Mathematical and computational methods for economists
PART IV

Prof. Mauro Maria Baldi

Department of Economics and Law,
University of Macerata,
mauromaria.baldi@unimc.it

AGENDA

- Solving nonlinear equations
- Matrices
- Solving linear equations
- Solving linear programming problems
- Solving (mixed) integer linear programming problems
- Solving unconstrained optimization problems
- Solving constrained optimization problems

ACKNOWLEDGEMENT

This material was prepared also taking inspiration from some slides by Professor Elisabetta Michetti, to whom my thanks go.

NON-LINEAR EQUATIONS

Let's introduce non-linear equations in Matlab with an economical example. Suppose you invest 1000€ at an unknown interest rate i for the first two years. In the third year, the interest rate doubles. After the third year, you earn 2000€. What is the interest rate i?

To address this question, we must solve the following equation:

$$1000\,(1+i)^2(1+2i) = 2000$$

This equation can be reformulated as:

$$(1+i)^2(1+2i) - 2 = 0$$

A RELEVANT DRAWBACK

- We are dealing with a third-degree equation for which a formula exists to find its roots. However, this formula goes beyond the scope of our course. Moreover, most problems require the solution of a transcendental equation for which no solving formulas exist, only numerical approaches.

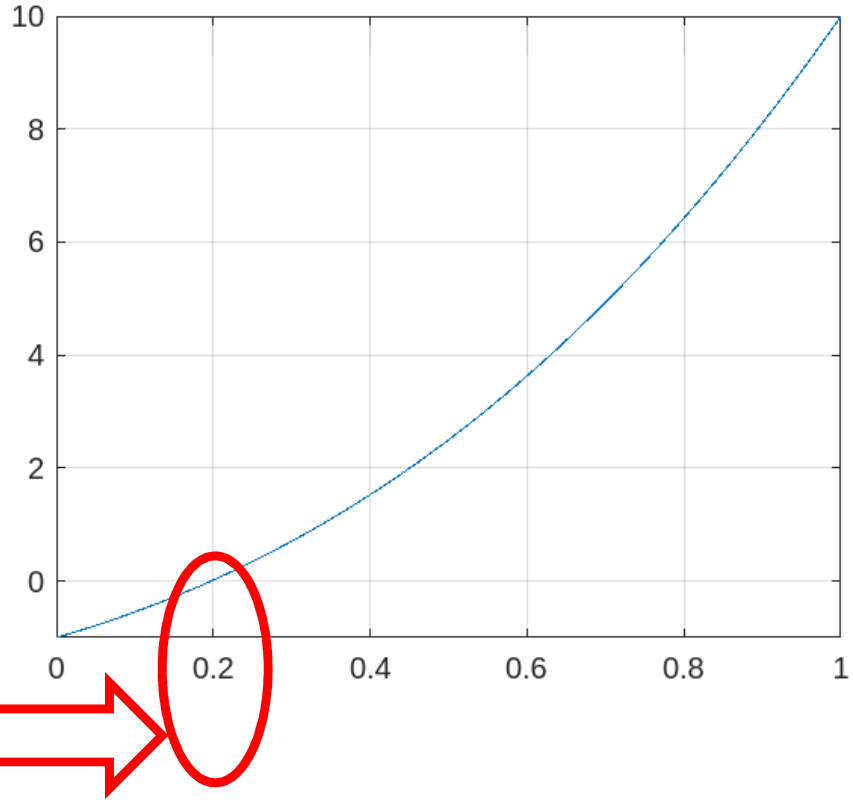
- Opting for numerical approaches, we need to have an idea of the locations of the zeros of the function comprising the left-hand side of the previous equation. However, we know how to tackle this issue.

- In fact, it is enough to plot the graph of the function:

$$y = (1+x)^2(1+2x) - 2$$

GRAPHICAL SOLUTION

```
>> f = @(x) (1 + x).^2.*(1 + 2*x) - 2;
>> fplot(f, [0, 1])
>> grid on
```



With this graphical (albeit approximate) approach, we can estimate that the interest rate i is approximately 0.2.

CAN WE DO BETTER?

Matlab offers the fzero command, which requires a function handle as a parameter, along with either an initial point close to the root or an interval containing the zero of the specified function.

```
>> ii = fzero(f, 0.3)

ii =

0.1974

>> ii = fzero(f, [0 1])

ii =

0.1974
```

Here, 0.3 is employed as the initial point.

Here, [0, 1] is employed as the initial interval.

ANOTHER ECONOMICAL PROBLEM

A car rental company wants to buy 25 trucks with a total capacity (volume) of 2,800 cubic meters. On the market, there are trucks with capacities of 35 cubic meters, 70 cubic meters, and 140 cubic meters. How many trucks of each type should the company buy?

A POSSIBLE SOLUTION

To solve this problem, we can set up a system of equations based on the volume requirements. Let x, y, and z represent the number of trucks with capacities of 35, 70, and 140 cubic meters, respectively. The total volume requirement is given as 2,800 cubic meters:

$$35x + 70y + 140z = 2800$$

The company wants to buy a total of 25 trucks:

$$x + y + z = 25$$

Now you have a system of two equations that we can solve to find the values of x, y, and z. Once we have these values, we'll know how many trucks of each type the company should buy to meet its volume and quantity requirements.

A LINEAR SYSTEM

To summarize, we need to solve the following (linear) system of equations:

$$\begin{cases} 35x + 70y + 140z = 2800 \\ x + y + z = 25 \end{cases}$$

This system has three unknowns but only two equations. Consequently, it possesses an infinite number of solutions. Within this set of solutions, our focus is on identifying integer and positive solutions, given that our variables represent the number of trucks.

AN ASSUMPTION

We make an additional assumption: the car rental company decides to purchase 16 trucks with a capacity of 140 cubic meters. This involves substituting the value of 16 into the variable z. Thus, our linear system become:

$$\begin{cases} 35x + 70y + 140 \cdot 16 = 2800 \\ x + y + 16 = 25 \end{cases}$$

i.e.:

$$\begin{cases} 35x + 70y = 560 \\ x + y = 9 \end{cases}$$

We all know how to solve this linear system. However, our goal now is to demonstrate how to solve it using Matlab. The same procedure can be applied to larger linear systems.

A REVIEW OF LINEAR ALGEBRA - I

The linear system

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2 \\ \vdots \qquad\qquad \vdots \quad \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n = b_m \end{cases}$$

can be expressed in matrix form as $\boldsymbol{Ax} = \boldsymbol{b}$, where

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \text{ and } \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

A REVIEW OF LINEAR ALGEBRA - II

- The most common method for manually solving the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ is Gaussian elimination through elementary operations. This process is akin to calculating $\mathbf{x}$ as $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$, where $\boldsymbol{A}^{-1}$ represents the inverse of the matrix $\boldsymbol{A}$.

- It's worth noting that the inversion of a matrix is not always possible. The existence of the inverse matrix $\boldsymbol{A}$ is contingent upon the nonzero rank of the matrix $\boldsymbol{A}$ itself.

- The Rouché-Capelli theorem offers insights into the number of solutions based on the ranks of the matrix $\boldsymbol{A}$ and the augmented matrix $[\boldsymbol{A} \mid \boldsymbol{b}]$.

- $\boldsymbol{x}$ and $\boldsymbol{b}$ are vectors, and we already know how to handle vectors in Matlab. However, to solve our linear system in the previous example, we also need to work with matrices. Therefore, before we proceed further, let's take some time to introduce matrices in Matlab.

SAVE A MATRIX

Consider a Matrix having m rows and n columns

- To define a matrix the elements must be written **row by row**, between square brakets

- When changing the row use the **semicolon**

The obtained matrix is of mxn kind, that is its dimension is mxn

EX. Save the following Matrix: $A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$

```
>> A=[1 3 5; 2 4 6]
```

To obtain the dimension of a matrix A the command is: **size(A)**

To obtain the **transpose matrix** (by changing rows with columns) the following command must be given:

```
>> A'
```

WORK WITH MATRICES - I

Notice: A matrix can be saved by using row vectors or column vectors previously defined: it can be considered as a **column vector having elements given by row vectors or as a row vector in which each element is a column vector**

EX. For the previously given matrix A:

```
>> u=1:2:5;
>> v=2:2:6;
>> A=[u;v]
```

Notice: each element of a matrix is identified by two indexes **the row index and the column index**

For instance in the previous examples the element 6 belongs to the second row and third column (it has indexes 2,3)

Hence: to **select an element of a matrix** the name of the matrix must be followed by the indexes between round brackets:

```
>> A(2,3)
```

WORK WITH MATRICES - II

Notice: by considering the indexes of an element it is possible to **change the value of one element of the matrix**

```
>> A=[1 3 5; 2 4 6]
```

EX. Substitute the element having value 3 with the new value -1

```
>> A(1,2)=-1
```

```
A =

     1    -1     5
     2     4     6
```

WORK WITH MATRICES - III

(1) To **select a row vector or a column vector** of the given matrix it must be used the operator **:**

A(:,j) gives column j of matrix A, A(i,:) gives row i of matrix A

EX.

- select the first column of A

```
>> A(:,1)
```

- select the second row of A

```
>> A(2,:)
```

WORK WITH MATRICES - IV

(2) To **select a sub matrix** it must be specified the interval of the row-indexes or column-indexes

EX.

Select the matrix from A composed by all the row of matrix A and the first two columns of matrix A

```
>> A(:,1:2)
```

Substitute the second row of A with an equally spaced vector given by elements from 2 to 6 with step 2

```
>> A(2,:)=2:2:6
```

Select the matrix from A composed by all the row of matrix A and the first and third column of A

```
>> A(:,[1,3])
```

WORK WITH MATRICES - V

SUMMARIZING

A(i,j)	select the element **(i,j)**
A(i,:)	select **i-row**
A(:,j)	select **j-column**
A(:,[n,m])	select **n-column and m-column**

WORK WITH MATRICES - VI

(3) It is possible to **delete a row or a column** (thus changing the dimension of the matrix)

EX.

Consider matrix

```
>> A=[1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9
```

Delete the first row

```
>> A(1,:)=[]
```

PROPOSED EXERCISE

- Save the following matrix by indicating its elements:

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 0 \end{pmatrix}$$

- Save the following vectors v1=(1,2,3), v2=(3,4,5), v3=(-1,0,-1). Define matrix B whose rows are given by v1, v2 and v3.

-Select form A the submatrix C: the rows are given by the first two rows of A while the columns are given by the last two columns of A

- Substitute the first row of B with vector [1,1,1], and the null element of B with the value 5

- Create form B the matrix D by deleting the second row. Which is the dimension of D?

- Obtain matrix E by transposing D

1) OPERATION: SUM

Let A and B be two matrices with same dimension (same number of row and columns). Then it is possible the sum **A+B** thus obtain a matrix in which each element is given by the sum of the lements of A and B having the same indexes

```
>> A=[1 2 3; -3 -1 2];
>> B=[-1 2 0.5; 3 -3 6];
>> C=A+B

C =

         0    4.0000    3.5000
         0   -4.0000    8.0000
```

Notice: similarly it is possible to compute A-B

If A+k (k real number) is computed, then the number k is added to each element of A

2) PRODUCT BETWEEN A SCALAR AND A MATRIX

A number (scalar) can be multiplied with a matrix by using *

In the resulting matrix each element is multiplied to the given scalar.

```
>> A=[1 2 3; -3 -1 2];
>> B=2*A

B =

     2     4     6
    -6    -2     4
```

PROPOSED EXERCISE

1) Save the 3×5 matrix A with:

- first row: vector of equally spaced elements from 0 to 12

- second row: all elements equal to 1

- third row: equally spaced elements from 2 to -2

2) save the number b=ln(2) and compute the scalar product between b and A thus obtaining B

3) Compute C by summing A and B

4) Compute D=A-2C+1

3) PUNCTUAL PRODUCT AND PUNCTUAL DIVISION

Consider two matrices A and B having the same dimension. It is possible the **punctual product (and punctual division)** by using **.*** (**./**)

```
>> A=[0 3; -4 2];
>> B=[7 2; -3 -1];
>> C=A.*B

C =

     0     6
    12    -2

>> D=A./B

D =

         0    1.5000
    1.3333   -2.0000
```

1) The element (i,j) of C is given by the product between the element (i,j) of matrices A and B
2) The element (i,j) of D is given by the division between the element (i,j) of matrice A over B.

Notice that Inf or NaN will be notified if the division is not possible in set R.

4) PUNCTUAL POWER

Consider a matrix A and a real number a. then it is possible to compute the power-a of each element of A by using **.^**

```
>> A=[1 2; 3 4];
>> B=A.^2

B =

     1     4
     9    16
```

Notice: Do not forget the point, in fact A^2 gives a different result!

```
>> C=A^2

C =

     7    10
    15    22
```

Similarly each element of a matrix A can be elevated to the corresponding element of a matrix B having the same dimension:

A.^B

```
>> A=[1 2; 3 4];
>> B=[5 3; 3 2];
>> C=A.^B

C =

     1     8
    27    16
```

Observe: here C(i,j)=A(i,j)^B(i,j)

4) FUNCTION APPLIED TO A MATRIX

Consider a matrix A and a function y=f(x). Then it is possible to calculate the **value associated by function f to each element of matrix A**, thus obtaining a matrix having the same dimension of A.

EX: Consider the following matrix

$$A = \begin{pmatrix} 1 & -1 & 0.5 \\ 1 & 0 & 1 \\ 3 & 2 & -4 \end{pmatrix}$$

and function $f(x)=\ln(x^2+4)/(x^3-8)$.

Obtain B by applying function f to matrix A.

Notice: Use correctly the syntax for elementary functions! Do not forget the punctual operators!

```
>> A=[1 -1 0.5; 1 0 1; 3 2 -4]

A =

    1.0000   -1.0000    0.5000
    1.0000         0    1.0000
    3.0000    2.0000   -4.0000

>> B=(log(A.^2+4))./(A.^3-8)

B =

   -0.2299   -0.1788   -0.1837
   -0.2299   -0.1733   -0.2299
    0.1350       Inf   -0.0416
```

Notice: B(3,2) is not a number since the division by 0 has been met.

AN EXAMPLE

1) Save the following Matrix A 3x3:

- first column: vector of 3 equally spaced elements from 1 to -5

- second column: vector of elements 2,4,5

- third column: elements 1, ln(4), e^5

2) Obtain matrix B by trasposing matrix A

3) Calculate matrix C by elevating each element of matrix A to the correspondent element of matrix B

4) Compute D=-3A+2B/C

5) Let $f(x) = \frac{\sqrt{x-1}}{\ln|x+3|}$. Obtain matrix E by applying function f to matrix D

SOLUTION - I

```
% Question 1
c1 = linspace(1, -5, 3)';
c2 = [2 4 5]';
c3 = [1 log(4) exp(5)]';
A = [c1 c2 c3]

% Question 2
B = A'

% Question 3
C = A.^B

% Question 4
D = -3*A + 2*B./C

% Question 5
f = @(x) sqrt(x - 1)./log(abs(x + 3));
E = f(D)
```

SOLUTION - II

A =

1.0000 2.0000 1.0000
-2.0000 4.0000 1.3863
-5.0000 5.0000 148.4132

B =

1.0000 -2.0000 -5.0000
2.0000 4.0000 5.0000
1.0000 1.3863 148.4132

C =

1.0000 0.2500 1.0000
4.0000 256.0000 5.1201
-5.0000 9.3106 Inf

D =

-1.0000 -22.0000 -13.0000
7.0000 -11.9688 -2.2058
14.6000 -14.7022 -445.2395

E =

Columns 1 through 2

0.0000 + 2.0403i 0.0000 + 1.6288i
1.0638 + 0.0000i 0.0000 + 1.6416i
1.2859 + 0.0000i 0.0000 + 1.6110i

Column 3

0.0000 + 1.6250i
0.0000 - 7.7709i
0.0000 + 3.4676i

PROPOSED EXERCISES

1

Save the following vectors:

A is a row vector with elements 1,4,7,2

B is a row vector with 4 equally spaced elements from 7 to -5

C is a row vector with elements 0,½,7.8,-3

A. Construct matrix X whose rows are the vectors A, B, C

B. Obtain Y by trasposing X and change the element (2,1) with 0

C. Obtain Z by extracting the matrix having the first two rows and the first two columns from Y

D. Obtain W by extracting the matrix having the last two rows and the last two columns from X

E. Calculate $U=2W+Z^2-\sqrt{W}$

Notice: use the punctual operators when it is necessary!

PROPOSED EXERCISES

2 Save the matrices:

$$A=\begin{pmatrix}1 & e^{0.5} & 0\\ 3 & 4 & 0.2\end{pmatrix}, B=\begin{pmatrix}0 & 1\\ 3 & -2\\ \ln 5 & 4\end{pmatrix}$$

A. Which is the dimension of A? And of B?

B. Substitute the element (2,2) of A with -3

C. Delete the second column of A and the first row of B

D. Compute C=3A-2B+6

E. Let $f(x)=\left(\sqrt[3]{x}+1\right)^2$, calculate $D=f(C)$

PROPOSED EXERCISES

3 Create the column vector X with 7 equally speced elements from 10 to 25, the column vector Y with equally speced elements from -10 to 50 step 10, the column vector Z with 7 elements all equal to 2

A. Create matrix A having columns X,Y and Z

B. Compute B=$A^{0.5}$

C. Observe the elements of B, are all real numbers? Why?

D. Calculate the punctual product $C = A\mathrm{B}$ and the punctual division D=1/A.

E. Observe the elements of D, are all real numbers? Why?

F. Let $f(x) = |\ln(x+1)^2|$, calculate $f(A)$

PROPOSED EXERCISES

4 Save the following matrices

$$A=\begin{pmatrix}3 & -1\\ 0 & 4\end{pmatrix}, B=\begin{pmatrix}0 & 1\\ 3 & -1\\ \sqrt{3} & 0\end{pmatrix}$$

A. Is it possible to compute A+B? Why?

B. Delete from B the second row. Now, is it possible to compute C=A+B?

C. Traspose matrix A and then change the element (1,1) with -3

D. Compute D=(3A)/(e^2B)

BACK TO THE CAR RENTAL

Let's go back to our car rental example. We were in the process of solving the following linear system:

$$\begin{cases} 35x + 70y = 560 \\ x + y = 9 \end{cases}$$

We can start by writing the coefficient matrix $\boldsymbol{A}$ as well as the vector of known terms $\boldsymbol{b}$.

```
>> A = [35, 70; 1, 1];
>> b = [560; 9];
```

THE RANK AND THE INVERSE OF A MATRIX

We mentioned that solving this linear system is equivalent to computing A^-1 b (assuming the matrix A is invertible). However, how do we determine the rank of A in Matlab, and, more importantly, how can we find the inverse of matrix A? The commands **rank** and **inv** provide answers to these two questions.

```
>> rank(A)

ans =

     2

>> inv(A)

ans =

   -0.0286    2.0000
    0.0286   -1.0000
```

Since the rank of the matrix A is 2, it is invertible and we can invoke the **inv** command without any issues.

SOLVING THE SYSTEM

Now that we know $\boldsymbol{A}^{-1}$, we can easily solve the system as follows:

```
>> x = inv(A)*b

x =

2
7
```

A MORE EFFICIENT APPROACH

Fundamentally, our resolution consisted in the following steps:

- ❖ We computed the inverse of the matrix $\boldsymbol{A}$
- ❖ We multiplied this matrix by the vector $\boldsymbol{b}$

Generally, computing the inverse of a matrix is a time-computing procedure. For this reason, Matlab offers the command '\' that solves a linear system without computing the inverse of the matrix $\boldsymbol{A}$.

```
>> x = A\b

x =

2
7
```

WHAT'S IN YOUR HAMBURGER?

Valceci has the following ingredients available to prepare vegan burgers:

Ingredient	Cost (€/kg)	Availability (kg)
Chickpeas	5	40
Spinach	3	30
Aubergines	2	15
Breadcrumbs	1	10

Valceci produces two types of burgers, each weighing 50 g (0.05 kg): classic ones (with at least 40% chickpeas) and protein-rich ones (with at least 60% chickpeas). The percentage of breadcrumbs in each burger cannot exceed 10%. Valceci aims to meet a demand for 500 classic burgers and 200 protein-rich burgers. Formulate a linear programming model that allows Valceci to determine the optimal composition of each burger and minimize production costs.

THE VARIABLES

We introduce the following variables to formulate a linear model:

- $x_{c,c}$: kilograms of chickpeas in a classic burger
- $x_{c,p}$: kilograms of chickpeas in a rich-protein burger
- $x_{s,c}$: kilograms of spinach in a classic burger
- $x_{s,p}$: kilograms of spinach in a rich-protein burger
- $x_{a,c}$: kilograms of aubergines in a classic burger
- $x_{a,p}$: kilograms of aubergines in a rich-protein burger
- $x_{b,c}$: kilograms of breadcrumb in a classic burger
- $x_{b,p}$: kilograms of breadcrumb in a rich-protein burger.

THE OBJECTIVE FUNCTION

Having introduced the variables, we are ready to formulate the objective function z:

$$z = 5x_{c,c} + 5x_{c,p} + 3x_{s,c} + 3x_{s,p} + 2x_{a,c} + 2x_{a,p} + x_{b,c} + x_{b,p}$$

THE WEIGHT

Both types of burgers should weight 50g (i.e., 0.05 kg). Thus:

$$x_{c,c} + x_{s,c} + x_{a,c} + x_{b,c} = 0.05$$

$$x_{c,p} + x_{s,p} + x_{a,p} + x_{b,p} = 0.05$$

CONSTRAINTS ON THE CHICKPEAS

At least 40% chickpeas in the classic burger:

$$\frac{x_{c,c}}{x_{c,c} + x_{s,c} + x_{a,c} + x_{b,c}} \geq \frac{40}{100}$$

Since $x_{c,c} + x_{s,c} + x_{a,c} + x_{b,c} > 0$, we can reformulate this constraint as:

$$x_{c,c} \geq 0.4\left(x_{c,c} + x_{s,c} + x_{a,c} + x_{b,c}\right), \text{ i.e.:}$$

$$-0.6\, x_{c,c} + 0.4x_{s,c} + 0.4x_{a,c} + 0.4x_{b,c} \leq 0.$$

Likewise, for the rich-protein burger we get:

$$-0.4x_{c,p} + 0.6x_{s,p} + 0.6x_{a,p} + 0.6x_{b,p} \leq 0.$$

CONSTRAINTS ON THE BREADCRUM

We proceed as we did for the chickpeas and find:

$$\frac{x_{b,c}}{x_{c,c} + x_{s,c} + x_{a,c} + x_{b,c}} \leq \frac{10}{100}$$

$$\frac{x_{b,p}}{x_{c,p} + x_{s,p} + x_{a,p} + x_{b,p}} \leq \frac{10}{100}$$

After some manipulations, we get:

$$-0.1x_{c,c} - 0.1x_{s,c} - 0.1x_{a,c} + 0.9x_{b,c} \leq 0$$

$$-0.1x_{c,p} - 0.1x_{s,p} - 0.1x_{a,p} + 0.9x_{b,p} \leq 0.$$

CONSTRAINTS ON THE DEMAND

Since we want to produce 500 classic burgers and 200 protein-rich burgers, the amount of chickpeas consumed will be:

$$500x_{c,c} + 200x_{c,p}.$$

This quantity canno exceed the availability of chickpeas, i.e., 40 kg. Thus, we have the following constraint:

$$500x_{c,c} + 200x_{c,p} \leq 40.$$

Similarly, for the remaining ingredients we have:

$$500x_{s,c} + 200x_{s,p} \leq 30$$
$$500x_{a,c} + 200x_{a,p} \leq 15$$
$$500x_{b,c} + 200x_{b,p} \leq 10.$$

THE MODEL

Putting all together, we get the following linear-programming model:

$$\min z = 5x_{c,c} + 5x_{c,p} + 3x_{s,c} + 3x_{s,p} + 2x_{a,c} + 2x_{a,p} + x_{b,c} + x_{b,p}$$

$$\text{s.t.: } -0.6\,x_{c,c} + 0.4x_{s,c} + 0.4x_{a,c} + 0.4x_{b,c} \leq 0$$

$$-0.4x_{c,p} + 0.6x_{s,p} + 0.6x_{a,p} + 0.6x_{b,p} \leq 0$$

$$-0.1x_{c,c} - 0.1x_{s,c} - 0.1x_{a,c} + 0.9x_{b,c} \leq 0$$

$$-0.1x_{c,p} - 0.1x_{s,p} - 0.1x_{a,p} + 0.9x_{b,p} \leq 0$$

$$500x_{c,c} + 200x_{c,p} \leq 40$$

$$500x_{s,c} + 200x_{s,p} \leq 30$$

$$500x_{a,c} + 200x_{a,p} \leq 15$$

$$500x_{b,c} + 200x_{b,p} \leq 10$$

$$x_{c,c} + x_{s,c} + x_{a,c} + x_{b,c} = 0.05$$

$$x_{c,p} + x_{s,p} + x_{a,p} + x_{b,p} = 0.05$$

$$x_{c,c},\, x_{c,p}, x_{s,c}, x_{s,p}, x_{a,c}, x_{a,p}, x_{b,c}, x_{b,p} \geq 0.$$

THE SOLVER

To solve the model, we can use the **linprog** function provided by Matlab and whose documentation is available at the following link:

https://it.mathworks.com/help/optim/ug/linprog.html

In particular, we are interested in the following function call:

[x, fval] = linprog(f, A, b, Aeq, beq, lb, ub),

where the parameters of the function refer to the following minimization problem:

$$\min f^T x$$
$$s.t.: Ax \leq b$$
$$Aeq\ x = beq$$
$$lb \leq x \leq ub.$$

AN INTERMEDIATE TRANSFORMATION

To build the vector and matrices to pass to the linprog function, let's rename the variable subscripts as follows:

$$
\begin{aligned}
c, c &\rightarrow 1 \\
c, p &\rightarrow 2 \\
s, c &\rightarrow 3 \\
s, p &\rightarrow 4 \\
a, c &\rightarrow 5 \\
a, p &\rightarrow 6 \\
b, c &\rightarrow 7 \\
b, p &\rightarrow 8.
\end{aligned}
$$

THE REFORMULATED MODEL

With the new labels, we have:

$$\min z = 5x_1 + 5x_2 + 3x_3 + 3x_4 + 2x_5 + 2x_6 + x_7 + x_8$$

$$\text{s.t.: } -0.6\,x_1 + 0.4x_3 + 0.4x_5 + 0.4x_7 \leq 0$$

$$-0.4x_2 + 0.6x_4 + 0.6x_6 + 0.6x_8 \leq 0$$

$$-0.1x_1 - 0.1x_3 - 0.1x_5 + 0.9x_7 \leq 0$$

$$-0.1x_2 - 0.1x_4 - 0.1x_6 + 0.9x_8 \leq 0$$

$$500x_1 + 200x_2 \leq 40$$

$$500x_3 + 200x_4 \leq 30$$

$$500x_5 + 200x_6 \leq 15$$

$$500x_7 + 200x_8 \leq 10$$

$$x_1 + x_3 + x_5 + x_7 = 0.05$$

$$x_2 + x_4 + x_6 + x_8 = 0.05$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \geq 0.$$

THE CODE

```
f = [5 5 3 3 2 2 1 1];
A = [-0.6, 0.0, 0.4, 0.0, 0.4, 0.0, 0.4, 0.0;
0.0, -0.4, 0.0, 0.6, 0.0, 0.6, 0.0, 0.6;
-0.1, 0.0, -0.1, 0.0, -0.1, 0.0, 0.9, 0.0;
0.0, -0.1, 0.0, -0.1, 0.0, -0.1, 0.0, 0.9;
500, 200, 0, 0, 0, 0, 0, 0;
0, 0, 500, 200, 0, 0, 0, 0;
0, 0, 0, 0, 500, 200, 0, 0;
0, 0, 0, 0, 0, 0, 500, 200];
b = [0, 0, 0, 0, 40, 30, 15, 10];
Aeq = [1, 0, 1, 0, 1, 0, 1, 0;
0, 1, 0, 1, 0, 1, 0, 1];
beq = [0.05, 0.05];
lb = zeros(8, 1);
ub = [];
[x, fval] = linprog(f, A, b, Aeq, beq, lb, ub)
```

THE SOLUTION

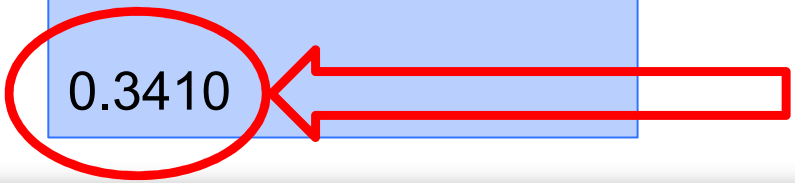
Optimal solution found.

x =

0.0200
0.0300
0.0010
0
0.0240
0.0150
0.0050
0.0050

fval =

0.3410



Valceci spends about 35 cents of euro for a classic and a rich-protein burger.

A PORTFOLIO PROBLEM

The company EtaBeta wants to build a portfolio of contracts for the supply of its stores from five commercial offers. Each offer requires a fixed purchase cost, which corresponds to an economic return. The contract names, profits, and costs in thousands of euros are listed in the next table. Formulate an appropriate mathematical programming model that maximizes the total profit subject to the budget constraint of 30,000 €.

Offer	Cost (K€)	Profits (K€)
1	10	20
2	5	8
3	4	3
4	23	23
5	5	15

THE MODEL

Let x_i be a binary (boolean, or logical) variable equal to 1 if the contract i is accepted, 0 otherwise. Then, the corresponding model is:

$$\max z = 20x_1 + 8x_2 + 3x_3 + 23x_4 + 15x_5$$
$$s.t.: 10x_1 + 5x_2 + 4x_3 + 23x_4 + 5x_5 \leq 30$$
$$x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}.$$

THE SOLVER

To solve the model, we can use the **intlinprog** function provided by Matlab and whose documentation is available at the following link:

https://it.mathworks.com/help/optim/ug/intlinprog.html

In particular, we are interested in the following function call:

[x, fval] = linprog(f, intcon, A, b, Aeq, beq, lb),

where the parameters of the function refer to the following minimization problem:

$$\min f^T x$$
$$s.t.: Ax \leq b$$
$$Aeq\ x = beq$$
$$lb \leq x \leq ub$$
$$x(intcon) \text{ are integers}$$

A TRICKERY

Maximizing f is equivalent to minimizing –f. Thus, we can write the following script:

```
f = [-20 -8 -3 -23 -15];
A = [10 5 4 23 5];
b = 30;
intcon = 1:5;
Aeq = [];
beq = [];
lb = [0 0 0 0 0];
[x, fval] = intlinprog(f, intcon, A, b, Aeq, beq, lb)
```

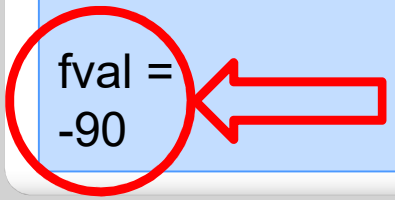
THE OUTPUT

LP: Optimal objective value is -90.000000.

Optimal solution found.
Intlinprog stopped at the root node because the objective value is within a gap tolerance of the optimal value, options.AbsoluteGapTolerance = 0. The intcon variables are integer within tolerance, options.IntegerTolerance = 1e-05.

x =
0
0
0
0
6

fval =
-90



We are minimizing $-f$, so the real value of the objective function is 90, i.e., 90,000€ purchasing six type-5 contracts.

THE CHEAPEST SWIMMING POOL

A swimming pool with a square bottom has a volume of 32 m^3. What dimensions should a swimming pool have to use the least amount of materials for its building?

FROM THE PROBLEM TO THE MODEL

Let x be the length (and the width) of the bottom and y the height of the swimming pool. Then, we have to solve the following constrained optimization problem:

$$\min x^2 + 4xy$$
$$s.t.: x^2 y = 32$$
$$x \geq 0, y \geq 0.$$

A FIRST APPROACH

From the equality constraint, we can express y w.r.t. x and substitute into the objective function. Consequently, the problem transforms into:

$$\min x^2 + \frac{128}{x}.$$

We can easily solve this problem using calculus as an unconstrained optimization problem. Of course, we need to verify that $x \geq 0$ and $y \geq 0$, rejecting solutions that do not satisfy these non-negativity constraints.

However, the focus of these slides it to introduce automatic resolutive methods in Matlab.
Therefore, we begin by introducing the **fminsearch** function.

THE fminsearch FUNCTION

The fminsearch function is provided by Matlab and is capable of addressing unconstrained optimization problems. Its documentation is available at:

https://it.mathworks.com/help/matlab/ref/fminsearch.html

In particular, we are interested in the following call:

[x, fval] = fminsearch(fun, x0),

where x0 is an initial point close enough to the minimum. The choice of the initial point is crucial because in the case of non-convex problems, the procedure can end at a local but non global minimum or follow an unbounded direction.

THE OUTPUT

```
>> f = @(x) x^2 + 128/x;
>> x0 = 1;
>> [xval, fval] = fminsearch(f, x0)

xval =

    4.0000

fval =

   48
```

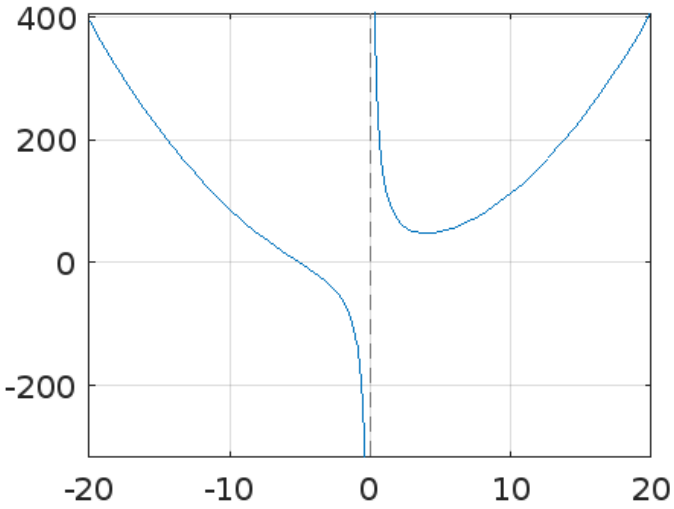
The bottom square's side length should be 4m.

The overall surface area of the swimming pool is 48 m^2.

A SUBTLE DETAIL

The function we are minimizing is: $f(x) = x^2 + \frac{128}{8}$. As we already know, to plot the graph of this function, we would type in Matlab something like this:

```
>> f = @(x) x.^2 + 128./x;
>> fplot(f, [-20 20])
>> grid on
```



However, when we called the fminsearch function, we defined the function as f = @(x) x^2 + 128/x. The reason for this slight difference will become clear when we apply the fminsearch command to multivariate minimization problems.

PAY ATTENTION TO THE STARTING POINT!

As mentioned earlier, the choice of the initial point is crucial and can result in unpleasant surprises if not made wisely. For instance, if we had not plotted the graph of f beforehand and selected a negative initial point (e.g., x0 = -1), the result would be:

```
>> [xval, fval] = fminsearch(f, -1)

Exiting: Maximum number of function evaluations has been exceeded
- increase MaxFunEvals option.
Current function value: -24019198012642588 8.000000

xval =
-5.3291e-16

fval =
-2.4019e+17
```

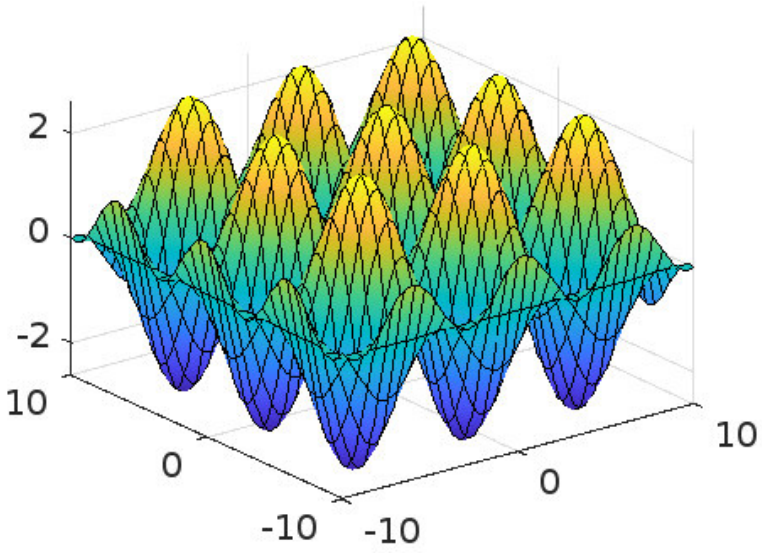
MULTIVARIABLE PROBLEMS - I

Suppose we want to minimize the function

$$f(x, y) = \sin(x) + \sin(y) + \sin(x + y).$$

Let's start with a plot.

```
>> f = @(x, y) sin(x) + sin(y) + sin(x + y);
>> fsurf(f, [-10 10 -10 10])
```



MULTIVARIABLE PROBLEMS - II

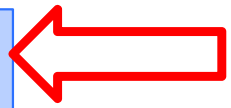
As we could see from the previous graph, the function f has an infinite number of minima. Thus, we expect to find different local minima depending on the starting point.

```
>> f = @(x) sin(x(1)) + sin(x(2)) + sin(x(1) + x(2));
>> [xval, fval] = fminsearch(f, [0, 0])
xval =
-1.0472    -1.0472

fval =
-2.5981

>> [xval, fval] = fminsearch(f, [2, 0])
xval =
5.2360    -1.0472

fval =
-2.5981
```



Again, observe the difference from the previous declaration used for plotting f. This time, we use x(1) and x(2) to respectively denote the variables x and y.

A SECOND APPROACH

In the first approach, we addressed the swimming pool problem by transforming it into an unconstrained minimization problem. This time, we explore a new MATLAB function that allows us to tackle the original constrained optimization problem. This function is called **fmincon**.

Let's go back to our original model:

$$\min x^2 + 4xy$$
$$s.t.: x^2 y = 32$$
$$x \geq 0, y \geq 0.$$

THE fmincon FUNCTION - I

The fmincon function in MATLAB is designed to find the minimum of constrained non-linear multivariable problems. Its documentation can be found at:

https://it.mathworks.com/help/optim/ug/fmincon.html

In particular, we are interested in the following call:

[x, fval] = fmincon(fun, x0, A, b, Aeq, beq, lb, ub, nonlcon)

THE fmincon FUNCTION - II

The parameters of the function refer to the following general non-linear constrained optimization problem:

$$\min f(\boldsymbol{x})$$
$$s.t.: c(\boldsymbol{x}) \leq 0$$
$$ceq(\boldsymbol{x}) = 0$$
$$\boldsymbol{Ax} \leq \boldsymbol{b}$$
$$\boldsymbol{Aeq\ x} = \boldsymbol{beq}$$
$$\boldsymbol{lb} \leq \boldsymbol{x} \leq \boldsymbol{b}.$$

SOLVING THE SWIMMING POOL PROBLEM WITH THE fmincon FUNCTION

```
f = @(x) x(1)^2 + 4*x(1)*x(2);
A = []; b = []; Aeq = []; beq = [];
nonlcon = @mynonlcon;
lb = [0, 0];
ub = [];
x0 = [2 2];
[xval, fval] = fmincon(f, x0, A, b, Aeq, beq, lb, ub, nonlcon)

% We write the nonlinear constraints
% within a function
function [c, ceq] = mynonlcon(x)
    c = [];
    ceq = x(1)^2*x(2) - 32;
end
```

THE OUTPUT

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

xval =

4.0000 2.0000

fval =

48.0000

CONCLUSION

- Is that all?
- Will I be an expert programmer after attending this course?
- Have we explored everything about Economy in Matlab?

The answer to all these questions is a resounding no! What we have explored so far is just the tip of the iceberg. The purpose of these slides is merely to provide a rough idea of the power of Matlab. Interested readers/students are encouraged to deepen their knowledge through additional materials and gain hands-on experience by frequently practicing programming.