Winfried Auzinger
Dirk Praetorius

Sommersemester 2018
23. Mai 2018

Übungsaufgaben zur VU Computermathematik
Serie 8

We use the packages `plots`, `LinearAlgebra` and mainly the data types `Vector` and `Matrix`. In the last two exercises we also use the packages `plottools` and `geometry`.

Exercise 8.1: *Es ist ein Kreuz mit dieser Matrix . . .*

Let

$$A = \begin{pmatrix} a & 0 & 0 & 0 & 0 & b \\ 0 & a & 0 & 0 & b & 0 \\ 0 & 0 & a & b & 0 & 0 \\ 0 & 0 & b & a & 0 & 0 \\ 0 & b & 0 & 0 & a & 0 \\ b & 0 & 0 & 0 & 0 & a \end{pmatrix} \in \mathbb{C}^{6\times 6}$$

where a and b are arbitrary complex numbers. This means that the variables `a` and `b` have no values assigned.

a) *Design a procedure which returns the matrix A.*

b) *Compute the symbolic expression for A^{-1} in dependence on* `a` *and* `b`*. In which case (depending on* `a` *and* `b`*) is A invertible?*

c) A real or complex number λ is called an eigenvalue of a matrix A if $A - \lambda I$ is singular.
Determine the eigenvalues of A.

d) Consider the matrix $A - \lambda I$, with all possible values[19] λ from **c)**. By construction, this matrix is singular.
Use `LinearSolve` *to determine the kernel of $A - \lambda I$.*

Remark: An element x from the kernel satisfies $A\,x = \lambda\,x$; it is called an eigenvector of A associated with the eigenvalue λ.

Exercise 8.2: *Pascal, schau owa.*

We consider lower triangular $n\times n$ - matrices of the form

$$P = \begin{pmatrix} 1 & & & & & & \\ 1 & 1 & & & & & \\ 1 & 2 & 1 & & & & \\ 1 & 3 & 3 & 1 & & & \\ 1 & 4 & 6 & 4 & 1 & & \\ 1 & 5 & 10 & 10 & 5 & 1 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

involving a table of binomial coefficients.

a) *Design a procedure which returns the matrix P for given n.*

b) *Experiment with $n = 1, 2, 3, 4, \ldots$ to find out how the inverse P^{-1} looks like.*

Remark: This experiment gives you the straightforward idea about the inverse. The rigorous proof for general n is a different story (one will use induction on n).

c) *Use Maple to verify for $n = 1, 2, 3, 4, \ldots$, that P has only one eigenvalue λ. What is λ ?*

Remark: Cf. **8.1c)**. With sufficient background in linear algebra the answer is simple.

[19] These values λ are expressions in the variables `a` and `b`.

d) Consider the matrix $P - \lambda I$ with the eigenvalue λ from **c)**. By construction, this matrix is singular.
Use `LinearSolve` *to determine its kernel.* (Again, consider $n = 1, 2, 3, 4, \ldots$)

Exercise 8.3: ***Linear mappings and matrix representation.***

Let a family of linear mappings $f = f_{m,n} \colon \mathbb{R}^n \to \mathbb{R}^m$ be given, where $m, n \in \mathbb{N}$ are arbitrary, and where these mappings share a common definition with rational coefficients, e.g.,

$$(f(x))_k := \sum_{j=1}^{n} \frac{j-k}{j+k}\, x_j\,, \quad k = 1 \ldots m,$$

or whatever you may choose for testing.

a) *Design a procedure* `f(x::Vector,m::posint)` *which expects an object* `x` *of type* `Vector` *and a positive integer* `m` *as its arguments and returns the value $f(x)$ in form of a* `Vector` *of dimension* `m`*.*

Remark: `n` is determined by `Dimension(x)`.

The syntax

```
f := proc(x::Vector,m::posint)
```

means that the arguments passed to the procedure must have the corresponding types, otherwise the procedure will exit with an error message (*check this*).

b) *Design a procedure* `fmatrix(f::procedure,m::posint,n::posint)` *which returns the corresponding $m \times n$ coefficient matrix of the linear mapping $f = f_{m,n}$ (see* **a)***) as an object of type* `Matrix`*. Check that a call of* `f` *from gives the same result as the corresponding matrix-vector multiplication.*

Exercise 8.4: ***Cayley, schau owa.***

In linear algebra it is shown that the inverse of a regular $n \times n$-matrix A can be written as [20]

$$A^{-1} = p(A) = c_0 I + c_1 A + \ldots c_{n-1} A^{n-1}$$

where

$$p(\lambda) = c_0 + c_1 \lambda + \ldots + c_{n-1} \lambda^{n-1}$$

is a polynomial of degree $\leq n-1$, with coefficients depending on A. Here we investigate this property for $n = 2$ only.

a) *Define a general* 2×2 *- matrix*

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

using

```
A := Matrix(2,2,symbol='a')
```

and compute the expression for its inverse.

b) *Use the ansatz $p(A) = c_0 I + c_1 A$ for the inverse A^{-1} and find c_0 and c_1 depending on the matrix entries $a_{i,j}$ such that $p(A) = A^{-1}$ indeed holds true* (assuming that A is indeed invertible).

Exercise 8.5: ***Playing Lego with matrices.***

Assume that an $n \times n$-matrix A is given in block structure

$$A = \left(\begin{array}{c|c} A_{1,1} & A_{1,2} \\ \hline A_{2,1} & A_{2,2} \end{array}\right),$$

where $A_{1,1}$ is $k \times k$ and $A_{2,2}$ is $(n-k) \times (n-k)$. In some applications it is advantageous to solve a linear system $A\,x = b$ or compute the inverse A^{-1} by means of 'blockwise Gaussian elimination'. For the inverse this leads to the representation [21]

[20] Keywords: Characteristic polynomial of a matrix, Cayley-Hamilton Theorem. In general, one needs to know all eigenvalues of A to determine the polynomial p.

[21] This is a generalization of the representation for the inverse from **8.4**.

$$A^{-1} = \left(\begin{array}{c|c} A_{1,1}^{-1}\big(I + A_{1,2}\,S_{2,2}^{-1}\,A_{2,1}\,A_{1,1}^{-1}\big) & -A_{1,1}^{-1}\,A_{1,2}\,S_{2,2}^{-1} \\ \hline -S_{2,2}^{-1}\,A_{2,1}\,A_{1,1}^{-1} & S_{2,2}^{-1} \end{array} \right), \qquad \text{(S)}$$

where I is the $k \times k$ identity matrix, and where the $(n-k)\times(n-k)$ - matrix

$$S_{2,2} = A_{2,2} - A_{2,1}\,A_{1,1}^{-1}\,A_{1,2}$$

is called the Schur complement of $A_{1,1}$. Here it is assumed that $A_{1,1}$ and $S_{2,2}$ are invertible. In this way the inverse A^{-1} is expressed in terms of smaller inverses.

a) *Design a procedure which takes a quadratic matrix A and a positive integer k as its arguments and which returns the inverse using formula* (S). *As you see, there are several subexpressions (e.g., $A_{1,1}^{-1}$) which are used several times. Therefore it is reasonable to compute them only once and store them for later use. Assemble all blocks of the inverse* (S) *and build them together.*

Choose an numerical example (with integer data) and test, comparing with the standard Maple inverse from `LinearAlgebra`*.*

(The parameter k determines the blocking; of course k must be smaller than the row dimension of A.)

b) $(*)$ Block elimination:

Design a procedure which uses the same blocking for A and an analogous blocking for a given vector b of dimension n, and which computes $A^{-1}b$ on the basis of (S)*. Avoid the explicit computation of (small) inverses as far as possible and make use of* `LinearSolve` *instead. Test.*

Exercise 8.6: *Ned scho wieda!* 😱 *A two-dimensional quadrature formula.*

Applying 'Quadrature' means numerical approximation of definite integrals. In the denotation of Exercise **7.7**, we consider the integral $I(f)$ of a function $f\colon \Delta_{\text{ref}} \to \mathbb{R}$ over Δ_{ref}. and we want to approximate it by a weighted sum of the form [22]

$$Q(f) = \omega_1\,f(P_1) + \omega_2\,f(P_2) + \omega_3\,f(P_3) + \omega_4\,f(P_{1,2}) + \omega_5\,f(P_{1,3}) + \omega_6\,f(P_{2,3}),$$

where the coefficients ω_j are to be determined. We want to find these coefficients according to the criterion that [23] $Q(p)$ is exact for arbitrary polynomials of degree ≤ 2, i.e., for the polynomials

$$p(x,y) \;=\; 1,\; x,\; y\;, x^2,\; x\,y,\; y^2\,.$$

Hint: The exact integrals of these polynomials over Δ_{ref} are easy to compute using `int`.

a) *Set up and solve a system of linear equations in matrix/vector form, such that its solution delivers the coefficients ω_j.*

Hint: Use exact rational arithmetic and apply `LinearSolve`.

b) $(*)$ *Compute*

$$\iint_\Delta (x+y)^2\,dy\,dx$$

making use of **a)** *and the substitution introduced in Exercise* **7.7**, *over the triangle Δ with vertices $A=(0,0)$, $B=(2,0)$, and $C=(-1,1)$. Check that your result is correct.*

Exercise 8.7: *A project on projectors. Using the package* `plottools`.

The `plottools` package provides representations of various basic geometric objects in form of ready-cooked plot structures.

In this example we use floating point arithmetic.

a) *Choose three linearly independent vectors $a,b,c \in \mathbb{R}^3$, and generate the matrix $P \in \mathbb{R}^{3\times 3}$ which represents the linear projector, in the direction of c, onto the plane E containing $(0,0,0)$ and spanned by $\{a,b\}$.*

Hint: P leaves each point on the plane E invariant. What is $P\,c$? Make a sketch.

[22] Here the P_j denote the vertices of Δ_{ref}, and the $P_{j,k}$ denote the midpoints of its edges.

[23] Here we extend the approach from Exercise **7.7**.

b) *Use the package* `plottools` *to visualize the* `? parallelepiped` *with edges represented by* a, b, *and* c, *and its image under the projector* P.

c) *Same as in* **b)**, *but with another, arbitrarily, chosen parallelepiped.*

d) *Same as in* **b), c)**, *with* c *orthogonal to* a *and* b.

Remark: In this case P is an orthogonal projector onto E; it is symmetric (*check*).

Exercise 8.8: *Wer rotiert hier in seinem Grab? Using the package* `geometry`*.*

Geometrically, a matrix of the form

$$A = \begin{pmatrix} c & -s \\ s & c \end{pmatrix}$$

with $c = \cos\varphi$, $s = \sin\varphi$ represents a rotation with center $(0, 0)$ and rotation angle φ. For a visualization of such a mapping we consider the square S with vertices $(-1, -1)$, $(1, -1)$, $(1, 1)$, $(-1, 1)$.

a) *Take a look at* `? geometry`.

In the following we use the commands `point`, `square`, `ellipse`, `rotation`, and `draw` from this package. The data structures used by `geometry` represent various objects from two-dimensional Euclidean geometry. These are not plot structures (like in `plottools`), but they can be easily plotted using `draw`.

b) *Generate the square* S, *choose a small angle* φ, *and use a loop* ($k = 0, 1, 2, \ldots$) *to*

- *rotate*[24] S *by* $k\,\varphi$;
- *generate a plot of the rotated object and assign this plot structure to the indexed variable* `pic[k]`;
- *use* `plots[display]` *with option* `insequence=true` *to produce an animated plot of* `[pic[0],pic[1],pic[2],...]`.

  Hint: You see the first picture, and using the context menu of the plot window you can start and control the animation.

c) *Repeat* **b)** *for an ellipse instead of a square, or whatever geometry object you may choose.*

Hint: `draw` accepts essentially the same options as `plot`, as for instance `thickness`.

Warning: There seems to be no simple way to copy objects which have been generated using `geometry` (?). Also it appears that when transforming such a geometry object, e.g., using `rotation`, it cannot be ‘overwritten’ but a new one (with another name) has to be generated.

Das wär’s für heute – viel Spaß . . .

[24] This corresponds to application of the mapping represented by A. The command `rotation` does the job.