N = null(B);
x = x + N(:,2) + N(:,3) + N(:,5);

n = length(x);
y=[1; 1; 1; 1];
s=f-B'*y;

y_ref = [0; 200; 400; 400];

% IPM:
[x_IPM, y_IPM, s_IPM, iter_IPM] = InteriorPointMethod_v2(f,B,b,x,y,s,0.6,1e-8);

% error:
erry = zeros(iter_IPM,1);
errdmeas = zeros(iter_IPM,1);
for i=0:iter_IPM
    erry(i+1) = norm(y_IPM(:,i+1)-y_ref);
    errdmeas(i+1) = x_IPM(:,i+1)'*s_IPM(:,i+1)/n;
end
Ydisp = ['After ',num2str(iter_IPM),' iterations, the error to the reference solution is ',num2
disp(Ydisp)

%plot error:
figure(1)
semilogy(0:iter_IPM,erry,'LineWidth',2,'Color','blue');
hold on
semilogy(0:iter_IPM,errdmeas,'LineWidth',2,'Color','red');
legend('Error ||y-yref|| to reference solution','Duality measure tau =(x^T s)/n')
title("Error plot of interior-point method");
xlabel("Iteration");
ylabel("Error");
set(gca,'fontsize', 16);
grid on

% number of boxes:
box_A = zeros(iter_IPM,1);
box_B = zeros(iter_IPM,1);
box_C = zeros(iter_IPM,1);
box_D = zeros(iter_IPM,1);
for i=0:iter_IPM
    box_A(i+1) = y_IPM(1,i+1);
    box_B(i+1) = y_IPM(2,i+1);
    box_C(i+1) = y_IPM(3,i+1);
    box_D(i+1) = y_IPM(4,i+1);
end
```

```
% plot number of boxes:
figure(2)
plot(0:iter_IPM,box_A,'LineWidth',2,'Color','blue');
hold on
plot(0:iter_IPM,box_B,'LineWidth',2,'Color','red');
plot(0:iter_IPM,box_C,'LineWidth',2,'Color','green');
plot(0:iter_IPM,box_D,'LineWidth',2,'Color','magenta');
legend('#box A','#box B','#box C','#box D')
title("Number of boxes");
xlabel("Iteration");
ylabel("#boxes of each kind");
set(gca,'fontsize', 16);
grid on
```



Remark: Here, we also allowed floating point numbers as solutions. However, since these are indivisible Lego boxes, it would make sense to define this problem as an integer linear programming problem, i.e. $y \in \mathbb{Z}^4$. But we don't want to dive into the theory of ILP problems here, since this wasn't part of the lecture either. Gladly, the solution y^* is in $\mathbb{Z}^4$.