Winfried Auzinger
Kevin Sturm

Sommersemester 2019
27.03.2019

Übungsaufgaben zur VU Computermathematik

Serie 3

Exercise 3.1: ***Two simple recursions.***

a) *Design a* <u>*recursive*</u> *procedure* `p(n)` *which produces the following output (using* `print(...)`:

```
n^2
   .
    .
     .
      16
       9
       4
       1
      0
       1
      2
     3
    4
   .
  .
  .
  n
```

Your procedure produces printed output but returns no value. This means that no `return` is necessary (one may also use `return` without specifying a return value).

b) (cf. Exercise 1.1.) A list `L` is called <u>palindromic</u> if `L[`i`]=L[`n+1−i`]` for $i = 1 \ldots n$, where n denotes the length of `L`.

Design a <u>*recursive*</u> *procedure* `ispalindromic(L)` *which expects a list* `L` *as its argument and returns* `true` *if* `L` *is palindromic, otherwise* `false`*.*

Special cases: `[]` and a list of length 1 are palindromic.

Exercise 3.2: ***Partial integration.***

a) *Design a procedure* `myintparts(`f,g`)` *which expects two functions f and g as its arguments and computes the indefinite integral*

$$\int f(t)\, g(t)\, dt$$

by means of partial integration.

Hint: Recall the the well-known formula for partial integration. You need to differentiate f and integrate g (or vice versa). Compare your result with the answer delivered by `int`.

b) *Use your procedure from* **a)** *to compute*

$$\int \log_2(t)\, dt.$$

Compare your result with the answer delivered by `int`*.*

Exercise 3.3: ***Recursion for a sequence of definite integrals.***

a) *Use partial integration (by hand) to derive a recursion ($n-1 \to n$) for*

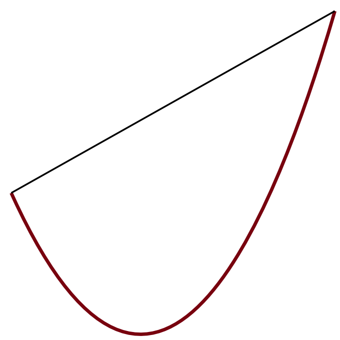
$$I_n = \int_0^1 t^n \, \mathrm{e}^{\lambda\, t} \, dt \qquad (\lambda \neq 0, \ n \in \mathbb{N}_0),$$

and implement this recursion in form of a procedure `IR(n)`. *Compare your results for $n = 0, 1, 2, 3, \ldots$ with the results delivered by* `int`.

b) Maple knows an explicit formula for I_n, $n \in N_0$. *Check this, using* `assume(n,nonnegint)`, *and compare with* **a)**.

Exercise 3.4: ***Convex minimization: a numerical bisection algorithm.***

Design a procedure `find_minimum(f,a,b,accuracy)` *which finds the unique minimum of a strictly convex real function $f\colon [a,b] \to \mathbb{R}$ by the searching algorithm indicated below. Here,* `accuracy` *is a small positive number specifying how much the search should be refined. The procedure returns an interval of length $\leq$* `accuracy` *(in form of a list) which contains the position $x_{\min}$ where the minimum is attained. All numerical computations are performed in floating point arithmetic.*



We assume that f and its derivatives are continuous, $f'(a) < 0$, $f'(b) > 0$, and $f''(x) > 0$ for all $x \in (a,b)$. Then, by elementary calculus, f has a unique minimum in (a,b). This can be found numerically by a bisection strategy : Let $c := (a+b)/2$.

(i) If $f'(c) = 0$, the minimum is located at c.

(ii) If $f'(c) > 0$, the minimum is contained in (a,c).

(ii) If $f'(c) < 0$, the minimum is contained in (c,b).

This leads, in an an obvious way, to a simple bisection algorithm for identifying an interval of length $\leq$ `accuracy` in which $x_{\min}$ is located. You may formulate it in an iterative or recursive way.

Exercise 3.5: ***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

$$\begin{aligned} x(\theta,\phi) &= \cos\theta\, \cos\phi \\ y(\theta,\phi) &= \cos\theta\, \sin\phi \\ z(\theta,\phi) &= \sin\theta \end{aligned}$$

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):

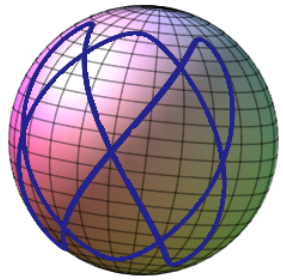
```
p[1]:=plot3d(...):  p[2]:=spacecurve(...):
```

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.*



Exercise 3.6: ***Continued fractions.***

A continued fraction is an (infinite) expression of the form

$$a_0 + b_1/\Big(a_1 + b_2/\big(a_2 + b_3/(a_3 + b_4/\ldots)\big)\Big)$$

a) Assume that the values a_k and b_k are given by a pair of functions `a(.)` and `b(.)`. *Design a procedure* `CFR(a,b,n,mode)` *which evaluates the truncated continuous fraction, stopping at depth* `n`*. Here,* `mode` *should be an option for evaluation (exact or float).*

b) Let $a_0 = 3$ and $a_k = 6,\ b_k = (2\,k-1)^2\ \ (k \geq 1)$. *Find out experimentally to which limit this continued fraction converges.*

Exercise 3.7: ***Animated graphs.***

a) *Prepare a nice example demonstrating the use of* `? animate`.

b) *Prepare a nice example demonstrating the use of* `? animate3d`.

Choose your own examples (as cool as possible).

Exercise 3.8: ***Your favorite package?***

Look at the help page `? index`, and select `packages.` Here you see a complete list of available packages.

Choose one of them, have a closer look, and prepare a small demo of its basic features.

If you have no other special preference, you may take a closer look at `plottools`, `geometry`. Aficionados of combinatorics may look at `combinat` (see also `combstruct`). And there are many, many more, like for instance `GraphTheory`.