Winfried Auzinger
Klaus Felsenstein
Dirk Praetorius

SS 2011
(14. Mai 2010)

Übungsaufgaben zur VL Computermathematik

Serie 10

Aufgabe 10.1*. Im Paket `LinearAlgebra` gibt es eine Funktion, die das charakteristische Polynom $p(z) := \det(zI - A)$ einer quadratischen Matrix $A \in \mathbb{R}^{n\times n}$ erzeugt (I = Einheitsmatrix). Verwenden Sie dies, um anhand einiger Beispiele den *Satz von Cayley-Hamilton* experimentell zu 'verifizieren': $p(A) = 0$ (Nullmatrix). (Was mit $p(A)$ gemeint ist, sollte klar sein – ein 'Matrixpolynom'.)

Erstellen Sie sodann eine Prozedur `CHsolve(A::Matrix,b::Vector)`, die die Lösung $x = A^{-1}b$ eines linearen Gleichungssystems $A\,x = b$ unter der Ausnutzung der Identität $p(A) = 0$ berechnet und zurückgibt (A wird als invertierbar angenommen). `CHsolve` soll, unter Zuhilfenahme des charakteristischen Polynoms von A, intern nur Matrix-Vektor-Multiplikationen $y \mapsto A\,y$ und Vektoradditionen, aber keine Matrixmultiplikationen ausführen. (Überlegen Sie selbst, wie das zu realisieren ist.)

Anmerkung: Dies hat eher theoretische Bedeutung; in der hier beschriebenen Form handelt es sich nicht um einen praktisch sinnvoll einsetzbaren Algorithmus.

Aufgabe 10.2*.

– Erstellen Sie eine Prozedur `checkproj(P::Matrix)`, die überprüft, ob eine gegebene rationale $n \times n$-Matrix P eine *[Orthogonal]-Projektion* auf einen Unterraum U des $\mathbb{R}^n$ repräsentiert. Dies ist genau dann der Fall wenn gilt $P^2 = P\,[= P^T]$, wobei die Symmetrieeigenschaft $P = P^T$ die Orthogonalität des Projektors charakterisiert. (Die Matrix $Q := I - P$ ist dann auch ein [Orthogonal]-Projektor; P und Q projizieren komplementär zueinander.) `checkproj` gibt die betreffende Information in geeigneter Weise zurück (z.B. 0 = kein Projektor, 1 = Projektor (jedoch 'schief'), 2 = Orthogonalprojektor). Falls P nicht quadratisch ist, Abbruch mit `error`.

– Wie konstruiert man einen Orthogonalprojektor auf einen Unterraum? – Geben Sie irgendeine rationale Matrix $B \in \mathbb{R}^{n\times m}$ vor ($m < n$), B soll vollen Rang m haben (`?LinearAlgebra[Rank]`). Erstellen Sie eine Hilfsprozedur `MGramSchmidt(B)`, die die Spalten einer derartigen Matrix B mittels `?LinearAlgebra[GramSchmidt]` orthonormalisiert und das entstehende orthonormale Vektorsystem wieder spaltenweise in einer Matrix $V \in \mathbb{R}^{n\times m}$ zurückliefert. Dann repräsentiert V eine Basis des Bildraumes U von B, und $P := V{\cdot}V^T$ ist der Orthogonalprojektor auf U. Dies ist leicht zu beweisen [1] – überprüfen Sie es auf jeden Fall mittels ihrer Prozedur `checkproj`.

Bei dieser Aufgabe verwenden wir nur rationale Daten und exakte Arithmetik, also keine Berücksichtigung von Rundungseffekten.

Aufgabe 10.3*. Eine *Givens-Rotation* um den Winkel φ in einem 2-dimensionalen Unterraum des $\mathbb{R}^n$ ist eine lineare Abbildung $G = G(i,j,\varphi) : \mathbb{R}^n \to \mathbb{R}^n$, definiert wie folgt ($i,j \in \{1,\ldots,n\}$, $i \neq j$):

- Für $k \neq i$, $k \neq j$ ist $(Gx)_k = x_k$.
- Jede Vektorkomponente $(x_i, x_j)^T$ in dem vom i-ten und j-ten kartesischen Einheitsvektor aufgespannten 2-dimensionalen Unterraum des $\mathbb{R}^n$ wird in diesem Unterraum um den Winkel φ verdreht (geometrisch gedeutet entgegen dem Uhrzeigersinn).

Schreiben Sie eine Maple-Prozedur

`mgivens(x::Vector,i::list,j::list,phi::list)`,

die für gegebenes $x \in \mathbb{R}^n$ eine Folge von Givens-Rotationen ausführt, d.h. zu berechnen ist der Ergebnisvektor

$$G(i_m, j_m, \varphi_m) \cdot G(i_{m-1}, j_{m-1}, \varphi_{m-1}) \cdots G(i_1, j_1, \varphi_1) \cdot x\,.$$

Die Parameter i_μ, j_μ und φ_μ werden über gleich lange Listen Listen `i`, `j` und `phi` der Länge m übergeben. Rechnung in Gleitpunktarithmetik (`evalf`).

Aufgabe 10.4*. Erstellen Sie eine Prozedur `Jacobian(F::procedure,u::Vector)`, die zu einer gegebenen Funktion $F\colon \mathbb{R}^n \to \mathbb{R}^m$ und gegebenem $u \in \mathbb{R}^n$ die Jacobi-Matrix (Matrix der partiellen Ableitungen)

$$\begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m\times n},$$

ausgewertet an der Stelle $x = u$ zurückliefert.

[1] Sehen Sie wie? Beachten Sie $V{\cdot}V^T = \sum v_i{\cdot}v_i^T$ mit den Spalten v_i von V.

Bezüglich `F` nehmen Sie an, dass der Aufruf `F('x')` einen Ergebnisvektor zurückliefert, der alle Komponenten von `F` als differenzierbare Funktionen in den Variablen `x[1],x[2],...` dargestellt. Beispiel ($m = n = 2$):

```
> F:=proc(x);
> return Vector([x[1]*x[2],x[1]^2+x[2]^2]):
> end proc:

> F('x');
          [    x_1 x_2    ]
          [ x_1^2 + x_2^2 ]
```

Hinweis: Unterscheiden Sie zwischen `'x'` als 'Variablennamen' und dem Argument `u` beim Aufruf von `Jacobian` – die Komponenten von `u` können aus Symbolen (Namen) oder aus Zahlen zusammengesetzt sein (die Stelle, wo man die Jacobi-Matrix auswerten will).

Aufgabe 10.5. Fortsetzung von Aufgabe 4: Eine analoge Prozedur `Hessian(f::procedure,u::Vector)` für die Matrix der (gemischten) zweiten partiellen Ableitungen (Hesse-Matrix $\in \mathbb{R}^{n\times n}$)

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix} \in \mathbb{R}^{m\times n}$$

für eine gegebene Funktion $f\colon \mathbb{R}^n \to \mathbb{R}$.

Aufgabe 10.6. 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, einfache Rechnung.)

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).

Man implementiere diese Art der Berechnung von $(A + UV^T)^{-1}$ in einer Maple-Prozedur

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

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

Aufgabe 10.7. In MATLAB gibt es den Befehl `spy` zur grafischen Darstellung der Besetzungsstruktur schwach besetzter Matrizen (etwa so: weißes Kasterl = Nullelement, schwarzes Kasterl = ungleich Null).

Schreiben Sie eine Maple-Prozedur

`spy(A::Matrix,...)`

die für Objekte vom Typ `Matrix` etwas Ähnliches macht. Design und Werkzeuge, allfällige optionale Parameter etc. seien Ihrer persönlichen Kreativität überlassen.

Aufgabe 10.8. Erweiteren Sie die Prozedur `arnoldi` aus Teil IV dahingehend, dass zusätzlich zu der Matrix $V \in \mathbb{R}^{n\times m}$ die Matrizen $V{\cdot}V^T$ und $H := V^T{\cdot}A{\cdot}V$ zurückgeben werden. $V{\cdot}V^T \in \mathbb{R}^{n\times n}$ ist der Orthogonalprojektor auf den betreffenden Unterraum U,[2] und $H = V^T{\cdot}A{\cdot}V \in \mathbb{R}^{m\times m}$ ist 'verwandt' zu A auf dem Unterraum U (keine Details).

Testen Sie ein paar Beispiele: Die entstehende Matrix H hat immer eine spezielle Struktur.[3] Welche ist das? Verwenden Sie auch symmetrische Beispielmatrizen A. Wie sieht H nun aus? Wenn Sie nun `arnoldi(A,b,m)` mit der vollen Dimension `m=n` aufrufen, insbesondere für symmetrisches A, erhalten Sie eine Matrix $H \in \mathbb{R}^{n\times n}$, die zu A orthogonal ähnlich aber einfacher gebaut ist. Dies ist z.B. günstig für numerische Algorithmen zur Bestimmung der Eigenwerte von A = Eigenwerte von H (keine Details).

Überprüfen Sie dann noch an einem Beispiel, dass tatsächlich Eigenwerte von `evalf(A)` = Eigenwerte von `evalf(H)` (`?LinearAlgebra[Eigenvalues]`).

Anmerkung: Bei der Arnoldi-Iteration kann es – abhängig von der Wahl des Startvektors b – passieren, dass sie vorzeitig abbricht, weil einer der internen w-Vektoren Null wird. Das sind aber Sonderfälle, die Sie im Code nicht zu berücksichtigen brauchen.

---

[2] Vgl. Aufgabe 2.

[3] Abgesehen von Kontaminationen durch Rundungsfehler (kleine Elemente statt Nullen).