Winfried Auzinger
Dirk Praetorius

Sommersemester 2017
03./04. Mai 2017

Übungsaufgaben zur VU Computermathematik

Serie 6

Exercise 6.1: ***Limits and infinite series.***

a) Compute the following limits or infinite series:

$$\lim_{x\to 0}\frac{e^x-1}{x} \qquad \lim_{x\to 0}\frac{e^{|x|}-1}{x} \qquad \lim_{n\to\infty}\Big(\frac{n-1}{n+1}\Big)^n$$

$$\sum_{k=1}^{\infty}\frac{1}{k} \qquad \lim_{n\to\infty}\Big(\sum_{k=1}^{n}\frac{1}{k}\ -\ \ln n\Big) \quad \text{(is this finite?)}$$

$$\sum_{k=1}^{\infty}\frac{1}{k^{\,2}} \qquad \sum_{k=1}^{\infty}\frac{1}{k^{\,22}}$$

b) Geometric series: First declare that x represents some real value $\in(-1,1)$ via

```
assume(x>-1,x<1)
```

Now, compute the values of the following series:

$$\sum_{k=1}^{\infty}x^k \qquad \sum_{k=1}^{\infty}k^2\,x^k \qquad \sum_{k=1}^{\infty}k^{22}\,x^k$$

c) Some further series (maybe you already know the results): [5]

$$\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{k} \qquad \sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{k^2} \qquad \sum_{k=0}^{\infty}\binom{c}{k}x^k \ \text{ for } c \text{ arbitrary, } x\in(-1,1)$$

Exercise 6.2: ***Calculus with real functions.***

a) Use Maple as a computational tool for analyzing the real function (*Kurvendiskussion*)

$$f(x)=\ln(x^2+\sqrt{3}\,x+1)$$

including nice plots of the function and its first and second derivatives.

b) Compute the indefinite integral of the function

$$f(x)=\frac{1}{1+x^6}$$

c) $(*)$ Verify that the result obtained in **b)** is indeed correct by differentiating it. Use `simplify` – or whatever turns out to be useful (this appears not to be straightforward).

---

[5] Here, $\binom{c}{k}=\dfrac{c(c-1)\cdots(c-k+1)}{k!}$ is the generalized binomial coefficient, a polynomial of degree k in the variable $c\in\mathbb{R}$. Use `binomial`.

Exercise 6.3: ***Taylor expansion.***

a) Use `taylor` to compute the Taylor expansion up to degree $n = 10$ about $x = 0$ for the function

$$f(x) = \sqrt{1+x}$$

and verify that this is in compliance with the binomial series expansion from Exercise 6.1 c).

b) Use `convert(...,polynom)` to convert the Taylor expansion from **a)** into a polynomial $p_{10}(x)$ of degree $n = 10$. Plot this polynomial together with $f(x)$, e.g., first for $x \in [0,1]$ and then for $x \in [0,1.1]$.

What do you conclude from this plot? Also try to increase degree of the Taylor polynomial, e.g., up to $n = 20$. What do you observe?

c) $(*)$ The convergence of these Taylor approximations $p_n(x)$ of degree n, for $n \to \infty$, at $x = 1$ somewhat delicate to decide (of course, the answer to this question is well known).

Investigate this via numerical experiment, using `evalf` *to compute* $p_n(1) - f(1)$ *up to* $n = 100$ *or even higher. What do you observe? Can you 'decide' on convergence for* $n \to \infty$ *on the basis of this numerical experiment?*

Exercise 6.4: ***Taylor approximation.***

Let f be a function with at least $n+1$ continuous derivatives f', f'', $\ldots$, $f^{(n+1)}$. For the Taylor polynomial $p_n(x;x_0)$ of degree n w.r.t. some point x_0, we know

$$f(x) - p_n(x;x_0) = \frac{1}{n!}\int_{x_0}^{x} f^{(n+1)}(\xi)\,(x-\xi)^n\,d\xi$$

a) *Design a function* `tayerr` *which expects a function* f, x_0, x, *and* n *as its arguments and which evaluates this error integral. Choose an example to test its correctness.*

b) Taylor polynomials are often used for practical numerical approximation of function values. For the function $f(x)$ from Exercise 6.3 this is not necessary since this can be directly evaluated in an accurate way. Now we consider the function

$$f(x) = \sqrt{1+x} - 1$$

satisfying $f(0) = 0$.

Set `Digits:=10` *and use* `evalf` *to evaluate* $f(x)$ *numerically for* $x = 2.0^{-10}$ *and* $x = 2.0^{-20}$. *Compare the results with the 'exact' values obtained by setting* `Digits:=20`, *and compute the relative errors. We may expect 10 correct digits, i.e., a relative error somewhere near* 10^{-10}. *What do you observe?*[6]

c) *Here, Taylor approximation helps. Use the formula from* **a)** *to decide what degree* n *you need such that the Taylor approximation* $p_n(x;0)$ *has a relative approximation error* $\approx 10^{-10}$ *at* $x = 2.0^{-10}$ *and* $x = 2.0^{-20}$, *respectively. Test these approximations using* `Digits:=10`.

d) *Same question as in* **b)** *for* $x = 0.5$. *What degree* n *would be required here? Anyway, is Taylor approximation required in this case in order to get 10 correct digits?*

[6] You learn more about this effect in a numerical analysis course.

Exercise 6.5: *A function defined in a piecewise way.*

We consider the functions $T_n\colon \mathbb{R} \to \mathbb{R}$ defined by [7]

$$T_n(x) = \begin{cases} \cos(n \arccos x), & |x| \le 1 \\ \cosh(n \operatorname{arcosh} x), & |x| > 1 \end{cases} \qquad (n \in \mathbb{N}_0)$$

a) *Design a function* `T` *which expects* `x` *and* `n` *as its arguments and which returns* $T_n(x)$*. Use* `ifelse`*. What happens if you call this function with a symbolic or a numerical argument* `x` *?*

b) *Set, e.g.,* $n = 5$ *and try* `plot(T(x,n),x=-1.1..1.1)`*. What do you observe?*

Then, try `plot('T(x,n)',x=-1.1..1.1)`*. Can you explain this effect?*

c) *Can you differentiate your function* `T` *?*

d) *Use the* `? piecewise` *construct to define the same function. Call* `T(x,n)` *and see what happens. Can you differentiate this function?*

Try `plot(T(x,n),x=-1.1..1.1)`*.*

e) $(*)$ Actually, $T_n(x)$ is a polynomial of degree n, the so-called *Chebyshev polynomial* of the first kind.

Verify this by playing around with $\cos(n \arccos x)$ *and* $\cosh(n \operatorname{arcosh} x)$*, e.g., for* $n = 5$*, using* `expand` *and* `simplify[trig]`*.*

Exercise 6.6: *A sequence of functions defined in a recursive way.*

Functions or procedures can be defined recursively. As a very simple example, we consider the sequence of polynomials defined by $T_0(x) = 1,\ T_1(x) = x$, and [8]

$$T_n(x) = 2\,x\,T_{n-1}(x) - T_{n-2}(x) \quad \text{for} \ \ n \ge 2$$

a) *Implement* $T_n(x)$ *in form of a recursive Maple function. Use* `ifelse`*.*

b) *Implement* $T_n(x)$ *in form of a recursive Maple procedure. Use* `proc` *and* `if ... else ... end if`*.*

c) *Check the running time for evaluation of, e.g.,* $T_{30}(x)$ *for both version. For this purpose, use the timer:*

```
> n:=30; start:=time(): T(x,n); runtime:=time()-start;
```

d) *Repeat* **b), c)***, adding the declaration* `option remember:` *immediately after* `proc(...)`*. What do you observe?*

Hint: `option remember` activates a so-called *remember table*. Each already computed function value is stored in an internal table (this requires some additional memory). Each time a function value is computed within the recursion, a table lookup is performed. If this value already has been computed, it is simply copied from the table.

Can you explain why the version with remember table runs much faster? Argue that the implementations **a)** *and* **b)** *are stupid.*

e) Actually, implementing this in form of a recursion is an overkill. You can realize this by a simple `do` loop.

Implement this version in form of a procedure and compare the running time with that of version **d)***, e.g., for* $n = 1000$*.*

---

[7] For some odd reason, the Maple implementation of arcosh has the name `arccosh`.
Remark: $\operatorname{arcosh} x$ is complex for $x < -1$, but $\cosh(\operatorname{arcosh} x)$ is again real.

[8] These functions are identical with the functions $T_n(x)$ from Exercise 6.5.

Exercise 6.7: ***An animated plot.***

a) *Use* `plots[animate]` *to produce a video displaying the functions $T_n(x)$ from Exercise 6.6 for $x \in [-1, 1]$ and n from* 0 *to* 20.

Hint: Use `'f(x,n)'` instead of `f(x,n)`. With right mouse click on the initial frame you get access to the control panel where you can start the animation.

b) *Increase the resolution of the plot via setting the paramater* `numpoints` *to* 1000. *Also, play with the other plot parameters in order to generate a nice animation. Export your animation in form of an animated* `.gif` *file.*

c) You can also combine several single plots in the following way:

```
p[1] := plot(...): # save first plot structure
p[2] := plot(...): # save second plot structure
  ...
plots[display](p[1],p[2],...)
```

Use such a version to display several of the $T(x, n)$ in a single plot. With the option `insequence=true`, `display` *again produces an animated plot.*

d) *Repeat* **c)** *using different colors for the different* `plots` *s.*

Hint: There are many ways to set the color. For instance, you may use the `plot/color` option together with the random number generator to produce random RGB values:

```
plot(...,color=ColorTools:-Color([rand()/10^12,rand()/10^12,rand()/10^12])
```

Exercise 6.8: ***Graphics packages.***

Check the contents of the packages `? plots` *and* `? plottools`. *Try some examples and prepare a presentation.*