Winfried Auzinger
Dirk Praetorius

Sommersemester 2017
17./18. Mai 2017

Übungsaufgaben zur VU Computermathematik

Serie 8

We use the package `LinearAlgebra` and mainly the data types `Vector` and `Matrix`. The terminology 'numerical' refers to the fact that a computation with numerical (real) data is to be performed. 'Integer' means that integer data are assumed.

Note that most (not all) of the exercises use numerical floating point computations and could also be realized, e.g., in MATLAB in an analogous way.

Exercise 8.1: ***Simple linear systems. Use of*** `try`***.***

A plane in $\mathbb{R}^3$ is implicitly described by an equation of the form

$$a\,x + b\,y + c\,z = d$$

for given values a, b, c, d. A pair of such equations describes the intersection of two planes, which is (in general) a straight line.

In the following we assume that all coefficients a, b, c, d take integer values.

a) *Design a procedure* `isline` *which takes an integer matrix A of the form*

$$A = \begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \end{pmatrix}$$

as its argument, where each row represents a plane in $\mathbb{R}^3$. Decide whether these planes

1. *are identical,*
2. *are parallel but not identical, or*
3. *intersect along a straight line,*

and return the value 1, 2 or 3. (Alternatively, in case 3. you may return a representation of the intersection line, using `LinearSolve`*).*

b) *Design a procedure* `intersection_point` *which expect two integer matrices (as in* **a)***) as its arguments. Use* **a)** *to check whether these represent straight lines. If so, decide whether these lines have an intersection point, and return the coordinates of this point. Otherwise (which is of course the generic case) exit (e.g.) with an error message.*

Hint: Use `LinearSolve`. If no solution is found, `LinearSolve` exits with an error message. You can control this event by 'catching' it and produce your own error message (or whatever you want to do in this case). To this end, use `? try`:

```
try
  X:=LinearSolve(...,...): # if it is O.K. then it is O.K.
catch:
  # specify what has to be done if try has failed, e.g.
  error("no intersection"):
end try:
```

Remark: In the special case where lines are identical, `LinearSolve` does not fail (check). In this case you may return the answer, but of course it is does not represent a single intersection point. A more special treatment of this case is possible but not compulsory. (Of course you may try it, e.g. using `is(...,numeric)`.)

Exercise 8.2: ***Rotation in 3D.***

Let $k \in \mathbb{R}^3$ be a column vector of Euclidean length 1 and $\varphi \in \mathbb{R}$ represent an angle. Furthermore, let K be the skew-symmetric 3×3 matrix satisfying $K \cdot x = k \times x$ for all x (cross product). The action of the following 3×3-matrix R describes a rotation with angle φ around the rotation axis defined by k. We do not study it theoretically but perform some experiments.

a) *Design a procedure* `rm(k,phi)` *which expects*[12] k *and* φ *as its (numerical) arguments and which returns the rotation matrix ('Rodriguez matrix')*[13]

$$R = I + \sin\varphi\, K + (1 - \cos\varphi)\,(k \cdot k^T - I)$$

without explicitly building the matrix K*. Use a double loop to build* R*.*

b) *Check by experiment that* R^T *is the inverse*[14] *of* R*, which in turn is the same as* R *with* φ *replaced by* $-\varphi$*.*

c) *Design a procedure* `rx` *which expects* k*,* φ *and a vector* x *as its (numerical) arguments and which returns the rotated vector* $R \cdot x$*, without explicitly building the matrix* R*. Use one inner product and one cross product.*

Hint: Realize the matrix-vector-product $(k \cdot k^T) \cdot x$ in a simplified way, avoiding matrix-vector multiplication. What is the geometrical interpretation of this operation?

Exercise 8.3: ***Numerical quadrature: Construction of an approximation formula for an integral.***

Assume that for a real function $f\colon [0,1] \to \mathbb{R}$ you know the $n+1$ values $f(0)$, $f(\frac{1}{n}), f(\frac{2}{n}), \ldots, f(1)$ (for given $n \in \mathbb{N}$). We want to find an approximation formula for

$$\int_0^1 f(x)\,dx$$

of the form

$$\omega_0 f(0) + \omega_1 f(\tfrac{1}{n}) + \omega_2 f(\tfrac{2}{n}) + \ldots + \omega_n f(1)$$

A general principle behind such constructions, ubiquitous in numerical analysis, is to require that the approximation is exact for arbitrary polynomials $f(x)$ of degree $\leq n$. Simplest examples:

$n = 1$: $\omega_0 = \frac{1}{2}$, $\omega_1 = \frac{1}{2}$

$n = 2$: $\omega_0 = \frac{1}{6}$, $\omega_1 = \frac{4}{6}$, $\omega_2 = \frac{1}{6}$

[12] It will be convenient not to require that k has length 1 but to normalize k within the procedure.

[13] Note that $k \cdot k^T$ is a 3×3-matrix of rank 1.

[14] This means that R is orthogonal. (A rotation is an example of an orthogonal mapping.)

a) *This construction principle is equivalent to the requirement that the approximation is exact for all monomials* $f(x) = x^0, x^1, x^2, \ldots, x^n$. *Design a procedure* `quad_coeffs` *which expects* $n \in \mathbb{N}_0$ *as its argument and which returns the coefficients* $\omega_0 \ldots, \omega_n \in \mathbb{Q}$ *in form of a* `Vector`. *For some values of* n, *check that your result is correct.*

Hint: Use `LinearSolve` to compute the exact solution of a linear system with rational data which you can generate automatically.

b) Assume you integrate over $[0, h]$ instead of $[0, 1]$, using $n+1$ function values $f(0),\ f(\frac{h}{n}), f(\frac{2h}{n}), \ldots, f(h)$. Show how to obtain the appropriate values for the coefficients ω_j, assuming you have solved **a)**.

(This is not a programming example but a simple exercise in analysis.)

Exercise 8.4: ***Reconstructing a linear mapping from observations.***

Assume that, for a linear mapping $y = A\,x$ from $\mathbb{R}^n$ to $\mathbb{R}^n$ represented by an (unknown) $n \times n$-matrix A, we know the images $y_j \in \mathbb{R}^n$ of n different vectors $x_j \in \mathbb{R}^n$ under the mapping. The question is how to reconstruct the matrix A from this information.

a) *Choose an example in form of two matrices containing the (numerical) vectors* x_j *and* y_j *as its columns and use* `LinearSolve` *for solving an appropriate matrix equation. Under what condition on the* x_j *or* y_j *is the solution unique?*

b) *Generalize the problem to the case of a linear mapping from* $\mathbb{R}^n$ *to* $\mathbb{R}^m$, *for arbitrary* $m, n \in \mathbb{N}$. *How many observations are required? Solve a problem with* $m > n$ *and another one with* $m < n$.

Exercise 8.5: ***Interpolation.***

The practical approximation of (real) functions $y = f(x)$ by polynomials is often based on *interpolation:* Assume that a set of values $\{(x_j, y_j),\ j = 0 \ldots n\}$ ($n \in \mathbb{N},\ x_j$ distinct) is given. Then the polynomial of degree n satisfying $p(x_j) = y_j,\ j = 0 \ldots n$, is called the corresponding interpolation polynomial. If $y_j = f(x_j)$ for some given function f, then p is an approximation to f.

a) *Design a procedure* `interp(x,y)` *which expects the (numerical) data* x_j *and* y_j *as its arguments (in some appropriate format) and which returns the interpolating polynomial* $p(x)$ *in form of a Maple function.*

Hint: Make the ansatz $p(x) = c_0 + c_1 x + \ldots + c_n x^n$, and use `LinearSolve` to determine the coefficients c_k. (Alternatively, you may also use `? VandermondeMatrix`.)

b) *Choose an example (*$f = \sin, \cos, \exp$*, or whatever), degree* $n = 10$*, and compute the interpolating polynomial* p *for the data* $\{(x_j, y_j = f(x_j)),\ j = 0 \ldots n\}$, *where the* x_j *are equidistant points in* $[0, 1]$ *(with* $x_0 = 0$ *and* $x_n = 1$*).*

In order to graphically explore the approximation quality, i.e., the behavior of the error $p - f$ *and its its derivatives* $p' - f',\ p'' - f'', \ldots,$ *plot* p, f *together in one graph over* $[0, 1]$ *and do the same for* p', f', p'', f''.

Exercise 8.6: ***A simple block system.***

Assume that a (numerically given) $mn \times mn$ - matrix M has the bidiagonal block structure

$$M = \begin{pmatrix} A_1 & B_1 & & & & \\ & A_2 & B_2 & & & \\ & & A_3 & B_3 & & \\ & & & \ddots & \ddots & \\ & & & & A_{n-1} & B_{n-1} \\ & & & & & A_n \end{pmatrix}$$

with subblocks A_j, B_j of dimension $m \times m$. We assume that the diagonal blocks A_j are invertible.

a) *Choose a representation of M where only the A_j and B_j are stored in some appropriate format but not M as a full* `Matrix`.

b) *Choose an example, and also vector $b \in \mathbb{R}^{mn}$, and solve the linear system of equations $M\,x = b$ via a* `do` *- loop and applying* `LinearSolve` *to an $m \times m$ - subsystem in each iteration step.*

Exercise 8.7: ***The inverse of a block-bidiagonal matrix.***

Compute the inverse of your matrix M from Exercise 8 b) using your algorithm and store it as an object of type `Matrix`. *What is its shape?*

Remark: The result shows that solving $M\,x = b$ via multiplication of M^{-1} by b would be rather stupid.

Exercise 8.8: ***A special linear solution procedure.***

We consider $n \times n$ - matrices A of the form

$$A = Q\,\Lambda\,Q^T$$

where Q is orthogonal, i.e., $Q^T Q = I$, and where Λ is diagonal. Thus, A is uniquely specified if Q and Λ are given. [15]

a) *Design a (numerical) procedure which expects Q, a vector λ representing the diagonal of Λ, and a vector $x \in \mathbb{R}^n$ as its arguments and which returns the vector $A\,x$, without computing the matrix A. Do not use matrix-matrix multiplication.*

Addendum: Choose one of the q_j and compute $A\,q_j - \lambda_j\,q_j$. What do you observe? Also prove that what you observe is true for all j (this is very simple).

b) We now also assume $\lambda_j \neq 0$, $j = 1 \ldots n$.

Design a (numerical) procedure which expects Q, a vector λ representing the diagonal of Λ, and a vector $b \in \mathbb{R}^n$ as its arguments and which returns the solution x of the linear system $A\,x = b$, without computing the matrix A. Do not use matrix-matrix multiplication.

Note that, in general, the solution of a linear system $A\,x = b$ requires $O(n^3)$ operations. In the case considered here, only $O(n^2)$ operations are required. *When presenting your example, provide an explanation of this fact.*

Remark: Note that a matrix A of the form considered here is symmetric. In linear algebra it is shown that, conversely, each symmetric matrix can be written in such a form. The λ_j are the *eigenvalues* of A and the columns q_j of Q are the corresponding *eigenvectors*, forming an orthonormal basis of $\mathbb{R}^n$. However, in the general case (where A is given but Q and Λ are not known a priori), first solving the eigenproblem, i.e., computing Q and Λ, and then proceeding as in **a)** is not an efficient algorithm in general.

[15] Such a situation sometimes occurs, e.g., in the context of the numerical solution of differential equations.