Exploring Math $\sum math$ with EIGENMATH

# Linear Algebra *Interactive*! with Eigenmath

## Part 1

**Matrix Algebra**

**Dr. Wolfgang Lindner**
LindnerW@t-online.de
Leichlingen, Germany
2020

# Contents

# Preface

These small scripts want to introduce the reader or student to the elementary Linear Algebra and at the same time into the use of CAS Eigenmath.

These booklets grew out from a series of lessons that I developed between 2001 and 2006 at Mercator University of Duisburg and at FernUniversität Hagen in Germany. The material was repeatedly tested at a German hight school resp. college. The learning parcourse was originally developed using notebooks compiled with CAS MuPAD 3.1 and in a very early state using the free CAS MuPAD$^{Light}$ 2.5.3 with accompanying learning materials in PDF mini booklets. Some figures in this text are therefore produced using MuPAD 3.1.

This series of scripts use now the CAS Eigenmath and presents a refreshed and revised version of that learning sequence, which is intended for students in K12 and K13. The underlying didactical concept is that of a CAS microworld in the setting of the APOS theory of constructivist learning.

In Eigenmath laboratories we explore decisive phenomena via only a few model problems. We verify or falsify hypotheses and would like to encourage ongoing dialogical practice in CAS language communication skills with the Eigenmath assistance.

Therefore the accompanying linguistic comments are deliberately short. If possible, all CAS dialog sequences - which are shown in blue `typewriter font` - should be performed live on the computer. If you cover up the Eigenmath output region with a postcard while going down step by step from an Eigenmath command to the next in the input region, you can uncover the intended answer in the Eigenmath output window line by line. This allows yourself a short pause for a reflection and you can simulate a communication process as a Q&A play in a rudimentary way.

An interdisciplinary aspect occurs sometimes through the use of elementary methods of software engineering in the bottom-up development and step-by-step refinement of Eigenmath-functions to construct mathematical concepts. Techniques of this kind can often be used in CAS and train algorithmic oriented constructive thinking. The Eigenmath commands used and the textual representation should be elementary enough to serve as a companion while reading basic or advanced courses or as help system for independent individual work.

These small booklets have fulfilled its purpose if the reader has learned to express himself in the mathematically-related symbolic CAS Eigenmath-language as a mathematical language of communication and if he/she can use it to formulate problems as discussed here or to tackle own tasks in dialogue with this CAS. Thus the mathematical requirements to read this text are minimalistic.

Using this booklet with Eigenmath$^{online}$, no installation of any software is necessary, *everything runs directly online via WLAN*: a click on a link in this text is enough and the calculation is made[1], allowing further free inputs form the user. If you own a Mac,

---

[1]The output is printed below the input window – therefore sometimes one has to scroll to the left to see the output region with the result.

there is the option to install the app EIGENMATH free of charge and run the scripts by *mark–copy–paste* into the EIGENMATH window.

**Why EIGENMATH?**

EIGENMATH is a small but well designed and powerful computer algebra system (CAS), that can be used to solve problems in mathematics and the natural and engineering sciences. It is a personal resource for students, teachers and scientists. EIGENMATH is compact, capable and free.

EIGENMATH ...

- focuses straight to the point, no frills, no gimmicks,

- makes easy learning: only 44 pages manual [6], only 100 commands,
    mostly you will use only a good dozen of them for your work - that's all!

- has an intuitive user platform (IDE) with a two window frame,

- has mathematical language output in professional looking LATEX printing,

- allows distraction-free experiences in Mathematics,

- is ideally suited for rapid prototyping.

Yet EIGENMATH is powerful ...

- allows simple programming with well-chosen commands (do, for, test, check),

- has an intuitive function concept, but also allows advanced recursion,

- is an ideal free resource for undergraduates and graduates as well,

- simple work flow: just Write it down - and let it Run, that's it.

EIGENMATH's motto sentence could be EINSTEIN's:

> "*Everything should be made as simple as possible,*
>     *but not simpler.*"

Any feedback from the user is very welcome.

Being retired and no native speaker, I have no support from colleges at high school or university, therefore the reader may excuse me for my grammatical and spelling mistakes.

Wolfgang Lindner
Leichlingen, Germany
December 2020

# 1   First impressions - the 1$^{rst}$ model problem

The following problem introduces to the elementary linear algebra, which is about solving linear equations. We encounter typical questions, concepts and solution methods, which are important e.g. for a degree in economics, sociology, psychology, statistics, etc..
The following first problem is meant for orientation and attunement.

## 1.1   Population development

Around the year 2000 and 2005 the population of AUStralia remains constant with almost 25 million inhabitants. In early 2001, Sydney had a population of approximately 5 million. It is assumed that the following "migration" (per year) takes place:



We ask some questions about the situation depicted in the graphic. We answer or solve these questions in the following sections using the CAS EIGENMATH as assistant.

**Q1:** Briefly interpret the graphic.

**Q2:** We will describe ("*model*") the above situation mathematically with equations; therefore complete[2] the following approach:

$$u = 0.8x + 0.1y$$
$$v =$$

$$x + y = 25$$
$$u + v =$$

    a. What do $x, y, u, v$ mean?
    b. Why do we use two "pairs of equations"?

**Q3:** What was the population in and outside of Sydney at the end of 2001? And at the end of 2002? End of 2003?

---

[2]We give the solution to this question later. Think at first for yourself. ♡
Then look at the end of this section ...

**Q4:** What was the population in and outside of Sydney in early 2000? And in early 1999?

**Q5:** A "linear system" (LS) of equations (i.e. 2 equations thought as 1 thing) such as

$$1x + 2y = 3$$
$$4x + 5y = 6$$

can be written in matrix shape

$$\begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

$$A \quad * \quad X = B$$

Identifying the '2-by-2'-*matrix* $A = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$ and *'vector'* $B = \begin{pmatrix} 3 \\ 6 \end{pmatrix}$ and the unknown vector $X = \begin{pmatrix} x \\ y \end{pmatrix}$ the LS can be solved easily with EIGENMATH. There exists different methods, whose scope and differences we'll get to know over time. One solution idea mimics the method of 'cutting out according to X' in the matrix equation by means of the so-called *'inverse'* matrix of $A$, noted in Math: $A^{-1}$ EIGENMATH: `inv(A)` and by which the matrix equation 'is divided':[3]

```
A   * X = B              | 'divide' through A
        X = 1/A * B      | 1/A = inv(A)
```

We use the following lexicon and try this recipe in EIGENMATH:[4]

| Math | | EIGENMATH |
|---|---|---|
| $A^{-1}$ | | `inv(A)` |
| $A * X$ | | `dot(A,X)` |
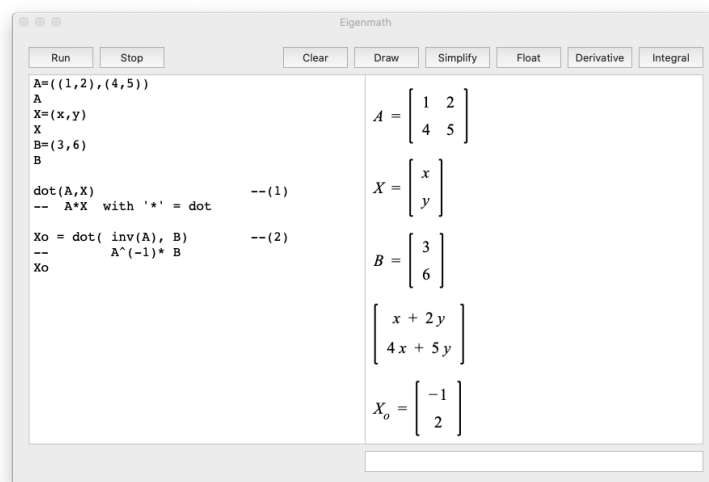


    a. *If you use* EIGENMATH *app on MacOS*: Type the commands (lines) on the left

---

[3]This serves as motivation and 'advanced organizer'. We justify it and expand on it it in the sequel.

[4]You can download the CAS EIGENMATH for free for iMac at `https://apps.apple.com/de/app/eigenmath/id584939279?mt=12`. The link for the executable online Demo follows further below.

side of the splitted screen line by line into your EIGENMATH window.
Press ENTER after every single command. Then press [Run].
Watch each output on the right hand side ('RHS') window.

*Otherwise*: use EIGENMATH$^{online}$ . ▷   *Click here to run Eigenmath online.*

b. Argue and explain, what `dot(A,X)` obviously does looking at line (1).
Do you see a rule how to calculate `dot(A,X)`?

c. In line (2) we name the solution `Xo`. EIGENMATH asserts, that $X_o = \binom{-1}{2}$.
Check the validity of this answer by brain using both systems equations.
How could you check this using EIGENMATH?[5]

d. Look at this analogy with 1x1-matrices, i.e. ordinary real numbers. That is we
discuss here[6] a linear system of 1 equation for 1 unknown. Explain the outputs:
_____   EIGENMATH _____

```
A=(2)
X=(x)
B=(1)

A*X
Xo = 1/A * B
Xo
Xo = A^(-1) * B
Xo
Xo =dot( inv(A), B)
Xo
```

- What looks the LS like? Solve in the head! Do you see 3 different solution methods?

Q**6:** One can look at this system of two linear equations

$$1x + 2y = 3$$
$$4x + 5y = 6$$

also in the so-called "vector form" $x \cdot A1 + y \cdot A2 = B$ and thus express the "coupling"
of the 2 individual equations into *one* (you see only one single '=') whole thing :

$$x \cdot \begin{bmatrix} 1 \\ 4 \end{bmatrix} + y \cdot \begin{bmatrix} 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

In EIGENMATH we write down this vector form of the LS in this way:

_____

[5]Ok, one writes the command `dot(A,Xo)` (read "$A$ times $X_o$"), which should give $B$.
[6]**Blue** lines are EIGENMATH input lines, which can be copied into the [Run] window.

```
A1 = (1,4)              -- 1. column
A2 = (2,5)              -- 2. column
A  = (A1,A2)            -- Matrix of both columns
A                       -- (?)
B = (3,6)
Xbad = dot( inv(A), B)
Xbad                    -- (1)
```

a. Run these commands[7] in the EIGENMATH input window,
e.g. ▷   *Click here to run the script* online.
What do you observe in line (?)? What about the offered 'solution' at line (1) ?

b. We have observed in a. that the *column* vectors A1 and A2 are registered as *rows*
in matrix $A = (A1, A2)$ - i.e. they are 'mirrored' at the left-above-right-down [↘]
diagonal! This must be corrected i.e. reversed and is exactly the job of the command
transpose[8]. With this correction we get the same solution $X_{good} = X_o$:



c. Using EIGENMATH app on the iMac or EIGENMATH$^{online9}$ :
Append and execute the last 4 lines of the above commands into the input region.

**Q7:** The solution of the LS in Q6, when given in this vector form, can be interpreted
geometrically as a so-called *linear combination* depicted in Fig. 1.
What do you observe? Where "is" the solution?

a. In your own words: what is the geometric interpretation of the solution $X_o = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$?

b. What happens in the figure on the right, if you form the linear combination of
A1 and A2 using any other number pair e.g. $(0, 3)$ or $(1, 2.5)$ instead of the numbers
$(-1, 2)$ from the solution vector?

---

[7]The double hyphen '--' signals a comment. See the EIGENMATH-manual.
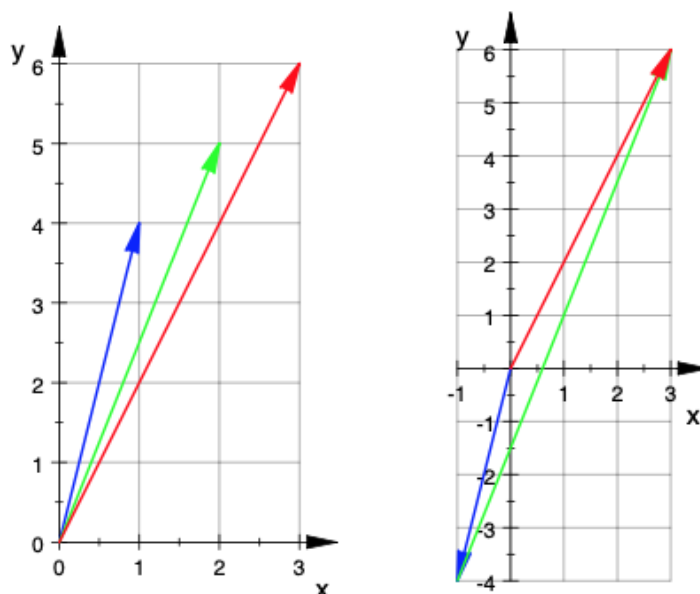[8]Read about *transpose* in the EIGENMATH help or in the EIGENMATH handbook.
[9]https://georgeweigt.github.io/eigenmath-demo.html

Figure 1: On the left figure we see the vector $A1$ painted in blue, $A2$ in. . . .
The figure on the right "shows" the solution vector $X_o = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$ in masked
form: the first coordinate -1 of $X_o$ directs $A1$ in opposite direction to the point
$(-1, -4)$, from there we have to go 2 (=the second coordinate of $X_o$) times
along $A2$ i.e. 4 steps eastward and 10 steps to the north to reach the target
point $B$ at $(3, 6)$.

  c. What do you think is a "linear combination" of the columns $A1$ and $A2$?

**Q8:** Let us change this LS at one number:
$$1x+2y=3$$
$$4x+8y=6$$

Try to calculate the solution with paper and pencil and with EIGENMATH.
What happens? Do you have an explanation for the phenomenon?

**Q9:** Let us change this LS at two numbers:
$$1x+2y=\ \ 3$$
$$4x+8y=12$$

Try to calculate the solution with paper and pencil and with EIGENMATH.
What happens? Do you have an explanation for the phenomenon?

**Q10:** What is the behavior of the population distribution from 2000 to 2005?
Can one make forecasts for the "distant" future?

**Q11:** Is there a population distribution inside and outside of Sydney, for which there is
no overall change in population from one year to the next year - even though 20% of
the Sydney's emigrate and 10% immigrate from outside?

In Q5 to Q6 we initially deviated from the actual problem, but will come back to it after these interim considerations.
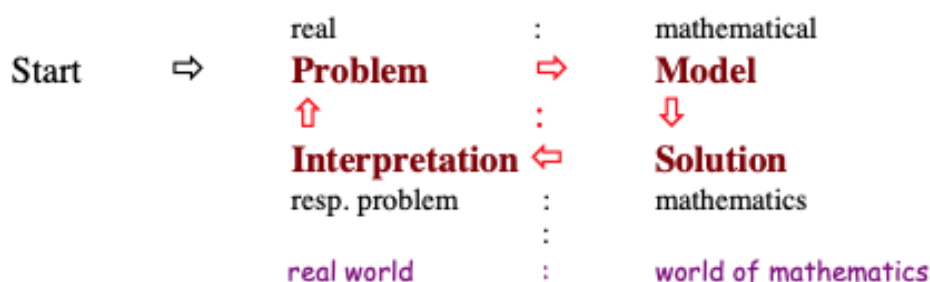
### Model.

*A **model** is a conceptual construct, which is interpreted into the actually existing reality in order to be able to apply mathematical methods and findings.*
*Mathematical models are often formulated in the form of equations.*

Here is a diagrammatic representation of a typical "Modeling cycle":

|  |  | real | : | mathematical |
|---|---|---|---|---|
| Start | ⇨ | **Problem** | ⇨ | **Model** |
|  |  | ⇧ | : | ⇩ |
|  |  | **Interpretation** ⇦ |  | **Solution** |
|  |  | resp. problem | : | mathematics |
|  |  |  | : |  |
|  |  | real world | : | world of mathematics |

a. Illustrate the model building process using the example of population development.
b. Reflect: at which phase of the cycle
- is human mental arithmetic (still) useful,
- is EIGENMATH-supported computing effective and increase our productivity?

In the next section we start a deeper understanding of the concepts and methods we have only indicated and vaguely outlined in this first encounter.[10] We begin with the concept of a matrix.

---

[10]Here is the solution to **Q2**: $u = 0.8x + 0.1y$, $v = 0.2x + 0.9y$, $x + y = 25$, $u + v = 25$.

## 1.2   Population development - modeling and solution

We now solve problem 1.1. For the mathematical modeling of the problem, we get to know the concept of a 'matrix', with their help we can build "coupled" linear equations in a compact manner - so-called "*linear systems of equations*" (LS for short). We learn to know the "star operator" $*$ of Mathematics, i.e. the EIGENMATH function `dot()`, as an important tool.

### 1.2.1   modeling the situation – back to Q2.

In many sciences such as ecology, economics and engineering, one needs mathematical models to describe dynamic systems that change over time. Discrete states of the system are measured at certain times and the changes are studied. We now solve Q2 together with the help of EIGENMATH as our assistant.

> ((Remember: It is estimated that between 2000 and 2005 the population of Australia remains constant with almost 25 million inhabitants. In early 2001, the region of Sydney had a population of approximately 5 million. We will assume that the following "migration" (per year) takes place.))

Step **1**    *To solve the problem, we first set up a mathematical model, in words.*

We write down the statements of the graph as equations line by line to formulate the problem clearly:
- $x$ million of 25 live in and $y$ million outside of Sydney at the beginning of the year.
- At the end of the year, 80% of the $x$ stay in Sydney, but 20% of the $x$ move out.
- At the end of the year, 90% of $y$ stay outside, but 10% of $y$ move in.

Step **2**    Present the problem more clearly *in tabular form.*

|  | inner Sydney | | outer · Sydney | |
| --- | --- | --- | --- | --- |
| begin of year | x | | y | |
|  | take 80% | take 20% | take 10% | take 90% |
| splitting | 0.80x | 0.20x | 0.10y | 0.90y |
| distribution | 0.80x | 0.10y | 0.20x | 0.90y |
| end of year | 0.8x+0.1y | | 0.2x+0.9y | |
|  | =u | | =v | |

Step **3**    "model" the problem *exactly with mathematical equations.*

$$
\begin{aligned}
u &= 0.8x + 0.1y & &\text{remain in Sydney at end of year}\\
v &= 0.2x + 0.9y & &\text{live outside of Sydney at end of year}\\
&\text{and}\\
x + y &= 25\\
u + v &= 25
\end{aligned}
$$

*Exercise.* Using the modeling, calculate the population inside and outside of Sydney at the
end of 2001, when 5 million people lived in Sydney at the beginning of the year.

### 1.2.2   compact modeling using matrices

Such systems of equations appear very frequently in the following. In practice, these can
consist of more than 1000 equations. We therefore want to write such large systems of
equations compactly and clearly and learn to solve them with EIGENMATH.
For this purpose, a system of linear equations is reduced to its essential data and compressed
("zipped") to a so-called "matrix equation":

<div style="text-align:center;">

```
0.8x + 0.1y = u
0.2x + 0.9y = v
```

</div>

Compression of the LS to one matrix equa-
tion A * X = B. The components of the LS
i.e. data A, unknown X and right side B
remain distinguishable.

$$\begin{pmatrix} 0.8 & 0.1 \\ 0.2 & 0.9 \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix}$$

$$\begin{array}{ccc}
\downarrow & \downarrow & \downarrow \\
A & * \; X = & B \\
data & unknown & right\ side
\end{array}$$

Repeated compression of the LS to a single equation
matrix M. The unknowns of the LS are left out.
Data A and Right Hand Side (RHS) B must mentally
be separated.

$$\begin{pmatrix} 0.8 & 0.1 & u \\ 0.2 & 0.9 & v \end{pmatrix}$$

Both types of compression can be written down in EIGENMATH:

```
A=((0.8,0.1),(0.2,0.9))    -- LHS of system
A
A=((0.8, 0.1),             -- same more readable
   (0.2, 0.9))
A
X=(x,y)                    -- the unknowns
B=(u,v)                    -- RHS of system

M=((0.8, 0.1, u),          -- equation matrix
   (0.2, 0.9, v))
M

X = (5,20)
U = dot(A,X)               -- dot produces the result
U
```

$$A = \begin{bmatrix} 0.8 & 0.1 \\ 0.2 & 0.9 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.8 & 0.1 \\ 0.2 & 0.9 \end{bmatrix}$$

$$M = \begin{bmatrix} 0.8 & 0.1 & u \\ 0.2 & 0.9 & v \end{bmatrix}$$

$$U = \begin{bmatrix} 6 \\ 19 \end{bmatrix}$$

▷   *Click here to run the script.*

a. Look, how the black box machine `dot(.)` produces the solution for the end of 2001.
b. Calculate the distribution of the inhabitants at the end of 2002 and 2003.
c. Arrgue, how `dot(.)` may work.

### 1.2.3   partial modeling using a function – back to Q3.

((What behavior does the population distribution show over 6 years? Make forecasts for the future.))

We will further explore and try to detect how `dot(.)` works. To this end we discuss at first a solution of **Q3** by means of a function.
According to Q2 we can use our mathematical model

$$0.8x + 0.1y = u$$
$$0.2x + 0.9y = v$$

to calculate a final state $(u, v)$ for a given start pair $(x, y)$ and take this $(u, v)$ as the new start state $(x, y)$. This procedure is iterated in the same way.

a. *Exercise.* Verify, that the first 3 states are $(50, 220), (62, 208), (70.4, 199.6)$.

We now formulate this process in tree different ways.
... as a *mental machine* named $f$, which gets $(x, y)$ as input and gives $(u, v)$ back:

$$(x, y)$$



**f = „insert in the left side of LS and calculate"**
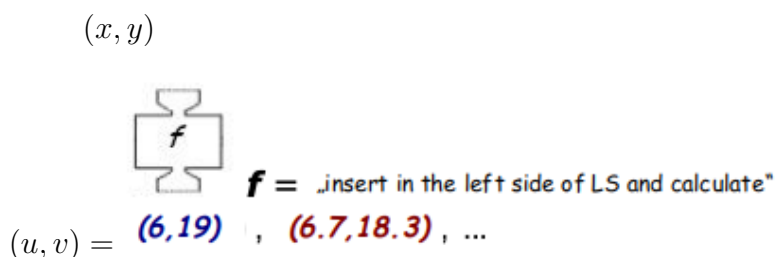
$(u, v) =$  **(6,19)**  ,  **(6.7,18.3)** , ...

... in *mathematical notation* as function $f$ :

$$
\begin{aligned}
f: (x, y) &\mapsto (0.8 \cdot x + 0.1 \cdot y, 0.2 \cdot x + 0.9 \cdot y) = (u, v) \\
(5, 20) &\mapsto (6, 19) \\
(6, 19) &\mapsto (6.7, 18.3) \\
(6.7, 18.3) &\mapsto \ \ ...
\end{aligned}
$$

... in EIGENMATH notation as function $f$:
_____ EIGENMATH _____

```
f(x,y) = (0.8*x + 0.1*y, 0.2*x + 0.9*y)

f(5,20)
last                    --(1)
```

```
    f(last[1],last[2])    --(2)
    last
    f(last[1],last[2])    --(3)
    f(last[1],last[2])
```

EIGENMATH output: last=[6,19], last=[6.7,18.3], last=[7.533,17.467].
▷   *Click here to run the script.*

**Comment.** In (1) $f$ gives $(6, 19) = (u, v)$ back for the start pair $(x, y) = (5, 20)$, which is saved on the system variable `last` 'value', i.e. `last`= $(6, 19)$ and its first component is `last`$[1] = 6$. Dropping these last value components into $f$ again gives the next value $(6.7, 18.3)$ and so on.

b. *Exercise.* Calculate the first 6 value pairs using `last`, which solves **Q3**.

### 1.2.4   Enhancing `last` using matrix techniques – back to Q10.

When using the EIGENMATH "flash memory" named `last`, you do not need to read the intermediate results off the screen at all in order to calculate successive pairs of values with $f$. But it is annoying to count the number of repetition cycles of the calculating machine in your head or on a paper sheet. Therefore we look at turbo techniques to compute the future values of our model problem,

> ((**Q10**: to output a table of the population development from 2001 to 2020. Get a forecast for the population of Sydney in 2100. In which year does the last observable change take place?))

Step **1**     *using a matrix as container for the f-values:*

```
f(x,y) = (0.8*x + 0.1*y, 0.2*x + 0.9*y)

f(5,20)           -- (1)

f(5,20)[1]      -- (2)
f(5,20)[2]      -- (3)

M = zero(4,2)      -- as container of values
M

M[1] = ( f(5,20)[1],f(5,20)[2] )   -- start line
M[1]                -- (4)
M                   -- (5)

M[2] = f(M[1,1],M[1,2])   --(6)
M[3] = f(M[2,1],M[2,2])
M[4] = f(M[3,1],M[3,2])
M                         --(7)
```

$$\begin{bmatrix} 6 \\ 19 \end{bmatrix}$$

$6$

$19$

$$M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 6 \\ 19 \end{bmatrix}$$

and further
$$M = \begin{bmatrix} 6 & 19 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad M = \begin{bmatrix} 6 & 19 \\ 6.7 & 18.3 \\ 7.19 & 17.81 \\ 7.533 & 17.467 \end{bmatrix}$$

**Comment.** In (1) $f$ gives $(6, 19)$ back, ok we know that. in (2) and (3) we pick the first and second coordinate of that value using the indexed access [..]. Then we set up a matrix named $M$ with 4 rows and 2 columns filled with zero's - you see $M$ in the output window as the 4th entry. In (4) we feed the 1st row $M[1]$ of $M$ with a start value and let EIGENMATH present this row and the matrix ("table") with the changed 1st row. In (6) to (7) we fill this table with the next $f$-calculated pairs.

• Do you see a pattern in (6)?        ▷    *Click here to run the script.*

Step **2**     *automating the iteration using an* EIGENMATH-`for`-*loop*

Let's get a profit of the recognized pattern in step**1** to put the $f$-values into the table $M$ and set up a loop. This way we abstract and generalize the pattern in step**1**.
——————— EIGENMATH ———————

```
f(x,y) = (0.8*x + 0.1*y, 0.2*x + 0.9*y)
M = zero(4,2)

M[1] = ( f(5,20)[1], f(5,20)[2] )

for(i,2,4,
          M[i] = f( M[i-1,1], M[i-1,2] )) --(1)
M[4]                                      --(2)
```

EIGENMATH user 0utput:      $\begin{bmatrix} 7.533 \\ 17.467 \end{bmatrix}$

▷    *Click here to run the script.*

**Comment.** Ok, first we define the modeling function $f$, set up the zero-table $M$ and put in the start pair as row $M[1]$. The `for` loop in (1) is running from $i = 1$ to $i = 4$ calculating the new row $M[i]$ using the previous coordinates $M[i-1, ..]$ of the row before with index $i - 1$. In (2) we give back the 4th row i.e. the population state in 2004.

• Could you enhance the matrix table with a third column showing the year of the status?

Step **3**     *... tuning with the matrix turbo*
——————— EIGENMATH ———————

```
A=((0.8,0.1),(0.2,0.9))
A
X=(5,20)
```

```
X

dot(A,X)                --(1)
dot(A,A,X)              --(2)
dot(A,A,A,X)            --(3)

dot(A^9,((5,20)))       --(4)
```

$$A = \begin{bmatrix} 0.8 & 0.1 \\ 0.2 & 0.9 \end{bmatrix} \qquad \begin{bmatrix} 6.7 \\ 18.3 \end{bmatrix}$$

$$X = \begin{bmatrix} 5 \\ 20 \end{bmatrix} \qquad \begin{bmatrix} 7.19 \\ 17.81 \end{bmatrix}$$

EIGENMATH user 0utput:

$$\begin{bmatrix} 6 \\ 19 \end{bmatrix} \quad \dots \quad \begin{bmatrix} 8.19882 \\ 16.8012 \end{bmatrix}$$

▷    *Click here to run the script.*

The cascade $(1)(2)(3)(4)$ of $dot(A, .., X)$ commands calculates *most easily the results*! Therefore we do not need any more a for loop[11] to calculate an end status using *all* computed previous values. Especially (4) demonstrates the impact of the 9-th power of system matrix $A$ on the start vector $(5, 20)$. We may use these tabulated values to draw a figure of the population development:



$$\begin{bmatrix} 5, 20 \end{bmatrix}$$
$$\begin{bmatrix} 6, 19 \end{bmatrix}$$
$$\begin{bmatrix} 6.7, 18.3 \end{bmatrix}$$
$$\begin{bmatrix} 7.19, 17.81 \end{bmatrix}$$
$$\begin{bmatrix} 7.533, 17.467 \end{bmatrix}$$

⋈

---
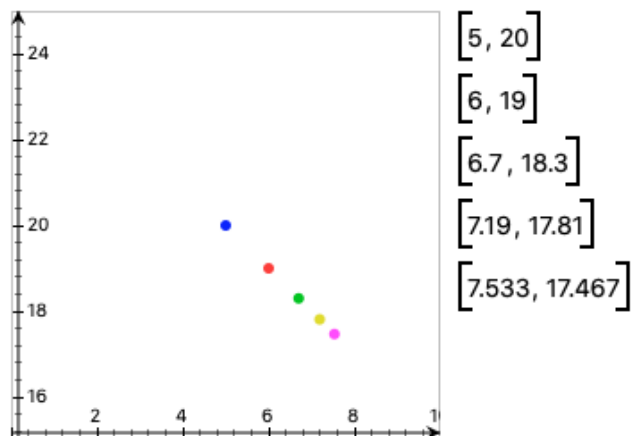
[11]I am grateful to George WEIGT for a tip, which demonstrates the power of EIGENMATH to allow an advanced recursive solution. This may be of interest to the hobby programmer:

```
f(x,y) = (0.8x+0.1y, 0.2x+0.9y)
g(n,x,y) = test( n < 2, f(x,y), g(n - 1,f(x,y)[1],f(x,y)[2]) )
g(4, 5,20)
```

which results again in $(7.19, 17.81)$.

In this introduction we have presented first glimpses in a more or less informal manner. We hope to arouse the readers interest e.g. what is inside the black box `dot(.)`. This and a bit formal matrix algebra is topic of the next section.

## 1.3   Problems

**P1. The search for the fixed state – back to Q11.**   Determine a population distribution $(x, y)$ in and out of Sydney, for which there is no further change in population from one year to the next, although 20% of Sydney's emigrate and 10% immigrate from outside.

> Just as a fixed star stands unchanged in the sky, in this way a "fixed" state (the "fixed vector") of the population let the population remain stable.

a. How is the stability of the population distribution expressed in the equation model?
b. Solve the problem first without the help of EIGENMATH. *Hint:* $f(x, y) \overset{!}{=} (x, y)$
c. Solve the problem with the help of EIGENMATH.


**P2.**    Here is a new machine $A = \begin{pmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{pmatrix}$.
Take the same start population $(x, y) = (5, 20)$.
Repeat the interesting parts of the investigation above with this new "population" matrix.
What do you notice. Give reasons for your considerations.


**P3.**    Here is another transformation matrix ("machine") $A = \begin{pmatrix} 0.8 & 0.15 \\ 0.15 & 0.9 \end{pmatrix}$.
a. Repeat the investigation from P1.2 with this new matrix and the old $X = (50, 20)$.
What do you notice. Give reasons for your considerations.
b. Make experiments with your own matrices.


**P4. Population one year before in 1999 – back to Q4.**
Solve **Q4**, which is an *turning back problem*, see P1.8.
Try many different solution methods.
Also use paper and pencil.


**P5. How does `dot(.)` works?**   This exercise will help you to explore the effect of the important EIGENMATH function `dot(.)`.
a. Look back on the function $f(x, y) = (0.8x + 0.1y, 0.2x + 0.9y)$.
Describe how you calculate the function value for the input $(x, y) = (10, 20)$ in your head.
b. How to calculate $f(a, b)$? Result?
c. Look at 1.2.4 step**3**. What is the value of `dot(A,X)`? What is $A$?
Compute the value `dot(A, (10,20))` with EIGENMATH and compare with a..
d. Show with the help of EIGENMATH that $f(a, b) = $ `dot(A, (a,b))`!
e. Now, if you have a *variable* input $X = (x, y)$, what is the value of    `dot(A, (a,b))`?
Can you give a recipe in your own words how to calculate `dot(A, X)`?
If $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, what is `dot(A, X)` for $X = \begin{pmatrix} x \\ y \end{pmatrix}$. Formulate a formula or an recipe.

Remember:

$$
\begin{array}{c|c}
Math & \text{EIGENMATH} \\
A * X & \texttt{dot(A,X)}
\end{array}
$$

f. How to calculate in your head $\begin{pmatrix} a\ b \\ c\ d \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix}$? Formulate a formula for calculating $A * X$!
Use your own words.

**P6. Andrei Andreyevich Markov.**
a. Read about `Markov` in `https://mathshistory.st-andrews.ac.uk/Biographies/Markov/`
or    `https://en.wikipedia.org/wiki/Andrey_Markov`.
Markov studied population matrices like the one from P1.1. Such matrices are therefore
also called *Markov matrices* in his honor.
b. Which property/s must a matrix have to be a Markov matrix?
c. Give examples and counterexamples of Markov matrices.
d*. ♡Only for EIGENMATH enthusiasts♡: Write a function  `isMarkov` that tests whether
a given matrix $M$ is a Markov matrix.

**Info**: *The modeling of a long-term development process with Markov matrices $M$ is called
a 'Markov process'.*

**P7.    Strategy considerations and research questions.**    The problems from this
first section **1** represent typical questions that mathematicians often ask themselves:

    1:   How do I represent a problem mathematically?    (*Modeling question*)
    2:   Is there a solution to the problem at all?    (*Existence question*)
    3:   Is there only (exactly) *one* solution?    (*Uniqueness question*)
    4:   Are other approaches to the solution conceivable?    (*Method question*)

• Comment on these key strategic questions using the example of population development.

**P8.**  *Review*: **Systems of linear equations modeling many different problems.**
This section 1 discusses e.g. the following sub-problems:

| | | |
|---:|---|---|
| Q2; 1.2.1: | a linear model | (*"model problem"*) |
| Q2; 1.2.2: | a linear substitution problem | (*"replace problem"*) |
| Q4: | a linear inversion problem | (*"turn back problem"*) |
| Q11: | a linear eigenvalue problem | (*"fixed vector search"*) |
| P1/2: | a linear Markov problem | (*"Markov matrix sequence"*) |

The complex problem **Q1** thus offers an initial insight and overview of typical tasks from Linear Algebra. We will learn more about this and the omnipresent ∗-Operator resp. the `dot(.)` function which ...

... models the individual process step in our model problem as a computing machine
... realize $A^n$ as an abbreviation for $A * A * ... * A$ (with n repetitions, i.e. n "factors")

in the next section.

# 2 Matrix Arithmetic

A matrix can describe a dynamic process such as a population growth or a static object e.g. a system of equations matrix. This section deals specifically with the '*star*' operator $*$, which is known as *matrix multiplication*. We learn its usefulness in solving specific Linear Systems. First we study the arithmetic of matrices. We learn how to add and multiply two matrices and also to calculate the multiple of a matrix.

The last chapter had demonstrated, that the `dot(.)` function of EIGENMATH plays a central role. Therefore we begin our exploration with the solution to P5.

**P9. *Exploration*: How does `dot(.)` works?.**

```
dot( ((a,b),(c,d)), (x,y) )
--    A            * X

dot( ((a,b),(c,d)), ((m,n),(p,q)) )
--    A            * B
```

$$\begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

$$\begin{bmatrix} am + bp & an + bq \\ cm + dp & cn + dq \end{bmatrix}$$

a. In your own words: how to calculate the "*dot*" of two matrices $A$ and $B$?

b. Let $A = \begin{pmatrix} 1 & 2 \\ c & d \end{pmatrix}$ and $B = \begin{pmatrix} 9 & 6 \\ 8 & 5 \end{pmatrix}$. Looking at the script, the result `dot(A,B)` should be again a $2 \times 2$ matrix. Which value is at position $[1,1]$ of that matrix, i.e at their left-above corner? What is the result at position $[2,2]$? - Check your answer with EIGENMATH.

## 2.1 *Definition*: product, sum, multiple of matrices

**Let** $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ and $B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$ be arbitrary $2 \times 2$ matrices, $m$ an arbitrary[12] number,

**then** we define

the *product* $\mathbf{A} * \mathbf{B}$ of $A$ and $B$,

the *sum* $\mathbf{A} + \mathbf{B}$ of $A$ and $B$, and

the $m$-fold *multiple* $\mathbf{m} \cdot \mathbf{A}$ of $A$ through:

$$\text{New Math concept} \overset{\text{def}}{=} \textit{which is defined through this formula:} \tag{2.1}$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \overset{\text{def}}{=} \begin{pmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} & a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} & a_{21} \cdot b_{12} + a_{22} \cdot b_{22} \end{pmatrix} \tag{2.2}$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \overset{\text{def}}{=} \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{pmatrix} \tag{2.3}$$

$$m \cdot \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \overset{\text{def}}{=} \begin{pmatrix} m \cdot a_{11} & m \cdot a_{12} \\ m \cdot a_{21} & m \cdot a_{22} \end{pmatrix} \tag{2.4}$$

---

[12]EIGENMATH number, i.e. integer, rational, real or complex.

To deepen your concept image[4] of the *product of matrices* beyond the symbolic-formal view in (2.2) we recommend to think of it as ...

**...** a **rule in words**: the entry at position $(A * B)[i, j]$ of the product is the "sum-product" of the $i$-th row of $A$ with the $j$-the column of $B$, i.e. "$A_{i\bullet} \bullet B_{\bullet j}$", mentally in your head:

$$\equiv * ||| \quad = \quad \vdots\vdots\vdots$$



**...** a **zipping idea**: each red row is $\overset{\bullet}{+}$ - zipped with each blue column, i.e.



red row $\overset{\bullet}{+}$ blue col

**...** a FALK[13] **scheme**: use a tableaux for paper & pencil calculation



**Remark.** Think of the symbol $*$ as 'overlay' $\overset{\times}{+}$ of the zipping operations *multiply* $\times$ *and add* $+$, i.e. the request to "*sum up the products*".

### 2.1.1   FALK **scheme simulation in** EIGENMATH.

——————— EIGENMATH ———————

```
dot( ((1,2,3),              -- FALK arrangement
      (4,5,6),              -- think A arranged as rows
      (7,8,9)),
                 ((9,8,7),   -- think B
                  (6,5,4),   -- arranged
                  (3,3,1)) ) -- as columns
```

---

[13]Sigurd FALK (*6. May 1921; †Mai 2016), German mathematician and engineer.

$$\begin{bmatrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & 90 \end{bmatrix}$$

EIGENMATH output:

▷   *Click here to run the script.*

This script demonstrates, that the smart input editor of EIGENMATH allows to arrange a FALK calculation scheme for a matrix product $A * B$.

a. How would the input line looks like, if you use a *one* 'horizontal' matrix input line?

b. Calculate `dot(((1,-1,1),(4,-5,3),(2,7,-2)),((7,-8,9),(-3,2,1),(6,5,-4)))` without EIGENMATH. Then check your result with EIGENMATH, using a FALK–like structured input line $\begin{smallmatrix} & * & B \\ A & & C \end{smallmatrix}$ .

## 2.2   *Training*: **product, sum, multiple**

| Math | | EIGENMATH | concept |
|---|---|---|---|
| $A + B$ | \| | `A+B` | **sum** of matrices |
| $A * B$ | \| | `dot(A,B)` | **product** of matrices |
| $X \bullet Y$ | \| | `dot(X,Y)` | **scalarproduct** of vectors |
| $m \cdot A$ | \| | `m*A` | m-times **multiple** of matrix |

In this training, we expand our previous experience in computing with matrices, which we gained in connection with our model problem of linear systems of equations. We have not systematically met two arithmetic operations with matrices that will appear in this exercise.

First solve each task in your head tentatively and "intuitively", as it makes sense to you. Then use EIGENMATH as assistant to check your mathematical intuition as well as the results. Be warned:

- the mathematical operations $A * B$ and $m \cdot A$ are both noted with $*$ in EIGENMATH, so *watch the context.*

- sometimes an operation is not executable, because of an '*typ mismatch*': a matrix is of *typ* $m \times n$ (in short $\overset{m \times n}{A}$), if $A$ has $m$ rows and $n$ cols.

- we can check the *typ* of a matrix using the EIGENMATH-function `dim(.)`, [14] e.g. the matrix `A=((1,2),(3,4),(4,5))` has typ $3 \times 2$, therefore $\overset{3 \times 2}{A}$ or $\overset{[\equiv]\,3}{_2}$:

---

[14] *dim*: short for *dimension*, i.e. here the expansion in two directions.,

```
A=((1,2),(3,4),(4,5))
A

dim(A,1)
dim(A,2)
```

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 4 & 5 \end{bmatrix}$$

3

2

**Exercise.** a. `A=((0,-2,1),(-1,1,3))` and `B=((3,2,1),(-1,-2,-3))`. Calculate $A + B$.
b. $B - 2 * A = ?$ with $A, B$ from a.
c. `C =((0,1),(2,3),(-4,5))`. What is $A + C$? - Why does EIGENMATH reports an *error*?
d. Let `X=(2,-1,3)` be a vector. Calculate $A - X$.
e. Compute $A * B$. - *Hint*: Error: dimensions do not match!
f. Compute $A * C$.
g. Let `Y=(3,2,-4)` be another vector. Calculate $A * Y$.
h. Compute $A * A$ using the FALK scheme. Check via `dot(A,A)`.
i. Let `Z = ((3,1,4),(-6,-2,0),(5,8,-7))`. Set up the FALK scheme and calculate $A * Z$ with paper& pencil. Check your result with EIGENMATH.
j. Compute $Z * Z$ using the FALK scheme. Check via `dot(Z,Z)`.
k. Compute $Z^2$. - Solution: `dot(Z,Z)` == `Z^2`.

```
Z = ((3,1,4),(-6,-2,0),(5,8,-7))
Z

dot(Z,Z)
Z^2
```

$$Z = \begin{bmatrix} 3 & 1 & 4 \\ -6 & -2 & 0 \\ 5 & 8 & -7 \end{bmatrix}$$

$$\begin{bmatrix} 23 & 33 & -16 \\ -6 & -2 & -24 \\ -68 & -67 & 69 \end{bmatrix}$$

$$\begin{bmatrix} 23 & 33 & -16 \\ -6 & -2 & -24 \\ -68 & -67 & 69 \end{bmatrix}$$

▷    *Click here to invoke* EIGENMATH*online*.

ℓ. Compute $(A + 2 * B) * C$.
Solution: ♡ please - try it for yourself before you consult he footnote[15].

---

[15]The first $*$ is a multiple $2 \cdot B$. The `(..)` is then a matrix of same typ like $A$. Therefore $(..) * C$ is a constructible matrix. In EIGENMATH we write this   `dot(A+2*B,C)`

## 2.3   Matrices as geometrical figures and transformations

A matrix can describe a dynamic process or a static object. Here we consider geometric problems in which matrices can be interpreted dynamically as images of transformations (e.g. mirroring) as well as statically as figures (objects).

So far we have got to know matrices as tools for compact solving of linear systems of equations. We are now studying matrices

– as tools for saving a static geometric objects e.g. a square □ and

– to describe geometrical actions such as reflections   .|.   on axes.[16]

### 2.3.1   matrices as geometric figures (”*objects*”)

We interpret the rows of matrix $F = ((0,0),(1,0),(1,1),(0,1),(0,0))$ as points of a *polygon* in the plane $\mathbb{R}^2$: they form the unit square □. The origin $(0,0)$ is repeated in this list $\overset{5\times 2}{F}$ of points, because we *draw this figure without lifting the pencil off the paper* while going from point to point and at the end go back from $(0,1)$ to the origin.
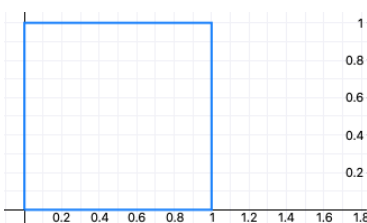
―――――――――― EIGENMATH ――――――――――

```
F = transpose(((0,0),(1,0),(1,1),(0,1),(0,0)))

F                    -- standard square at origin
```

▷   *Click here to run the script.*

In EIGENMATH we took the transpose of the list $F$ to have the points as a vector list. We plot figure $F$ e.g. with paper & pencil:



### 2.3.2   matrices as geometric actions (”*transformations*”)

We apply a simple geometric action to the original object figure $F$. The next script shows, what matrix $A = ((1,0),(0.5,1))$ causes, when it 'acts' on figure $F$ from the left via *Math:* $A*F$ .
EIGENMATH: dot(A,F)

―――――――――――――――――――――――――

[16]Much more on this in my booklet *Linear Transformations interactive!* with EIGENMATH. To appear in this series.

```
F = transpose(((0,0),(1,0),(1,1),(0,1),(0,0)))

F                          -- standard square at origin

A=((1,0),(0.5,1))    -- transformation of figure
A

dot(A, F)            -- transformed figure F(4)'
```

$$F = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 \\ 0.5 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0.5 & 1.5 & 1 & 0 \end{bmatrix}$$

The script shows in (4) the table of coordinates of the transformed points $A * F$, which gives the red image figure in the following plot:



○   Put the geometric effect of *transformation matrix* $A$ in words:
How does the figure $A * F$ arise from the original figure $F$?

**Remark.** We note:

- factor matrices $A$ on the left of the matrix product $A*X$[17] act as transformations

- factors $X$ on the right of $A*X$ are figures to be mapped ("geometric object")

- the calculated product value $A * X$ represents the image object.

In a nutshell:



matrix *A* as
*process,*
*transformation,*
*machine ..*

matrix *X* as
*object,*
*figure,*
*data table ..*

_____
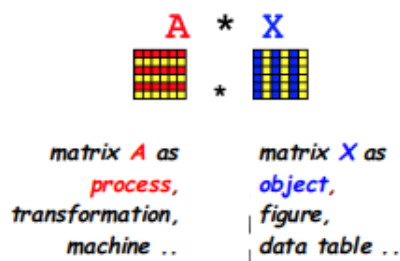
[17]The notation $A * X$ remembers at a function value f(x) i.e. $\frac{A*X}{f(x)}$ .

### 2.3.3   Compositions of transformations as matrix products.

```
scale(r,s) =((r,0),(0,s))          -- (1)
scale(2, 0.5)
```

$$\begin{bmatrix} 2 & 0 \\ 0 & 0.5 \end{bmatrix}$$

```
dot(scale(2,0.5), F)               -- (2)
```

$$\begin{bmatrix} 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 \end{bmatrix}$$

```
dot(A, scale(2, 0.5), F)           -- (3)
```

$$\begin{bmatrix} 0 & 2 & 2 & 0 & 0 \\ 0 & 1 & 1.5 & 0.5 & 0 \end{bmatrix}$$

```
dot(scale(2, 0.5), A, F)           -- (4)
```

$$\begin{bmatrix} 0 & 2 & 2 & 0 & 0 \\ 0 & 0.25 & 0.75 & 0.5 & 0 \end{bmatrix}$$

▷   *Click here to run the script.*

Geometric actions can be executed one after the other. This corresponds algebraically to the multiplication of the associated transformation matrices. The previous EIGENMATH script demonstrates such a geometric interlinking process using the scaling and shearing (e.g. transformation $A$ in 2.3.2) of figures.

**Comment.** In (1) we define the *scal*ing transformation `scale(r,s)` for arbitrary values $r$ and $s$ and use a special exemplar  `scale(2,1/2)`   for the next concrete calculations. (2) calculates the scaled image  `scale(2, 1/2) * F`  of figure $F$. Check the coordinates.
In script line (3) the scaled figure $F$ is sheared by means of matrix $A$ to give a new figure $F''$. In summa we have the mapping:
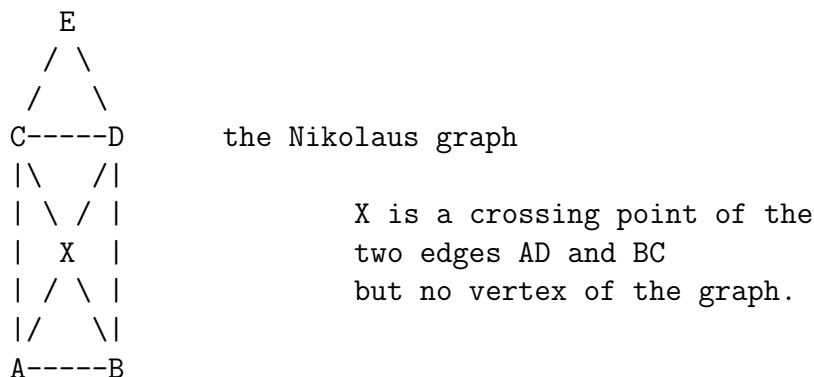
$$F \xmapsto{scale} F' \xmapsto{A} F''$$

**Exercise.** a. Sketch the images of actions (3) and (4). Draw the picture in your head.
b. Formulate expression (4) as a mathematical term using $*$.
Calculate the image $F''$ via a FALK scheme.

**P10. The house of St Nicholas.** [18]
a. Draw the Nikolaus puzzle without lifting your pencil off the paper or going along the same line twice. Count the number of possibilities.
Note: In the sketch all lines are meant *not be dashed*.

```
    E
   / \
  /   \
 C-----D        the Nikolaus graph
 |\   /|
 | \ / |              X is a crossing point of the
 |  X  |              two edges AD and BC
 | / \ |              but no vertex of the graph.
 |/   \|
 A-----B
```

b. Give one solution in matrix form as point list 'Niko' like 2.3.1.
c. Scale the figure `Nico` with `scale(0.5,0.5)` and plot/draw the result.
d. Formulate the reflection-transformation on the line $AB$ as a matrix and calculate the image of the scaled-reflected `Niko`. Use EIGENMATH.

## 2.4  Problems

**P11. Lightning computing.**   Calculate in your head.
Check - if necessary - with EIGENMATH.

$$(1\ 0) * \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} =?  \qquad \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} =?  \qquad \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \end{pmatrix} =?$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} =?  \qquad \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} =?  \qquad \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \end{pmatrix} =?$$

$$(1\ 0) * \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 1 \\ 2 \end{pmatrix} =?  \qquad \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} =?$$

**P12. Checking solutions.**

```
1x+2y = 5
3x+4y = 3
```

a. Write this linear system as a matrix equation $A * X = B$.
b. Verify: $X1 = (10, -5.5)$ and $X2 = (-7, 6)$ are two solutions of the linear system.
c. Stack both solutions in *one* matrix `S=transpose(X1,X2)` [19] and check the solution with EIGENMATH simultaneously for both solutions using a matrix product.

---

[18] We quote `https://www.math.uni-bielefeld.de/~sillke/PUZZLES/nikolaus`: "In Germany this problem is known under the name 'The house of the St Nicholas'. A person drawing this figure is saying the rhyme '*This is the house of Ni-ko-laus*' at each stroke one syllable."
[19] Why the `transpose`? Try without ...

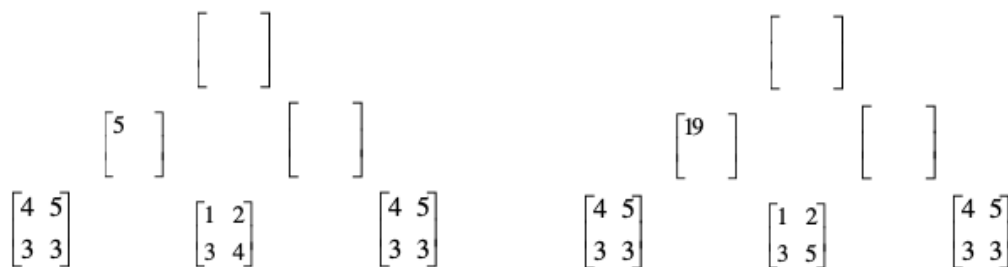**P13. Matrix Walls.**   Here is a "matrix tower": stack the other matrices stones.



Figure 2:                                          matrix stones *multiplicatively* stacked
matrix stones *additively* stacked


a.  How to choose the numbers in the "stone" matrix at the bottom right of the additive wall (left figure), so that the stone above consists of all zeros?

b.  How do you choose the numbers in the matrix "stone" at the bottom right of the multiplicative wall (right figure), so that the left middle "19" stone appears unchanged as the keystone at the top?


**P14. Matrix equations.**

a. Determine $u, v$ in the matrix equation $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} -7 & -1 \\ 6 & 1 \end{pmatrix} = \begin{pmatrix} u & 3 \\ 1 & v \end{pmatrix}$.

a. Determine $x$ and $y$ in $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 2 & x \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} y & 3 \\ 10 & 5 \end{pmatrix}$.

a. Determine the four unkowns $u, v, x, y$ in the matrix equation $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} x & u \\ y & v \end{pmatrix} = \begin{pmatrix} u & 3 \\ 1 & v \end{pmatrix}$.


**P15. Two pairs of LS.**   Here are two pairs of linear systems.

$$\begin{array}{ll} \texttt{1x+2y = 5} & \texttt{1u+2v = -1} \\ \texttt{3x+4y = 6} & \texttt{3u+4v = 8} \end{array}$$

a. Write both pairs of systems of equations as *one* matrix equation.

b.  Use it to solve both pairs of equations at the same time, i.e. calculate the searched numbers $x, y$ and $u, v$ at one stroke. If necessary, don't forget a test w/o EIGENMATH.


**P16. LU decomposition I.**



We can split the number 6 e.g. into the factors 2 and 3 $(6 = 2 \cdot 3 = 1 \cdot 6)$.
Let's try to factor a given matrix into "advantageous" factors.
Q: Does this have any influence of the calculation of the solutions of the associated linear equations?

a. Determine all the unknows:

$$\begin{pmatrix} 6 & -2 \\ 12 & 8 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} * \begin{pmatrix} x & y \\ 0 & z \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 4 & 2 \end{pmatrix} * \begin{pmatrix} u & v \\ 0 & w \end{pmatrix}$$

$$\quad A \qquad\qquad L1 \quad\ U1 \qquad\quad L2 \quad\ U2$$

Check the equality with EIGENMATH.

b. What is special about the matrices $L$ and $U$?  *Hint*: "**U**pper triangular" vs. "**L**ower triangular"

c. In a. we have found factor matrices $L1$ and $U1$ or $L2$ and $U2$, so that $A$ is "LU-factored", i.e. broken down into two factors $L$ and $U$. We see: this factorization is *not* unique.
Check with EIGENMATH whether the *commutative law* $a \cdot b = b \cdot a$ of multiplying numbers also applies for the matrix product: $L * U \overset{?}{=} U * L$.

d.   Determine the missing numbers $u$ and $v$ in the matrix factor $U$ so that an LU-decomposition of $A$ results:

$$
\begin{pmatrix} 2 & 1 & 3 \\ 4 & 1 & 7 \\ -6 & -2 & -12 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -1 & 1 \end{pmatrix} * \begin{pmatrix} 2 & u & 3 \\ 0 & -1 & 1 \\ 0 & 0 & v \end{pmatrix}
$$
$$
\quad\quad A \quad\quad\quad\quad\quad L \quad\quad\quad\quad\quad U
$$

Check your answer with EIGENMATH.

▷    *Click here to invoke* EIGENMATH$^{online}$.

## P17. LU decomposition II.

$$
\begin{array}{rcl}
1x & = 1 \\
1y & = 1 \\
2x - y + z & = 1
\end{array}
\qquad
\begin{array}{rcl}
1u + 1v + 1w & = 1 \\
1v + 2w & = 1 \\
3w & = 0
\end{array}
$$

a. Calculate the solution $(x, y, z)$ and $(u, v, w)$ of the red/blue system by head.

b. Write the red LS as a matrix equation $L * Y = B$ and solve it with EIGENMATH.

c. Write the blue LS as a matrix equation $U * X = C$ and solve it with EIGENMATH.

d. Determine the unknown numbers $x, y, z$ of the green LS:

$$
\begin{array}{rcl}
1x + 1y + 1z & = 1 \\
1y + 2z & = 1 \\
2x + 1y + 3z & = 1
\end{array}
$$

e. Write the green LS as a matrix equation $A * X = B$ and solve it.

e. Argue:

– the solution $Y$ of $L * Y = B$ is equal to the right-hand side $C$ of $U * X = C$.

– the following applies: $L * U = A$.

f. Suppose you had known from the beginning ("a priori") that $L * U = A$.
Would you then have been able to easily solve the equation $A * X = B$ for $X$?
Justify your answer; recalculate $X$ according to this recipe.

g. Now that you know in retrospect ("a posteriori") that in P16.d we have $L * U = A$, solve the following magenta system of equations as economically as possible.

$$
\begin{array}{rcl}
2x + y + 3z & = -1 \\
4x + y + 7z & = 5 \\
-6x - 2y - 12z & = -2
\end{array}
$$

**P18.** *Recreation*: **Falk scheme.**

$$
\begin{array}{ccc|cc}
 & & \begin{matrix} 1 & 0 & 2 \\ & 2 & 1 \\ 3 & 1 & 5 \end{matrix} \\
\hline
\begin{matrix} 25 & 5 & 1 \\ 8 & 4 & 1 \\ 100 & 10 & 1 \end{matrix} & & \begin{matrix} 38 & & 70 \\ 19 & 5 & \\ 123 & & \end{matrix}
\end{array}
$$

Determine the missing numbers.

**P19. Linear substitution as matrix multiplication.**

$$
\begin{array}{ll}
2x \;-\; 3y \;+4z \;=\; 5 & \qquad x \;=\; 3u \;-\; v \\
4x \qquad\quad -\; z \;=-2 & \qquad y \;=\; 2u \;+5v \\
 & \qquad z \;=\; -u \;+2v
\end{array}
$$

a. Are there two numbers $u, v$ for the blue LS, such that the red LS can be solved with
the corresponding numbers for $x, y, z$? First consider or calculate without EIGENMATH.
b. Now study the following EIGENMATH script. We write the red LS as $A * X = Y$ and
the blue one as $B * Z = X$. In symbols:

$$
Z \xmapsto{\;B\;} X \xmapsto{\;A\;} Y
$$

Then we look at the *linear replacement* $A * (B * Z) = Y$.
– What is the value of left hand side $A * (B * Z)$ of the new linear system? Advantage?
– The solution $U$ of the substituted LS $A * U = Y$ is $U = (-16/47, -19/47)$.
Watch, how EIGENMATH helps in the solution process:

—————————  EIGENMATH  —————————

```
A=((2,-3,4),(4,0,-1))    -- red LS
X=(x,y,z)                -- red unknowns
dot(A,X)                 -- A*X

B=((3,-1),(2,5),(-1,2))  -- blue LS
Z=(u,v)                  -- blue unknowns
dot(B,Z)                 -- B*Z

dot(A,B,Z)               -- reduced LS for U=(u,v)
U=(-16/47, -19/47)       -- solution for U

dot(B,U)                 -- values for X
dot(A,B,U)               -- solves red LG
Y=(5,-2)

dot(A,B,U)==Y            -- check
```

▷   *Click here to run the script.* You should see the following output:

```
Run      Stop                                                    Clear
```

```
A=((2,-3,4),(4,0,-1))    -- red LS
X=(x,y,z)                -- red unknowns
dot(A,X)                 -- A*X
```

$$\begin{bmatrix} 2x - 3y + 4z \\ 4x - z \end{bmatrix}$$

```
B=((3,-1),(2,5),(-1,2))  -- blue LS
Z=(u,v)                  -- blue unknowns
dot(B,Z)                 -- B*Z
```

$$\begin{bmatrix} 3u - v \\ 2u + 5v \\ -u + 2v \end{bmatrix}$$

```
dot(A,B,Z)               -- reduced LS for U=(u,v)
U=(-16/47, -19/47)       -- solution for U
```

$$\begin{bmatrix} -4u - 9v \\ 13u - 6v \end{bmatrix}$$

```
dot(B,U)                 -- values for X
```

$$\begin{bmatrix} -\dfrac{29}{47} \\ -\dfrac{127}{47} \\ -\dfrac{22}{47} \end{bmatrix}$$

```
dot(A,B,U)               -- solves red LG
Y=(5,-2)
```

$$\begin{bmatrix} 5 \\ -2 \end{bmatrix}$$

```
dot(A,B,U)==Y            -- check
```
1

**P20. Magic squares.**   Here you can see "magic" 3-by-3 squares:

$$\begin{bmatrix} 2 & 6 & 7 \\ 10 & 5 & 0 \\ 3 & 4 & 8 \end{bmatrix}, \quad \begin{bmatrix} 4 & 9 & 2 \\ 3 & 5 & 7 \\ 8 & 1 & 6 \end{bmatrix}, \quad \begin{bmatrix} -1 & 4 & -3 \\ -2 & 0 & 2 \\ 3 & -4 & 1 \end{bmatrix}, \quad \dots$$
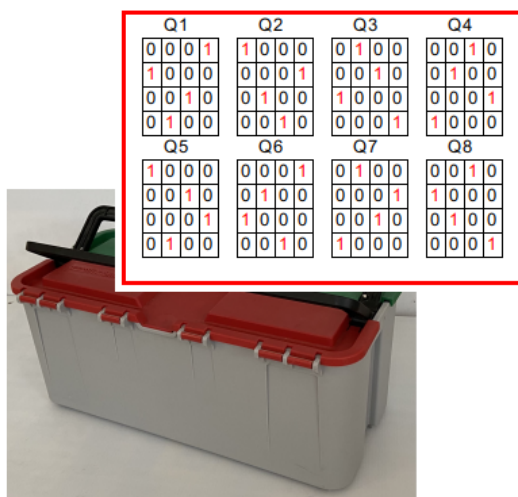
a. Reconstruct the first 3x3 magic square only knowing its first column:
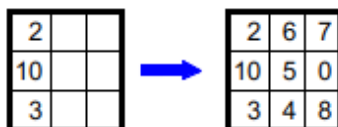


b. Give a few more 3x3 magic squares. What is "magical" here?
c. Can you make new magic squares from them? Explain. Give examples.
d. Here is a toolkit for putting together magic 4x4 matrices; it contains the "basic" magic 4x4 squares:

↖Dürer's magic □

e. Check that $Q1$ through $Q8$ are magical.

f. According to which systematic pattern were $Q1$ to $Q8$ constructed?

g. Try to calculate with these magic squares: What is $2*Q1 + 5*Q3 - 2*Q5$? Is this square magical again?

h. Try to "mix" Dürer's magic square in the copperplate "Melancholia" (1514) from the eight listed magic squares $Q1$ to $Q8$ of the toolkit.

• We now **implement the building set ('kit')** for magic squares in Eigenmath.
Here is a start script. Complete it with the other magic squares $Q[3]$ until $Q[8]$; do *not forget to adapt* the container $Q$ in (1)!



▷   *Click here to run the script.*

**Remark.** The container matrix in (1) is an example of a so-called *tensor*: both entries of $Q$ are itself matrices! Think of this list of matrices $Q$ as 3-dimensional layered: matrix

$Q[1]$ is then the yellow matrix in front of the matrix cube  .
EIGENMATH's output shows the matrices stacked second under the first.

• We next **program the magic check** as an EIGENMATH- function `isMagic`, which should make work easier.

——————————  EIGENMATH  ——————————

```
Q = zero(2,4,4) -- (1) magic squares container
Q[1]= ((0,0,0,1), (1,0,0,0), (0,0,1,0),(0,1,0,0))
Q[2]= ((1,0,0,0), (0,0,0,1), (0,1,0,0),(0,0,1,0))
-- (2) here comes Q[3] .. Q[8]

isMagic(Q) = do(
             n = dim(Q,1),
             s = sum(j,1,n, Q[1,j]), -- (3) the magic sum
             print(s), -- (4) print this sum
             for(i,2,n, check( sum(j,1,n, Q[i,j])==s )), --(5)
             for(j,1,n, check( sum(i,1,n, Q[i,j])==s )), --(6)
             check( sum(i,1,n, Q[i,i]) == s ),           --(7)
             check( sum(i,1,n, Q[i,n-i+1]) == s ),       --(8)
             "is magic")                                 --(9)

isMagic(Q[1])                      --(10)
isMagic(Q[2])

Qok=((2,6,7),(10,5,0),(3,4,8))  --(11)
Qok
isMagic(Qok)

Qbad=((2,6,7),(10,5,0),(3,4,7)) --(12)
Qbad
isMagic(Qbad)
```

▷   *Click here to run the script.*

**Comment.** Identifier $n$ saves the typ of the magic square. Line (3) save the sum of all the entries of the 1st row in identifier $s$. This value should be the same for all row-sums, column-sums and of both diagonal-sums: that's makes it 'magic'. Comment line (4) out, if you do not want to see this value. The for-loop in (5) `check`s, whether all row-sums (without the first one) have the same value $s$. The same check does line (6) for all column-sums. Line (7) checks, whether the elements of the NW-SE-diagonal add up to the magic-sum $s$. The same does line (8) for the NE-SW-diagonal. (9) gives back the

string "is magic", if all tests passed. Otherwise the function stops, because one check did not pass. The lines (10), (11) and (12) show some example runs.

k. Check, whether all $Q1$ til $Q8$ are magic using a `for`-loop with `isMagic` calls.

$\ell$. Derive an equation to calculate DÜRER's magic square from integer multiples of $Qi$.

m. Formulate this equation as a matrix equation $A * X = Duerer$.

n. Specify a basic kit for assembling 3x3 magic squares.

p. Can you build any magic 4x4 square from $Q1$ to $Q8$?

## P21. Matrix factory I - the HILBERT–matrix.

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}$$

Sometimes we want to produce matrices automatically.

This is the pattern-generated so-called 3-by-3 *Hilbert* matrix.

a. In your head: What is the 4x4 HILBERT matrix?

b. The 4x4-HILBERT matrix is created according to the following construction plan:



c. Here is a little script, that implements an EIGENMATH function `Hilbert`[20] to calculate such a matrix automatically for any dimension $n$:

---

[20]Please note: there is a build-in function `hilbert` with lowercase letter 'h'. EIGENMATH is case-sensitive, therefore we must use an uppercase 'H' for the name of our user-defined function. Btw, per convention *all build-in* EIGENMATH *functions start with a lowercase letter*, so we should avoid naming problems and associated runtime errors by *always using user-defined identifiers with first letter in* **U***ppercase*.

```
-- the Hilbert procedure

hilb3 = zero(3,3)
hilb3                                    --(1)

for(i,1,3,
    for(j,1,3,
            hilb3[i,j]=1/(i+j-1)) )   --(2)

hilb3


                -- the Hilbert function
Hilbert(n)= do(
            HILB=zero(n,n),          --(3)
            for(i,1,n,
              for(j,1,n, HILB[i,j]=1/(i+j-1))),--(4)
            HILB )                   --(5)

Hilbert(3)
```

$$h_{ilb3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$h_{ilb3} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

▷   *Click here to run the script.*

**Comment.** The script has two parts. The first part shows the implementation steps by example: define the container matrix `hilb3`, which is 'empty' at the beginning, see (1). Then each entry `hilb3[i,j]` of the table `hilb3` is filled according to the matrix-filling "generating" assignment rule `1/(i + j-1)`. Observe the important ',' at the end of line (4)![21]

d. A call `Hilbert(0)` or `Hilbert(1)` gives an error message. Why? What to do?

e. We can calculate individual values using a little helper function.

───────────  EIGENMATH  ───────────

```
Hilb(x,y) = 1/(x+y-1)
Hilb(3,3)
```

─────────────────────────────────

– What is the value of `Hilb(3,3)`?
– What is `Hilbert(6)[4,3]`? Calculate the same value using function `Hilb`.
– Discuss pro's and con's of both functions.

---
[21]Slogan: *"for" doesn't return a value like "do" does*!

**P22. Matrix factory II - the PASCAL–matrix.**

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{pmatrix}$$

This is the pattern-generated so-called $4 \times 4$-`Pascal` matrix.

a. Write up the 5-by-5-`Pascal` matrix in your head.

b. The `Pascal` matrices are created with the construction rule `binomial(i+j-2, j-1)`. Code a helper function `Pas3(i,j)` and a matrix generating function `Pascal(n)` according to the pattern of *P.21*. Make use of the build-in EIGENMATH function `binomial`.

**P23. Matrix factory III - RANDOM–matrix generators.**

Sometimes we are interested in random values for test reasons. Currently there is no build-in generator for random numbers $x_n$ in EIGENMATH. Therefore we program the so-called *Minimal Standard Generator* for EIGENMATH, a special case of an LCG (*Linear Congruential Generator*)[22] based on the recurrence relation $x_n = a \cdot x_{n-1} \; mod \; m$:

―――――――― EIGENMATH ――――――――

```
-- the Minimal Standard Generator MSG (recursive version)
m    = 2147483647  -- the modul
a    = 16807        -- the multiplier
seed = 1043618065  -- the starting value x1

gn(n) = test( n=1, seed,
              mod( a * gn(n-1), m) )
```

▷ *Click here to run the script.*

This recursive function can also be coded iterative[23]

―――――――― EIGENMATH ――――――――

```
-- the Minimal Standard Generator MSG (iterativ version)
m    = 2147483647
a    = 16807
seed = 1043618065

xn(n) = do( Ntmp = seed,
            for(i,2,n, Ntmp = mod(a * Ntmp, m)),
            Ntmp)

xn(9)     --  result: 461468216
```

▷ *Click here to run the script.*

―――――――――――――――――――

[22]see [2, p. 112 ff]

[23]I thank George WEIGT for telling me this version.

**Remark.** The *recursive* version `gn` of the MSG is very near to the mathematical formulation as recurrence relation, whereas the *iterative* version `xn` is more qualified for the effective calculation of long sequences of (pseudo) random numbers and should be preferred.

a. Call the 249th random number via `xn(249)` and through the call `gn(249)`.
Call also `xn(300)` and `gn(300)`. Compare.
b. Display the sequence of the first 5 MSG-random numbers via  `for(i,1,5, print(xn(i)))`.
Do the same call again. What do you observe? - Try another value for *seed*. Redo.
c. Test the following function `Random(.)`  to output an MSG-random matrix:

| Run | Stop | | Clear | Draw | Simplify | Float | Derivative |
|---|---|---|---|---|---|---|---|

```
m    = 2147483647
a    = 16807
seed = 1043618065

xn(n)=test(n=1, seed, mod( a *xn(n - 1), m))

Random(n)= do(
          RND=zero(n,n),
          for(i,1,n,
          for(j,1,n, RND[i,j] = xn(i+j) )),
          RND)

Random(3)
```

$$\begin{bmatrix} 1589873406 & 2010798668 & 543060237 \\ 2010798668 & 543060237 & 407903509 \\ 543060237 & 407903509 & 866474539 \end{bmatrix}$$

▷    *Click here to run the script.*

○    This user-defined function `Random(.)`-matrix has a weakness. Why? What to do?

d. In statistics we often need random values $u_n$, which fall into the *unit* intervall $[0; 1]$, i.e. we want to have $0 \leq u_n \leq 1$. Define $u_n$ and let `Random(.)`  produce a unit random matrix.

e. Define an LCG in EIGENMATH, which is based on the recurrence relation

$$y_n = (a \cdot y_{n-1} + c) \quad mod \quad m$$

where *mod* is EIGENMATH's modulo operator, which returns the remainder of $a$ divided by $b^{24}$. Pick the *multiplier* $a = 3$, the *increment* $c = 5$, the *modulus* $m = 11$ and the *seed* (start-value of the recurrence) $x_o = 9$. Verify, that the first dozen values are *9 10 2 0 5 9 10 2 0 5 9 10*. That means: this LCG has period 5 – which is bad. Choose the set $(a, c, m, seed) = (7, 5, 18, 12)$. What is its period? These random numbers "looks a little bit more randomly distributed".

---

[24]Lexicon: $\dfrac{Math \qquad\quad : 12\ mod\ 5 = 2}{\text{EIGENMATH: } \texttt{mod(12,5)=2}}$

f*. The FIBONACCI random generator is based on the recurrence

$$z_n = (z_{i-1} + z_{i-2}) \quad mod \quad m$$

Define this random generator in EIGENMATH. Test it with the set $(m, y1, y2) = (19, 17, 13)$. If you want to experiment with another test set $(m, y1, y2)$, so try only integers $m \geq 2$ and $y1, y2 \in \{0, ..., m-1\}$ with $gcd(m, y1, y2) = 1$.

> Fibonacci generators are not very suitable as pseudo random number generators. For example, if you were to try to generate a random point cloud in a cube, all points would be on one level.

g*. *TI-59 congruence generator.* Here is the famous mixed congruential generator used in the *Ti-59* pocket calculator by Texas Instruments:

$$t_n = (24298 \cdot t_{n-1} + 99991) \quad mod \quad 199017$$

Test its usefulness in competition with the other LCG's.

**P24. Junction − a look ahead.** [25] An inner-city traffic junction P-Q-R-S is constructed according to the figure; the figures given are estimates of the number of cars expected per hour in the network of one-way streets:
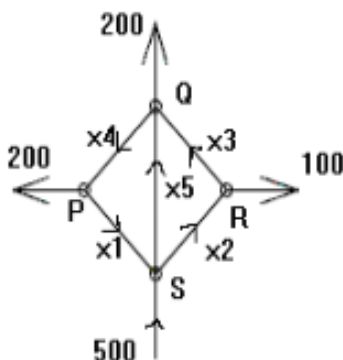


x1 ... x5 denotes the number of vehicles on the respective route section.
a. Under which conditions ("equations") is a traffic flow free of stowage ?
b. The description of the given situation by equations is a 'modeling' of the real situation. Describe the model and the assumptions on which your mathematical approach is based.
c. Write down the equations found in b. clearly and compactly with matrices.
d. Solve the problem, i.e. calculate the unknown number of vehicles.

---

[25]See GLASER et. al.: "SIGMA. Linear Algebra/Analytische Geometrie." Stuttgart: Klett 1987. p.106. Problems of this sort are theme of the next script in this series , Part 2.

# References

[1] ARTMANN, B. & TŌRNER, G. (1980): *Lineare Algebra. Grund- und Leistungskurs.* Goettingen: Vandenhoeck & Ruprecht.

[2] FLAMIG, B. (1995): *Practical Algorithms in C++.* New York: John Wiley.

[3] LINDNER, W. (2003): "CAS-supported Multiple Representations in Elementary Linear Algebra - The Case of the Gaussian Algorithm." In: *ZDM* Vol. 35 (2), S. 36 - 42.

[4] TALL, D. & VINNER, S. (1991): "Concept Images and Concept Definition in Mathematics with Particular reference to Limits and Continuity." In: *NCTM Educational Studies in Mathematics*, no. 12, p. 49-63.

[5] WEIGT, G. (2020): EIGENMATH *online Demo.*
url: `https://georgeweigt.github.io/eigenmath-demo.html`

[6] WEIGT, G. (2020): EIGENMATH *Manual.*
url: `https://georgeweigt.github.io/eigenmath.pdf`

⋈

Links checked 12.12.2020, wL

Dr. Wolfgang Lindner
Leichlingen, Germany
dr.w.g.Lindner@gmail.com
2020