

Linear Systems of Constraints

Linear program:

By example. It is desired to find x_1 and x_2 such that

$$-2x_1 - x_2$$

is minimized subject to the constraints

$$3x_1 + 4x_2 \leq 5$$

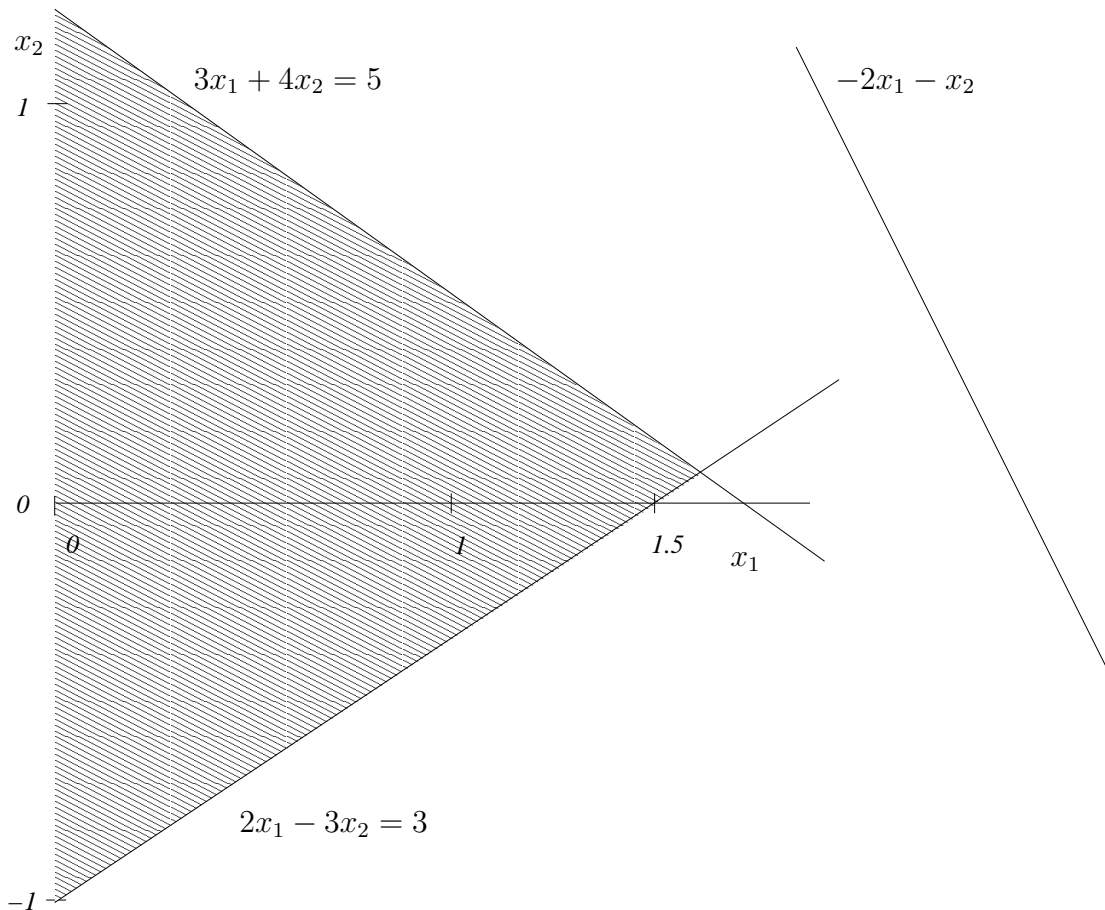
$$2x_1 - 3x_2 \leq 3$$

The first expression is called an *objective function*. The solution to this system can be found by solving the following matrix formulation:

$$\text{minimize: } (-2 \ -1) * \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\text{subject to: } \begin{pmatrix} 3 & 4 \\ 2 & -3 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 5 \\ 3 \end{pmatrix}$$

where $*$ is the matrix multiplication operator. The graphical interpretation of this is shown in the figure below.



The shaded area shows all the $\langle x_1, x_2 \rangle$ points that satisfy the constraints (it is sufficient to show only the region $x_1 \geq 0$ and we do this for practical reasons). The area is bounded from above by the line representing the first constraint and is bounded from below by the line representing the second constraint. The objective function is shown as the line marked $-2x_1 - x_2$. The value of the objective is what we are trying to minimize; it positions the line somewhere in x_1, x_2 space without affecting its slope. The position of the line, as shown in the figure, obviously captures no $\langle x_1, x_2 \rangle$ points that satisfy the constraints (the value of the objective is too small). So the line is slid to the left until it touches the rightmost point of the shaded triangle (sliding to the left increases the value of the objective). That position of the line sets the minimum acceptable value of the objective function.

Consider again the matrix representation. We can use Matlab's `linprog` function to solve the matrix version of the problem. There are several prototypes for this function but we mention two here:

```
function ans = linprog(f,A,b)
function ans = linprog(f,A,b,Aeq,beq)
```

where **A** is the constraint matrix which is used for all constraints for which \leq applies, **Aeq** is the constraint matrix for all constraints for which $=$ applies instead of \leq , **b** is the vertical vector of constants to the right of \leq , **beq** is the vertical vector of constants to the right of $=$, if such constraints exist, and **f** is the vertical vector of objective function coefficients. The output **ans** is a vector of assignments to the variables that solves the given problem. For example, the following Matlab program solves the system of the previous page:

```
function ans = littest()
    f = [-2 -1]';
    A = [ 3 4 ; 2 -3];
    b = [5 3]';
    ans = linprog(f,A,b);
end
```

Running this program results in the following output:

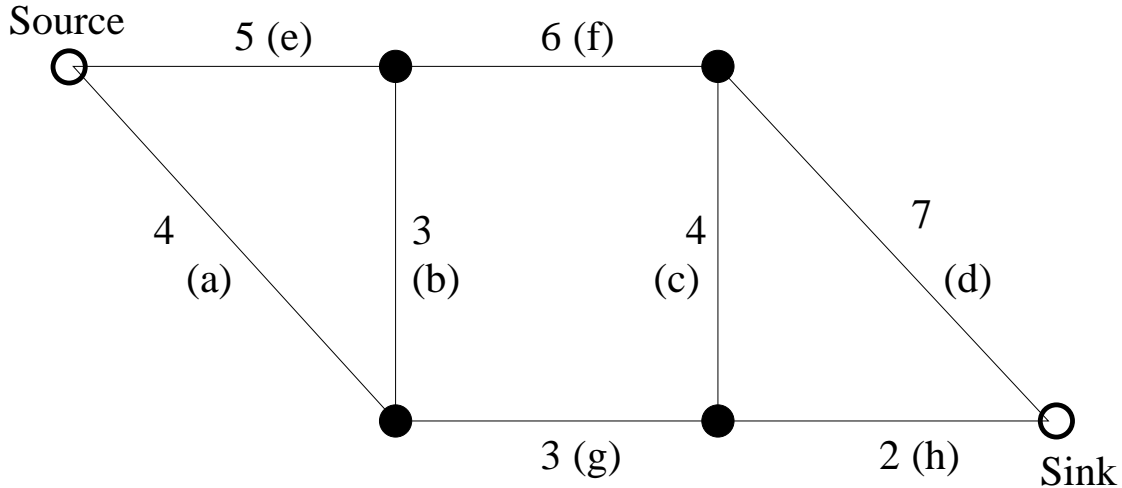
```
ans = {
    1.588235
    0.058824
}
```

An example - The Network Flow Problem:

Given a network of pipes connecting a flow source and a flow sink, each pipe with its own maximum flow capacity. What is the maximum flow possible in the network? The restrictions:

1. The flow through any pipe cannot exceed its capacity.
2. The sum of the flows into a junction of 3 or more pipes is 0.

The figure below shows an example of a simple network of pipes and identifies the source and sink of the network.



In the figure, pipe junctions are identified by dots. Source and sink are special junctions identified by white dots with thick black borders. All other junctions are represented by solid black dots. Numbers show pipe capacity for each pipe connecting two junctions, and letters (in parens) will be used as variables to represent the actual flow in a particular pipe. Not shown is another variable S which is used to represent the flow into the source and out of the sink.

The network problem is represented mathematically as minimizing

$$-S$$

subject to the constraints

$$S - a - e = 0$$

$$e - b - f = 0$$

$$f - c - d = 0$$

$$a + b - g = 0$$

$$c + g - h = 0$$

$$d + h - S = 0$$

and

$$\begin{array}{cccccccc} a \leq 4 & b \leq 3 & c \leq 4 & d \leq 7 & e \leq 5 & f \leq 6 & g \leq 3 & h \leq 2 \\ -a \leq 4 & -b \leq 3 & -c \leq 4 & -d \leq 7 & -e \leq 5 & -f \leq 6 & -g \leq 3 & -h \leq 2 \end{array}$$

Expressed as matrices these become:
 minimize

$$(0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ -1) * \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ S \end{pmatrix}$$

subject to

$$\begin{pmatrix} -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} * \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ S \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} * \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ S \end{pmatrix} \leq \begin{pmatrix} 4 \\ 4 \\ 3 \\ 3 \\ 4 \\ 4 \\ 7 \\ 7 \\ 5 \\ 5 \\ 6 \\ 6 \\ 3 \\ 3 \\ 2 \\ 2 \end{pmatrix}$$

A Matlab program for solving this system is

```
function ans = netflow()
    f = [0 0 0 0 0 0 0 0 -1]';
    A = [1 0 0 0 0 0 0 0 0 ; \
        -1 0 0 0 0 0 0 0 0 ; \
         0 1 0 0 0 0 0 0 0 ; \
         0 -1 0 0 0 0 0 0 0 ; \
         0 0 1 0 0 0 0 0 0 ; \
         0 0 -1 0 0 0 0 0 0 ; \
         0 0 0 1 0 0 0 0 0 ; \
         0 0 0 -1 0 0 0 0 0 ; \
         0 0 0 0 1 0 0 0 0 ; \
         0 0 0 0 -1 0 0 0 0 ; \
         0 0 0 0 0 1 0 0 0 ; \
         0 0 0 0 0 -1 0 0 0 ; \
         0 0 0 0 0 0 1 0 0 ; \
         0 0 0 0 0 0 -1 0 0 ; \
         0 0 0 0 0 0 0 1 0 ; \
         0 0 0 0 0 0 0 -1 0 ] ;
    b = [4 4 3 3 4 4 7 7 5 5 6 6 3 3 2 2]';

    Aeq = [ -1 0 0 0 -1 0 0 0 1 ; \
            0 -1 0 0 1 -1 0 0 0 ; \
            0 0 -1 -1 0 1 0 0 0 ; \
            1 1 0 0 0 0 -1 0 0 ; \
            0 0 1 0 0 0 1 -1 0 ; \
            0 0 0 1 0 0 0 1 -1 ] ;
    beq = [0 0 0 0 0 0]';

    ans = linprog(f, A, b, Aeq, beq);
    fprintf('Incoming=%4.2f Outgoing=%4.2f\n',ans(1)+ans(5),ans(4)+ans(8));
    fprintf('Pipe Capacity Flow\n');
    fprintf('---- - - - - - - - - - - - - - - \n');
    for i=1:8
        fprintf(' %c %6d %8.2f\n',char(i-1+'a'),b(2*i-1),ans(i));
    end
end
end
```

An example - The Knapsack Problem:

Given a collection of items, each with a size and a value, and given a *knapsack* of size B . A number of items can be placed in the knapsack so long as the sum of the sizes of the items is no greater than B . The total value of items in the knapsack is the sum of the values of the items. The problem is to find the subset of items that fits into the knapsack and

has maximum total value among all subsets that fit into the knapsack. For example, the following table lists 5 items:

<u>Item</u>	<u>Value</u>	<u>Size</u>
1	23	3
2	19	4
3	21	2
4	17	2
5	18	2

If the given knapsack has capacity $B = 6$, then the optimal (maximum value) subset consists of items 3,4,5 for a total value of 56 and total size equal to the knapsack capacity.

Let x_1 be a variable whose value will be 1 if item 1 is in the solution and value 0 if item 1 is not in the solution. Create variables $x_2, x_3, x_4,$ and x_5 accordingly, for each of the other items. Then this knapsack problem is represented mathematically as minimizing

$$-23x_1 - 19x_2 - 21x_3 - 17x_4 - 18x_5$$

subject to the single constraint

$$3x_1 + 4x_2 + 2x_3 + 2x_4 + 2x_5 \leq 6$$

Expressed as matrices this becomes:
minimize

$$(-23 \ -19 \ -21 \ -17 \ -18) * \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

subject to

$$(3 \ 4 \ 2 \ 2 \ 2) * \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq (6)$$

x_1, x_2, x_3, x_4, x_5 take value 0 or 1

A Matlab program for solving this system is

```
function ans = knapsack()
    f = [-23 -19 -21 -17 -18]';
    A = [3 4 2 2 2];
    b = [6];
    ans = bintprog(f,A,b);
end
```

The Matlab function `bintprog` is used instead of `linprog` because the variables are assumed to take value 0 or 1 only. Notice that the program includes no constraints that force this occurrence.

An example - A Simple Scheduling Problem:

A number of speakers need to be assigned to rooms at particular times for a future one-day conference. The final schedule is subject to the following restrictions:

1. There are t consecutive speaking periods.
2. There are parallel sessions, or tracks. In other words, for each period there are several rooms in which speakers may speak. The number r of rooms needed equals the number of tracks.
3. There are n speakers.
4. Each speaker has a set number of speeches that he or she will deliver during the day. This number is fixed for each speaker and all numbers are known before scheduling.
5. Every speech lasts one period.
6. All speakers are available the entire day for speaking.
7. No two speakers are scheduled to speak in the same room at the same time.
8. Every speaker is scheduled to at most one speech in a period.

The problem is to determine the minimum number r of rooms needed for the conference. For example, suppose the following table shows a list of speakers and the number of speeches they must give:

<u>Speaker</u>	<u>Speeches</u>
1	2
2	3
3	4
4	3
5	2
6	3

and the number of periods is 5. Then the following table shows a schedule that satisfies all requirements in four tracks (the minimum):

	Tracks			
Period	1	2	3	4
1	3	5	1	
2	3	1	4	2
3	3	4	2	6
4	2	3		6
5	5	4		6

Create variables

$$S_{i,j,k} = \begin{cases} 1 & \text{if speaker } i \text{ is scheduled for room } j \text{ in period } k \\ 0 & \text{otherwise} \end{cases}$$

Then the restriction that there must be at most one speaker scheduled for each room in a single period is expressed by the following constraints:

$$S_{1,j,k} + S_{2,j,k} + S_{3,j,k} + \dots + S_{n,j,k} \leq 1 \quad (1)$$

of which there are $r \cdot t$. The restriction that every speaker gives at most one speech in a period is expressed by the following constraints:

$$S_{i,1,k} + S_{i,2,k} + S_{i,3,k} + \dots + S_{i,r,k} \leq 1 \quad (2)$$

of which there are $n \cdot t$. To satisfy the restriction that there are a set number of speeches for each speaker we add the following constraints:

$$S_{i,1,1} + \dots + S_{i,1,t} + S_{i,2,1} + \dots + S_{i,2,t} + \dots + S_{i,r,1} + \dots + S_{i,r,t} = T_i \quad (3)$$

where T_i is the number of speeches a speaker must give that day. There are n of these constraints. Any assignment of 0-1 values to all $S_{i,j,k}$ variables that satisfies the above inequalities will also satisfy the requirements of the problem. The only thing holding us back now is that we need to “flatten” the indices of those variables. That is, we need to create variables indexed like this: x_1, x_2, \dots from all the $S_{i,j,k}$ variables. This is easy: the number

$$idx = k + (j - 1) \cdot t + (i - 1) \cdot t \cdot r;$$

provides the translation to a “flattened” index space and

$$\begin{aligned} i &= \text{floor}((idx - 1)/(t \cdot r)) + 1; && \% \text{ Speaker} \\ j &= \text{floor}(\text{mod}((idx - 1), t \cdot r)/t) + 1; && \% \text{ Room} \\ k &= \text{mod}((idx - 1), t) + 1; && \% \text{ Period} \end{aligned}$$

provide the translation back. Writing the Matlab code is Lab10.