Winfried Auzinger
Dirk Praetorius

Sommersemester 2016
19. April 2016

Übungsaufgaben zur VU Computermathematik
Serie 6

Einige plot-Befehle und einige weitere Befehle (z.B. `piecewise`) wurden in der VO noch nicht genauer besprochen. Dafür werden jeweils Hinweise gegeben. Konsultieren Sie die Maple-Hilfe für genauere Details.

Exercise 6.1: ***Parametric plots.***

a) In spherical coordinates (θ, ϕ), a parametrization of the unit sphere $\{(x,y,z) \in \mathbb{R}^3 : x^2+y^2+z^2=1\}$ is given by

$$x(\theta,\phi) = \cos\theta\,\cos\phi$$
$$y(\theta,\phi) = \cos\theta\,\sin\phi$$
$$z(\theta,\phi) = \sin\theta$$

where $\theta = -\frac{\pi}{2}\ldots\frac{\pi}{2}$ and $\phi = -\pi\ldots\pi$.

Use `plot3d` *and play with plot parameters in order to produce a nice plot:*

```
plot3d([x(theta,phi),y(theta,phi),z(theta,phi)],theta=-Pi/2..Pi/2,phi=-Pi..Pi,...,...)
```

b) Let C be a curve in the (x,y)-plane, specified by two functions $x(t)$ and $y(t)$, where t is a real parameter, $t = a\ldots b$. You may expect that this can be plotted analogously as in **a)** using `plot` in the form

```
plot([x(t),y(t)],t=a..b,...)
```

Try out – what happens? Consult the help page for `plot` *to check how to realize such a parametric 2D plot. Play with plot parameters in order to produce a nice plot. Choose your own functions $x(t)$ and $y(t)$.*

c) Combination of **a)** and **b)**: Assume that two functions $\phi(t)$ and $\theta(t)$ define a curve in the (θ,ϕ)-plane. Then,

$$\big(x(\theta(t),\phi(t)), y(\theta(t),\phi(t)), z(\theta(t),\phi(t))\big)$$

(with $x(\theta,\phi), y(\theta,\phi), z(\theta,\phi)$ from **a)**) represents a spatial curve on the unit sphere.

Use `plots[spacecurve]` *to produce a nice plot of such a curve. Play with parameters.*

d) Each plot command produces a special plot structure representing the data of the plot. Normally, the plot is immediately displayed. But you can also store the plot data by assigning them to a variable, e.g. (for two 3D plots):

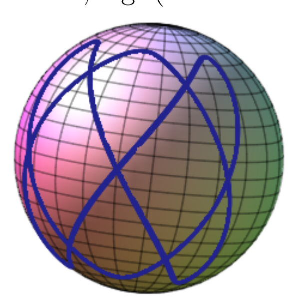
```
p[1] := plot3d(...):  p[2]:=spacecurve(...):
```

Then you may use `plots[display]` to render the plots together:

```
plots[display]([p[1],p[2]],...)
```

Combine **a)** *and* **c)** *in this way.*

Again, play with plot parameters in `display` *to produce a nice plot.*



Hint: You also can manipulate a plot interactively in various ways using the right mouse button or using the context menu shown on top of the screen when you click into the plot area.

Exercise 6.2: ***[Graphical] study of a real function.***

Consider the real function

$$f(x) = \sqrt{1 - \sqrt{2 - \sqrt{3 - x}}} \qquad (\sqrt{\ } = +\sqrt{\ })$$

a) f is well-defined for all x such that each $\sqrt{\ }$ - evaluation involved has a nonnegative argument. If you are too lazy to check the domain of definition of f by hand, use Maple:

Use `plot` *to visualize the function f. Choose the x - range appropriately, then you will immediately see for what interval $x \in [a, b]$ the function f is well-defined.*

b) *Use* `display` *to produce a plot showing the graph of f and its derivatives up to order 3. Do you think this is a very reasonable way of visualizing these data?*

c) For data ranging over several orders of magnitudes it is more reasonable to plot logarithmic values (usually using $\log_{10}$).

Use `plots[logplot]` *to repeat* **b)**. *What problem now appears? Suggest a fix for this problem.*

Exercise 6.3: ***Ex. 6.2 continued.***

Consider the function f from **6.2**.

a) *When you plot f you see that it is strictly monotonously decreasing (and therefore injective). Compute f' and use it for a strict proof of this fact. (This involves some manual inspection.)*

b) *Let $[a, b]$ be the domain of definition of f. Then, the inverse $f^{-1} : f([a, b]) \to [a, b]$ is well-defined. Compute this inverse and verify that $f^{-1}(f(x)) = x$ indeed holds true.*

Hint: Use `solve`.

c) *When you plot f'' you see that f has a turning point (Wendepunkt). Compute it using Maple (also its decimal value) and check that it is indeed a turning point.*

d) *Check whether Maple is able to integrate the function f.*

Exercise 6.4: ***A polynomial approximation of the exponential function.***

We compute a simple polynomial approximation for the exponential function $f(x) := e^x$ on the interval $[0, \ln 2]$. For a polynomial $p(x)$ of degree 3,

$$p(x) = a_0 + a_1\, x + a_2\, x^2 + a_3\, x^3$$

we require

$$p(0) = f(0), \quad p'(0) = f'(0), \quad p(\ln 2) = f(\ln 2), \quad p'(\ln 2) = f'(\ln 2)\,.$$

a) *Compute the coefficients of p using* `solve`.

b) *Check the accuracy of the resulting approximation $p(x) \approx f(x)$ graphically.*

c) As you see, the error $p(x) - f(x)$ is nicely small, negative, and it has a minimum in the interior of the interval $[0, \ln 2]$, and here the absolute approximation error $|p(x) - f(x)|$ becomes maximal. [5]

- *Use* `solve` *to determine the location of this minimum.*
- *If this does not work, use the numerical solver* `fsolve`. *What do you observe?*
- *You can inform* `fsolve` *about an interval where the solution should be found (see* `? fsolve`*). Use this to determine the location of the minimum, and compute the approximation error $p(x) - f(x)$ at this point.*

[5] In numerical analysis (interpolation theory) it is shown how the error of such approximations can be estimated theoretically. But this is not a issue here.

Exercise 6.5: ***Ex. 6.4 continued.***

Outside the interval $[0, \ln 2]$, the polynomial $p(x)$ from **6.4** is not a reasonable approximation for $f(x) = e^x$ (*check this using* `plot`). A global approximation for e^x can be constructed in a modified way: For arbitrary [6] $x > 0$, determine $k \in \mathbb{N}$ such that

$$x = k \ln 2 + r, \quad \text{with} \quad r \in [0, \ln 2)\,.$$

Then,

$$e^x = 2^k e^r, \quad \text{and therefore,} \quad e^x \approx 2^k p(r)\,.$$

a) *Implement this approximation in form of a procedure* `q(x)` *expecting a numerical argument* x*, using* p *from* **6.4.**

Hint: Use `evalf` and `floor`, and copy/paste the (numerical) coefficients of p obtained in **6.4** into your procedure.

b) *Extend your procedure from* **a)** *such that also negative values of* x *are correctly handled.*

Hint: Use `if ... else ... end if`; observe $e^{-x} = 1/e^x$.

c) In this way we have defined a (numerical) function $p(x) \approx e^x$ for 'arbitrary' x.

Check the relative error $(p(x) - e^x)/e^x$ *of this approximation by plotting it, e.g., over the interval* $[-100, 100]$. *Try to explain the uniform relative accuracy obtained.*

$(*)$ Let us play a little bit with these functions which are defined via procedures:

d) You may also try to call `q` from **a)** with a symbolic argument, `q(x)`, where `x` has not been assigned a value.

Try this. Can you you differentiate the resulting expression? Can you plot the derivative?

Hint: First try to differentiate the function `x*floor(x)`, plot it, and explain the outcome. See `? floor`.

e) *Same question as in* **d)** *for the extended procedure from* **b)**. *What do you observe?*

Exercise 6.6: ***Functions defined in a piecewise way.***

Let a function $f(x)$ defined in a piecewise manner, e.g.

$$f(x) := \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ x^2, & x > 0 \end{cases}$$

a) *Design a function which, using* `if ... end if` *or* `` `if`(...) ``*, implements this (or a similar) example.*

Can you differentiate or integrate this function using `diff`, `D`, *or* `int` *?*

Can you plot it? If not, try `plot('f(x)',...)`

b) Alternative: *Define the same function using the* `piecewise` *construct (look at the help page), and try again. Evaluate its indefinite integral and a definite integral.*

c) Assume that g is a given function, X is a strictly monotonically increasing list of numbers representing points on the real line, and let Y represent a list of function values, with `numelems(`Y`)` = `numelems(`X`)-1`.

Design a procedure `stepfunction(`X,Y`)` *which uses the* `piecewise` *construct to return the corresponding step function (*Treppenfunktion*)*

$$f(x) := \begin{cases} 0, & x < x_1 \quad (\textit{i.e., for } x \in (-\infty, x_1)\,) \\ g(y_1), & x < x_2 \quad (\textit{i.e., for } x \in [x_1, x_2)\,) \\ g(y_2), & x < x_3 \quad (\textit{i.e., for } x \in [x_2, x_3)\,) \\ \vdots & \vdots \\ g(y_{n-1}), & x < x_n \quad (\textit{i.e., for } x \in [x_{n-1}, x_n)\,) \\ 0, & \textit{otherwise} \quad (\textit{i.e., for } x \geq x_n\,) \end{cases} \qquad (n = \texttt{numelems(X)}\,)$$

Choose an example and plot the resulting step function.

Hint: Use `seq` to generate the `piecewise` construct.

[6] In practice, 'x arbitrary' means 'x not too large'. Observe that, for instance, $e^{700} \approx 10^{304}$ which is close to overflow in double precision arithmetic.

Exercise 6.7: ***A sequence of functions defined in a recursive way.***

a) Functions or procedures can be defined recursively. As an example, we consider the function sequence

$$f_0(x) := \frac{e^x - 1}{x}\,, \qquad f_n(x) := \frac{1}{x}\Big(-1 + \sum_{j=0}^{n-1} \frac{f_j(x)}{n-j} \Big), \quad n = 1,2,3,\ldots$$

Design a function `f(x,n)` *which implements this recursion.*

Hint: Use '`if`'.

b) *Verify experimentally for* $n = 0,1,2,3$ *that*

$$f_n(x) = (-1)^n \int_0^1 e^{(1-t)\,x} \binom{-t}{n}\, dt\,.$$

Here,

$$\binom{\tau}{n} = \frac{\tau(\tau-1)\cdots(\tau-n+1)}{n!}\,, \qquad \tau \in \mathbb{R},\ n \in \mathbb{N}$$

is the generalized binomial coefficient (`binomial(`*tau*,*n*`)`), a polynomial of degree n in τ.

Remark/hint: This identity is true for all $n \in \mathbb{N}$, but the proof is not so easy.

Use `int(...,t=0..1)` for evaluating the integrals. For internal implementation reasons concerning `binomial`, you have to `expand` this expression under the integral to make it work (try without and with `expand`).

c) *Realize a version of* **a)** *using a procedure instead of a function.*

Hint: Here you can use `if ... else ... end if`.

Exercise 6.8: $(*)$ ***A procedure manipulating a list.***

(See Exercise **5.5**.) Let *LL* be a list consisting of lists of the same length with entries 0 or 1. For simplicity, you may assume that the elements in *LL* are pairwise different (no multiple entries).

• *Design a procedure* `p(...)` *which scans the list LL and for each element L in LL checks whether some other element K occurring later in LL together with L forms a twin (see* **5.5***). In this case, K is removed from the list. The modified list is returned. Selfies are not removed.*

Do not encode `istwin` *directly into your procedure but use a function argument which may also represent some other relation between two elements L and K. This is more flexible.*

Example: `p([[0,0,1,0],[1,1,0,0],[1,0,1,1]],istwin)` returns `[[0,0,1,0],[1,1,0,0]]`.

Hint: Use two passes – marking twin elements in an appropriate way in the first pass and then building the modified list in the second pass.

Remark: Many standard list operations (like finding elements, removing double entries, etc., etc.) are implemented in the package `ListTools`. But none out of the functions from `ListTools` seems to be applicable here.