Winfried Auzinger
Dirk Praetorius

Sommersemester 2013
2./8.05.2013

Übungsaufgaben zur VU Computermathematik
Serie 6

Exercise 6.1. Study the help pages

```
? interface
```

(concerning control of the user interface), and

```
? kernelopts
```

(concerning control of the behavior of the kernel), and get an overview. Try out what you find interesting and collect this material in a worksheet.

Exercise 6.2.

a) Do you know how whether the following series are convergent and what the respective values are? Maple knows – test it:[1]

$$\sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{2k-1} = \frac{\pi}{4}\,, \qquad \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{2k} = \tfrac{1}{2}\,\ln 2\,. \tag{1}$$

b) Use `taylor` to compute the first terms of the Taylor expansion of the functions

$$f(x) = \arctan x, \quad f(x) = \ln(1+x)$$

with respect to $x_0 = 0$. From the result it is evident how the Taylor series of these functions look like. Note that the arctan - series converges to $\arctan x$ for all $x \in \mathbb{R}$, and the $\ln(1+x)$ - series converges to $\ln(1+x)$ for $-1 < x \le 1$.

Now, turn the situation around: Specify these series in Maple using `sum`, summing up to `infinity`, and check whether Maple is able to identify them with the corresponding functions. Does Maple care about the convergence domain (which is finite for $\ln(1+x)$)?

Furthermore, use these series to verify the formulas (1) by appropriate choice of x.

c) Design a procedure `taycoe(f,x0,n)` which, for a given (infinitely differentiable) function $f(x)$, returns its n-th Taylor coefficient

$$\frac{f^{(n)}(x_0)}{n!}$$

for given $n \in \mathbb{N}$ and $x_0 \in \mathbb{R}$, and use it to design your own version of `taylor`, e.g., `mytaylor(f,n,x0)` generating the Taylor polynomial of degree n. Your procedure should return the polynomial function

$$h \mapsto \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}\, h^k$$

in this way:

---

[1] $\infty =$ `infinity`

```
return h->add(...);
```

Remark: This amounts to a naive, brute-force computation. E.g., to compute the 1001*-th Taylor coefficient of* $f(x) = \arctan x$*, your procedure* `taycoe` *will compute the* 1001*-th derivative (a high-degree rational function) and evaluate it at* $x_0 = 0$*.*

In contrast, the general form of the Taylor series of many standard functions like, e.g., $\arctan x$*, is built-in to the Maple in an internal database as static knowledge. Furthermore, a number of more or less tricky symbolic algorithms for yielding general formulas for Taylor coefficients is internally implemented.*

Exercise 6.3.

a) Check by experiment the validity of the Leibniz formula for the n-th derivative of a product of two functions,

$$\frac{d^n}{dx^n}\,(f\cdot g)(x) \;=\; \sum_{k=0}^{n} \binom{n}{k}\,\frac{d^k}{dx^k}\,f(x)\cdot\frac{d^{n-k}}{dx^{n-k}}\,g(x)\,,$$

for $n = 1, 2, 3, \ldots$. Use the derivative operator `D`. (In Maple, the binomial coefficient is `binomial(n,k)`.)

c) Design a procedure `htvol(n)` which computes the n-dimensional volume of a hypertetraeder $\subseteq \mathbb{R}^n$,

$$V_n = \int_{x_1=0}^{1}\Big(\int_{x_2=0}^{x_1}\Big(\cdots\int_{x_n=0}^{x_{n-1}} 1\; dx_n\,\ldots\Big)\,dx_2\Big)\,dx_1$$

for given $n \in \mathbb{N}$. Testing $n = 1, 2, 3, \ldots$ you see how the general solution V_n will look like (it can be easily verified by induction).

Exercise 6.4. Consider the recursion

$$x_n := c\,x_{n-1} + a_n\,, \quad n = 1, 2, 3, \ldots$$

with a parameter c and a given initial value x_0.

a) Implement this in form of a recursive procedure,

```
xrec := proc(n,x0,c,a) ... end proc;
```

Your procedure expects a function `a := n-> ...` Check what happens when this function is not explicitly specified.

b) Call your procedure with $n = 0, 1, 2, 3, \ldots$ and use `expand` in order to find the general formula for the solution x_n. This is easy to see (and easy to prove by induction for general n).

c) Consider numerical evaluation of the sequence x_n, e.g.,

```
seq(xrec(n,x0,c,a),n=0..N);
```

with numerically specified data `c` and `a(.)`, for larger values of N, e.g., $N = 1000$. Is this efficient? Compare this with a simple loop for computing the sequence. Compare computing times using the CPU clock `time()`.

Exercise 6.5.

a) Consider the two-step recursion

$$x_n := a\,x_{n-1} + b\,x_{n-2}\,, \quad n = 2, 3, 4, \ldots \tag{2}$$

where starting values x_0 and x_1 are given in some way. We wish to find the general form if the solution. To this end we use the ansatz

$$x_n = \lambda^n$$

with some parameter λ and plug it into (2). Conclude that there are two possible values $\lambda = \lambda_1$ and $\lambda = \lambda_2$ such that the ansatz works. Use Maple to express λ_1 and λ_2 in terms of the arbitrary parameters a and b. (Depending on a and b, the solution may be real or complex).

Then, the general solution of recursion (2) is given by

$$c_1\,\lambda_1^n + c_2\,\lambda_2^n$$

with arbitrary constants c_1, c_2.

b) Use **a)** in order to generate an explicit formula for the *Fibonacci numbers* F_n defined by $F_0 = 0$, $F_1 = 1$, and

$$F_n := F_{n-1} + F_{n-2}\,, \quad n = 2, 3, 4, \ldots \tag{3}$$

Verify that your F_n indeed satisfy (3).

c) Use `plots[pointplot]` to plot the coordinate pairs $(x, y) = (0, F_0), (1, F_1), (2, F_2), (3, F_3), \ldots$.

Exercise 6.6. Proceed in an analogous way as in 6.5 to solve the recursion

$$x_n = x_{n-1} - x_{n-2} + x_{n-3}\,, \quad n = 3, 4, 5, \ldots$$

with given initial values $x_0 = 0$, $x_1 = 1$, and $x_2 = 2$. (You may use variables with indices, `lambda[1]`, `lambda[2]`, `lambda[3]`, and `c[1],c[2],c[3]`.)

For two of the λ's you will obtain complex values,[2] but the solution x_n is of course real. You should be able to verify by simplification and testing the recursion that the solution is given by

$$x_n = 1 - \cos\frac{n\,\pi}{2}\,, \quad n = 0, 1, 2, 3, \ldots$$

In contrast to the Fibonacci sequence (see 6.5), this solution is bounded and oscillatory.

Exercise 6.7. Design a procedure `myrooti(x,n,tol,maxiter)` for numerical approximation of $y = \sqrt[n]{x}$ by means of a *interval bisection method*. Here we assume that $x \in (0, 1)$ and $n \in \mathbb{N}$ is given. We know that $\sqrt[n]{x} \in (0, 1)$, and the function $\sqrt[n]{x}$ is strictly monotonously increasing. The bisection method starts with $y = \frac{1}{2}$. If $y^n = x$, we are done. Otherwise, the solution is either contained in $[0, y]$ or in $[y, 1]$, respectively, depending on the sign of the residual $y^n - x$, and so on. Use this idea to program the bisection method, and realize it using a loop.

Implement this algorithm using exact (rational) arithmetic for given rational x. However, the solution will be irrational (algebraic) in general. Stop the iteration when the length of the current interval containing the true solution is $\leq$ `tol`, where `tol` is a given tolerance, e.g., `tol` = 1E-3 for moderate accuracy. The iteration should also be stopped when `maxiter` bisection steps have been performed without satisfying the tolerance.

---

[2] The imaginary unit is represented by the built-in variable `I`.

Note that `tol` and `maxiter` are actually not independent, because you can *predict* how many iterations will be sufficient to satisfy the given tolerance. The purpose of the parameter `maxiter` is simply to implement an 'emergency exit'.

Procedure `myrooti` should return

`[[yleft,yright],converged]` or `y`

In the first case, `[yleft,yright]` represents an interval containing the true solution, and `converged=true` if it satisfies the tolerance, otherwise `converged=false` (this occurs if the iteration stops prematurely at `maxiter` iterations). The second case indicates that the *exact* solution `y` has been found (special case).

(Note that in practice, such an algorithm will be implemented in floating point arithmetic, e.g., by using `evalf`. Also note that bisection converges because $\sqrt[n]{x}$ is monotone, but it converges rather slowly.)

Exercise 6.8. The package `plots`, to be activated by

```
with[plots];
```

contains many different plotting functions:

```
animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d, conformal,
conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot, display,
dualaxisplot, fieldplot, fieldplot3d, gradplot, gradplot3d, implicitplot, implicitplot3d,
inequal, interactive, interactiveparams, intersectplot,
listcontplot, listcontplot3d, listdensityplot,
listplot, listplot3d, loglogplot, logplot, matrixplot, multiple, odeplot, pareto,
plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d, polyhedra_supported,
polyhedraplot, rootlocus, semilogplot, setcolors, setoptions, setoptions3d, spacecurve,
sparsematrixplot, surfdata, textplot, textplot3d, tubeplot
```

Look at the documentation of this package and collect some cases of interest to you in a worksheet. Also play with options in order to produce 'nice-looking' results.

In particular, this package contains the useful function `display` which allows you to display several plots simultaneously, e.g., like in

```
p0 := plot(0,x=0..1);
pf := plot(x^2,x=0..1);
display(p0,pf);
```
