Winfried Auzinger
Dirk Praetorius

Sommersemester 2014
21.–23. Mai 2014

Übungsaufgaben zur VU Computermathematik
Serie 8

In all examples we use the package `LinearAlgebra`*.*

Exercise 8.1. We design a procedure for computing the orthogonal projection onto a linear subspace $\mathcal{U} \subseteq \mathbb{R}^n$. Let $m < n$ linearly independent vectors $(u_1, \ldots, u_m)$ be given. We apply `GramSchmidt` to compute an orthonormal basis $(q_1, \ldots, q_m)$ which spans the same linear space $\mathcal{U}$ as $(u_1, \ldots, u_m)$.

Then, the orthogonal projection of $x \in \mathbb{R}^n$ onto $\mathcal{U}$ is given by

$$P\,x = \sum_{j=1}^{m} (x \cdot q_j)\, q_j\,,$$

where $x \cdot y = x^T y = y^T x$.

The following makes only sense in floating point:

a) Design a procedure `mgs(U::matrix)` which expects a matrix `U` as its argument and which calls `GramSchmidt` to orthonormalize the columns of `U`. The procedure returns a matrix `Q` with the corresponding orthonormal columns.

b) Design a procedure `orthoproj(x::Vector,Q::{Vector,Matrix})` which expects a vector `x` and a matrix `Q` (according to **a)**) or a single vector[1] `Q` as its arguments and returns the orthogonal projection $P\,x$. If dimensions are incompatible, stop with an error message.

Remark on the mathematical background: The projection $P\,x$ is the best approximation of x within $\mathcal{U}$, i.e., $\|u - x\|_2$ becomes minimal for $u = P\,x$. A matrix representation of the projector is given by $P = Q\,Q^T$ satisfying $PP = P$ (projector property) and $P = P^T$ (this characterizes orthogonal projectors). Can you verify the latter properties? You may also 'verify' them experimentally.

Exercise 8.2. Let $\mathcal{U}$ be a linear subspace of $\mathbb{R}^3$ of dimension 2 (i.e., a plane containing 0). We wish to determine the matrix representation of the projector P which projects $x \in \mathbb{R}^3$ onto $\mathcal{U}$ in direction of a given vector $0 \neq w \notin \mathcal{U}$. P is uniquely determined by the requirements (make a sketch)

$$P\,u = u, \quad P\,v = v, \quad P\,w = 0,$$

where $u, v \in \mathcal{U}$ are any linearly independent vectors spanning $\mathcal{U}$.

Design a procedure

```
screw_projector(u::Vector,v::Vector,w::Vector)
```

which returns the matrix P in form of an object of type `Matrix`. Use `LinearSolve` to solve the corresponding matrix equation. What happens if $w \in \mathcal{U}$ or if u, v are linearly dependent?

Remark: If $w \perp \mathcal{U}$, then the outcome is the orthogonal projector onto $\mathcal{U}$.

---

[1] This makes sense for the special case $m = 1$. If `Q` is a vector, orthonormalize it.

Exercise 8.3. Let $U \in \mathbb{R}^{n\times m}$ and $Q \in \mathbb{R}^{n\times n}$ be matrices in the sense of **8.1**, with $n > m$. The function `QRDecomposition` also converts U into Q: It returns the matrix Q together with a matrix $R \in \mathbb{R}^{m\times m}$ such that[2] $U = Q\,R$, with R upper triangular.

a) Consider the problem of finding $x \in \mathbb{R}^m$ such that the residual norm $\|U\,x - b\|_2$ becomes minimal for given $U \in \mathbb{R}^{n\times m}$ and $b \in \mathbb{R}^n$. The solution x can be obtained by solving the *normal equations*

$$U^T(U\,x) = U^T b\,,$$

which, due to $Q^TQ = I$, is equivalent to

$$R^T(R\,x) = R^T(Q^T b)\,.$$

Design a procedure `leastsquares(U,b)` which returns the solution x. Use `QRDecomposition` and `LinearSolve`, *without* explicitly computing R^TR or R^TQ, which would be inefficient.

b) Solve a simple problem for some matrix $U \in \mathbb{R}^{3\times 2}$ and some vector $b \in \mathbb{R}^3$. Compare your results with the outcome of `LeastSquares`, which does the same job.

Exercise 8.4. A differentiable vector field $S:\ \mathbb{R}^2 \supseteq G \to \mathbb{R}^3$ defines a parametrization of a smooth surface $\mathcal{S} \subseteq \mathbb{R}^3$. The points on the surface are given by $(x,y,z) = S(u,v) = (S_1(u,v), S_2(u,v), S_3(u,v))$ with $(u,v) \in G$.

Let $G = [a,b] \times [c,d]$ be a rectangle in (u,v) - plane. Then, the area of the surface is given by the double integral

$$\int_{\mathcal{S}} d\sigma := \int_{v=c}^{d}\int_{u=a}^{b} \sqrt{\mu(u,v)}\; du\,dv, \qquad \text{with} \quad \mu(u,v) := \det\begin{pmatrix} \frac{\partial S}{\partial u}\cdot\frac{\partial S}{\partial u} & \frac{\partial S}{\partial u}\cdot\frac{\partial S}{\partial v} \\ \frac{\partial S}{\partial u}\cdot\frac{\partial S}{\partial v} & \frac{\partial S}{\partial v}\cdot\frac{\partial S}{\partial v} \end{pmatrix}.$$

Here,

$$\frac{\partial S}{\partial u} = \begin{pmatrix} \frac{\partial S_1}{\partial u} \\ \frac{\partial S_2}{\partial u} \\ \frac{\partial S_3}{\partial u} \end{pmatrix}$$

(analogously for v), and $\cdot$ is the dot product (= Euclidean inner product).

Use this formula in Maple to compute the area of a section of a hyperboloid in $\mathbb{R}^3$, given by

$$S(x,y) = \begin{pmatrix} x \\ y \\ x\,y \end{pmatrix}, \quad (x,y) \in [-1,1]\times[-1,1]\,.$$

This means that the hyperboloid is parametrized by the Cartesian coordinates in the (x,y) - plane, i.e., it is simply represented as the graph of the function $z(x,y) = x\,y$, and x, y play the role of u, v. (You may use `plot3d` to visualize this surface.)

This integral is nontrivial. Use `evalf` to find: area ≈ 5.123.

Exercise 8.5. Let a family of linear mappings $\psi = \psi_{m,n}:\ \mathbb{R}^n \to \mathbb{R}^m$ be given, where $m, n \in \mathbb{N}$ is arbitrary, and where these mappings share a common definition, e.g.,

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

or whatever you may choose for testing.

[2] This is equivalent to Gram-Schmidt applied to the columns of U. The upper triangular matrix R contains the coefficients of the representation of the columns of U in terms of the columns of Q.

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

Remark: `n` is determined from `Dimension(x)`. The syntax

```
psi := 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 (try out).

b) Design a procedure `psimatrix(psi::procedure,m::posint,n::posint)` which returns the corresponding $m \times n$ coefficient matrix of the mapping $\psi_{m,n}$ as an object of type `Matrix`. Check that a call of `psi` gives the same result as the corresponding matrix-vector multiplication.

Exercise 8.6. An $n \times n$ matrix $H = (h_{i,j})$ is called *upper Hessenberg* if $h_{i,j} = 0$ for $j < i - 1$.

a) Design a recursive procedure which computes the determinant of an upper Hessenberg matrix:[3] By expanding the determinant along the first column (Laplace expansion theorem), evaluation of the determinant for dimension n is reduced to 2 evaluations of the determinants of submatrices of dimension $n-1$ which are also upper Hessenberg. (Make a sketch.)

Choose an example and compare with `Determinant(...)`. Also use `time()` to observe computing times for $n = 10, 20, 30$. What do you observe? Explain the effect. How many recursive calls are performed?

Hint: For extracting a submatrix you may use vector index notation using index lists. For instance, `H[[1,3..n],[2..n]]` removes the second row and the first column. This can also be written as `H[[1,3..n],2..n]`.

b) The algorithm from **a)** is a nice exercise but it is stupid. Write another one: Let H_k = `H[k,..]` denote the k-th row of H.

- Replace H_2 by a linear combination of H_1 and H_2 such that the new H_2 satisfies $H_{2,1} = 0$, i.e., set `H[2,..] := H[2,..]` + α `H[1,..]` with the appropriate value for α.
- Replace H_3 by a linear combination of (the new) H_2 and H_3 such that the new H_3 satisfies $H_{3,2} = 0$.
- ... (As you know, the determinant is invariant under these operations.)
- After $n-1$ steps, return the determinant of the resulting modified matrix H.

You may assume that no division by zero occurs; otherwise the algorithm would have to be modified. But insert an `error` branch which monitors this case. Repeat the tests from **a)**.

Exercise 8.7.

a) With `plots[arrow]` you can draw arrows. Use this to visualize the behavior of a linear mapping $\psi\colon \mathbb{R}^3 \to \mathbb{R}^3$ represented by a coefficient matrix A, by drawing the parallelepiped spanned by the image of the unit vectors $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ under the mapping. Produce a nice plot.

---

[3] We do not need this procedure because Maple readily can compute determinants. However, it may be necessary to do this in some other programming language.

b) Another visualization is provided by the image of the unit sphere under the mapping. To this end, use spherical coordinates

$$x = \cos\theta\,\cos\varphi,$$
$$y = \cos\theta\,\sin\varphi,$$
$$z = \sin\theta,$$

with $\varphi \in [0, 2\,\pi]$ and $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and use the `plot3d` syntax for parametric surfaces. (See `?plot3d`, 'Plotting a parametric surface'.)

Produce a nice plot. Also use `display[3d]` to combine this with a plot of the unit sphere. Use different colors and set the option `transparency=0.5`.

Hint: With `convert(...,list)` you can convert a `Vector` into a list.

Exercise 8.8. For a square matrix A, the *matrix exponential* is defined as the convergent power series $e^A := \sum_{k=0}^{\infty} A^k/k!$.

a) Use[4] $r(t) =$ `numapprox[pade](exp(t),t,[3,3])` to approximate e^A by $r(A)$. With $r(t) = p(t)/q(t)$ and the matrix polynomials $p(A)$ and $q(A)$ this amounts to solving the linear matrix equation (use `LinearSolve`)

$$q(A) \cdot X = p(A) \quad \Rightarrow \quad X = r(A)\,.$$

To evaluate $p(A)$ and $q(A)$, use a `for` loop realizing the so-called *Horner scheme,*[5]

$$p(A) = c_0 + A \cdot \big(c_1 + A \cdot \big(c_2 + \ldots + A \cdot (c_{n-1} + A \cdot c_n)\big)\big)$$

Implement this in form of a procedure `ratexp(A::Matrix)`. Use $A =$ `evalf(HilbertMatrix(10))` for testing. Compare with `MatrixExponential(A)`.

b) If A is 'large', the approximation quality may be rather bad. Due to $e^A = e^{A/2+A/2} = \big(e^{A/2}\big)^2$ we may use the (better) approximation $r(A/2)^2$. More generally, one may use $r(A/n)^n$ with $n \in \mathbb{N}$. This is called *scaling and squaring.* Modify your procedure from **a)** to include the parameter k such that scaling and squaring is performed with $n = 2^k$. Compute the n-th matrix power in an efficient way.

For the test example from **a)**, determine experimentally the smallest k such that

| `ratexp(A,k) - MatrixExponential(A)` | $< 10^{-10}$.

Here | `B` | = `max(abs(B))` denotes the size of the largest element in `B` (by absolute value).

---

[4] `pade` delivers a rational approximation, a sp-called *Padé approximation.* This is a rational analog of a Taylor polynomial.

[5] for $p(t) = c_0 + c_1\,t + \ldots + c_n\,t^n$