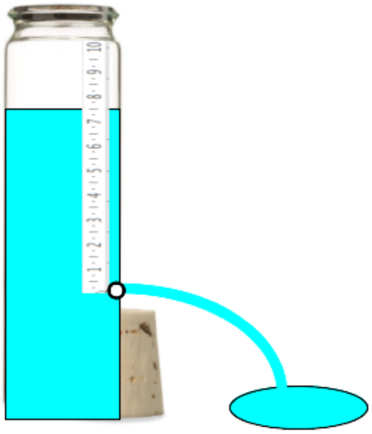
Exploring Math Σ *math* with EIGENMATH

Linear Algebra *Interactive*! with Eigenmath

Part 2

Regular Linear Systems • GAUSS Algorithm • RREF



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

2020

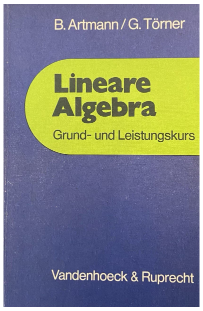
Contents

Preface

This is part 2 of a series of interactive learning parcours on Elementary Linear Algebra. It is about the Gauss-Jordan algorithm, with is done using CAS Eigenmath and explores the solution of regular linear systems.
These booklet 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. This learning environment was originally developed using notebooks compiled with CAS MuPAD 3.1, its free predecessor CAS MuPAD^{Light} 2.5.3 and in a very early status using the CAS Derive 3.0 with accompanying learning materials in PDF mini booklets.
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, there is the option to install the app Eigenmath free of charge and run the scripts by *mark–copy–paste* into the Eigenmath window.
I owe many suggestions on the subject to my doctoral supervisor Prof. Günter Törner and the book[2], that he wrote together with his doctoral supervisor Prof. Benno Artmann. The didactical obligation of the book was Gilbert Strang's motto

> "*explain rather than deduce.*"

to which I also feel obliged. The photo was taken at the beginning of the 2000's in the Mathematical Institute of the University of Göttingen.


B. Artmann / G. Törner
Lineare Algebra
Grund- und Leistungskurs
Vandenhoeck & Ruprecht

$bA^{wL}\, gT$

Any feedback from the user is very welcome.

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

Wolfgang Lindner
Leichlingen, Germany
December 2020

[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.

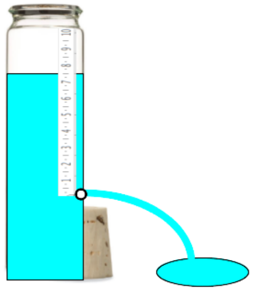
3 The 2^{nd} model problem

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 problem leads to the investigation of linear solution methods for systems of linear equations ("LS" for short).

3.1 Water towers - the reservoir problem

Cities store water in towers and from there the municipal utilities distribute the water to households and industrial plants via pipe systems. If no pumps are used, this physical method is called a "gravity system". Here is the model of a water tower:

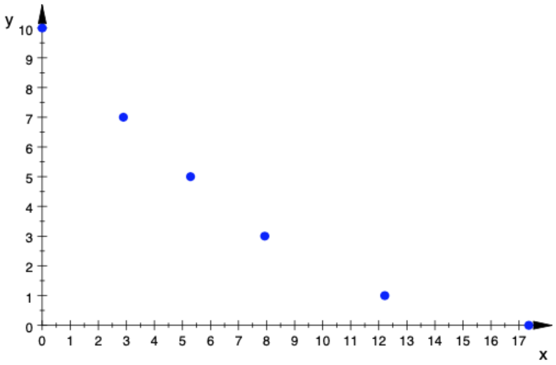


Q1: *Experiment*: get a plastic bottle with an approximately constant cross-sectional area analogous to the photo. Put a "0" mark a few centimeters above the bottom of the bottle and drill a small hole at the level of this zero mark "0".

- Mark the height in 1 cm steps from the 0 mark to the height of 10 cm. Hold your finger on the hole and fill the bottle with colored water up to the 10 cm mark.
- Run a stopwatch as soon as you take your finger off the hole.
- Measure the time it takes for the water surface to reach the 8 cm, 6 cm and 4 cm hole.
- Repeat the experiment several times and average your measured values.
- Enter your measured values in a table; here is a sample:

Height in cm	10	7	5	3	1	0
Time in sec	0	2.8	5.3	7.9	12.2	17.3

We draw the measured values e.g. on millimeter paper:



Q2: *Research questions*:

- Is it possible to "calculate" the time required to reach a certain height beforehand?
- Are all measured values on a function graph, i.e. models a law ("formula") this experiment?
- How could such a formula be obtained?[2]
- Can the measuring points - that would be the simplest case - e.g. lie on a straight line?[3]
- Repeat the experiment: estimate and measure the time it takes to reach the middle of the scale and the zero level.
  How accurate was your prediction?
  Find more questions for yourself and investigate them.
- The problem at hand was explored for the first time by E. Torricelli (1608-1647).
  Get a short biography and a photo of Torricelli (e.g. from the Internet)?
  Inquire about Torricelli's experiments and his findings.

  – What influence does the size or shape of the borehole have? – What influence does the composition of the liquid have?

  – What influence does the cross-sectional area of the bottle have?

  – What influence does the air pressure have? ...

Strategy considerations and research questions.

This problem represents typical questions that mathematicians often ask:

1: Can I describe the problem by an equation(s)? (*Modeling question*)
2: Are there systematic solution methods for the problem? (*Algorithm development*)

[2] By finding a function that goes through the points.

[3] No, because the water does not run out of the bottle constantly but slows down towards the end.

3.2 Partial solution of Q1 and Q2

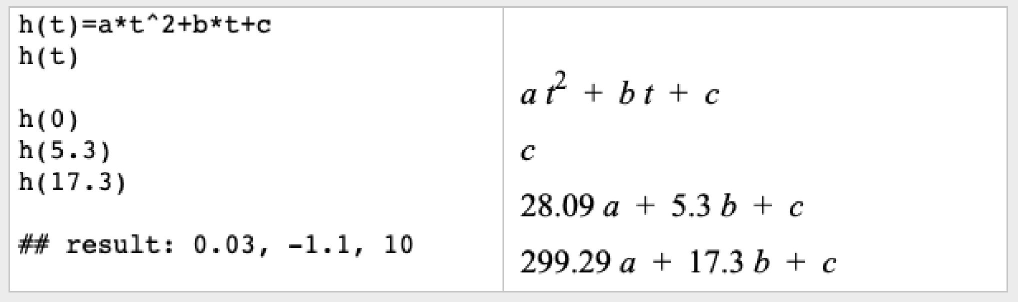
Here are the experimental data from a measurement (you can use your own data here):

Height in cm	10	7	5	3	. 1	0
Time in sec	0	2.8	5.3	7.9	12.2	17.3

a. TORRICELLI's considerations led to a quadratic relationship between the measured variables time t and height h of the water level above zero level:

$$h = a \cdot t^2 + b \cdot t + c$$

b. Calculate the parameter a, b, c using pairs from the table.[4] *Hint*: EIGENMATH:



```
h(t)=a*t^2+b*t+c
h(t)

h(0)
h(5.3)
h(17.3)

## result: 0.03, -1.1, 10
```

$$a\,t^2 + b\,t + c$$

$$c$$

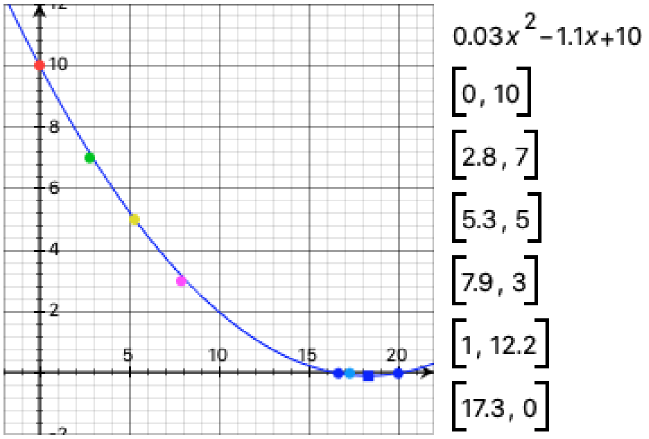
$$28.09\,a + 5.3\,b + c$$

$$299.29\,a + 17.3\,b + c$$

c. Which predictions result from the formula for reaching the middle of the scale or the zero mark?

d. Draw the graph of the function $h = f(t)$, enter the measuring points and also mark the values calculated with $f(t)$. Compare.

Hint: For the drawing[5] we use the lexicon $\begin{array}{c}\text{text: t h}\\ \text{graph: x y}\end{array}$.



e. Repeat b. and d. using the 2nd, 3rd and 4th data points. What do you observe?
Hint: Let help you e.g. using a Wolfram widget[6].

[4] Taking e.g. the 1st, the 3rd and the last data pair we get $a \approx 0.03$, $b \approx -1.1$ and $c = 17.3$.

[5] The plot was done with PocketCAS for iMac with the plot setting `//!x=-2..22,y=-2..12`.

[6] `https://www.wolframalpha.com/widgets/view.jsp?id=f7439c2089333993290cb74f148ddced`

4 Solving Linear Systems – The Gauss Algorithm

We ask: which method or which algorithm did the software e.g. in 2.2.e use to calculate the solution? To shed light upon this black box and to construct a systematic and semi- or fully automated solution for our model problem, we first consider a few simple special cases.

4.1 Solving a matrix equation – the row image

We want to study the solution process of a linear system using matrices. Therefore we consider for the moment a 2-by-2 linear system and look at it from different perspectives.

$$\begin{aligned} 1x + 2y &= 3 \\ 4x + 5y &= 6 \end{aligned} \qquad \text{– written as one matrix equation} \rightarrow \qquad \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

First we solve the system by multiplying simultaneously both sides of the equation with transformation matrices in order to guarantee that *the solution set of the system remains the same.*

1. scene:

linear equations to solve:

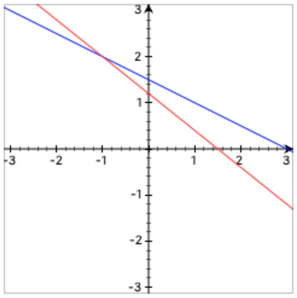
$$\begin{aligned} 1x + 2y &= 3 \\ 4x + 5y &= 6 \end{aligned}$$

the matrix equation:

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

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

visualization as *row image*:



2. scene:

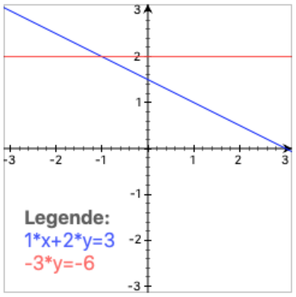
- Hereinafter $R1$ denotes the 1st row and $R2$ the 2nd row of the actual matrix. E.g. for matrix $A = \left[\begin{smallmatrix} 1 & 2 \\ 4 & 5 \end{smallmatrix}\right]$ we have $R1 = [1\ 2]$ und $R2 = [4\ 5]$ and the notation "$-4*R1+R2 \rightarrow R2$" means the instruction "add the -4 multiple of row1 to row2 and put it into the matrix as new 2nd row". This is called a "*linear combination* of row1 and row2". For example, $-4 * R1 + R2 \rightarrow R2$ means: please do the following calculation:
  $LHS = -4 * R1 + R2 = -4 * [1\ 2] + 1 * [4\ 5] = [-4\ \ -8] + [4\ 5] = [0\ \ -3] = R_{2new}$
  $RHS = -4 * R1 + R2 = -4 * [3] + 1 * [6] = [-12] + [6] = [-6] = R_{2new}$
- Multiply the matrix equation from the left with $\left[\begin{smallmatrix} 1 & 0 \\ -4 & 1 \end{smallmatrix}\right]$, i.e. $-4 * R1 + R2 \rightarrow R2$:

$$\begin{aligned}1x+2y &= 3\\ 0x-3y &= -6\end{aligned}$$

$$\begin{bmatrix}1&0\\-4&1\end{bmatrix}*\begin{bmatrix}1&2\\4&5\end{bmatrix}*\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}-4&0\\0&1\end{bmatrix}*\begin{bmatrix}3\\6\end{bmatrix}$$

$$-4*R1+R2\to R2$$

$$\begin{bmatrix}1&2\\0&-3\end{bmatrix}*\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}3\\-6\end{bmatrix}$$



3. scene:

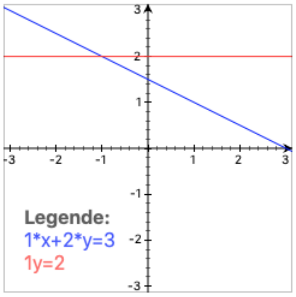
- Multiply the new matrix equation from the left with $\left[\begin{smallmatrix}1&0\\0&-1/3\end{smallmatrix}\right], i.e. \frac{-1}{3}*R2\to R2$:

$$\begin{aligned}1x+2y &= 3\\ 0x+1y &= 2\end{aligned}$$

$$\begin{bmatrix}1&0\\0&-1/3\end{bmatrix}*\begin{bmatrix}1&2\\0&-3\end{bmatrix}*\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}1&0\\0&-1/3\end{bmatrix}*\begin{bmatrix}3\\-6\end{bmatrix}$$

$$\frac{-1}{3}*R2\to R2$$

$$\begin{bmatrix}1&2\\0&1\end{bmatrix}*\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}3\\2\end{bmatrix}$$



4. scene:

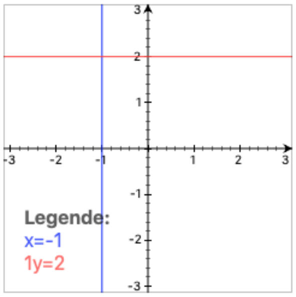
- Multiply the new matrix equation from the left with $\left[\begin{smallmatrix}1&-2\\0&1\end{smallmatrix}\right]$, i.e. $-2*R2+R1\to R1$:

$$\begin{aligned}1x+0y &= -1\\ 0x+1y &= 2\end{aligned}$$

$$\begin{bmatrix}1&-2\\0&1\end{bmatrix}*\begin{bmatrix}1&2\\0&1\end{bmatrix}*\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}1&-2\\0&1\end{bmatrix}*\begin{bmatrix}3\\2\end{bmatrix}$$

$$-2*R2+R1\to R1$$

$$\begin{bmatrix}1&0\\0&1\end{bmatrix}*\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}-1\\2\end{bmatrix}$$



- What do we see? The two straight lines, which belong to the original system of 2 equations (Figure in scene 1) have a *common intersection point*, the solution of the system. But this intersection is not to be read off clearly! During the transformation process this lines are stepwise alined with the coordinate axes. In the last figure the straight lines are parallel to the coordinate axes, where the intersection can plainly read off: $x=-1,\ y=2$, i.e. $\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right]=\left[\begin{smallmatrix}-1\\2\end{smallmatrix}\right]$.

4.2 Solving a matrix equation – the column image

In order to solve the matrix equation $A * X = B$ and taking the perspective of an *column image*, we write the linear system $A * X = B$ in form of a single matrix, which has as parts the system matrix A as well as the RHS B, i.e. we build the so-called *augmented matrix* $(A \vdots B)$, which allows a compact solution process and prepares for the use of EIGENMATH:

$$\begin{aligned} 1x + 2y &= 3 \\ 4x + 5y &= 6 \end{aligned} \qquad \text{– written as augmented matrix} \rightarrow \qquad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

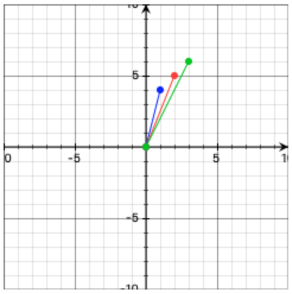
1. scene:

linear equations to solve: augmented system matrix: visualization *column image*:

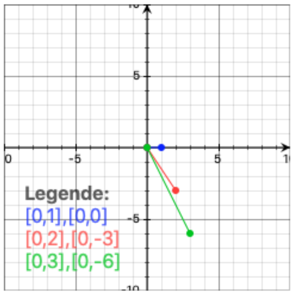
$$\begin{aligned} 1x + 2y &= 3 \\ 4x + 5y &= 6 \end{aligned}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

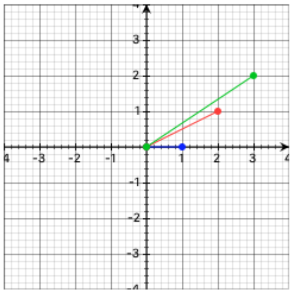
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$



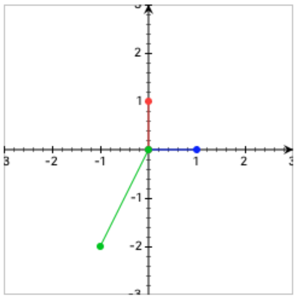
2. scene: $\begin{bmatrix} 1 & 0 \\ \mathbf{-4} & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \end{bmatrix}$ $\quad -4 * R1 + R2 \rightarrow R2$



3. scene: $\begin{bmatrix} 1 & 0 \\ 0 & \mathbf{-1/3} \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix}$ $\quad -1/3 * R2 \rightarrow R2$

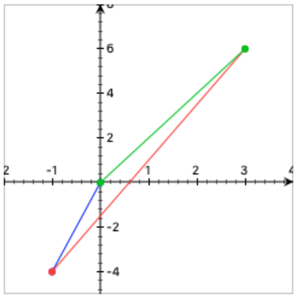


4. scene: $\begin{bmatrix}1 & \mathbf{-2}\\ 0 & 1\end{bmatrix} * \begin{bmatrix}1 & 2 & 3\\ 0 & 1 & 2\end{bmatrix} = \begin{bmatrix}1 & 0 & -1\\ 0 & 1 & 2\end{bmatrix}$ $\qquad -2{*}R2{+}R1 \to R1$



- What do we see? In the sequence of figures, which accompany the steps, the 1st column $\left[\begin{smallmatrix}1\\4\end{smallmatrix}\right]$ of $AB = \left[\begin{smallmatrix}1&2&3\\4&5&6\end{smallmatrix}\right]$ and their new instances is colored blue, the 2nd in red and the 3dr (the solution) in green. **The 4th step shows**, that the solution is reached, when the 1st and 2nd column vectors of AB are the transformed into the unit vectors e_1 and e_2 of the coordinate system: the *solution can then be read off as coordinates of the green solution vector*: $\left[\begin{smallmatrix}-1\\2\end{smallmatrix}\right]$.

5. scene: $\begin{bmatrix}1 & \mathbf{-2}\\ 0 & 1\end{bmatrix} .* \begin{bmatrix}1 & 0\\ 0 & \mathbf{-1/3}\end{bmatrix} * \begin{bmatrix}1 & 0\\ \mathbf{-4} & 1\end{bmatrix} * \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} = \begin{bmatrix}1 & 0 & -1\\ 0 & 1 & 2\end{bmatrix}$



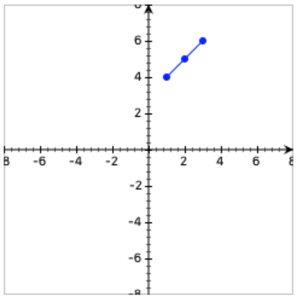
- Btw: The figure in scene 5 shows a second interpretation of the solution: $\left[\begin{smallmatrix}-1\\2\end{smallmatrix}\right]$ are the multiples to linear combine the RHS $B = \left[\begin{smallmatrix}3\\6\end{smallmatrix}\right]$ of the system using the two columns $A_1 = \left[\begin{smallmatrix}1\\4\end{smallmatrix}\right]$ and $A_2 = \left[\begin{smallmatrix}2\\5\end{smallmatrix}\right]$ of A as direction vectors, i.e. one has to go 1 step backwarts (minus sign in -1!) in direction of A_1 and from that point 2 steps in length and direction of A_2 to reach the RHS B, that is:

$$-1 * \begin{bmatrix}1\\4\end{bmatrix} + 2 * \begin{bmatrix}2\\5\end{bmatrix} = \begin{bmatrix}3\\6\end{bmatrix}$$

4.3 Solving a matrix equation – the matrix image

1: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

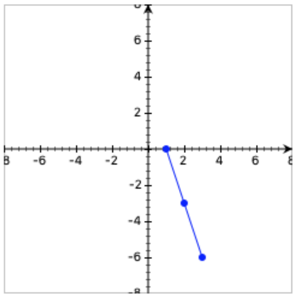
$-4 * R1 + R2 \rightarrow R2$



$\begin{bmatrix} 1 & 0 \\ \mathbf{-4} & 1 \end{bmatrix} \downarrow$ *eliminate*

2: $\begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \end{bmatrix}$

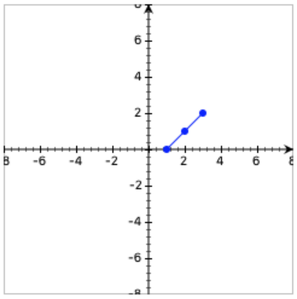
$-1/3 * R2 \rightarrow R2$



$\begin{bmatrix} 1 & 0 \\ 0 & \mathbf{-1/3} \end{bmatrix} \downarrow$ *scale*

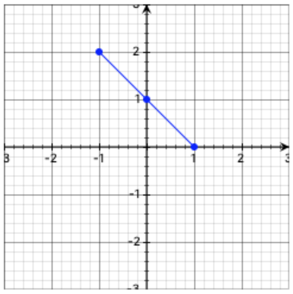
3: $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix}$

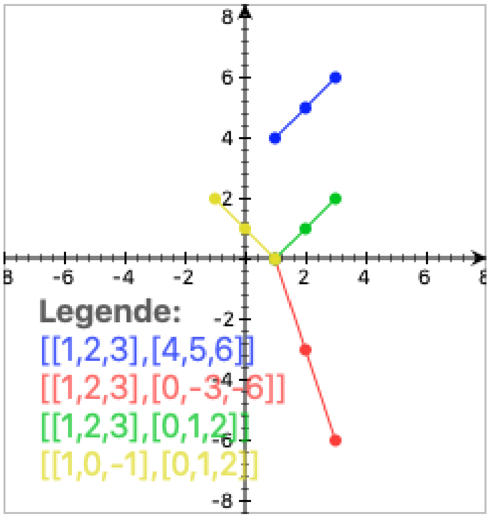
$-2 * R2 + R1 \rightarrow R1$



$\begin{bmatrix} 1 & \mathbf{-2} \\ 0 & 1 \end{bmatrix} \downarrow$ *eliminate*

4: $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \end{bmatrix}$





- ○ What do we see? In the sequence of figures, which accompany the steps, every column of the augmented system matrix AB is point of a polygon. During the solution process this polygon is transformed step by step into **the final position 4:**. The solution is reached, when the 1st and 2nd column vectors of AB are the unit vectors e_1 and e_2 of the coordinate system, i.e. when the polygon is locked at the points $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ – the *solution can then be read off as coordinates of the last point of the polygon (the yellow one)*: $\left[\begin{smallmatrix}-1\\2\end{smallmatrix}\right]$.

- ○ The end configuration of the matrix in step **4:** shows a typical pattern $\begin{smallmatrix}\square & Z\\ X & Y\end{smallmatrix}$ of the system matrix A, which gives an indication to control the solution process. Looking back at the solution steps we see, that we try to get a zero(s) at position X of matrix A, then to get a one(s) at position Y and then a zero(s) above the Y at Z. We will elaborate in detail on this Gauss scheme soon.

P25. (To the solution using transformation matrices) Here is a second example.
a. Fill in the missing matrices [...], which give the correct intermediate results with *left multiplication*. An example is given in step 1.
b. Observe the positioning and the values of the entries in the transformation matrices [...] carefully and try to recognize relationships.
c. Sketch a matrix image as illustration of the solution process.

1:

$$\begin{aligned} 3x + 6y &= 6 \\ 2x + 3y &= 3.5 \end{aligned} \qquad \begin{bmatrix} 3 & 6 & 6 \\ 2 & 3 & 3.5 \end{bmatrix}$$

$$\textit{scale} \quad \downarrow \quad \left[\begin{smallmatrix}1/3 & 0\\ 0 & 1\end{smallmatrix}\right] * \left[\begin{smallmatrix}3 & 6 & 6\\ 2 & 3 & 3.5\end{smallmatrix}\right] = \left[\begin{smallmatrix}1 & 2 & 2\\ 2 & 3 & 3.5\end{smallmatrix}\right] \quad \triangleleft\ 1/3 * R1 \to R1$$

2: .

$$\begin{aligned} x+2y &= 2 \\ 2x+3y &= 3.5 \end{aligned} \qquad\qquad \begin{bmatrix} 1 & 2 & 2 \\ 2 & 3 & 3.5 \end{bmatrix}$$

$$\textit{eliminate}\downarrow \quad \left[\qquad\right] * \left[\begin{smallmatrix} 1 & 2 & 2 \\ 2 & 3 & 3.5 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 1 & 2 & 2 \\ 0 & -1 & -0.5 \end{smallmatrix}\right] \quad \lhd -2*R1+R2 \to R2$$

3:

$$\begin{aligned} x+2y &= 2 \\ -1y &= -0.5 \end{aligned} \qquad\qquad \begin{bmatrix} 1 & 2 & 2 \\ 0 & -1 & -0.5 \end{bmatrix}$$

$$\textit{Skalieren} \quad \downarrow \quad \left[\qquad\right] * \left[\begin{smallmatrix} 1 & 2 & 2 \\ 0 & -1 & -0.5 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 1 & 2 & 2 \\ 0 & 1 & 0.5 \end{smallmatrix}\right] \quad \lhd -1*R2 \to R2$$

4:

$$\begin{aligned} x+2y &= 2 \\ 1y &= 0.5 \end{aligned} \qquad\qquad \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 0.5 \end{bmatrix}$$

$$\textit{Eliminieren} \quad \downarrow \quad \left[\qquad\right] * \left[\begin{smallmatrix} 1 & 2 & 2 \\ 0 & 1 & 0.5 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 1 & 0 & 1 \\ 0 & 1 & 0.5 \end{smallmatrix}\right] \quad \lhd -2*R2+R1 \to R1$$

5:

$$\begin{aligned} x \quad &= 1 \\ y &= 0.5 \end{aligned} \qquad\qquad \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0.5 \end{bmatrix}$$

Remark. In school we stopped half way at scene **3:** and then *go back for the solution.*

$$\begin{aligned} x+2y &= 2 \\ -1y &= -0.5 \end{aligned} \qquad \therefore \qquad \begin{aligned} -1y = -0.5 \quad &\rightsquigarrow \quad y = 0.5 && (4.1) \\ &\rightsquigarrow \quad x + 2 \cdot 0.5 = 2 && (4.2) \\ &\rightsquigarrow \quad x = 2-1 && (4.3) \\ &\to \quad x = 1 && (4.4) \end{aligned}$$

Equation (2.1) is equivalent to scene **4:** and (2.2) until (2.4) to scene **5:**. We will (semi)automate this ”Go back” - method later with Eigenmath as well. A first hint:

Eigenmath

```
-- BACK SUBSTITUTION                   works for 2x2 matrices
U = ((1,2,2),(0,-1,-0.5))           -- the intermediate matrix of scene 3:
y =  U[2,3] / U[2,2]                -- see (2.1)
y
x = (U[1,3] - U[1,2]*y) / U[2,2]    -- see (2.2) til (2.4)
x
```

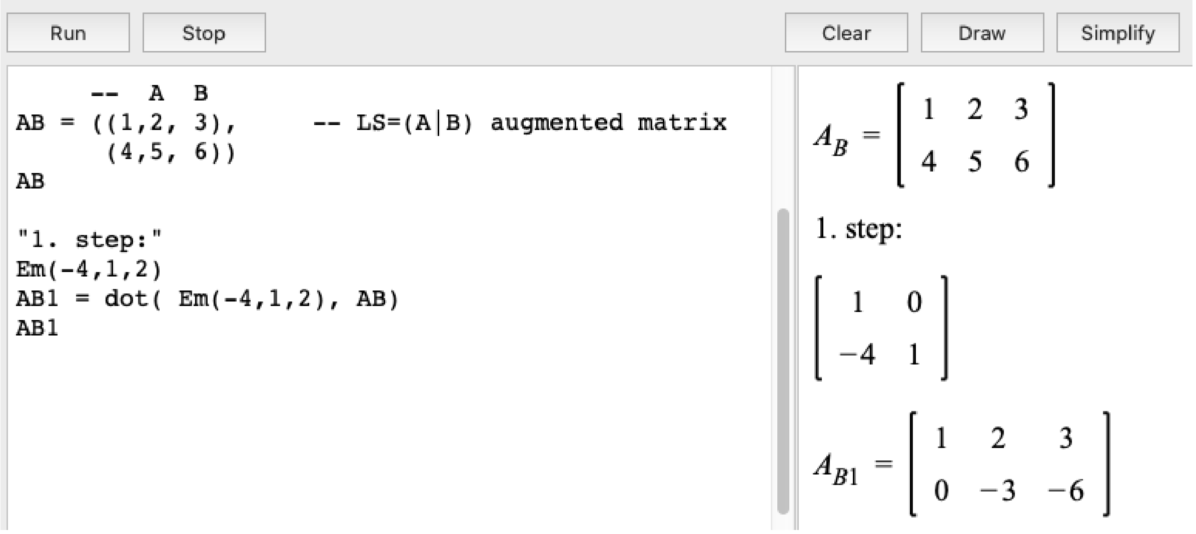
Try it: ▷ *Click here to run the script.*

4.4 Solving a matrix equation – use EIGENMATH!

We now solve the same linear system using EIGENMATH. We use so-called "**El**ementary **m**atrices" `Em(..)`, which we define resp. construct in the next definition. Here we study them in action. Look how `Em(x,i,j)` reflect the symbolic process description

$$x * Ri + Rj \to Rj$$

1: $-4 * R1 + R2 \to R2$ is encoded trough `Em(-4,1,2)`, i.e the LHS of "→":



```
     --  A   B
AB = ((1,2, 3),       -- LS=(A|B) augmented matrix
       (4,5, 6))
AB

"1. step:"
Em(-4,1,2)
AB1 = dot( Em(-4,1,2), AB)
AB1
```

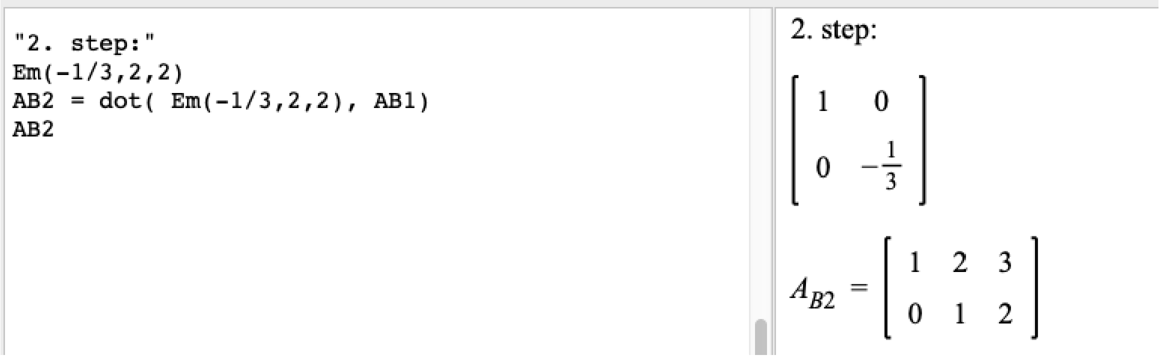
$$A_B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

1. step:

$$\begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix}$$

$$A_{B1} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \end{bmatrix}$$

2: $-1/3 * R2 \to R2$ is encoded trough `Em(-1/3,2,2)`:



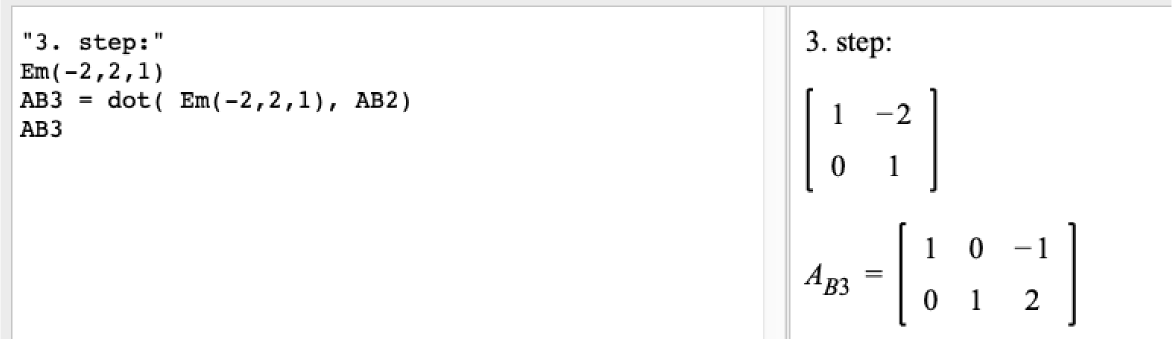
```
"2. step:"
Em(-1/3,2,2)
AB2 = dot( Em(-1/3,2,2), AB1)
AB2
```

2. step:

$$\begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{3} \end{bmatrix}$$

$$A_{B2} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix}$$

3: $-2 * R2 + R1 \to R1$ is encoded trough `Em(-2,2,1)`:



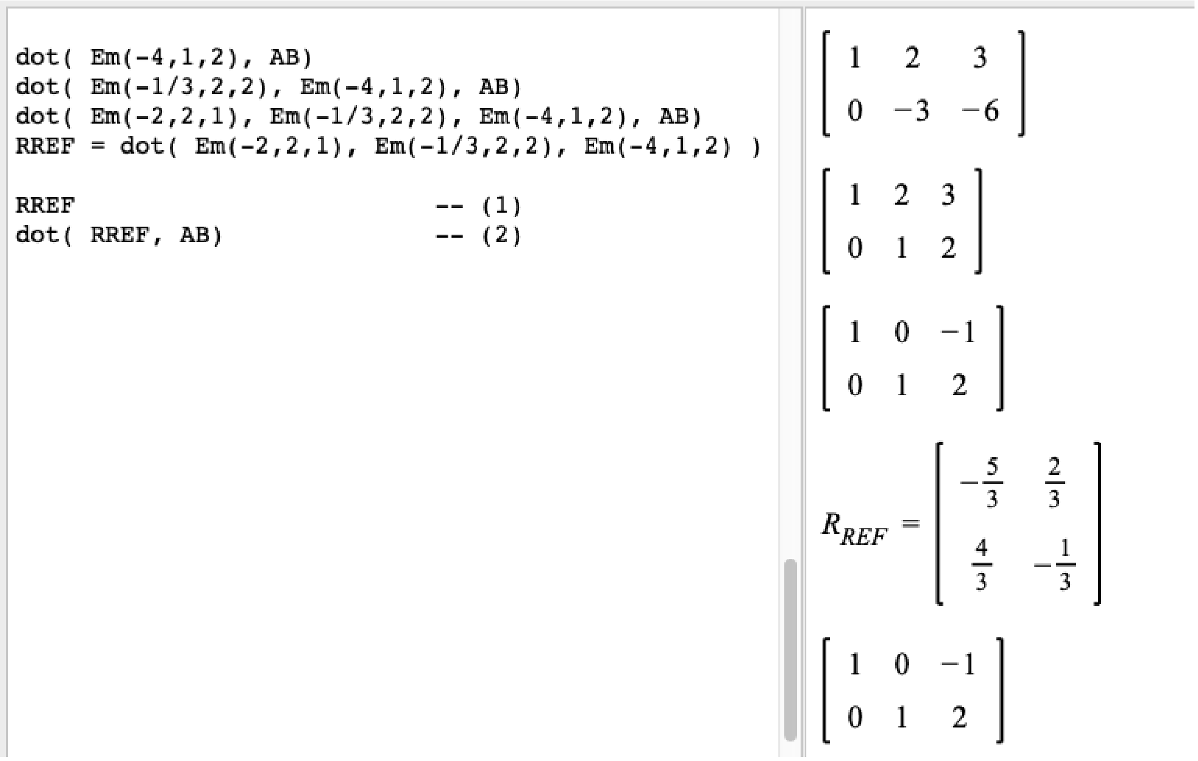
```
"3. step:"
Em(-2,2,1)
AB3 = dot( Em(-2,2,1), AB2)
AB3
```

3. step:

$$\begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}$$

$$A_{B3} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \end{bmatrix}$$

4: We can omit the intermediate steps using multiple matrix *dot*-products:



```
dot( Em(-4,1,2), AB)
dot( Em(-1/3,2,2), Em(-4,1,2), AB)
dot( Em(-2,2,1), Em(-1/3,2,2), Em(-4,1,2), AB)
RREF = dot( Em(-2,2,1), Em(-1/3,2,2), Em(-4,1,2) )

RREF                              -- (1)
dot( RREF, AB)                    -- (2)
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix}$$

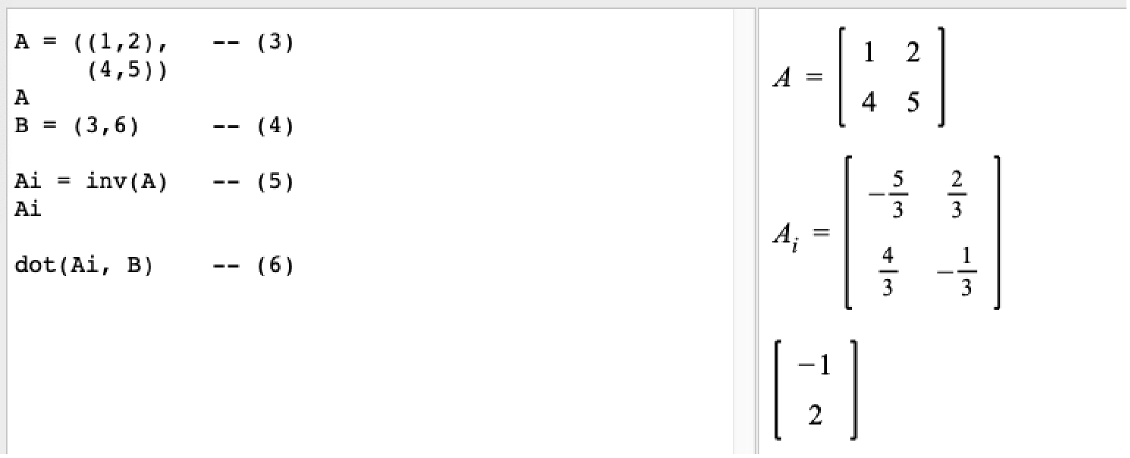
$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \end{bmatrix}$$

$$R_{REF} = \begin{bmatrix} -\frac{5}{3} & \frac{2}{3} \\ \frac{4}{3} & -\frac{1}{3} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \end{bmatrix}$$

Comment. The identifier *RREF* saves in (1) the whole elimination process, i.e. the product of the 3 elementary matrices `Em(k,i,j)`. Therefore we can unwind the process in (2) giving a compact short call with the solution of the linear system as answer. In the course of this *RREF* is an abbreviation for the process description "**r**ow **r**educed **e**chelon **f**orm".

A look ahead: looking at the left hand side (LHS) matrix A in (3) and the RHS matrix B in (4) of the linear system AB we call in (5) the so-called "`inv`"(erse) matrix of A. This inverse of A is decoded as the process result *RREF*, i.e `RREF(A)=inv(A)`! This is indeed a fact and no coincidence:



```
A = ((1,2),      -- (3)
     (4,5))
A
B = (3,6)        -- (4)

Ai = inv(A)      -- (5)
Ai

dot(Ai, B)       -- (6)
```

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

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

$$\begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

After considering some visual representations of the solution process of a LS, we now concentrate on the analysis of the underlying solution strategy and study the computational, *methodically strategic approach* in the solution process of a linear equation system.

- Is there an underlying guiding principle?
- What to do from one step to the next?

4.5 *Definition*: Elementary matrices in Eigenmath

If we want to describe the individual phases during the solution of a larger LS with the help of transformation matrices, the *wish for an automatic generation or easier input of the transformation matrices* arises, especially if the matrix dimension is greater than 2. This would make it possible to compute the solution of an LS by means of matrix multipliers *clearly and semi-automatically.*

4.5.1 *Definition*: identity matrices

Looking back at

$$\begin{pmatrix}3&6&6\\2&3&3.5\end{pmatrix} \rightsquigarrow \begin{pmatrix}1&0&1\\0&1&0.5\end{pmatrix}$$

$$\underbrace{\qquad}_{A}\quad\underbrace{\qquad}_{B}\qquad\qquad\underbrace{\qquad}_{E}\quad\underbrace{\qquad}_{R}$$

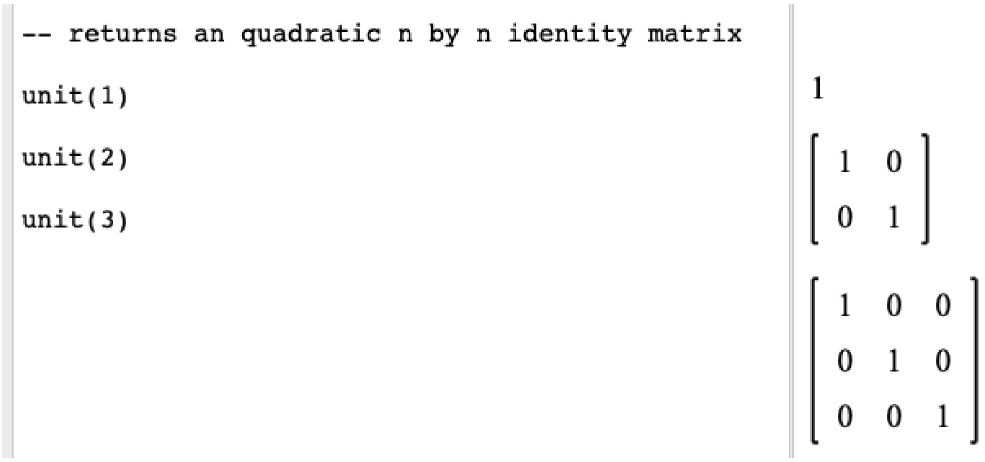
there was always a typically final status with the pattern `[E R]`, where $E = \left[\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right]$ is the so-called 2–*dimensional identity matrix* and $R = \left[\begin{smallmatrix}1\\0.5\end{smallmatrix}\right]$ is the result at the RHS of the linear equation. The following concept of an *identity matrix* enables us to provide a clear and compact description of insights and observations such as the above formulation of the solution process of a linear system.

Math: The *n-dimensional identity matrix* E_n for $n = 1, 2, 3, \ldots$ is defined through:

$$\begin{aligned}
E_1 &\overset{def}{=} (1)\\
E_2 &\overset{def}{=} \begin{pmatrix}1&0\\0&1\end{pmatrix}\\
E_3 &\overset{def}{=} \begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}\\
\ldots
\end{aligned}$$

Eigenmath:

$$E_n \overset{def}{=} \texttt{unit(n)}$$



```
-- returns an quadratic n by n identity matrix

unit(1)

unit(2)

unit(3)
```

$$1$$

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

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

Comment. 1. If there is no risk of confusion in reading the text, we write briefly E instead of E_n.

2. Unit matrices E are *quadratic* $n \times n$ matrices. They have all ones on the *main diagonal* and otherwise only contain zeros.

3. The identity matrix E plays the same role resp. $*$ as the number 1 in ordinary multiplication: E_n is the neutral element resp. $*$, i.e. for any square matrix M the following rule applies:

$$\mathbf{M} * \mathbf{E} = \mathbf{E} * \mathbf{M} = \mathbf{M}$$

The multiplication with the identity matrix E does not change M. We had seen that before in §2.1.

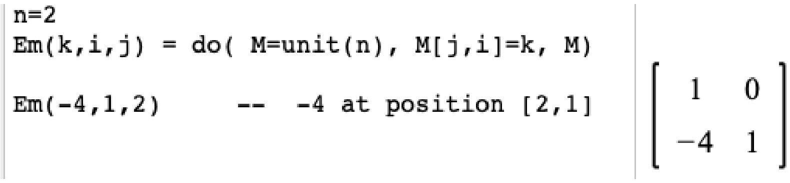
4.5.2 *Definition*: Elementary matrices in Eigenmath

If we change an identity matrix at exactly one position, we get a so-called *elementary matrix*.

Eigenmath

```
    Em(k,i,j) = do( M = unit(n), M[j,i] = k, M)
                  --  k*Ri + Rj -> Rj
```

Comment. The definition implements the transformation rule $k * Ri + Rj- > Rj$. First an n-by-n unit matrix M is elected, second the value k is placed in M at position $[j, i]$ and then the changed matrix M is returned.



```
n=2
Em(k,i,j) = do( M=unit(n), M[j,i]=k, M)

Em(-4,1,2)      --  -4 at position [2,1]
```

$$\begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix}$$

P26. (To the solution and back) We continue $P25$.

$$\begin{bmatrix}3&6&6\\2&3&3.5\end{bmatrix} \quad \overset{\textit{elim.}}{\underset{\mathbf{1}}{\longrightarrow}} \quad \overset{\textit{skal.}}{\underset{\mathbf{2}}{\longrightarrow}} \quad \overset{\textit{elim.}}{\underset{\mathbf{3}}{\longrightarrow}} \quad \begin{bmatrix}1&0&1\\0&1&0.5\end{bmatrix}$$

a. Realize the solution process $\left[\begin{smallmatrix}3&6&6\\2&3&3.5\end{smallmatrix}\right] \rightsquigarrow \left[\begin{smallmatrix}1&0&1\\0&1&0.5\end{smallmatrix}\right]$ using Elementary matrices $Em()$ in EIGENMATH. Think at the symbolic process description $x * Ri + Rj \to Rj$.
b. Define the $Em(?)$, which does the transformation in steps $1 - 2 - 3$.
c. Calculate the RREF like the one in 2.3.4 and solve the linear system in one stroke. Check your solution using $\begin{array}{ll}\textit{Math} & : A^{-1}\\ \text{EIGENMATH:} & \texttt{inv(A)}\end{array}$.
d. Sketch a suite of hand made matrix images to visualize the solution process.
f*. Give a sequence of transformations ($= Em$'s) to go back from the final solution status $\left[\begin{smallmatrix}1&0&1\\0&1&0.5\end{smallmatrix}\right]$ to the original start situation $\left[\begin{smallmatrix}3&6&6\\2&3&3.5\end{smallmatrix}\right]$:

$$\begin{bmatrix}3&6&6\\2&3&3.5\end{bmatrix} \quad \overset{?}{\underset{\mathbf{3}}{\longleftarrow}} \quad \overset{?}{\underset{\mathbf{2}}{\longleftarrow}} \quad \overset{?}{\underset{\mathbf{1}}{\longleftarrow}} \quad \begin{bmatrix}1&0&1\\0&1&0.5\end{bmatrix}$$

P27. (A mixing problem) Mix a 30% and a 50% alcoholic liquid to produce 2 liters of a liquid with a pure alcohol content of 45%.
a. Write the numbers of this 2×2 –LS as 2×3 – system matrix and form the first two columns of this system matrix into the matrix $\left[\begin{smallmatrix}1&0&\star\\0&1&\star\end{smallmatrix}\right]$ by suitable left multiplications with elementary matrices `Em`. Read off the solution from the LS. Do this exercise with/without EIGENMATH.
b. Describe the solution process algebraically by a product of transformation matrices `Em`. What is the matrix with the help of which the solution of the LS can be calculated *'in one putt'*?
c. Use EIGENMATH to calculate the inverse matrix of the coefficient matrix $A = \left[\begin{smallmatrix}1&1\\0.3&0.5\end{smallmatrix}\right]$. Compare with b.
d. Display the solution *process* graphically in the matrix image.

P28. a. Solve this 2×2– linear system using EIGENMATH and Em's:

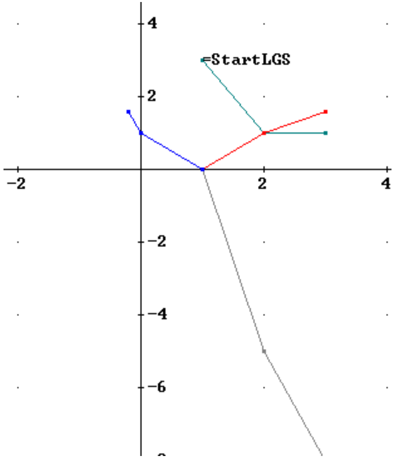
$$\begin{aligned} x - y &= 0\\ x + 2y &= 3 \end{aligned}$$

b. Alternatively use from the "half way status" (matrix U) the "Go back" (Gb) method.

EIGENMATH

```
-- BACK SUBSTITUTION                    works for 2x2 matrices
Gb(U) = do(x2 =  U[2,3] / U[2,2],
           x1 = (U[1,3] - U[1,2]*x2) / U[2,2],
           x  = (x1,x2) )
x
```

P29. (graphic representation of a solution process) Consider the following graphic representation of the solution process for a 2-by-2 linear system of equations



a. What is the original LS $A * X = B$ or $[AB]$, what is its solution?
b. Which transformation matrix G has transformed the LS "in one stroke" into the end position? Let EIGENMATH calculate the inverse matrix A^{-1} of the coefficient matrix A for comparison.
c. Reconstruct the above matrix image.

P30. (A historical problem from LIU HUI)[7] In ancient China there was a mathematics textbook in nine books, which was probably published as early as 180 BC by the mathematician SHANG CANG and used in the processing of LIU HUI from the year 263 AD. The eighth of these books deals with the systems of linear equations and develops a general solution method that was unknown to western mathematicians. The following system of equations had to be solved by a candidate for a higher civil service post in ancient China (in modern notation):

$$\begin{aligned} x + 3y + 2z + 8u + 5v &= 95 \\ 2x + 5y + 3z + 9u + 4v &= 112 \\ 3x + 5y + 7z + 6u + 4v &= 116 \\ 7x + 6y + 4z + 5u + 3v &= 128 \\ 9x + 7y + 3z + 2u + 5v &= 140 \end{aligned}$$

> This system of equations could in principle be solved using the addition method known from the intermediate level. The main difficulty in solving this task - and the reason why many very soon fail and give up (and thus could not become Chinese officials) - lies in the ability to organize the solution process appropriately.

a. Solve this linear system using Elementary matrices.
b. If the solution process in a. would be stopped at half time, one would have the following intermediate status:

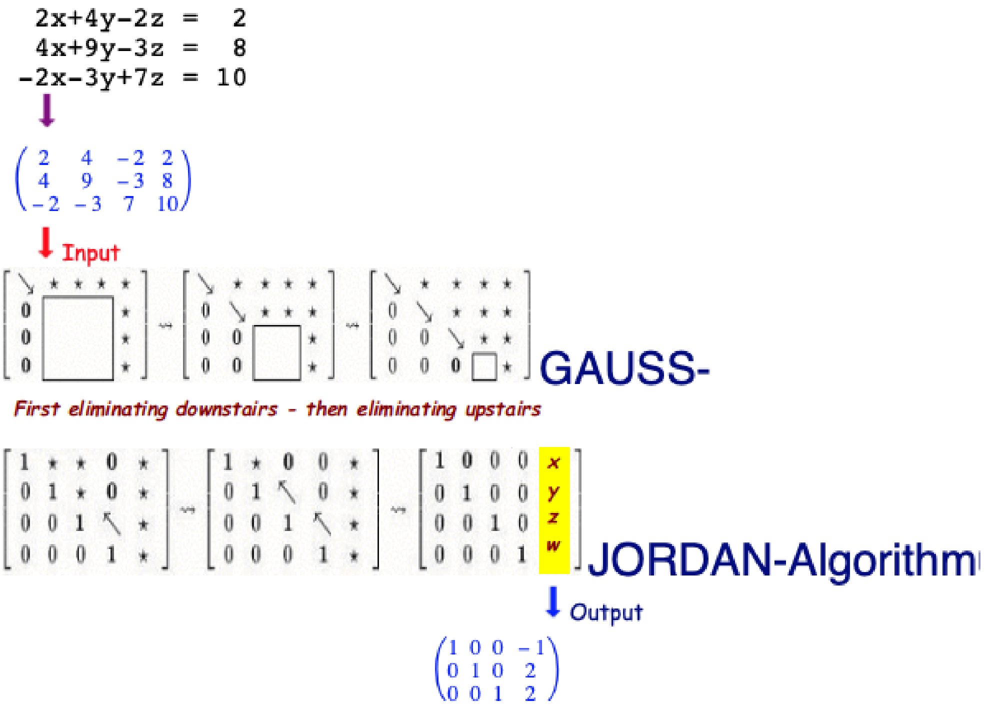
[7] See [2, p.15]

```
U=
    [1,  2,     3,   1,        4],
    [0,-10,   -13,  -4,      -18],
    [0,  0,-13/10,13/5,     21/5],
    [0,  0,     0,  -1,  -146/13],
    [0,  0,     0,   0,   -99/13]
```

♡ *Why is the original LS easily solvable in the form U?*

∘ Determine some of the unknowns x, y, z, u, v with a calculator. What could a solution strategy look like?

4.6 The Gauss-Jordan scheme



4.6.1 Tackling a 3-by-3 linear system with the Gauss-Jordan scheme

a. Reflect and ponder about the Gauss-Jordan scheme. What is the solution strategy?
b. Write the output matrix in 2.8 back as a classical system of linear equations.
c. What is the solution of the LS (do not calculate!)? Verify the solution on the LS.
d. Now solve the LS using elementary matrices with Eigenmath. Use the Gauss-Jordan-scheme as guiding principle. *Hint.* Before using $Em()$ you must set $n = 3$.
e. At the end of the Gauss part, i.e. at half time, we should have got the matrix status

$$\begin{pmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{pmatrix}$$

Calculate the unknowns by paper and pencil without using the Jordan part of the scheme and not using Eigenmath, too. Then check your calculation by a "Go back" $Gb()$ scheme.

P31. (Solution of the model problem Q1)
Calculate the values a, b, c of the parameters for the water tower problem in 2.2.e using Eigenmath. Use the Gauss-Jordan-scheme as guiding principle.
Hints: $h(2.8) = 7, h(5.3) = 5$ and $h(7.9) = 3$. This leads to the 3×3–linear system:

$$\begin{array}{rcl} 7.84a + 2.8b + c &=& 7 \\ 28.09a + 5.3b + cx + 2y &=& 5 \\ 62.41a + 7.9b + c &=& 3 \end{array}$$

○ Because the system matrix is of typ 3×3 we have to set $n = 3$ before using `Em()`.

4.7 *Definition*: Elimination matrices – Tuning the Em's

Thinking of the problem of Lui in *P*30, we have seen how laborious the single-step solution with elementary matrices can be for higher-dimensional matrices or linear systems. That is why we are now want to contruct an accelerated method. Therefore the following construction of the Gauss matrix alias *Elimination* matrix automatically 'cancels'[8] all matrix entries of a complete column.

4.7.1 Eigenmath *Definition* of the *Elimination* alias Gauss matrix

The following function `Gm(k,A)` ist called a **Gauss m***atrix* (short: '`Gm`') or a Gauss *transformation*. Sometimes ist is also called *Elimination matrix*.

Eigenmath

```
#######       Construction of GAUSS matrix 'Gm'
Gm(k,A) = do( n   = dim(A,1),
              Gmm = unit(n),
              for(i, k+1, n, Gmm[i,k] = - A[i,k]/A[k,k]),
              Gmm )
```

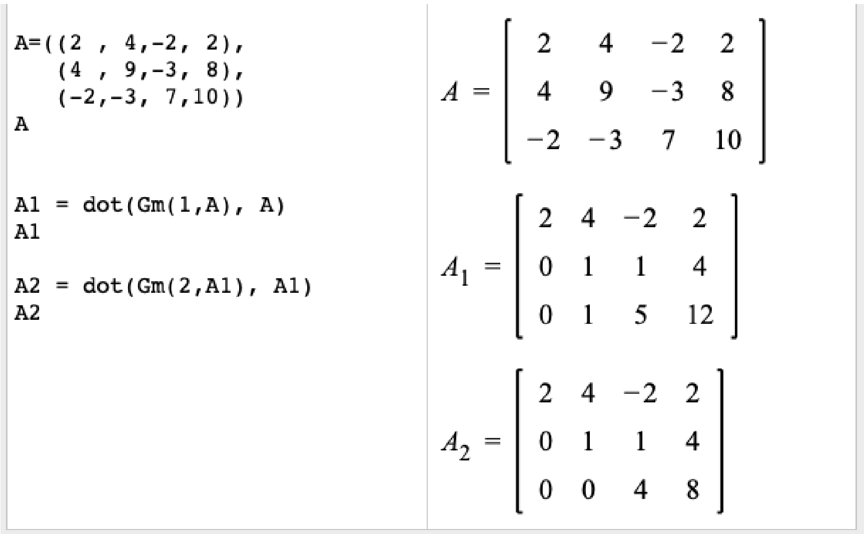
Comment. The function `Gm(k,A)` awaits two inputs:
– the number k of the column to be 'annulated' below row k and
– the (augmented) matrix A of the linear system.

The *do*-compound command first remembers the number of rows of A in identifier n, then keeps ready a quadratic $n \times n$ identity matrix `Gmm` to store the elimination steps and then executes the elimination steps inside a `for`-loop by changing the column positions $k + 1$ trough n of the identity matrix with the multipliers a_{ij}/a_{kk}.
Btw: The divisor a_{kk} on the main diagonal of A is called a *pivot element*. The value of the pivot is critical: it must be $\neq 0$.

[8] i.e. makes the corresponding matrix elements to 0.

4.7.2 *Example.*

The action of the Gauss matrices `Gm` is best understood by means of an example. Here we do again the Gauss scheme of $P31$.



```
A=((2 , 4,-2, 2),
   (4 , 9,-3, 8),
   (-2,-3, 7,10))
A

A1 = dot(Gm(1,A), A)
A1

A2 = dot(Gm(2,A1), A1)
A2
```

$$A = \begin{bmatrix} 2 & 4 & -2 & 2 \\ 4 & 9 & -3 & 8 \\ -2 & -3 & 7 & 10 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 1 & 5 & 12 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix}$$

The 1st call of `Gm` nullifies the complete 1st column of A below the 1st entry. The new status is saved in $A1$. Now one calls `Gm(2, A1)` to nullify the 2nd column of $A1$ below the 2nd entry. The new status is saved as $A2$. This terminates the Gauss part of the Gauss-Jordan scheme.

Remark. For the 'Jordan part' to finalize the solution process, we use the following '***G**o **b**ack*' scheme `Gb3`, which is adapted for 3×4-dimensional augmented system matrices:

Eigenmath

```
U=((2,4,-2,2),(0,1,1,4),(0,0,4,8))

Gb3 = do( x3 =  U[3,4] / U[3,3],
          x2 = (U[2,4] - U[2,3]*x3) / U[2,2],
          x1 = (U[1,4] - (U[1,2]*x2 + U[1,3]*x3)) / U[1,1],
          x  = (x1,x2,x3))

x  -- display x=(-1,2,2)
```

Try it: ▷ *Click here to run the script.*

Exercise. Determine the solution vector x using single-step elementary matrices.

P32. (Boosted solution of the Liu exercise)

This is the historical 5×5 linear system by Liu, we met in P30:

$$\begin{aligned} x + 3y + 2z + 8u + 5v &= 95 \\ 2x + 5y + 3z + 9u + 4v &= 112 \\ 3x + 5y + 7z + 6u + 4v &= 116 \\ 7x + 6y + 4z + 5u + 3v &= 128 \\ 9x + 7y + 3z + 2u + 5v &= 140 \end{aligned} \quad \rightsquigarrow \quad \begin{pmatrix} 1 & 3 & 2 & 8 & 5 & 95 \\ 2 & 5 & 3 & 9 & 4 & 112 \\ 3 & 5 & 7 & 6 & 4 & 116 \\ 7 & 6 & 4 & 5 & 3 & 128 \\ 9 & 7 & 3 & 2 & 5 & 140 \end{pmatrix}$$

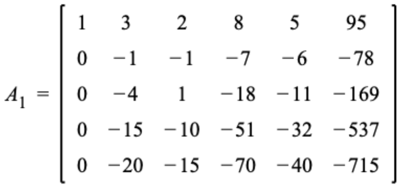
a. Do the Gauss part of the solution using Gauss matrices. Here is the start:

Eigenmath

```
Liu = ((1,3,2,8,5,  95),
       (2,5,3,9,4, 112),
       (3,5,7,6,4, 116),
       (7,6,4,5,3, 128),
       (9,7,3,2,5, 140))

A1 = dot(Gm(1,Liu), Liu)
A1
```

The output should be:



$$A_1 = \begin{bmatrix} 1 & 3 & 2 & 8 & 5 & 95 \\ 0 & -1 & -1 & -7 & -6 & -78 \\ 0 & -4 & 1 & -18 & -11 & -169 \\ 0 & -15 & -10 & -51 & -32 & -537 \\ 0 & -20 & -15 & -70 & -40 & -715 \end{bmatrix}$$

b. Do the Jordan part of the Gauss-Jordan procedure to finalize the solution process using single-step elementary matrices. Do not forget to set the global variable $n = 5$. In advance: How many Em's do you need?
c. If b. seems to slow-moving: write the '***Go back***' scheme for `Gb5` for dimension 5. Remember `Gb3`:

```
x3 =  U[3,4] / U[3,3]
x2 = (U[2,4] - U[2,3]*x3) / U[2,2]
x1 = (U[1,4] - (U[1,2]*x2 + U[1,3]*x3)) / U[1,1]
```

d. If that procedure seems also to long-winded think as a mathematician and ... ♡.

Math	Eigenmath
$x_i = \frac{b_i - \sum_{j=i+1}^{n} u_{ij} \cdot x_j}{u_{ii}}$	`x[i] = (U[i,n+1] - sum(j,i+1,n, U[i,j]*x[j])) / U[i,i])`

5 ⋆Programming the Gauss-Jordan algorithm

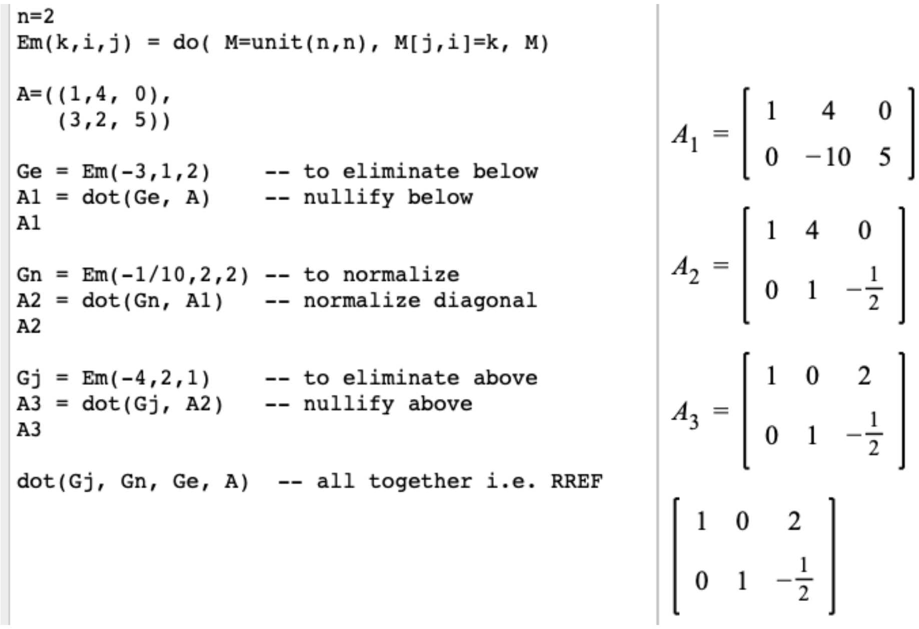
In the previous sections we gained first experience with the semi-automated solution of linear equations with low-dimensional matrices using Eigenmath. We will now use this experience to obtain a fully automated solution process for regular linear systems. In doing so, we do not bother about some pitfalls, because we want to see the main guidelines and also because Eigenmath warns us anyway, if something goes wrong. It is our aim to consolidate our programming skills in Eigenmath, not to compete with the fully fledged routines, that exists in other 'big' CAS anyway.

We proceed as follows while thinking at the Gauss-Jordan-scheme $[\begin{smallmatrix}\square & Z\\ X & Y\end{smallmatrix}]$ in §3.6 :

1. program recipe `Gup` for *quick* elimination *upstairs*: $[\begin{smallmatrix}\square & Z\\ X & Y\end{smallmatrix}] \rightsquigarrow [\begin{smallmatrix}\square & :\\ X & Y\end{smallmatrix}]$
2. program recipe `Ge` to eliminate completely *below* diagonal: $[\begin{smallmatrix}\square & Z\\ X & Y\end{smallmatrix}] \rightsquigarrow [\begin{smallmatrix}\square & Z\\ \dots & Y\end{smallmatrix}]$
3. program recipe `Gj` to eliminate completely *above* diagonal: $[\begin{smallmatrix}\square & Z\\ X & Y\end{smallmatrix}] \rightsquigarrow [\begin{smallmatrix}\square & \vdots\\ X & Y\end{smallmatrix}]$
4. program recipe `Gn` to *normalize* the diagonal: $[\begin{smallmatrix}\square & Z\\ X & Y\end{smallmatrix}] \rightsquigarrow [\begin{smallmatrix}\ddots & Z\\ X & \ddots\end{smallmatrix}]$
5. program recipe `RREF` to put *all together*: $[\begin{smallmatrix}\square & Z\\ X & Y\end{smallmatrix}] \rightsquigarrow [\begin{smallmatrix}\ddots & \vdots\\ \dots & \ddots\end{smallmatrix}]$

5.1 Inductive prologe - a first approach

In the case of 2×2 matrices we can see the recipes in action, because the Elementary matrices *Em*'s play their role. For example, if $[\begin{smallmatrix}\square & Z\\ X & Y\end{smallmatrix}] = [\begin{smallmatrix}1 & 4\\ 3 & 2\end{smallmatrix}]$ we have:



```
n=2
Em(k,i,j) = do( M=unit(n,n), M[j,i]=k, M)

A=((1,4, 0),
   (3,2, 5))

Ge = Em(-3,1,2)    -- to eliminate below
A1 = dot(Ge, A)    -- nullify below
A1

Gn = Em(-1/10,2,2) -- to normalize
A2 = dot(Gn, A1)   -- normalize diagonal
A2

Gj = Em(-4,2,1)    -- to eliminate above
A3 = dot(Gj, A2)   -- nullify above
A3

dot(Gj, Gn, Ge, A)  -- all together i.e. RREF
```

$$A_1 = \begin{bmatrix} 1 & 4 & 0\\ 0 & -10 & 5 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1 & 4 & 0\\ 0 & 1 & -\frac{1}{2} \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 2\\ 0 & 1 & -\frac{1}{2} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 2\\ 0 & 1 & -\frac{1}{2} \end{bmatrix}$$

Try it: ▷ *Click here to run the script.*

Comment. The tree principal phases of the GAUSS-JORDAN algorithm are demonstrated in the changing matrix A. The output $A1$ shows the *elimination below* (`Ge`), the *normalizing* (`Gn`, i.e. to get 1's at the main diagonal) in $A2$ and the *elimination above* (`Gj`) in $A3$. The last command (`RREF`, i.e. *row reduced echelon form*) does the whole process at once.
We now generalize this pattern to higher dimensions.

5.2 Abstraction to a general procedure

In what follows we use the matrix $A = ((2,4,-2,2),(4,9,-3,8),(-2,-3,7,10))$ of the GAUSS-JORDAN-scheme in 3.6 as test matrix. Remember the construction of the GAUSS matrix:

EIGENMATH

```
----      Construction of GAUSS matrix 'Gm' - Go down
Gm(k,A) = do( n   = dim(A,1), -- get the dim
              Gmm = unit(n),  -- k th column
              for(i,k+1,n, Gmm[i,k] = - A[i,k]/A[k,k]),
              Gmm )

A=((2,4,-2,  2), (4,9,-3,  8), (-2,-3,7,  10))

A = dot(Gm(1,A),A)
A = dot(Gm(2,A),A)
Ua = A
```

Try it: ▷ *Click here to run the script.*

The effect was discussed in 3.7.2:

$$A = \begin{pmatrix} 2 & 4 & -2 & 2 \\ 4 & 9 & -3 & 8 \\ -2 & -3 & 7 & 10 \end{pmatrix} \qquad \rightsquigarrow \qquad Ua = \begin{pmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{pmatrix}$$

5.2.1 Procedure `Gu` – going upstairs to eliminate

We are guided by the pattern of the GAUSS matrix `Gm`. But instead of counting forward from $i = k$ to $i = n$ like `Gm`'s for loop $for(i, k+1, n, ..)$, we are now counting backward[9] from $i = k+1$ to $i = 1$ in the `for`-loop of `Gup` to go upstairs in column with number k:

[9] EIGENMATH is smart enough to allow this backward counting!

```
----      Construction of quick elimination upstairs 'Gup' - Go up
Gu(k,A) = do( n = dim(A,1),
               Gmm = unit(n),
               for(i,k,1, Gmm[i,k] = - A[i,k]/A[k,k]),
               Gmm )

Ua=((2,4,-2,  2), (0,1,1,  4), (0,0,4,  8))
A3=dot(Gu(3,Ua),Ua)
A3
A2=dot(Gu(2,A3),A3)
A2
```

Here is the result:

$$U_a = \begin{pmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & -4 & -8 \end{pmatrix} \rightsquigarrow A_3 = \begin{pmatrix} 2 & 4 & 0 & 6 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 4 & 8 \end{pmatrix} \rightsquigarrow A_2 = \begin{pmatrix} 2 & 0 & 0 & -2 \\ 0 & -1 & 0 & -2 \\ 0 & 0 & -4 & -8 \end{pmatrix}$$

Notice, that in our example we also go back stepwise from column 3 til column 2.
Try this script online: ▷ *Click here to run the script.*

5.2.2 Procedure Gn – normalizing the diagonal of a matrix

This is easy. We reuse once again our pattern Gm and in the action part of that for-loop simply divide each diagonal element a_{ii} by itself[10] - which gives '1', i.e.:

```
----  Construction of normalization of the diagonal: 'Gn'
Gn(A) = do( n = dim(A,1), -- get the dim
               Gmm = unit(n),  -- k th column
               for(k,1,n, Gmm[k,k] = 1/A[k,k]),
               Gmm )

Ua = ((2,4,-2,  2), (0,1,1,  4), (0,0,4,  8))
Un = Gn(Ua)
Un
A2 = ((2,0,0, - 2), (0,-1,0, -2), (0,0,-4, -8))
A2n = dot(Gn(A2),A2)
A2n
```

Here is the result:

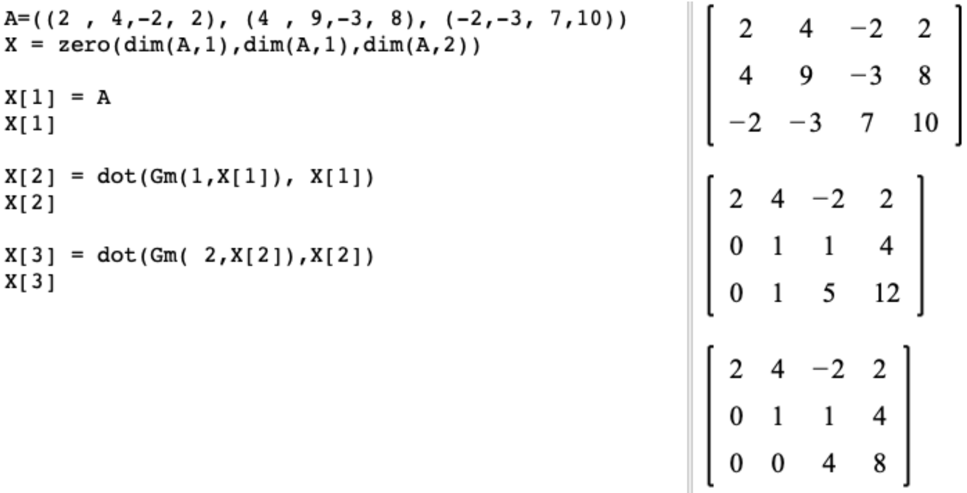
$$U_n = \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/4 \end{pmatrix} \therefore \quad A_2 = \begin{pmatrix} 2 & 0 & 0 & -2 \\ 0 & -1 & 0 & -2 \\ 0 & 0 & -4 & -8 \end{pmatrix} \rightsquigarrow A_{2n} = \begin{pmatrix} 2 & 0 & 0 & -2 \\ 0 & -1 & 0 & -2 \\ 0 & 0 & -4 & -8 \end{pmatrix}$$

Run this script online: ▷ *Click here to run the script.*

[10] Actually, here we had to check, if $a_{ii} \neq 0$. We let this as an exercise for the reader in order to keep our code simple and understandable. EIGENMATH would warn us anyway.

5.2.3 Procedure Ge – the Gauss part: eliminate below diagonal

This part is a bit more difficult. The aim is to get an upper triangular matrix[11], so we had to do several Gauss elimination steps one after the other. We program *bottom-up*, i.e. we first study a concrete example and then try to generalize it.



```
A=((2 , 4,-2, 2), (4 , 9,-3, 8), (-2,-3, 7,10))
X = zero(dim(A,1),dim(A,1),dim(A,2))

X[1] = A
X[1]

X[2] = dot(Gm(1,X[1]), X[1])
X[2]

X[3] = dot(Gm( 2,X[2]),X[2])
X[3]
```

$$\begin{bmatrix} 2 & 4 & -2 & 2 \\ 4 & 9 & -3 & 8 \\ -2 & -3 & 7 & 10 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 1 & 5 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix}$$

We recognize that matrix $X[2]$ has the wanted zeros in his 1st column and matrix $X[3]$ in his 2nd column. Now we construct the associated Eigenmath-procedure.

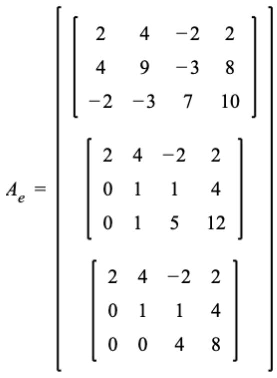
Eigenmath

```
   ----      (G)AUSS (e)limination - iterative version
   Ge(A) =  do( n=dim(A,1),
                 X = zero(dim(A,1),dim(A,1),dim(A,2)),          --(1)
                 X[1]=A,                                         --(2)
                 for( i,1,n-1, X[i+1]=dot(Gm(i,X[i]),X[i]) ),   --(3)
                 X)                                              --(4)

   A=((2,4,-2, 2), (4,9,-3, 8), (-2,-3,7,10))
   Ae=Ge(A)
   Ae
   Ae[3]          --(5)
```

The test run displays the output of the elimination sequence Ae and of its last status $Ae[3]$:

[11] Therefore there should be only zeros below the main diagonal of the input matrix.



$$A_e = \begin{bmatrix} \begin{bmatrix} 2 & 4 & -2 & 2 \\ 4 & 9 & -3 & 8 \\ -2 & -3 & 7 & 10 \end{bmatrix} \\ \begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 1 & 5 & 12 \end{bmatrix} \\ \begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix} \end{bmatrix}$$

$$A_e[3] = \begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix}$$

Comment. In (1) we define a $n \times n \times m^{12}$ *tensor* X to record the intermediate results, i.e. the $n-1$ elimination steps (calls of `Gm`). This container X consists at initialization time of n zero matrices. Think of the 'tensor' X like this .
To initialize the `for`-loop we set the first entry $X[1]$ to the given start matrix A itself.[13] In the i-st step the `for`-loop eliminates via `Gm(i,..)` all elements *below* the diagonal element of i-th column of the actual intermediate matrix $X[i]$. The result is saved in variable $X[i+1]$ as input for the next step.
In the example the procedure finish after 2 steps. It returns the tensor X, i.e. a matrix of 3 matrices of typ 3×4. We save the returned matrix X in Ae to pick at the end the last entry $Ae[3]$, which is a single matrix to start the next procedure.

▷ *Click here to run the script.*

Exercise. Normalize matrix $Ae[3]$ online by

```
Ane = dot( Gn(Ae[3]), Ae[3])     -- normalize the result
Ane
```

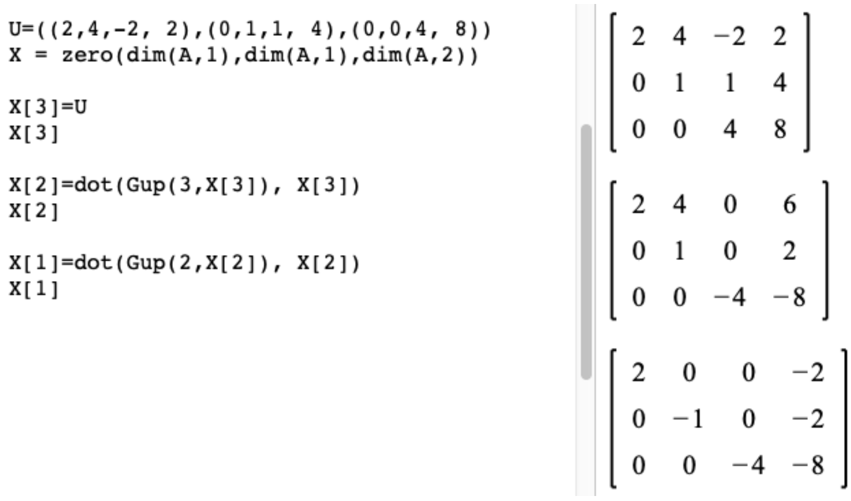
– In your head: What looks Ane like?

[12] In the example a $3 \times 3 \times 4$

[13] Now follow the further description by looking parallel at the step-by-step 3-dimensional hand-made run above.

5.2.4 Procedure `Gj` – the Jordan part: eliminate above diagonal

Now we do the Jordan part. Again we program *bottom-up*, i.e. we first study a concrete step-by-step example to see the difference to procedure `Ge` and then try to generalize it.



```
U=((2,4,-2, 2),(0,1,1, 4),(0,0,4, 8))
X = zero(dim(A,1),dim(A,1),dim(A,2))

X[3]=U
X[3]

X[2]=dot(Gup(3,X[3]), X[3])
X[2]

X[1]=dot(Gup(2,X[2]), X[2])
X[1]
```

$$\begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 4 & 0 & 6 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & -4 & -8 \end{bmatrix}$$

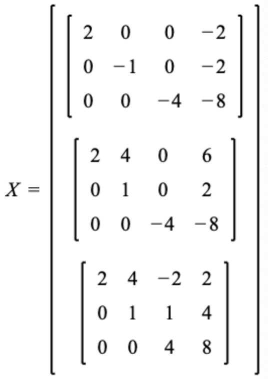
$$\begin{bmatrix} 2 & 0 & 0 & -2 \\ 0 & -1 & 0 & -2 \\ 0 & 0 & -4 & -8 \end{bmatrix}$$

We start with the intermediate result U, which is upper triangular. We recognize that matrix $X[2]$ has the wanted zeros in his 3rd column above his diagonal element -4 and matrix $X[3]$ in his 2nd column. Pay attention: we are counting the process backwards. Now we construct the associated EIGENMATH-procedure, which is abstracted from this small example:

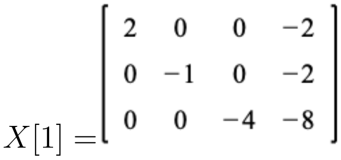
```
----     (G)AUSS (j)ordan part - iterative version
Gj(A) =  do( n = dim(A,1),
             X = zero(dim(A,1),dim(A,1),dim(A,2)),
             X[n] = A,                                          --(1)
             for( i,n-1,1, X[i] = dot(Gu(i+1,X[i+1]), X[i+1])),    --(2)
             X)

U = ((2,4,-2, 2),(0,1,1, 4),(0,0,4, 8))
X = Gj(U)
X           --(2)
X[1]        --(3)
```

Here is the output X and $X[1]$:



$$X = \begin{bmatrix} \begin{bmatrix} 2 & 0 & 0 & -2 \\ 0 & -1 & 0 & -2 \\ 0 & 0 & -4 & -8 \end{bmatrix} \\ \begin{bmatrix} 2 & 4 & 0 & 6 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & -4 & -8 \end{bmatrix} \\ \begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix} \end{bmatrix}$$



$$X[1] = \begin{bmatrix} 2 & 0 & 0 & -2 \\ 0 & -1 & 0 & -2 \\ 0 & 0 & -4 & -8 \end{bmatrix}$$

Comment. In (1) we save the input matrix A in the *last entry* of the container (tensor) X and initialize the `for`-loop with that matrix. Now we let the for loop *count down* (as in the 3-dimensial example) and in the i-st step it eliminates via `Gup(i,..)` all elements *above* the diagonal element of $i+1$-th column of the actual intermediate matrix $X[i+1]$. The result is saved in variable $X[i]$ as input for the next step.
In the example the procedure finish after 2 steps. It returns the tensor X, i.e. a matrix of 3 matrices of typ 3×4. We pick the *first* entry $X[1]$, which is a single matrix.

▷ *Click here to run the script.*

Exercise. Do the normalization of the result $X[1]$ online to get the solution of the linear system. – In your head: What looks *Ane* like?

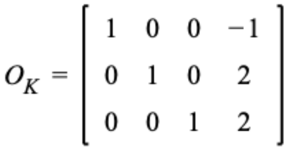
5.2.5 rref – composing the procedures

We now put all parts together in the function `RREF`:

```
RREF(A)= do( n = dim(A,1),
             U = Ge(A),               -- Gauss elimination
             V = Gj(U[n]),            -- Jordan elimination on last matrix
             X=dot(Gn(V[1]),V[1]),    -- normalization of first matrix
             X)                       -- return result

A = ((2,4,-2, 2), (4,9,-3, 8), (-2,-3,7,10))
OK = RREF(A)
OK
```

The call of `RREF` returns the result of the whole elimination processes including the normalization. Therefore, in the test case, we can directly read off the solution of the linear system A in his terminal status:



$$O_K = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

▷ *Click here to run the script.*
The solution of the linear system A is $(x, y, z) = (-1, 2, 2)$.

P33. (gjBox1 - the didactical version) Put all sub-procedures *Em, Gm, Gu, Gn, Ge, Gj* und *RREF* in a file called `gjBox1.txt`. Pay attention to the order of the individual parts: which procedure needs which other as a help function? EIGENMATH needs the correct call sequence, otherwise it reports an error.

If you use EIGENMATH^{online}, this toolbox of functions is already present and can be called using

```
run("gjBox.txt")
```

If you use the EIGENMATH app under MACOS, you *must* put this toolbox in the `Download` folder, because of *iMac*'s security conventions.
○ We name `gjBox1.txt` a *didactical* version, because their functions show all intermediate results. `gjBox1` is therefore recommended for use in learning phases or to search for mistakes in hand calculations, allowing a kind of trace.

P34. (gjBox - the functional version)
a. Alter the sub-procedures `Ge`, `Gj` und `RREF` so, that these functions only give back one result matrix. Then collect all (revised) sub-procedures *Em, Gm, Gu, Gn, Ge, Gj* and *RREF* in a file called `gjBox.txt`.
b. These revised toolbox functions `Ge` and `Gj` allows together with a small helper function `doGn(.)`[14] a very compact and meaningful formulation of the main function `RREF` in the so-called *functional style*:

```
doGn(M) = dot(Gn(M), M).           -- action of Gn
RREF(A) = doGn( Gj( Ge( A) ))      -- functional programming
```

Coded in this way the definition of `RREF` reflects 1:1 the mathematical calculation process

$$RREF\colon\ A \overset{elim.down}{\rightsquigarrow} A1 \overset{elim.up}{\rightsquigarrow} A2 \overset{norm.}{\rightsquigarrow} R \quad =\text{result}.$$

If you use EIGENMATH^{online}, this toolbox of functions is called with the command

```
run("gjBox.txt")
```

If you use the EIGENMATH app under MACOS, you *must* put this toolbox in the `Download` folder because of *iMac*'s security conventions.

[14] The definition of `doGn` is actually unnecessary and superfluous, but allows a cleaner and more aesthetic programming construct of `RREF`, it is in this sense a so-called '*syntactical sugar*'.

With the procedure `RREF` we are now able to solve linear systems of equation very handy. If difficulties arise we may fall back on the constituent subroutines `Ge`, `Gj`, `Gn` and even `Gm` or `Gup` or to do single steps with `Em` to observe, what caused the problem.

Subroutine `Gj` nullifies a upper triangular matrix, returning a diagonal matrix. So no backsubstitution is necessary any more. But to train our programming skills in EIGENMATH we will nevertheless offer a program for `backSubstitution`.

5.3 Back to backSubstitution

Backsubstitution for solving linear systems were studied before in concrete situations like P25, P27, P32.c./d. and section 3.7.2. Look at our GAUSS-JORDAN model problem:

$$A = \begin{pmatrix} 2 & 4 & -2 & 2 \\ 4 & 9 & -3 & 8 \\ -2 & -3 & 7 & 10 \end{pmatrix} \rightsquigarrow U = \begin{pmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{pmatrix} \quad \rightsquigarrow \quad S = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

At solution status U we now want to do a *backsubstitution scheme* to reach the final status S. For this lower dimensional linear problem, we had the following routine

```
x3 =  U[3,4] / U[3,3]
x2 = (U[2,4] - U[2,3]*x3) / U[2,2]
x1 = (U[1,4] - (U[1,2]*x2 + U[1,3]*x3)) / U[1,1]
```

which leaded directly to the final status S.
Here is the abstraction of this '***G**o **b**ack*' scheme for arbitrary dimension n.

```
backSubst(U) = do( n = dim(U,1),
                   Z = zero(2,n),
                   x = Z[1],
                   x[n] = U[n,n+1] / U[n,n],
                   for(i,n-1,1,
                    x[i] = (U[i,n+1] - sum(j,i+1,n, U[i,j]*x[j])) /U[i,i]),
                   x)

U = ((2,4,-2, 2),(0,1,1, 4),(0,0,4, 8))

S = backSubst(U)
S                        -- Solution

Bs(U)=backSubst(U)       -- alias for backSubst(itution)
Bs(U)
```

▷ *Click here to run the script.*

Comment. Our container is vector x, consisting of n zeros. We start with the *last* line and the *last* component $x[n]$ oof the solution, e.g. $x[3] = U[3,4]/U[3,3] = 8/4 = 2$. We now go back[15] with help of the `for`-loop until we reach the first solution component $x[1]$. Each $x[i]$ is calculated along the RHS of the formula, which itself is a *realization* of the mathematical formula for backsubstitution:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} u_{ij} \cdot x_j}{u_{ii}}$$

The b_i are the entries of the last column of matrix U.[16]

$\bowtie$

Summary.

In this optional section we programed (sub)routines to solve linear systems of equations. Along the journey we developed functions for the well-known Gauss-Jordan algorithm using Eigenmath.

Here are some of our programming guidelines:

- ○ *bottom-up development*, i.e. using experiences from concrete cases and abstract it.
- ○ *divide & conquer*, i.e. cut a bigger task into smaller handier pieces.
- ○ *functional programming*, i.e. try to concatenate available functions[17].
- ○ *modularization*, i.e. do a strategic–methodical structuring.
- ○ *stepwise refinement*, i.e. do a functional decomposition of a top-down design.

In summa:

Simplicity.
Clarity.
Generality.

... Automation.

Kernighan & Pike:
The Practice of Programming.

[15] The `for`-loop runs *backwards* from $i = n - 1 = 3$ to $i = 1$.

[16] See [1, pp. 8–14] for an implementation in C.

[17] See the coding of RREF.

6 Regular Linear Systems

Maybe you were not interested in programming with Eigenmath and you has skipped the last section. In this case you can nevertheless use all the functions of the toolbox `gjBox.txt` to solve linear systems of equations. You only need to load the tools of the box into the script area of Eigenmath using the command `run(.)`. This is done in this section.

P35. (Liu **III)**
Check the functions of the `gjBox.txt` for the Liu problem of P30 and P32.
Here is a start:

Eigenmath

```
Liu = ((1,3,2,8,5,  95),
       (2,5,3,9,4, 112),
       (3,5,7,6,4, 116),
       (7,6,4,5,3, 128),
       (9,7,3,2,5, 140))

RREF(Liu)
```

a. Verify the result $(x, y, z, u, v) = (7, 4, 3, 5, 6)$. Check it on the 5 equations.
b. What is the 'half time' matrix U?
c. Determine the solution using the suite $Ge \rightsquigarrow Gj \rightsquigarrow Gn$ in order to consider intermediate results.

6.1 Jacobi method

Jacobi's method is a simple so-called *iterative* ('for-loop') technique for solving linear systems. Let's begin with an example:[18]

Eigenmath

```
--  x  y  z  rhs         --Linear system:
A=((5, 1, 1, 10),        -- 5x+y+z=10
   (1, 6,-2,  7),        -- x+6y-2z=7
   (1,-3, 7, 16))        -- x-3y+7z=16

T=100                   -- 'Time' or counter
do(x=0, y=0, z=0)       -- initialization, start values
for(n,1,T,  do( x1= x, y1= y, z1= z),   -- (1)
                x=(10-y1-z1)/5,
                y=(7-x1+2*z1)/6,
```

[18] see [8, pp. 72–73] and [1, pp. 28–31]

```
                    z=(16-x1+3*y1)/7 )

     do(float(x), float(y), float(z))  -- return decimal numbers
```

▷ *Click here to run the script.*

Comment. Identifier T counts the number or repetitions of the calculations in the *for* loop. The *do* statement takes in (1) the current values of the run, starting first with the arbitrary guess $(x, y, z) = (0, 0, 0)$.

Exercise. a. What is the main idea of Jacobi's method?
b. Solve the linear systems from 3.6, 4.1 and P32 using Jacobi's method.
c$^{\heartsuit}$. Program the Jacobi method in Eigenmath.
Be inspired from the pseudocode[19] or the Python code or the mathematical formula

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n$$

in which $x^{(k)}$ is the k-th *approximation* to the solution x and $x^{(k+1)}$ is the next or $(k+1)$-iteration of x.
Check your code e.g. with the system $A \star X = B$ from that wikipedia page:

$$A = \begin{bmatrix} 2 & 1 \\ 5 & 7 \end{bmatrix}, \; B = \begin{bmatrix} 11 \\ 13 \end{bmatrix} \quad \text{and} \quad X^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

6.2 *Definition*: Determinant of a Linear System

The so-called *determinant of a matrix* is a number which *determines*, whether a linear system is unique solvable with the Gauss-Jordan-algorithm. We define the determinant `Det(A)` as the product of the diagonal elements in the upper triangular transformation of A.[20] There exist also a build-in determinant function `det`, which is denoted with a *lower initial* letter and is only defined for *square* $n \times n$ matrices $A = \square$.

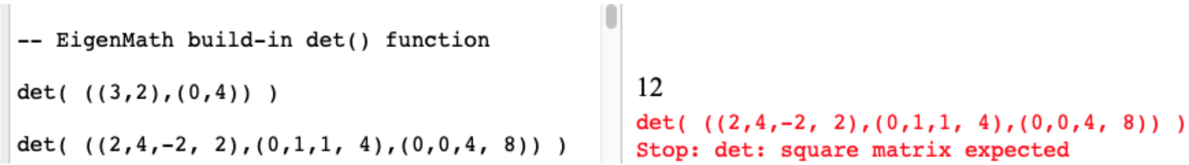
[19] `https://en.wikipedia.org/wiki/Jacobi_method`

[20] We show other cooler methods to define the determinant of a square matrix later, e.g. using concepts of the so-called *Geometric Algebra*.

6.2.1 Examples

$$
\begin{aligned}
Det(\begin{bmatrix} 3 & 7 \\ 0 & 4 \end{bmatrix}) &= 3 \cdot 4 = 12 \\
Det(\begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix}) &= 2 \cdot 1 \cdot 4 = 8 \\
Det(\begin{bmatrix} 2 & 4 & -2 & 2 \\ 4 & 9 & -3 & 8 \\ -2 & -3 & 7 & 10 \end{bmatrix}) &\overset{def}{=} Det(\begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix}) = 8
\end{aligned}
$$

EIGENMATH returns using its build-in function:



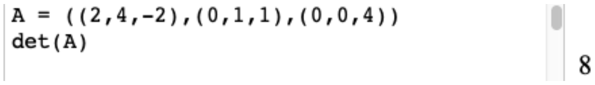
```
-- EigenMath build-in det() function

det( ((3,2),(0,4)) )

det( ((2,4,-2, 2),(0,1,1, 4),(0,0,4, 8)) )
```

```
12
det( ((2,4,-2, 2),(0,1,1, 4),(0,0,4, 8)) )
Stop: det: square matrix expected
```

Remark: Our definition matches with the build-in function `det` for square matrices, but has the advantage to work also for non-square augmented linear system matrices, which are of typ $m \times n$. Otherwise, every time we would had to detach the quadratic part on the LHS of the system in order to use the build-in determinant, e.g.



```
A = ((2,4,-2),(0,1,1),(0,0,4))
det(A)
```

```
8
```

6.2.2 Definition

The *determinant* `Det1(U)` of a linear system U in upper triangular shape is

-
  ```
  Det1(U)  = product(i,1,dim(U,1), U[i,i])          -- (0)

  B = ((2,4,-2),    -- is Upper triangular
       (0,1, 1),
       (0,0, 4))
  Det1(B)           -- returns 8
  ```

  ▷ *Click here to run the script.*

- Matrix[21] A is *regular* $\overset{def}{\Longleftrightarrow}$ `Det1`$(A) \neq 0$.

- Matrix A is *singular* $\overset{def}{\Longleftrightarrow}$ `Det1`$(A) = 0$.

[21] e.g. a Linear system of equations in augmented matrix shape.

This definition is executable with Eigenmath and with paper & pencil.

a. Argue using Eigenmath, why the systems N and M are singular. Does the systems have a solution?

$$N = \begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix} \qquad M = \begin{bmatrix} 2 & 4 & -2 & 2 \\ 0 & 1 & 2 & 4 \\ 0 & 2 & 4 & 8 \end{bmatrix}$$

b. Determine `Det1` of the systems A in 4.1, 4.2 and `Liu` in P33. Notice, that the systems must be transformed to upper triangular form!
For comparison, use the build-in `det`. Notice: You must use their quadratic LHS.
c.

> Looking at b. it is annoying to detach the quadratic LHS for the use of `det` or to prepare the upper triangular form for the use of `Det1` in advance. Therefore we want to get rid of this condition of this 'semi-automatic' version and like to code a fully-automatic version, which do this preparatory step for us.

• Program a function `Det(A)`, which transforms A to upper triangular form U before calculating the product of the diagonal elements. *Hint*: look up the code for `RREF`.
• Program a function `det1(A)`, which calls `det` with the LHS of augmented system A.

Determinant:

build-in	*user defined*	
`det`		`normally used det for arbitrary` □
	`Det1`	`simple version for upper triangle matrices/LS`
	`Det`	`version for arbitrary LS`
	`det1`	`enhanced det for arbitrary LS`

From now on we only use `det` or `Det`.

6.3 Row exchange and partial pivoting

In order to avoid a division by zero error, caused by a possible diagonal element of value zero, we sometimes must enhance the Gauss algorithm with pivoting.

> ”The *pivoting procedure* consists in interchanging rows (*partial* pivpting) or rows and columns (*full* pivoting) so as to put a particularly desirable element (the *pivot*) in the diagonal position. In practice, the pivot corresponds to the largest (in magnitude) available element.

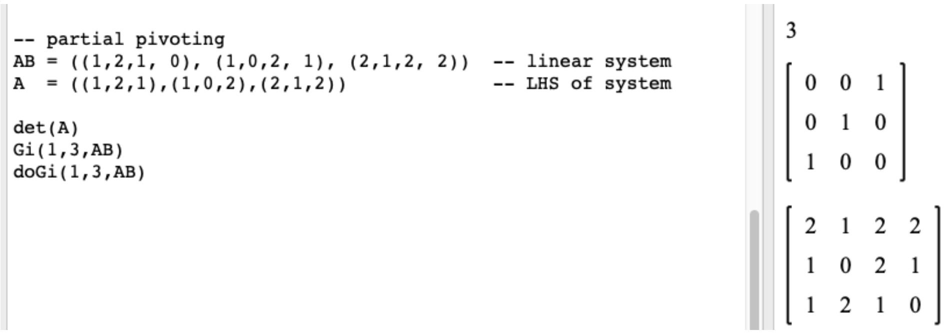
To make partial pivoting available we define the interchanging of rows in Eigenmath.

```
-- Gauss (i)nterchange rows
Gi(i,j,A)=do( n   = dim(A,1),
              Gii=unit(n),
              N= Gii[i],
              M= Gii[j],
              Gii[j]=N,
              Gii[i]=M,
              Gii)

doGi(i,j,A) = dot(Gi(i,j,A), A)  -- a bit syntactic sugar: Gi(i,j,A) * A

-- partial pivoting
AB = ((1,2,1, 0), (1,0,2, 1), (2,1,2, 2))  -- linear system
Gi(1,3,AB)    -- (1) transformation matrix to interchange row1 and row3
doGi(1,3,AB)  -- do it
```

Output:



```
-- partial pivoting
AB = ((1,2,1, 0), (1,0,2, 1), (2,1,2, 2))  -- linear system
A  = ((1,2,1),(1,0,2),(2,1,2))             -- LHS of system

det(A)
Gi(1,3,AB)
doGi(1,3,AB)
```

3

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

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

▷ *Click here to run the script.*

Comment. The small helper function `doGi` let the interchanging matrix Gi operate on the matrix A via `dot(.)`. The output displays the matrix Gi and the system matrix AB with interchanged rows 1 and 3. The pivot (i.e. the maximum of the values $1, 1, 2$ of the first column) is the number 2 at the former position $[1, 3]$. It is swapped at the diagonal position $[1, 1]$.

a. Verify that $\det(A) = 3$, the system AB is therefore regular and should have one solution. Solve the system with a call to `RREF`.

```
A  = ((1,2,1),(1,0,2),(2,1,2))      -- LHS of system
det(A)   -- (1)
```

b. The function Gi does the swap of the two rows i and j using two memory slots M and N. Reduce the 4 lines of the swap code to 3 by using only *one* memory slot M.
Hint: `https://en.wikipedia.org/wiki/Swap_(computer_programming)`

♡. Do you like to code a function `Gp(j,A)`, which search for the *pivot* in column j?

6.4 *Definition*: The inverse A^{-1} of a Matrix

We have touched this before in 3.4 and P26. We will now discuss the inverse of A in detail.

6.4.1 Prelude: some lower dimensional examples

First we do some analogy conclusions by means of three concrete examples in order to understand the solution X of a linear system $A * X = E$ to be the inverse A^{-1} of A.

1: $\frac{1}{2}$ is solution of the equation $2 \cdot x = 1$ in $\mathbb{R}$, because $x = 2^{-1} = 0.5$ fulfills $2 \cdot 2^{-1} = 1$,

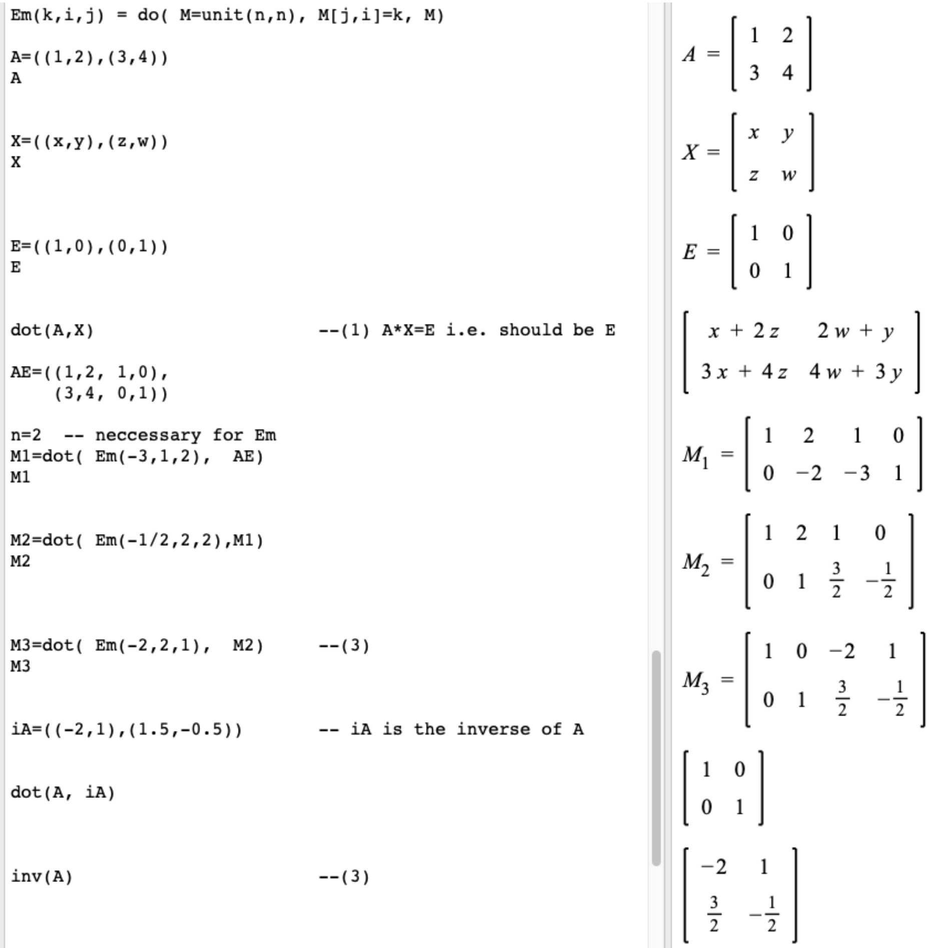
i.e. the matrix $[\frac{1}{2}]$ is the solution of the 1×1 linear system $[2] * [x] = [1]$ in $\mathbb{R}^{1\times 1}$,

i.e. is the solution of the system $A * X = E$ with $A = [2], X = [x], B = [1] = E$ and E being the 1-dimensional identity matrix.

We write: $X = A^{-1}$ and call it the *inverse* of A.

We have: $A * A^{-1} = [2] * [2^{-1}] = [2 \cdot \frac{1}{2}] = [1] = E \quad \overset{analog}{\sim} \quad 2 \cdot 2^{-1} = 1.$

2: In $\mathbb{R}^{2\times 2}$, we solve $A * X = E$ for $X \overset{def}{=} A^{-1}$:



```
Em(k,i,j) = do( M=unit(n,n), M[j,i]=k, M)

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

X=((x,y),(z,w))
X

E=((1,0),(0,1))
E

dot(A,X)                        --(1) A*X=E i.e. should be E

AE=((1,2, 1,0),
    (3,4, 0,1))

n=2  -- neccessary for Em
M1=dot( Em(-3,1,2),  AE)
M1

M2=dot( Em(-1/2,2,2),M1)
M2

M3=dot( Em(-2,2,1),  M2)        --(3)
M3

iA=((-2,1),(1.5,-0.5))          -- iA is the inverse of A

dot(A, iA)

inv(A)                          --(3)
```

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

$$X = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

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

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

$$M_1 = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 0 & -2 & -3 & 1 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 0 & 1 & \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

$$M_3 = \begin{bmatrix} 1 & 0 & -2 & 1 \\ 0 & 1 & \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

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

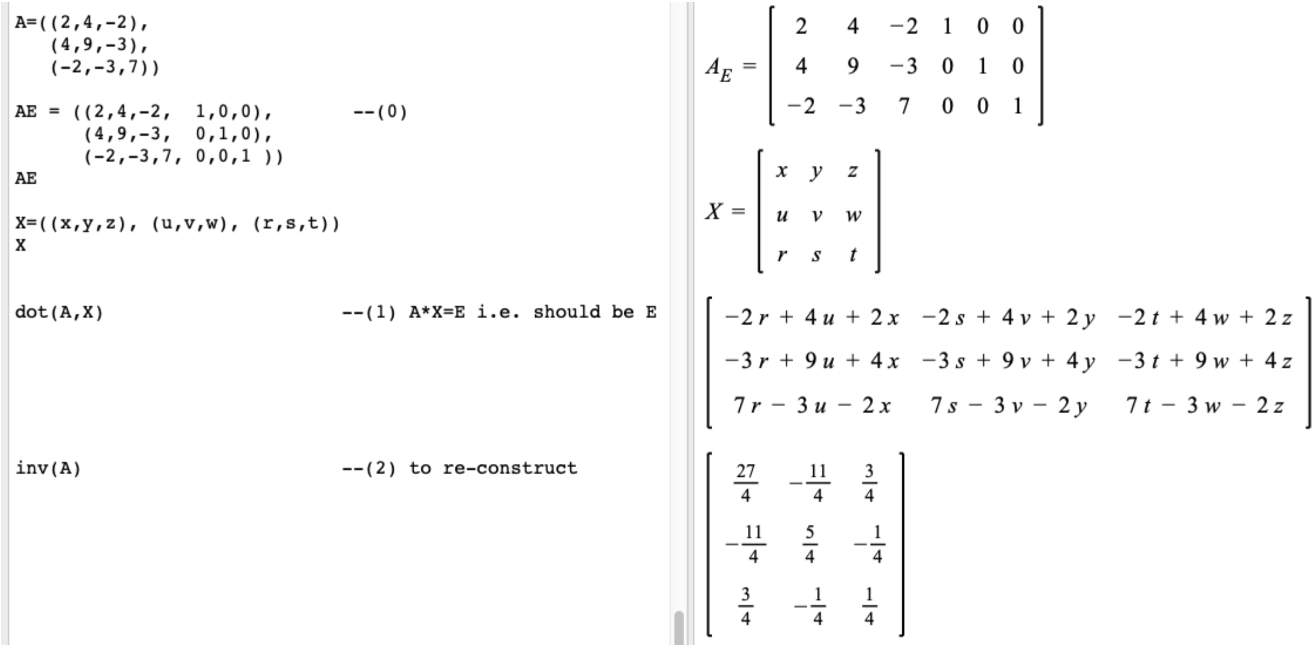
$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

Comment. We determine the inverse of A as solution (matrix) of the linear system $A * X = E$ with the identity matrix E as special RHS. To let the transformation matrices `Em` operate simultaneously on both sides of the linear system, we write the system as usual as *one augmented matrix* `AE` $= [A \;\; E]$. Three `Em`-steps solve the system for X giving $X = \left[\begin{smallmatrix}-2 & 1\\ 1.5 & -0.5\end{smallmatrix}\right] = A^{-1}$.

The build-in EIGENMATH function `inv(A)` returns the same value for the inverse of A directly. *But we had looked behind the scene* and know, how the inverse of a square matrix could be calculated: we transmuted the black box A^{-1} into a white box.

Intermezzo a. Calculate the inverse of A by hand looking at line (1).

3: In $\mathbb{R}^{3\times 3}$, we solve $A * X = E$ for $X \overset{def}{=} A^{-1}$ for the LHS of the demo linear system of the GAUSS-JORDAN scheme, see 3.6.. We make use of the GAUSS-JORDAN suite of functions in our `gjBox`. Here is the start:



```
A=((2,4,-2),
   (4,9,-3),
   (-2,-3,7))

AE = ((2,4,-2,  1,0,0),        --(0)
      (4,9,-3,  0,1,0),
      (-2,-3,7, 0,0,1 ))
AE

X=((x,y,z), (u,v,w), (r,s,t))
X

dot(A,X)                        --(1) A*X=E i.e. should be E

inv(A)                          --(2) to re-construct
```

$$A_E = \begin{bmatrix} 2 & 4 & -2 & 1 & 0 & 0 \\ 4 & 9 & -3 & 0 & 1 & 0 \\ -2 & -3 & 7 & 0 & 0 & 1 \end{bmatrix}$$

$$X = \begin{bmatrix} x & y & z \\ u & v & w \\ r & s & t \end{bmatrix}$$

$$\begin{bmatrix} -2r + 4u + 2x & -2s + 4v + 2y & -2t + 4w + 2z \\ -3r + 9u + 4x & -3s + 9v + 4y & -3t + 9w + 4z \\ 7r - 3u - 2x & 7s - 3v - 2y & 7t - 3w - 2z \end{bmatrix}$$

$$\begin{bmatrix} \frac{27}{4} & -\frac{11}{4} & \frac{3}{4} \\ -\frac{11}{4} & \frac{5}{4} & -\frac{1}{4} \\ \frac{3}{4} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

Comment. We determine the inverse of A as solution (matrix) of the linear system $A * X = E$, which we write in (0) as augmented matrix `AE` $= [A \; E]$. We want to solve the system for X. In principe we could do the solution by hand using the 9 linear equations for the 9 unknowns $x, y, ...t$ of X. The aim is the term in (2), calculated with the build-in function `inv(A)`. We will use the toolbox `gjBox` to re-construct this result in order to understand the procedure to calculate the inverse matrix. We proceed as follows and demonstrate 3 possibilities of different tempi.

- fast:

```
doGm(j, M) = dot(Gm(j, M), M)     --(2)  syntactic sugar ;)
doGu(j, M) = dot(Gu(j, M), M)
```

```
doGn( M) = dot(Gn( M), M)

M1=doGm(1,AE)
M1
M2=doGm(2,M1)
M2
M3=doGu(3,M2)
M3
M4=doGu(2,M3)
M4
iA=doGn(M4)
iA                             --  read off the solution with your eyes
```

▷ *Click here to run the script.*

In this version we get insight in the 'baby' solution steps watching the outputs. We learn: what are the effects of the actions of `Gm` and `Gu`. This version is much quicker than the use of `Em`, which gives the most basic and most detailed control of the single solution steps.

- faster:

```
doGn(M) = dot(Gn( M), M)

AE = ((2,4,-2,  1,0,0),
      (4,9,-3,  0,1,0),
      (-2,-3,7, 0,0,1 ))
AE

AE1=Ge(AE)
AE1
AE2=Gj(AE1[3])
AE2[1]
AE3=doGn(AE2[1])
AE3
```

▷ *Click here to run the script.*

Here we make only 3 bigger steps, but can watch some relevant outputs, too.

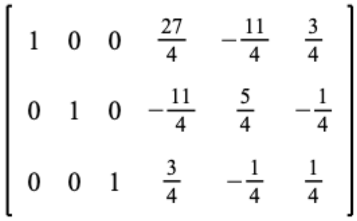
- fastest:

```
RREF(AE)                      --  read off the solution
```

▷ *Click here to run the script.*

The quickest version abstracts from all intermediate steps, making one 'giant' step directly to the solution. It is this version which leads us to our constructive definition of the inverse matrix.

The output in all three cases is the same:



$$\begin{bmatrix} 1 & 0 & 0 & \frac{27}{4} & -\frac{11}{4} & \frac{3}{4} \\ 0 & 1 & 0 & -\frac{11}{4} & \frac{5}{4} & -\frac{1}{4} \\ 0 & 0 & 1 & \frac{3}{4} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

6.4.2 Definition of the Inverse matrix A^{-1}

Let `E=unit(n)` be the identity matrix of same typ as the square matrix A of type $\mathbb{R}^{n\times n}$ with $det(A) \neq 0$. Then:

- ▷ *The* ***inverse*** *matrix* $\mathbf{A^{-1}}$ *of A is per definition the solution X of the linear system $A * X = E$.*
- ▷ *Therefore we have: $A * A^{-1} = E$,   i.e. A^{-1} 'neutralizes' the effect of A.*
- ▷ *If* `AE` $= [A\ E]$ *is the augmented matrix of the system $A * X = E$, then $A^{-1} = \square$ in* `RREF(AE)`$= [\square\square]$.
- ▷ *The solution X of the linear system $A * X = B$ is*

$$\boxed{\mathbf{X = A^{-1} * B}}$$

We have the analogies:

$$\begin{aligned} a \cdot x = 1 \quad &\Rightarrow \quad x = 1/a = a^{-1} \\ A * X = E \quad &\overset{analogy}{\rightsquigarrow} \quad X = \text{”}E/A = 1/A\text{“} = A^{-1} \\ A * X = B \quad &\overset{analogy}{\rightsquigarrow} \quad X = \text{”}1/A * B\text{“} = A^{-1} * B \end{aligned}$$

We have the lexicon:

	Math	EIGENMATH
inverse matrix	A^{-1}	`inv(A)`
linear system	A*X=B	`[A B]`
solution of LS	$A^{-1} * B$	`dot( inv(A), B)`

We see: The inverse of A plays with regard to the matrix multiplication $*$ the same role as the inverse number $\frac{1}{a} = a^{-1}$ to a number a in the usual multiplication: $\frac{1}{3} \cdot 3 = 3 \cdot \frac{1}{3} = 1$.

6.4.3 Exercise.

a. Determine the inverse of A in 5.4.1.2: analog zu 3: using the GAUSS-JORDAN function suite `G..` and especially `RREF`.
b. Calculate the solution of the model problem system in P30. using its inverse.
c. Calculate the the solution of the `Liu` system in P32. using its inverse.
♡. Do you like to code a user function `Inv(A)`, which peels the inverse of a matrix A out of the call `RREF(AE)`?[22]

P36. (Solution of an 3×3 linear system witn EIGENMATH.)

$$\begin{aligned} x+2y+3z &= 4 \\ 3x+4y+3z &= 2 \\ 2x+2y+3z &= 4 \end{aligned}$$

a. Write the numbers of this 3×3 - LGS in an 3×4 - system matrix `M = [A B]` and transform the first three columns of the system matrix into the identity matrix E_3 by means of suitable multiplications with transformation matrices:

$$\begin{bmatrix} 1 & 0 & 0 & \star \\ 0 & 1 & 0 & \star \\ 0 & 0 & 1 & \star \end{bmatrix}$$

Read off the solution for the LS and do the sample with/without EIGENMATH.

b. Represent the solution process algebraically as the product of transformation matrices. Possible intermediate results can be displayed with `G...`
What is the matrix `G` with $G * M = [E\ \Box]$?
Calculate with EIGENMATH the inverse matrix of the coefficient matrix $A =$

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

P37. (Determination of solutions of 3×3 linear system.)
Repeat the last exercise for:
a.

$$\begin{aligned} 2x-3y+z &= -8 \\ 2y+5z &= -6 \\ -2z &= 4 \end{aligned}$$

[22] You get the solution on demand via the authors email address. ♡

b.

$$\begin{aligned} 3x + 6y - 2z &= -4 \\ 3x + 2y + z &= 0 \\ \frac{3}{2}x + 5y - 5z &= -9 \end{aligned}$$

c.

$$\begin{aligned} x + 2y - z &= 2 \\ y + z &= 5 \\ x - y - z &= 2 \end{aligned}$$

P38. (The inverse and the solution of a 2×2 linear system.) The augmented system matrix `[A B] =`$\left[\begin{smallmatrix} 3 & 6 & 6 \\ 2 & 3 & 3.5 \end{smallmatrix}\right]$ of the LS $\begin{smallmatrix} 3x+6y=6 \\ 2x+3y=3.5 \end{smallmatrix}$ was transformed through an algebraic `RREF`– process into an final status `[E R]`, where it was possible to read off the solution $R = \left[\begin{smallmatrix} 1 \\ 0.5 \end{smallmatrix}\right]$ immediately:

$$\begin{bmatrix} 3 & 6 & 6 \\ 2 & 3 & 3.5 \end{bmatrix} \quad \underset{\text{scal}}{\longrightarrow} \underset{\text{elim}}{\longrightarrow} \underset{\text{scal}}{\longrightarrow} \underset{\text{elim}}{\longrightarrow} \quad \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0.5 \end{bmatrix}$$

This process was expressed with this matrix equation:

$$\underbrace{\begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} * \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & 1 \end{bmatrix}}_{=A^{-1}} * \begin{bmatrix} 3 & 6 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

abbreviated[23]

$$E_4 * \cdots * E_1 * A = E$$

We notice

$$A^{-1} = E_4 * \cdots * E_1 = \begin{bmatrix} -1 & 2 \\ 2/3 & -1 \end{bmatrix}$$

and

$$\begin{bmatrix} -1 & 2 \\ 2/3 & -1 \end{bmatrix} * \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

ergo:

$$\mathbf{A^{-1} * A = E}$$

○ *Basically, the LS was solved by calculating the inverse matrix A^{-1} of the coefficient matrix $A = \left[\begin{smallmatrix} 3 & 6 \\ 2 & 3 \end{smallmatrix}\right]$ of the system.*

a. Calculate the inverse of A by paper & pencil using the augmented matrix.
b. Calculate the inverse of A by using `inv(A)` and `RREF(AB)`.
Also determine the solution of AB most directly.

[23] Herein we denote for short the elementary matrices E_1 etc. of the solution process.

P39. (Calculating inverse matrices.) **a.** Calculate the inverse in *P.38* with one `dot`–call in EIGENMATH using the 4 transforming elementary matrices.
b. To check understanding, use the GAUSS-JORDAN algorithm to calculate the inverse matrices of

$$(1)\quad \begin{bmatrix}2 & 3\\ 3 & 5\end{bmatrix} \qquad (2)\quad \begin{bmatrix}2 & 5\\ -3 & -7\end{bmatrix}$$

$$(3)\quad \begin{bmatrix}2 & 1 & 1\\ 0 & -1 & -2\\ 0 & 0 & -4\end{bmatrix} \qquad (4)\quad \begin{bmatrix}0 & 1 & 2\\ 1 & 0 & 3\\ 4 & -3 & 8\end{bmatrix}$$

○ Look at (1) and (2). Vermutung? Conjecture?

P40. (Daily production.)[24] In a company there are two machines M and N for the production of goods A and B. The machine M can be used for 12 hours, the machine N for 16 hours. The processing of the product A requires 2 hours on the machine M and an additional hour on the machine N. For B the data are 2 and 3 hours, respectively.
a. What is the daily production?
b. What is the effect of purchasing another machine of type M, type N or both types? Interpret the results.

P41. (To be or not to be - that is the *existential question.*) Study the matrices

$$(1)\quad A = \begin{bmatrix}2 & 3\\ 3 & 6\end{bmatrix} \qquad (2)\quad B = \begin{bmatrix}1 & -2 & -1\\ -1 & 5 & 6\\ 5 & -4 & 5\end{bmatrix}$$

Try *by all means* with/without EIGENMATH to find inverse matrices of A or B.
Analyze possible resistances or problems with algebraic and/or geometric tools (line image, column image, matrix image, RREF . . .).

⋈

> The observation that many matrices does not have an inverse has serious consequences. If one interprets such a *inverseless matrix* matrix `A` as the left hand side of a linear system of equations $A * X = B$, then one cannot solve such an LS using an inverse "A^{-1}" - since such a matrix cannot be created at all. What to do in such situations - which often occur in the applications - is the topic of the next part.
>
> *The existence of an inverse is in any case an extremely fortunate circumstance for the solution of a linear system*

[24] See [2, p. 57]

6.5 Universal tool Gauss-Jordan-`scheme` ∼ `RREF`

Summary. The universal tool[25] Gauss-Jordan **scheme** realized in Eigenmath as `RREF` and its child's

– to solve linear systems of equations or

– to calculate the inverse matrix

is used as follows:

1. *input* a suitable matrix for processing: the human being thinks ...
2. *let work* the CAS-tool Eigenmath : man rests...
3. *interpret* the output and read off the solution: the human being thinks ...

So ask yourself:

Load with what...

↓

`RREF[???]   =  [!!!]`

↑ ↑

...start tool ...how to interpret?

♡

> The final status of Gauss-Jordan $\overset{alias}{\sim}$ `RREF`[!!!] should fluently be produced by hand of the learner *(for invoices without CAS!)* if necessary, e.g. in external exam situations without approved aids: *eliminate* → *normalize* → *backSubstitute.*

After that, the interpretive considerations proceed as usual - if you have calculated without errors and are no caught by calculation mistakes.

[25] so to speak the "Swiss Army Knife" of linear algebra...

6.6 Some problems – the practical minimum

Make your own selection from the following training tasks and solve them using as many different methods as possible with/without EIGENMATH. We cite only a very small selection of exercises, most of the problems are quoted from [2, P40, P47, P48], [3, P43, P45, P46, 49] and [4, P44, P52, P53, P54]. For German speaking users these books contain an abundance of additional exercises, for the English speaking community e.g. [5] is a source of many (solved) exercises, if there is need of further training material to reach your level of procedural competence.
For the solution, you can optionally use our EIGENMATH tools

○ GAUSS-JORDAN-suite `Gm, Gu, Ge, Gj, Gn`

○ `RREF(...)`

○ $(...)^{-1}$

As a general rule, we dispense with the detailed consideration with elementary matrices.

P42. (Active Summary)

① **The inverse matrix** A^{-1} of a [26] matrix A is calculated

...as product of Elementary matrices: $A^{-1} =$

...with the characterizing equation: $A*$

...with the RowReduction–ansatz: `[A`

② A **is invertible** (i.e has an inverse A^{-1}), if

...A is quadratic of type $n \times n$

...`RREF` transforms matrix A into : $A \overset{rref}{\rightsquigarrow} E$

③ **The matrix multiplication** $\cdots * \ldots$

j n

□□ is *commutative*

□□ is *assoziative*

□□ has for each element an inverse

□□ has a *neutral* element (∼ number 1 in $\mathbb{R}$)

[26] fill with: matrix with $det(A) \neq 0$ – identity matrix $unit(n)$

P43. Shelves from AEKI. A company makes three types of shelves, all of which are made from boards, strips and screws. The stacks of boards (B), strip packs (L) and screw sets (S) of the various shelf types (R1, R2, R3) are interchangeable, the relationship between shelf types and components being given in the following table:

	B	L	S
R1	1	2	4
R2	2	1	1
R3	1	2	1
La	380	170	230

In the warehouse (La) there are 380 stacks of boards, 170 packs of steel strips and 230 sets of screws. Since production is to be discontinued, the current inventory is to be sold out. *How many shelves of the different types should be offered* so that (if possible) no remainder remains in the warehouse? See [2, ?][27]

P44. Functions from conditions. The total costs of a production e.g. of pencils does not depend solely on the number of pencils produced, but we can assume that it may be modulated nevertheless by a third-degree polynomial. For example, the following values have been determined for a production:

production in Mio.	1	3	5	10
costs in US$	27,2	38,4	40	65

a. Argue, why these values results in the following 4×4 - LGS

$$\begin{aligned} a+b+c+d &= 27.2 \\ 27a+9b+3c+d &= 38.4 \\ 125a+25b+5c+d &= 40 \\ 1000a+100b+10c+d &= 65 \end{aligned}$$

b. Write the numbers of this 4×4 –LGS as 4×5 - system matrix and compute the solution with EIGENMATH in as many different ways as possible!

P45. Food requirements for survival training.
The daily food requirement of an adult is 5 g to 6 g of carbohydrates, about 0.9 g of protein and 1 g of fat per kg of body weight. During survival training, three types of A, B, C concentrate food are used.

Concentrate	A	B	C
Protein	5g	10g	7g
Carbohydrates	40g	30g	30g
Fat	5g	10g	13g

Each concentrate cube weighs 50 g and is crushed in water and mixed.
How can an adult (75 kg) use it to meet his daily food requirements (400 g carbohydrates, 70 g protein, 75 g fat)?

[27] Hint: We take the following classical system of equations from the task I: x +2y+4z = 380 II: 2x+1y+1z = 170 III: 1x+2y+1z = 230 ...

P46. Optimal fertilization.
For fertilization experiments, 10 kg of flower fertilizer should be mixed from the three types of fertilizer I, II and III, which contains 40% potassium, 35% nitrogen and 25% phosphorus. (The numbers in the table are in %.) What quantities are required?

	I	II	III
Potassium	40	30	50
Nitrogen	50	20	30
Phosphorus	10	50	20

P47. Handicraft supplies.
A hobby electronics technician buys **T**ransitors, **D**iodes, **C**ondensators and **R**esistors in the specified quantities on consecutive days and pays the specified total **P** price in US $.

	T	D	C	R	P (US $)
1st day	1	3	2	10	17
2nd day	2	7	1	22	32.5
3rd day	3	11	6	37	58.5
4th day	1	3	5	11	22

Determine the individual prices for the electronic components from the information.

P48. Number lotto.
Winning numbers of the number lottery on 1/30/78 was:
8, 9, 18, 29, 38, 46; additional number 32.
With a system tip, player A won one prize class IV and 20 times prize class V. He received a total of 160 $. The table shows the data for a further four players:

player	I	II	III	IV	V	profit ($)
A	-	-	-	1	20	160
B	-	-	2	5	-	6300
C	1	1	8	-	-	414000
D	-	2	-	10	-	80600
E	1	2	10	15	-	460900

How high were the winnings in the individual prize categories at that time?

P49. UVW economy – an Input-output analysis.
A simple economy is divided into three sectors U, V and W, with the following exchange relationships in WE:

	received sectors			
	U	**V**	**W**	*prod*
delivering **U**	2400	1500	600	8000
sectors **V**	1600	2000	600	5000
W	800	5000	1200	3000

a. What is the input matrix? Which final demand results from the given situation? **b.** How does the production level $X(=$ total output) change, if the market is supplied with $Y = [1500, 600, 1200] = [y_U, y_V, y_w]$?

P50. Design of a ski jump.
A ski jump should be built according to the following specifications of the client:
– the hill starts at a height of 100 m and ends at a height of 10 m above the base.
– the horizontal distance from start to finish is 120 m.

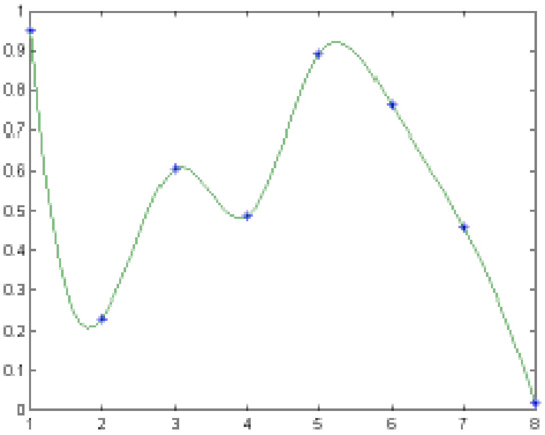
– a ski flyer takes off from a horizontal position and takes off at the end of the ramp table at an angle of 30^o to the horizontal.
○ *What is the cross-section of the ski jump?*

A common way to draw curves of a desired shape is to specify a few points on the path and then find a polynomial whose graph goes through those points.
For example, two given points clearly define a straight line, this is the graph of a polynomial of degree 1. Three non–collinear points generally determine uniquely a parabola, this is the graph of a polynomial of degree 2 (if the points have different x coordinates).
Suppose we are looking for a path through the points $(0; 7), (1; 6), (2; 9)$. So we are looking for the unique parabola with the equation

$$P(x) = ax^2 + bx + c$$

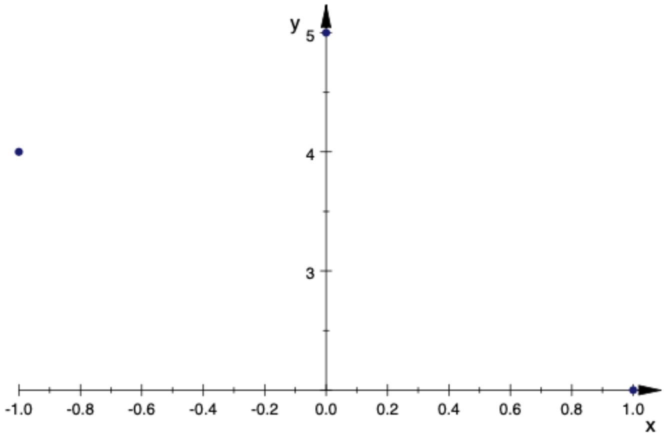
whose graph goes through these three points.
a. What is the associated linear system? Solve it with EIGENMATH.
b. Draw the graph of $P(x) = ax^2 + bx + c$ with the coefficients that were determined in a. and do a graphical test.
c. Calculate the jump with the client's data.
d. Calculate paths for self-selected data sets for cubic ($n = 3$) polynomials and polynomials of 4^{th} degrees. Observations?
e. Get the data for a real ski jump from the Internet: state the source, include photo(s) in your report, . . .

P51. ***Building a ski jump hill with cubic splines.**
As in the previous project, we are looking for a smooth curve that goes through some given data points. The previous project showed that usually curves with high oscillations arise. This is often undesirable.
○ A better technique is to use a *cubic spline*, in which successive points are connected by a cubic (= 3rd degree) polynomial and each interval uses a different cubic polynomial. Here is the graph of a cubic spline through eight data points with the coordinates $x = 1, 2, ...8$:



This spline uses seven cubic polynomials: s_1 is defined on $1 \leq x \leq 2$, s_2 on $2 \leq x \leq 3$, etc. The graph looks smooth because we also require neighboring cubics should have the same 1st and 2nd derivatives. E.g. we demand at the point $x = 2$ that $s_1'(2) = s_2'(2)$ and $s_1''(2) = s_2''(2)$ should apply in addition to $s_1(2) = y_2$ and $s_2(2) = y_2$.

a. First consider the point list $(-1.4), (0.5)$ and (1.2) as a data set:



b. Derive the equations of the cubic splines for these data points. There are 2 cubic polynomials:

$$\begin{aligned} s_1(x) &= a_1x^3 + a_2x^2 + a_3x + a_4, \quad -1 \leq x \leq 0 \\ s_2(x) &= a_5x^3 + a_6x^2 + a_7x + a_8, \quad 0 \leq x \leq 1 \end{aligned}$$

Find 6 equations for the 8 unknowns $a_1, \ldots, a_8$; note:

$$\begin{aligned} s_1(-1) &= 4 \\ s_1(0) &= 5 \\ \cdots &= \ldots \\ s_1''(0) &= s_2''(0) \end{aligned}$$

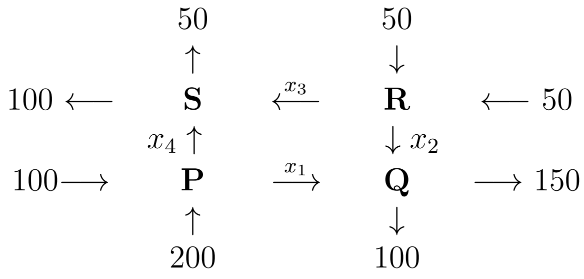
∘ We need *two* further equations for the *unambiguous* solution. An additional condition that is often required is that the second derivatives should be zero at the two end points. What does this mean in our case?

Solve the resulting 8×8 linear system and draw the cubic spline e g. with octave online.

∘ Draw a cubic spline that goes through 8 given points. Choose evenly distributed x coordinates and 8 randomly chosen y coordinates.

∘ Reproduce the figure from the assignment.

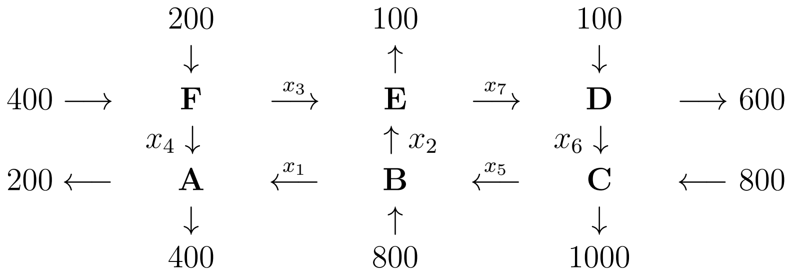
P52. Flow in a net I. The following digraph shows schematically the flow of traffic on four one-way streets in a city. In the case of a traffic count, the average number of vehicles per minute was determined in eight places and entered as data in the plan. The city planners want to determine the traffic densities x_1, x_2, x_3 and x_4.



○ What assumptions are to be made to the model?
○ Support the planners in determining the traffic densities x_1, x_2, x_3 and x_4 in the streets of the quarters.
○ Why can't the traffic flows x_i be negative? What is the minimum value for x_1, i.e. the traffic flow $\overrightarrow{PQ}$ from P to Q?
○ Construction work has to be carried out on the $\overrightarrow{PQ}$ route. Determine the smallest possible flow of traffic that does not lead to traffic jams.
○ For a diversion, the streets between R and Q or S and P will be reoriented. Construction work has to be carried out on the $\overrightarrow{PQ}$ route. Determine the smallest possible flow of traffic between the P and Q intersections.

P53. Flow in a net II.

The following digraph schematically shows the flow of traffic on a section of a one-way network of a city. Repair work is to be carried out on the sections $\overrightarrow{CB}$ and $\overrightarrow{DC}$:



○ What assumptions are to be made to the model?
○ How many vehicles still have to pass the route if there is no traffic jam in the road network?
○ Make a reasoned, planning-based decision on how to handle the traffic situation.

P54. The balance of unchangeable components.

Sulfur dioxide is e.g. used as a bleach for wool or as a preservative for fruit. In the extraction of sulfur dioxide, the ore pyrites (FeS_2) is roasted in an oxygen stream (O_2). This creates sulfur dioxide (SO_2) and gravel burn-off (Fe_2O_3), which can be further reduced to iron.

For a detailed description of this reaction, the question must be answered:

What are the exact proportions of the substances involved in this reaction?

$$a\mathrm{FeS}_2 + b\mathrm{O}_2 \quad \longrightarrow \quad c\mathrm{FeO}_3 + d\mathrm{SO}_2$$

> The mathematical model to answer this question assumes that three elements Fe, S, O do not change in the course of the reaction. The question of the correct proportions also implies that a complete conversion of the starting materials into the end products takes place, i.e. *the composition of each element is the same before and after the chemical reaction*:

The conditions are resumed in the following 3×4 - LGS

$$\begin{array}{rrcl} & \text{before} & = & \text{after} \\ \text{Fe:} & a & = & 2c \\ \text{S:} & 2a & = & d \\ \text{O:} & 2b & = & 3c + 2d \end{array}$$

a. Write the numbers of this 3×4 –LGS as 3×5 - augmented system matrix and compute the solution.

b. Interpret the solution in the chemical context, i.e. find the *smallest integer* solution of the LS and give the reaction equation with it.[28]

○ Problems? Reasons? Suggested solutions?

P55. *... **Exercises from Lay's book.**

Do as many exercises e.g. from [5] as you like. Use EIGENMATH to assist you or control your training. Good luck.

[28] Answer: $4\mathrm{FeS}_2 + 11\mathrm{O}_2 \quad \longrightarrow \quad 2\mathrm{FeO}_3 + 8\mathrm{SO}_2$

6.6.1 *Appendix* I: Source code of `gjBox1.txt`

```
####################
## Gauss Jordan Box1 -- (2020) Dr. W.Lindner, Leichlingen GE
####################

Em(k,i,j) = do( M=unit(n,n), M[j,i]=k, M)

-- Row interchange
Gi(i,j)=do( Gii=unit(n),
           N= Gii[i],
           M= Gii[j],
           Gii[j]=N,
           Gii[i]=M,
           Gii)

----      GAUSS matrix I, elimination downstairs
Gm(k,A) = do( n   = dim(A,1), -- get the dim
              Gmm = unit(n),  -- k th column
              for(i,k+1,n, Gmm[i,k] = - A[i,k]/A[k,k]),
              Gmm )

----       GAUSS matrix II, elimination upstairs
Gu(k,A) = do( n   = dim(A,1), -- get the dim
              Gmm = unit(n),  -- k th column
              for(i,k,1, Gmm[i,k] = - A[i,k]/A[k,k]),
              Gmm )


----    NORMalization of diagonal
Gn(A) = do( n = dim(A,1), -- get the dim
               Gmm = unit(n),  -- k th column
               for(k,1,n, Gmm[k,k] = 1/A[k,k]),
               Gmm )

----     GAUSS elimination
Ge(A) =  do( n=dim(A,1),
              X = zero(dim(A,1),dim(A,1),dim(A,2)),
              X[1]=A,
              for( i,1,n-1, X[i+1]=dot(Gm(i,X[i]),X[i]) ),
              X)

----     JORDAN elimination
Gj(A) =  do( n=dim(A,1),
              X = zero(dim(A,1),dim(A,1),dim(A,2)),
              X[n]=A,
              for( i,n-1,1, X[i]=dot(Gu(i+1,X[i+1]),X[i+1]) ),
              X)

----     Row Reduced Echelon Form
RREF(A)= do( n = dim(A,1),
             U = Ge(A),
             V = Gj(U[n]),
             X=dot(Gn(V[1]),V[1]),
             X)

backSubst(U) = do( n = dim(U,1),
                   Z = zero(2,n),
                   x = Z[1],
                   x[n] = U[n,n+1] / U[n,n],
                   for(i,n-1,1,
                    x[i] = (U[i,n+1] - sum(j,i+1,n, U[i,j]*x[j])) /U[i,i]),
                   x)

----     syntactic sugar
doGm(j, M) = dot(Gm(j, M), M)
doGu(j, M) = dot(Gu(j, M), M)
doGn(M) = dot(Gn( M), M)
doGi(i,j,M) = dot(Gi(i,j,M), M)
doGn(M) = dot(Gn(M), M)

################### End of gjBox1 ###################
```

6.6.2 *Appendix* II: Source code of gjBox.txt

```
###################
## Gauss Jordan Box -- (2020) Dr. W.Lindner, Leichlingen GE
###################

Em(k,i,j) = do( M=unit(n,n), M[j,i]=k, M)

-- Row interchange
Gi(i,j)=do( Gii=unit(n),
           N= Gii[i],
           M= Gii[j],
           Gii[j]=N,
           Gii[i]=M,
           Gii)

----      GAUSS matrix I, elimination downstairs
Gm(k,A) = do( n   = dim(A,1), -- get the dim
              Gmm = unit(n),  -- k th column
              for(i,k+1,n, Gmm[i,k] = - A[i,k]/A[k,k]),
              Gmm )

----       GAUSS matrix II, elimination upstairs
Gu(k,A) = do( n   = dim(A,1), -- get the dim
              Gmm = unit(n),  -- k th column
              for(i,k,1, Gmm[i,k] = - A[i,k]/A[k,k]),
              Gmm )


----    NORMalization of diagonal
Gn(A) = do( n = dim(A,1), -- get the dim
               Gmm = unit(n),  -- k th column
               for(k,1,n, Gmm[k,k] = 1/A[k,k]),
               Gmm )

----     GAUSS elimination
Ge(A) =  do( n=dim(A,1),
              X = zero(dim(A,1),dim(A,1),dim(A,2)),
              X[1]=A,
              for( i,1,n-1, X[i+1]=dot(Gm(i,X[i]),X[i]) ),
              X[n])

----     JORDAN elimination
Gj(A) =  do( n=dim(A,1),
              X = zero(dim(A,1),dim(A,1),dim(A,2)),
              X[n]=A,
              for( i,n-1,1, X[i]=dot(Gu(i+1,X[i+1]),X[i+1]) ),
              X[1])

----     syntactic sugar
doGm(j,M)   = dot(Gm(j,M), M)
doGu(j,M)   = dot(Gu(j,M), M)
doGn(M)     = dot(Gn(M), M)
doGi(i,j,M) = dot(Gi(i,j,M), M)
doGn(M)     = dot(Gn(M), M)

----     Row Reduced Echelon Form
RREF(A) = doGn( Gj( Ge(A) ))

backSubst(U) = do( n = dim(U,1),
                   Z = zero(2,n),
                   x = Z[1],
                   x[n] = U[n,n+1] / U[n,n],
                   for(i,n-1,1,
                    x[i] = (U[i,n+1] - sum(j,i+1,n, U[i,j]*x[j])) /U[i,i]),
                   x)

################### End of gjBox ###################
```

References

[1] Andrecut, M. (2000): *Introductory Numerical Analysis: Lecture Notes.* Parkland: Universal Publishers.

[2] Artmann, B. & Tōrner, G. (1980): *Lineare Algebra. Grund- und Leistungskurs.* Goettingen: Vandenhoeck & Ruprecht.

[3] Baum, M. and others (2000): *Lambacher-Schweizer: Lineare Algebra mit analytischer Geometrie - Grundkurs.* Stuttgart: Klett.

[4] Kroll, W. & Reiffert, H.-P & Vaupel, J. (1997): *Analytische Geometrie/Lineare Algebra. GK/LK.* Bonn: Dümmler.

[5] Lay, D.C. (1994): *Linear Algebra and its Applications.* Reading: Addison-Wesley.

[6] 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.

[7] Ralston, A. & Neill, H. (1997): *algorithms. Teach Yourself.* London: Hodder & Stoughton.

[8] Steeb, W.-H. (1992): *Algorithms and Computatuions with Turbo C.* Mannheim: BI-Wiss.-Verlag.

[9] 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.

[10] Weigt, G. (2020): Eigenmath *online Demo.*
url: `https://georgeweigt.github.io/eigenmath-demo.html`

[11] 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