Winfried Auzinger
Dirk Praetorius

SS 2012
(22. Mai 2012)

Übungsaufgaben zur VU Computermathematik

Serie 9

Vorbemerkung: `with(LinearAlgebra);` bauen Sie naheliegende Fehlerabfragen und `error ...` ein (z.B. Abfragen auf kompatible Dimensionen).

Alle Vektoren sind als Spaltenvektoren zu denken. Für einen Vektor u ist dann u^T ein Zeilenvektor. $u^T v = v^T u \in \mathbb{R}$ ist das innere Produkt (Skalarprodukt) von u und v (reeller Fall). $u\,v^T$ ist eine Matrix.

Manche der Aufgaben beziehen sich auf numerische Algorithmen und sind sinnvoll nur ein Gleitpunktarithmetik ausführbar. Beachten Sie die Hinweise. Sie können auch `hfloat`'s verwenden, aber das ist für die Aufgaben nicht wesentlich.

Aufgabe 9.1*.

Folgendes ist für reelle `float`-Daten gedacht (abgesehen von Trivialfällen, bzw. sehr niedriger Dimension):

Wie konstruiert man eine Orthogonalprojektion auf einen Unterraum des $\mathbb{R}^n$? – Geben Sie eine Matrix $B \in \mathbb{R}^{n\times m}$ vor ($m < n$). Erstellen Sie eine Hilfsprozedur `MGramSchmidt(B)`, die die Spalten b_i einer derartigen Matrix B mittels `LinearAlgebra[GramSchmidt]` orthonormalisiert und das entstehende orthonormale Vektorsystem $\{v_i\}$ wieder spaltenweise in einer Matrix $V \in \mathbb{R}^{n\times m}$ zurückliefert.[1] Die orthonormalisierten Vektoren v_i bilden dann eine Orthonormalbasis in dem von den b_i aufgespannten m-dimensionalen Unterraumm U, und für einen gegebenen Vektor $x \in \mathbb{R}^n$ ist die Orthogonalprojektion auf U gegeben durch

$$P\,x := \sum_{i=1}^{m} (x^T v_i)\, v_i\,.$$

Erstellen Sie eine Prozedur `orthproj(x,B)`, die `P*x` zurückliefert.

Anmerkung: In Matrix-Terminologie kann das ganze so ausdrücken: $U = \text{Bild}(B)$ ist der Bildraum der durch B definierten repräsentierten linearen Abbildung, und es gilt auch $U = \text{Bild}(V)$. Weiters gilt $V^T V = I_{m\times m}$ (Orthonormalität der v_i), und die Matrix $P := V\,V^T \in \mathbb{R}^{n\times n}$ repräsentiert den Orthogonalprojektor auf U: Es gilt nämlich $P\,x = V\,V^T x$ (überlegen Sie, warum).

Achtung: `GramSchmidt` normiert die orthogonalen Vektoren nicht automatisch auf Länge 1. Dafür ist die option `normalized` anzugeben.

Aufgabe 9.2*. Gegeben sei eine rationale Funktion $r(x) = p(x)/q(x)$ (p und q sind Polynome), und eine quadratische Matrix $A \in \mathbb{R}^{n\times n}$. Falls das Matrixpolynom $q(A)$ invertierbar ist, dann nennt man $r(A) = q^{-1}(A)\,p(A)$ die zugehörige rationale Matrixfunktion.

Die rationale Funktion $r(x)$ sei als Prozedur gegeben z.B. `r:=x->x^2/(1-x^3)`. Erstellen Sie eine Prozedur `rmatfun(r,A)`, die die Matrix $r(A)$ zurückliefert.

Hinweis: Falls $r(x)$ wie in obigem Beispiel angegeben wird, liefert die Auswertung `r(A)` eine Fehlermeldung. Überlegen Sie, wie Sie es angehen, entweder indem Sie `r` in anderer, mit Matrixarithmetik kompatibler Weise spezifizieren, oder indem Sie auf Zähler- und Nennerpolynom zurückgreifen (`numer, denom`).

[1] Anmerkung: Dies funktioniert nur, falls die b_i linear unabhängig sind. Auch bei 'fast linear abhängigen', z.B. annähernd parallelen b_i wird die numerische Rechnung unter Umständen sehr ungenau aufgrund von Rundungsfehlereffekten. Vermeiden Sie beim Test solche Fälle.

Aufgabe 9.3*. Fortsetzung von Aufgabe 9.2:

Erstellen Sie eine Prozedur `rmatfunv(r,A,v)`, die, zu einem zusätzlichen Vektorargument `v` den Vektor $r(A)v$ berechnet, ohne die Matrix `r(A)` explizit zu generieren. Lösen Sie zu diesm Zweck ein lineares Gleichungssystem mit Hilfe von `LinearSolve`. Dies funktioniert aber nur, wenn `r` so spezifiziert ist, dass Sie Zähler- und Nennerpolynom identifizieren können.

Sie können die beiden Aufgaben auch kombinieren, d.h. nur eine einzige Prozedur schreiben, indem Sie `v` als optionalen Parameter ansehen (vgl. dazu Aufgabe 8.7).

Aufgabe 9.4*. Zu einer gegebenen Matrix $A \in \mathbb{R}^{m\times n}$ mit $m > n$ (mehr Zeilen als Spalten) nennt man $A^+ := (A^T A)^{-1} A^T \in \mathbb{R}^{n\times m}$ die *Pesudoinverse* von A. Diese ist wohldefiniert, falls A vollen Rang hat.

Erstellen Sie eine Prozedur `pinv(A,v)`, die `v` als optionales Argument betrachtet und entweder A^+ oder A^+v zurückgibt. Falls `v` angegeben wird, berechnen Sie A^+ nicht explizit, sondern verwenden `LinearSolve`. Seien Sie tolerant und akzeptieren Sie auch ein Argument `v`, das in Form eines Zeilenvektors übergeben wird.

Anmerkung: $x = A^+v$ ist die Lösung der *Gauß'schen Normalgleichungen*, das ist der Vektor mit minimalem Residuum $\|A\,x - v\|_2$.

Aufgabe 9.5*. Für die Untersuchung von Matrizen auf ihre Eigenschaften können auch grafische Hilfsmittel nützlich sein. Schauen sie in die Hilfe zu `plots[sparsematrixplot]` und probieren Sie es an einem Beispiel aus. Definieren Sie weiters eine Matrix $A \in \mathbb{R}^{n\times n}$ durch

$$a_{i,j} = \begin{cases} 2\,h, & i = j \\ -h, & |i-j| = 1 \\ 0, & \text{sonst} \end{cases}$$

mit $h = 1/(n-1)$. Generieren Sie die Matrix, z.B. für $n = 10$, und visualisieren sie sie ihre Struktur mittels `plots[sparsematrixplot]` und die Gestalt ihrer Inversen mittels `plots[matrixplot]`. Sieht nett aus. Was passiert, wenn Sie die Diagonalelemente auf einen Wert > 2 verändern?

Anmerkung: A ist eine Tridiagonalmatrix. Für hohe Dimensionen würde man sie daher mittels

```
A:=Matrix(n,n,shape=symmetric,storage=band[1])
```

initialisieren, um Speicherplatz zu sparen. Die Inverse A^{-1} ist allerdings voll besetzt (keinerlei Null-Einträge).

Aufgabe 9.6. Die Funktion `JordanForm` bestimmt die Jordan'sche Normalform einer Matrix A, $A = Q\,J\,Q^{-1}$.

```
JordanForm(A)
```

ist äquivalent zu

```
JordanForm(A,output='J')
```

und liefert nur J. Mit

```
JordanForm(A,output=['J','Q'])
```

erhält man auch die Transformation Q.

Wählen Sie als Beispiel

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

und bestimmen Sie die Jordan-Form. Was passiert, wenn Sie im Ergebnis $\varepsilon = 0$ setzen? (Vergleichen Sie mit der Jordan-Form der Einheitsmatrix; diese ist trivial.)

Folgerung aus der Beobachtung: Für sehr kleine Werte von ε ist die Jordan'sche Normalform in diesem Beispiel zwar korrekt, aber ziemlich verrückt. Schauen Sie sich Zahlenwerte an. Probieren Sie auch

$$A = \begin{bmatrix} 1+\delta & \epsilon \\ 0 & 1 \end{bmatrix}, \quad \epsilon, \delta \text{ klein.}$$

'Verrückt' bedeutet: Die Jordan'sche Normalform ist nicht durchwegs eine stetige Funktion der Daten. Daher weigert sich Maple auch, diese für `float`-Daten zu berechnen (ausprobieren). Die Jordan-Form hat keine Bedeutung für allgemeine numerische Algorithmen.

Aufgabe 9.7. Sei $A \in \mathbb{R}^{n\times n}$ invertierbar und $U, V \in \mathbb{R}^{n\times k}$ gegeben. Dann gilt die *Sherman-Morrison-Woodbury*-Formel

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1}U)^{-1}V^T A^{-1}$$

in dem Sinn, dass $A + UV^T \in \mathbb{R}^{n\times n}$ genau dann invertierbar ist wenn $I + V^T A^{-1}U \in \mathbb{R}^{k\times k}$ invertierbar ist, und in diesem Fall gilt die behauptete Identität. (Die Richtigkeit dieser Formel ist direkt überprüfbar, ein bisschen Rechenarbeit).

Dies kann man z.B. dazu verwenden, um für bereits berechnetes A^{-1} die Inverse $(A + UV^T)^{-1}$ zu bestimmen, wobei man für $k < n$ nur zusätzlich die 'kleinere' Inverse $(I + V^T A^{-1}U)^{-1} \in \mathbb{R}^{k\times k}$ benötigt (abgesehen von den zusätzlich erforderlichen Matrixmultiplikationen).

Implementieren sie diese Art der Berechnung von $(A + UV^T)^{-1}$ als Prozedur

`SMW_Inverse(AI::Matrix,U,V::{Matrix,Vector[column]})`

Im Argument `AI` wird A^{-1} übergeben. Für U und V sollen im Spezialfall $k = 1$ auch Spaltenvektoren zulässig sein.

Aufgabe 9.8. Gegeben sei eine vektorwertige Funktion $f : \mathbb{R} \to \mathbb{R}^n$, d.h. `f` liefert zu jedem $t \in \mathbb{R}$ ein Objekt vom Typ `Vector` der Dimension n. Die Komponenten seien differenzierbare Funktionen; dann beschreibt die Menge $\{f(t) : t \in \mathbb{R}\}$ eine glatte Kurve im $\mathbb{R}^n$.

Erstellen Sie eine Prozedur `bogenlaenge(f,a,b)`, die die Boegenlänge eines Kurvenabschnittes C von ($t = a$ bis $t = b$) zurückgibt. Diese ist gegeben durch

$$\int_C ds := \int_a^b \|f'(t)\|_2 \, dt \,.$$

(Dabei repräsentiert – für $f(t) = (f_1(t), \ldots, f_n(t))$ – die Funktion $f'(t) = (f_1'(t), \ldots, f_n'(t))$ den Tangentialvektor an die Kurve.)

Beispiel ($n = 2$): Logarithmische Spirale $e^{-k^2\varphi}(\cos(\alpha\varphi), \sin(\alpha\varphi))$. Die Bogenlänge von $\varphi = 0$ bis $\varphi = \infty$ ist $1/k$. Wählen Sie auch ein Beispiel mit $n = 3$.

Aufgabe 9.9. In der Maple - Online-Hilfe findet man unter `index[packages]` eine Übersicht uber die in der Library vorhandenen Packages. Man gebe einen kurzen Überblick (Auswahl aus individueller Sicht), mit dem einen oder anderen Beispiel. Von allgemeinem Interesse sind z.B. die Packages `CodeGeneration, CurveFitting, geometry, Statistics, Student, VectorCalculus`, aber es gibt noch viel mehr Interessantes. Wählen Sie nach Geschmack.

Aufgabe 9.10. [Für `float`-Daten:]

Unter einer *data sparse* Darstellung einer linearen Abbildung bzw. der zugehörigen Koeffizientenmatrix A versteht man eine Darstellung, die die Abbildung repräsentiert, ohne dass A explizit gebildet wird. Dies funktioniert für gewisse Klassen von Abbildungen. Beispiel:

$$A = \prod_{k=1}^{m} (I - u_k\, v_k^T), \quad u_k, v_k \in \mathbb{R}^n\,.$$

Hier ist $u_k\, v_k^T \in \mathbb{R}^{n\times n}$ (Spaltenvektor mal Zeilenvektor), und die u_k, v_k enthalten die volle Information über die Abbildung A. Schreiben Sie eine Prozedur `mvmul(U,V,x)`, die die Multiplikation $A\,x$ ausführt, ohne die Matrix A explizit auszurechnen. Dabei entsprechen die Spalten von U und V den u_k und v_k. Ihre Prozedur soll *keinerlei Matrix-Vektor-Multiplikationen* verwenden.

Hinweis: Das Produkt ist wie folgt zu lesen (Notationskonvention; beachten Sie, dass Matrixmultiplikation nicht kommutativ ist):

$$\Big(\prod_{k=1}^{m} (I - u_k\, v_k^T)\Big) x = (I - u_m\, v_m^T) \cdots (I - u_1\, v_1^T)\, x$$

Beachten Sie: $(uv^T)\, x = u\,(v^T x) = (v^T x)\, u$, d.h. dafür braucht man nur ein inneres Produkt auszuwerten und den Wert mit u multiplizieren.

Diese Vorgangsweise ist insbesonderte dann rechenzeit- und speichereffizient, wenn n sehr groß ist, aber $m \ll n$.