Università della Svizzera italiana | Institute of Computing CI

Numerical Computing **2023**

Student: Kyla Kaplan Discussed with:

Solution for Project 5 **Due date:** Wednesday, December 20, 2023, 11:59 PM

Numerical Computing 2023 — Submission Instructions
(Please, notice that following instructions are mandatory: submissions that don't comply with, won't be considered)

- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, MATLAB). If you are using libraries, please add them in the file. Sources must be organized in directories called:
  Project_number_lastname_firstname
  and the file must be called:
  project_number_lastname_firstname.zip
  project_number_lastname_firstname.pdf
- The TAs will grade your project by reviewing your project write-up, and looking at the implementation you attempted, and benchmarking your code's performance.
- You are allowed to discuss all questions with anyone you like; however: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently.

The purpose of this project is to implement the Simplex Method to find the solution of linear programs, involving both the minimisation and the maximisation of the objective function.

1. Graphical Solution of Linear Programming Problems [20 points]

Please consider the following two problems:

(1)

$$\begin{aligned} \min \quad & z = 4x + y \\ \text{s.t.} \quad & x + 2y \leq 40 \\ & x + y \geq 30 \\ & 2x + 3y \geq 72 \\ & x,\, y \geq 0 \end{aligned}$$

(2) A tailor plans to sell two types of trousers, with production costs of 25 CHF and 40 CHF, respectively. The former type can be sold for 85 CHF, while the latter for 110 CHF. The tailor estimates a total monthly demand of 265 trousers. Find the number of units of each type of trousers that should be produced in order to maximise the net profit of the tailor, if we assume that the he cannot spend more than 7000 CHF in raw materials.

Start by writing problem (2) as a linear programming problem. Then complete the following tasks:

- Solve the system of inequalities.
- Plot the feasible region identified by the constraints.
- Find the optimal solution and the value of the objective function in that point.

1. Graphical Solution of Linear Programming Problems

1. Problem (2) re-written as a linear programming problem:

$$\begin{aligned} max \quad & z = 85x + 110y \\ s.t.: \quad & 25x + 40y \leq 7000 \\ & x + y \leq 265 \\ & x, y \geq 0 \end{aligned}$$

Solving its linear system of inequalities:

$$\begin{aligned} y &\leq \frac{1400 - 25x}{8} \\ y &\leq 265 - x \end{aligned}$$

Plotted:

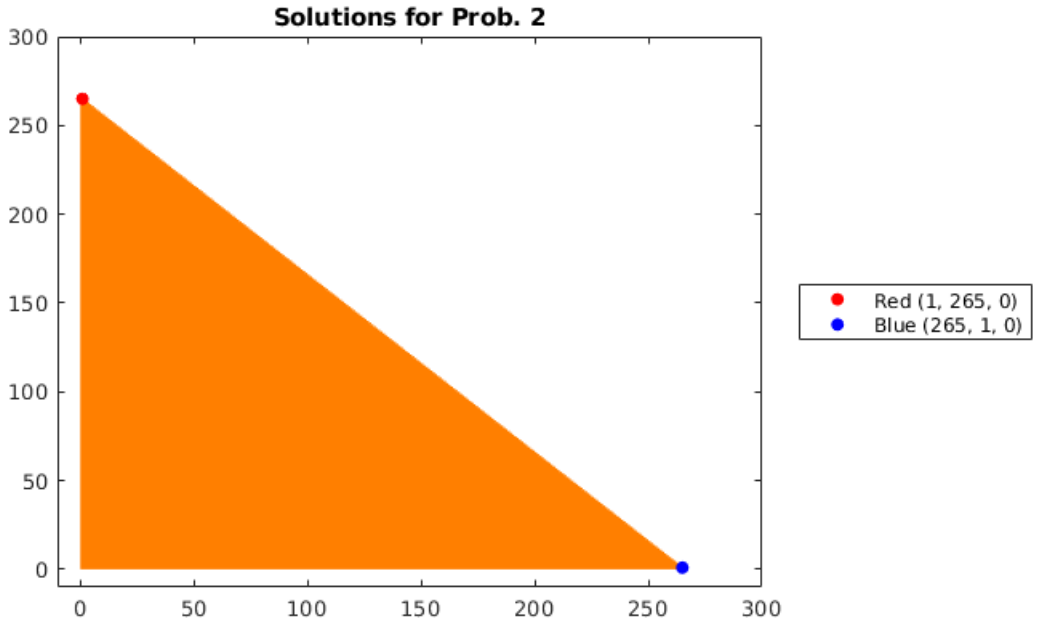

Figure 1: Feasible region for Problem 2

The code for this plot can be found as *ex1b.m* in the accompanying files.

```
% Define the range of values for x and y
x = -10:1:300;
y = -10:1:300;

figure(1);
```

```
% Create a meshgrid for x and y
[X, Y] = meshgrid(x, y);

% Define the inequality conditions
inequality_condition = (X >= 0) & (Y >= 0) & (Y <= 265 - X) & ((Y <= 1400 - 5
    .* X) ./ 8);

% Convert the logical array to double
inequality_condition = double(inequality_condition);

% Replace zeros with NaN to visualize only the valid solutions
inequality_condition(inequality_condition == 0) = NaN;
plot_handle = pcolor(X, Y, double(inequality_condition));
...
```

By highlighting the values of the two most optimal vertices from the plot, and plugging them into the objective function will result in:

$$z_1 = 1 * 85 + 110 * 265 \approx 40.24$$

$$z_2 = 265 * 85 + 110 * 1 \approx 22.64$$

Therefore, the point where $x = 1$, $y = 265$, is 40.235 which is > 22.635, providing the most optimal solution.

2. By following the solution of Problem(2), in Problem (1), it can be easily identified that the optimum **minimum** value of z is 106, at the point where $x = 24$ and $y = 8$.

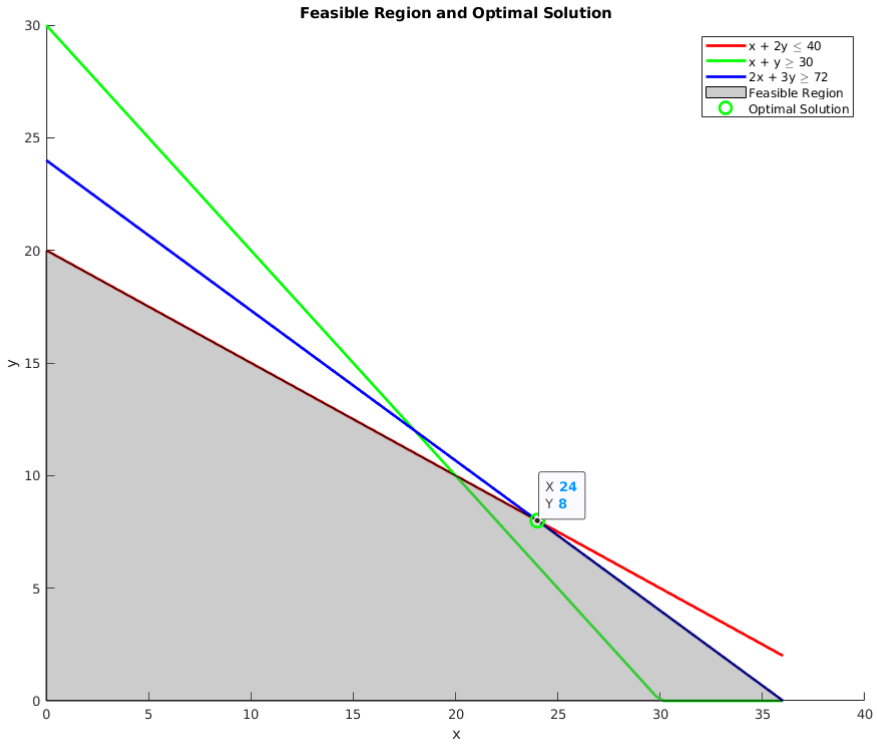

Figure 2: Feasible solution for Problem 1

The code for this plot can be found as *ex1a.m* in the accompanying files. The optimal solution was determined manually by locating the point with the maximum objective function value.

2. Implementation of the Simplex Method [30 points]

In this first part of the assignment, you are required to complete 2 functions which are part of a dummy implementation of the simplex method. Specifically you have to complete the TODOs in:

- *standardise.m*, which writes a maximisation or minimisation input problem in standard form;
- *simplexSolve.m*, which solves a maximisation or minimisation problem using the simplex method.

You are given also some already-implemented functions to help you in your task: *simplex.m* is a wrapper which calls all the functions necessary to find a solution to the linear program; *auxiliary.m* solves the auxiliary problem to find a feasible starting basic solution of the linear program; *printSol.m* is a function which prints the optimal solution found by the simplex algorithm. Finally, *testSimplex.m* presents a series of 6 problems to check if your implementation is correct, before moving to the next part of the assignment. Additional details to aid you in your implementation can be found in the comments inside the code.

2. Implementation of the Simplex Method

Following are implementations of the pre-given files. First, a partially-shown implementation of *standardize.m*:

```
function [A_aug,h,c_aug] = standardize(type,A,h,c,m,sign)
    ...
    aug_matrix = eye(m); % matrix corresponding to the slack/surplus variables
    ...
    % TODO: Correction on the sign of h for auxiliary problem (we want to ensure that
        h>=0, but we need to flip all the signs)
    for i = 1:m
        if(h(i)<0)
            A(i,:) = -A(i,:);
            h(i,:) = -h(i,:);
            aug_matrix(i,:) = -aug_matrix(i,:);
        end
    end

    c_aug = [c, zeros(1,m)];
    if(strcmp(type,'max'))
        % TODO: Extend matrix A by adding the slack variables
        A_aug = [A, aug_matrix];
    elseif(strcmp(type,'min'))
      % TODO: Extend matrix A by adding the surplus variables
        A_aug = [A, -aug_matrix];
    else
        error('Incorrect type specified. Choose either a maximisation (max) or
            minimisation (min) problem.')
    end
    ...
```

Following is a partially shown implementation of the *simplexSolve.m*:

```
function [x_B, c_B, index_B] = simplexSolve(type, B, D, c_B, c_D, h, x_B, x_D, index_B,
    index_D, itMax)
    % Solving a maximization problem with the simplex method
```

```
...
% Compute the reduced cost coefficients
r_D = c_D - c_B * BiD;
...
% Check the optimality condition, in order to skip the loop if the solution is
    already optimal
if strcmp(type, 'max')
    optCheck = sum(r_D < 0);
elseif strcmp(type, 'min')
    optCheck = sum(r_D > 0);
else
    error('Incorrect type specified. Choose either a maximisation (max) or
        minimisation (min) problem.')
end

% Changed, continue iterating until the optimality condition reaches the specified
    tolerance
while optCheck ~= tol
    % Find the index of the entering variable
    if strcmp(type, 'max')
        [~, idxIN] = max(r_D);
    elseif strcmp(type, 'min')
        [~, idxIN] = min(r_D);
    else
        error('Incorrect type specified. Choose either a maximisation (max) or
            minimisation (min) problem.')
end
...
% Evaluate the coefficients ratio for the column corresponding to the entering
    variable
ratio = Bih ./ BiD(:, idxIN);

% Find the smallest positive ratio
idxOUT = find(ratio == min(ratio(ratio >= 0)), 1);

out = B(:, idxOUT);
c_out = c_B(1, idxOUT);
index_out = index_B(1, idxOUT);

% Update the matrices by exchanging the columns
B(:, idxOUT) = in;
D(:, idxIN) = out;
c_B(1, idxOUT) = c_in;
c_D(1, idxIN) = c_out;
index_B(1, idxOUT) = index_in;
index_D(1, idxIN) = index_out;
...
% Compute the reduced cost coefficients
r_D = c_D - c_B * BiD;

% Check the optimality condition
if strcmp(type, 'max')
    optCheck = sum(r_D < 0);
elseif strcmp(type, 'min')
    optCheck = sum(r_D > 0);
else
    error('Incorrect type specified. Choose either a maximisation (max) or
        minimisation (min) problem.')
end
...
```

```
    % Compute the new x_B
    x_B = Bih - BiD * x_D;

    end

end
```

The implementations pass the 6 tests in the *testSimplex.m* file. **All completed code files can be found attached in the accompanying .zip file.**

3. Applications to Real-Life Example: Cargo Aircraft [25 points]

In this second part of the assignment, you are required to use the simplex method implementation to solve a real-life problem taken from economics (constrained profit maximisation).

A cargo aircraft has 4 compartments (indicated simply as $S_1, \ldots, S_4$) used to store the goods to be transported. Details about the weight capacity and storage capacity of the different compartments can be inferred from the data reported in the following table:

Compartment	Weight Capacity (t)	Storage Capacity (m^3)
S_1	18	11930
S_2	32	22552
S_3	25	11209
S_4	17	5870

The following four cargos are available for shipment during the next flight:

Cargo	Weight (t)	Volume (m^3/t)	Profit (CHF/t)
C_1	16	320	135
C_2	32	510	200
C_3	40	630	410
C_4	28	125	520

Any proportion of the four cargos can be accepted, and the profit obtained for each cargo is increased by 10% if it is put in S_2, by 20% if it is put in S_3 and by 30% if it is put in S_4, due to the better storage conditions. The objective of this problem is to determine which amount of the different cargos will be transported and how to allocate it among the different compartments, while maximising the profit of the owner of the cargo plane. Specifically you have to:

1. Formulate the problem above as a linear program: what is the objective function? What are the constraints? Write down all equations, with comments explaining what you are doing.

2. Create a script *exercise2.m* which uses the simplex method implemented in the previous exercise to solve the problem. What is the optimal solution? Visualise it graphically and briefly comment the results obtained (are you surprised of this outcome on the basis of your data?).

3. Application to Real-Life Example: Cargo Aircraft

1. In order to tackle the problem of formulating the problem as a linear program, we will represent with the following variables, the problem as an objective function:

$$max \quad z = 1.0(135x + 200a + 410e + 520p) +$$

$$1.1 * (135y + 200b + 410f + 520q) +$$

$$1.2 * (135l + 200c + 410g + 520r) +$$
$$1.3 * (135k + 200d + 410h + 520s)$$

Followed by the following constraints. The weights of the cargo:

$$s.t. \quad x + y + l + k \leq 16$$
$$a + b + c + d \leq 32$$
$$e + f + g + h \leq 40$$
$$p + q + r + s \leq 28$$

As well as the weights of the aircrafts:

$$x + a + e + p \leq 18$$
$$y + b + f + q \leq 32$$
$$l + c + g + r \leq 27$$
$$k + d + h + s \leq 17$$

And the storage capacities:

$$320x + 510a + 630e + 125p \leq 11930$$
$$320y + 510b + 630f + 125q \leq 22552$$
$$320l + 510c + 630g + 125r \leq 11209$$
$$320k + 510d + 630h + 125s \leq 5870$$

Ensuring non-negative values:

$$a, b, c, d, e, f, g, h, l, k, p, q, r, s \geq 0$$

2. **The implementation for this part of the exercise can be found in *exercise2.m*.** It utilizes the *simplex.m* method. Below is the Command Terminal output in Matlab:

```
Variables and optimal solution:

x5 = 18.000000
x6 = 6.000000
x10 = 26.000000
x11 = 14.000000
x15 = 11.000000
x16 = 17.000000
s5 = 2750.000000
s6 = 3112.000000
s7 = 1014.000000
s8 = 3745.000000
s9 = 16.000000
s10 = 8.000000

Optimal value of z = 41890
```

The simplex method determined the optimal solution for maximizing profit at 41890 CHF. In this solution, none of cargo 1 was moved, 24 out of 32 cargos from cargo 2 were transported, and the remaining cargos from cargo 3 and 4 were moved to fill all compartments. The profit increase aligns with the cargo's profit, reflecting the effective placement strategy. However, the actual maximum profit was lower than initially expected, given the capacity constraints and cargo distribution.

4. Cycling and Degeneracy [10 points]

Consider now the following simple problem:

$$\begin{aligned} \max \quad & z = 3x_1 + 4x_2, \\ \text{s.t.} \quad & 4x_1 + 3x_2 \leq 12 \\ & 4x_1 + x_2 \leq 8 \\ & 4x_1 + 2x_2 \leq 8 \\ & x_1, x_2 \geq 0. \end{aligned}$$

1. Create a script *exercise3.m* which uses the simplex method implemented above to solve this problem. Do you achieve convergence within the maximum number of iterations (given by the maximum number of possible basic solutions)? Do you notice any strange behaviour? (*hint:* check, e.g., the indices of the entering and departing variables)

2. Look at the number of constraints and at the number of unknowns: what can you notice about the underlying system of equations? Represent them graphically and try to use this information to explain the behaviour of your solver in the previous point.

4. Cycling and Degeneracy

1. The code below (**attached in file ex3.m**), shows the restructuring of the question:

```
type = "max";


c = [3, 4, 0, 0, 0];
A = [4, 3, 1, 0, 0; 4, 1, 0, 1, 0; 4, 2, 0, 0, 1];
h = [12; 8; 8];

sign = [-1, -1, -1];

[z,x_B,index_B] = simplex(type,A,h,c,sign);
```

The following gets printed in the Command Terminal (additional print statements were included in the simplex functions in order to track the indices):

```
itMax
    56

x_B
     4
     2
     0

Variables and optimal solution:

x1 = 2.000000
x2 = 0.000000
x3 = 4.000000

Optimal value of z = 6
```

However, I was only able to receive the following print statement after including the following bit of code in simplexSolve.m. Otherwise, I encountered an error in the convergence. By looking at the printings of the indices, it seems that they were cycling between the values of 2 and 4, hence, seemingly, not converging. If looking at the value of x_B, it is also visible that

the same solution persists over many iterations, making it a potential degenerate solution (containing 0), and additionally implying that there is an error in the constraints.

```
while optCheck ~= tol
    % disp(x_B);
    if optCheck
        break;
    end
```

2. This particular system of equations is comprised of 2 variables and 3 constraints, and thus it seems the method loops forever due to attempting the presence of three constraints. As mentioned, there is the possibility that the condition of $x_B \geq 0$ was not fulfilled, making the whole solution degenerate. Below is an attempt to visualize the feasible region (without the optimal solutions):

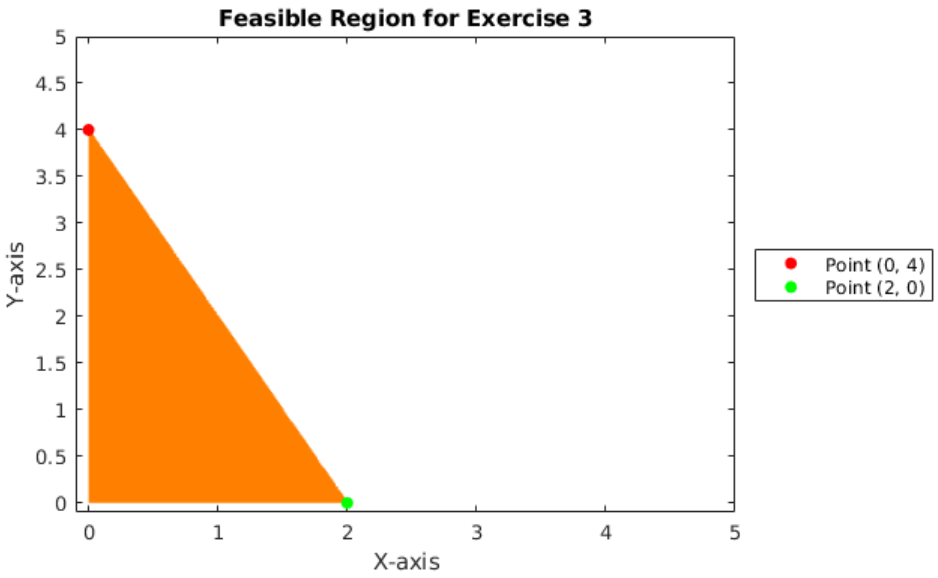

Figure 3: Feasible solution for Exercise 3

Since I am unable to graph the constraints and optimal solutions, there is a high chance the constraints are linearly dependent, as the simplex algorithm would most likely swap through the same set of 'solutions' (i.e. a cycle), and never converge to the actual final solution.