Winfried Auzinger
Klaus Felsenstein
Dirk Praetorius

SS 2010
(5. März 2010)

Übungsaufgaben zur VL Computermathematik

Serie 1

Die Aufgaben mit Stern (*) sind bis zur Übung in der kommenden Woche fix vorzubereiten und werden dort abgeprüft (Minimalerfordernis).

Kopieren Sie Ihre Worksheets auf Ihren Account auf `lva.student.tuwien.ac.at`*. Überprüfen Sie vor der Übung, ob Ihre Codes unter Maple 12 auf* `lva` *einwandfrei funktionieren. Es wird empfohlen, für jedes Beispiel im Verzeichnis* `serie01` *ein eigenes Worksheet mit dem Namen* `aufgabey-y.mw` *anzulegen. Alle zu verfassenden Prozeduren sind zu kommentieren und – je nach Angabe – mit verschiedenen Werten für die Parameter auszutesten.*

Verwenden Sie konsequent die Online-Hilfe – das ist wichtig für den erfolgreichen praktischen Umgang mit Maple. Insbesondere kommt es immer wieder vor, dass etwas in der Vorlesung [noch] nicht im Detail besprochen wurde – dann muss man sich zu helfen wissen. Explizite Verweise auf die Hilfe (z.B. `?set`*) deuten an, dass man sich die entsprechende Hilfe-Seite auf jeden Fall ansehen sollte. Manchen Aufgaben dienen auch der Vertiefung von Themen, die in der Vorlesung behandelt wurden; auch weitere Themen werden angesprochen (mit entsprechenden Hinweisen).*

'Perfekte' Maple-Prozeduren arbeiten mit konsequenter Überprüfung der Zulässigkeit der Eingangsparameter, insbesondere mittels `type`*. Bei den meisten für die Übung zu implementierenden Prozeduren verzichten wir jedoch darauf, es sei denn, es wird explizit verlangt.*

Aufgabe 1.1*. Eine endliche Menge (`?set`) wird in Maple repräsentiert als eine `exprseq` zwischen geschwungenen Klammern, im einfachsten Fall in der expliziten Gestalt

{ *element1*,*element2*, . . . , *elementn* }

Generieren Sie zwei Mengen M_1, M_2 von natürlichen Zahlen, [1]

```
M[1]:={...}; M[2]:={...};
```

und führen Sie folgende Operationen aus.

- $(M_1 \cup M_2) \setminus (M_1 \cap M_2)$
- Überprüfung, ob eine Zahl `m` genau in einer der beiden Mengen enthalten ist (mittels `evalb`)
- Berechnen Sie die Summe der Elemente einer derartigen Menge mittels `? add`
- Ist es mittels `subset` möglich, auf 'echte Teilmenge' zu testen? Falls nein, erstellen Sie eine Prozedur, die das realisiert.
- Erstellen Sie eine Prozedur `cartprod(A,B)` die das kartesische Produkt `A` × `B` als Menge von geordneten Paaren zurückliefert. Die geordneten Paare (a, b) werden als Listen `[a,b]` der Länge 2 dargestellt.

Aufgabe 1.2*. Schreiben Sie eine Prozedur `dsumsum(n::posint)`, die die Ziffernsumme der Ziffernsumme einer natürlichen Dezimalzahl `n` berechnet.

Wie lautet der betreffende maximale Wert für beliebige k-stellige Zahlen ($k = 1 \ldots 9$)? Das ist eine kleine Denksportaufgabe; überprüfen Sie Ihre Lösung *brute force* mittels Maple.

Aufgabe 1.3*. Die *Stirling'sche Formel* liefert eine einfach auswertbare asymptotische Näherung von $n!$ für große Werte von n:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \quad \text{für } n \to \infty.$$

Überprüfen Sie diese Aussage experimentell; stellen Sie den Verlauf mit wachsendem n in Form einer Grafik dar. (`with(plots); ?pointplot`)

Aufgabe 1.4*. Überlegen Sie, wie die Lösung (u_i) der rekursiv definierten Folge

$$u_i := a\,u_{i-1} + b_i, \quad i = 1, 2, \ldots; \quad u_0 \text{ gegeben}$$

aussieht,[2] und erstellen Sie eine Prozedur `linear_recursion(a,b,n)`, die zu gegebenem `a` und einer gegebenen Funktion `b`, die die Werte der Inhomogenität b_i liefert, die Lösungswerte $u_1, \ldots, u_n$ in einer Liste `u` zurückliefert.

[1] Hier werden sogenannte indizierte Namen für die Mengen verwendet. `M[1]` ist die Maple-Notation für M_1.

[2] Sie können dazu auch Maple verwenden (was passiert für $i = 1, 2, 3, \ldots$?). Für allgemeine i kann man diese Summenformel mittels vollständiger Induktion beweisen. Gelingt es Ihnen, den Induktionsschritt $i \mapsto i + 1$ mit Hilfe von Maple auszuführen?

Aufgabe 1.5. Die Funktion `time()` dient als Stoppuhr (liefert verbrauchte CPU-Zeit in s seit Beginn der aktuellen Maple-Session (entspricht `cputime` in MATLAB.) Schreiben Sie eine Prozedur `ftimer(f,n)`, die eine Funktion `f(x)` `n`-mal mit zufällig generierten Argumenten `x` auswertet und die verbrauchte CPU-Zeit zurückgibt. (Zeiten im ms-Bereich sind nicht sinnvoll beobachtbar.)

Hinweis: ? `rand` (Zufallszahlen). Für die Zwischenspeicherung der aktuellen CPU-Zeit am Beginn der Prozedur verwendet man eine lokale Variable, `local t;`

Gibt es auch eine *wall clock*? (MATLAB: `tic / toc`)

Aufgabe 1.6. Ein Praxis-Tipp: Manchmal ist es übersichtlicher, wenn eine Eingabe nicht sofort ausgewertet wird (ähnlich wie bei den trägen Befehlen). Dazu 'kapselt' man den Ausdruck zwischen `'...'`. Beispiel:

```
> v := 'cos(Pi)';
```

$$\cos(\pi)$$

```
> v;
```

$$-1$$

Die erste Auswertung 'entfernt nur die Quotes', danach erst wird wirklich ausgewertet. Manchmal ist das tatsächlich *notwendig*: Versuchen Sie einmal, die durch folgende Prozedur definierte Funktion zu plotten:

```
> f := proc(t) if t<0 then -1 else 1 end if end proc:
> plot(f(t),t=-1..1);
```

Überlegen Sie, warum das nicht funktioniert. Mittels geeigneter Setzung von Quotes funktioniert es.

Eine Alternative besteht in der Verwendung von

```
> f := proc(t) `if`(t<0,-1,1) end proc;
```

oder von `?piecewise` – recherchieren und ausprobieren. Beachten Sie auch den Effekt der Option `discont=...` beim Plotten unstetiger Funktionen. (`? plot/options`)

Checken Sie auch das Verhalten von `diff, D` und `int` bei den drei genannten Varianten.

Aufgabe 1.7. *Partielle Summation* ist ein diskretes Analogon zur partiellen Integration: Für zwei Folgen $(a_k), (b_k)$, $k = 0 \dots n$ gilt

$$\sum_{k=1}^{n} a_k(b_k - b_{k-1}) \;=\; a_n b_n - a_0 b_0 - \sum_{k=1}^{n} (a_k - a_{k-1})\, b_{k-1}$$

(Beweis mittels vollständiger Induktion bzw. durch Umformen auf eine Teleskopsumme). Verwenden Sie partielle Summation zur Herleitung einer Formel für die verallgemeinerte geometrische Summe

$$\sum_{k=1}^{n} k\, q^k, \quad q \neq 1.$$

Hinweis: $a_k = k$; b_k ist leicht zu bestimmen; Maple hilft Ihnen auch dabei: Berechnen Sie `sum(q^k,k)` und überlegen Sie, was die Ausgabe bedeutet. [3]

Implementieren Sie die so Formel in Maple und vergleichen Sie mit dem Ergebnis von `sum`. Wie lautet der Wert der konvergenten Reihe $\sum_{k=1}^{\infty} k\, q^k$ für $|q| < 1$?

Sie können sich auch an $\sum_{k=1}^{n} k^m\, q^k$ versuchen ($m = 2, 3, \dots$).

Aufgabe 1.8. Der Operator `@` dient zur Komposition von Funktionen, z.B. `sin@cos(x)` bedeutet dasselbe wie `sin(cos(x))`. `@@` steht für die funktionale Potenz, z.B. `sin@@2(x)` bedeutet dasselbe wie `sin(sin(x))`.

Definieren Sie mit Hilfe dieser Operatoren die von einem Parameter $k \in \mathbb{Z}$ abhängige Funktion $g(x,k) = e^{2\sin^k(\ln(x))}$ in der Form `g:=`*Funktionsausdruck*. Schreiben Sie zum Vergleich auch eine Prozedur

```
G:=proc(x,n::integer) ...
```

die dasselbe leistet, und verifizieren Sie, dass die beiden Funktionen identisch wirken. Testen Sie für $k = \dots -2, -1, 0, 1, 2, \dots$. Wenden Sie auf beide Varianten den Ableitungsoperator `D` an, und plotten Sie eine der Funktionen samt Ihrer Ableitung.[4]

---

[3] Zu beachten: $q = 1$ ist ein Sonderfall.

[4] Hinweis: Die Notation $f^k(x)$ wird nicht einheitlich verwendet. Für $k \geq 0$ bedeutet es manchmal $f(x)^k$ (Werte von f zur k-ten Potenz erhoben), oft jedoch die k-fache Anwendung $f(f(f(\dots(x)))$ (funktionale Potenz). Hier ist letzteres gemeint. Man bedenke auch, was $f^k(x)$ für negative k bedeutet.