Winfried Auzinger
Dirk Praetorius

Sommersemester 2015
06.–08.05.2015

Übungsaufgaben zur VU Computermathematik
Serie 7

Exercise 7.1. *Curves in 3D.*

a) A curve in 3D is specified by three coordinate functions $x(t), y(t), z(t)$, where the parameter t varies in some real interval $[a, b]$. Choose a curve (with differentiable coordinate functions $x(t), y(t), z(t)$) and plot it using `spacecurve` as well as using `tubeplot` from the `plots` package. Play with plot parameters in order to produce a nice plot.

b) Compute the arclength of your curve from **a)** according to the formula

$$\int_a^b \sqrt{x'(t)^2 + y'(t)^2 + z'(t)^2}\, dt$$

If `int` is not able to provide the exact result, compute a numerical approximation using the `int`-option `numeric` or using `evalf`.

c) Design a procedure `approximate_arclength(C,n)` which returns your own (very simple) numerical approximation of the arclength of a curve. Here:

- `C` is assumed to be a list of length 4, with the first entry representing the parameter interval $[a, b]$ and the other entries representing the coordinate functions $x(t), y(t), z(t)$. (I.e., `C[1]` is a list of length 2, and `C[2]`,`C[3]`,`C[4]` are Maple functions.)
- `n` $\in \mathbb{N}$ specifies that for the numerical integration the interval $[a, b]$ is divided into `n` subintervals of the same length $h = (b - a)/n$.

On each of these subinterval your procedure approximates the arclength over this subinterval by the trapezoidal rule (see **6.7 a)**). These values are summed up. Use `evalf` and compare with **b)**. How does the error of the approximation behave if you replace a given value n by $2n$, $4n$, $\ldots$?

Exercise 7.2. *Sudoku.*

Let a standard 9×9 Sudoku tableau be represented by a 9×9 `Matrix S` with entries `S[i,j]` $\in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Here, 0 means that the corresponding entry is void. (If no entry is 0, this means that the tableau is completely filled.)

Design a procedure `check_sudoku(S)` which returns `true` if the tableau `S` is valid according to the Sudoku rules, and `false` otherwise.

Exercise 7.3. *An* argmax *implementation.*

Design a procedure `argmax(A::{Vector,Matrix})` [1] which accepts an object `A` of type `Vector` or `Matrix` as its argument and returns the positions [2] of the maximal elements in `A`. For the case of a `Matrix`, the 'position' is the corresponding pair of indices. Return the answer in form of a list [of lists].

Include a check whether all elements of `A` have a numerical value (use `is(...,numeric)`). [3] If one of these tests fails, exit with an `error`-message.

Remark: In Maple, there is `max` but there seems not to exist something like `argmax`.

Hint: Using `type` you can determine the type of an object. In this way you can discern between `Vector` and `Matrix`.

---

[1] This syntax means that arguments of the type `Vector` or `Matrix` are accepted; otherwise the procedure will automatically exit with an error message (try). For accepting a single type only, e.g., `Vector`, one would use the syntax `A::Vector`.

[2] The maximal value may be attained several times.

[3] Data types are organized in a hierarchic way. E.g., the types `integer`, `rational`, `float` are sub-types of the type `numeric` representing any numerical real value.

Exercise 7.4. *Recursive procedures; application of the* `unapply` *command (see lecture, Part II).*

a) Let the numbers $B(n)$ defined by

$$B(0) = 1, \quad \text{and} \quad B(n) = \sum_{j=0}^{n-1} \binom{n-1}{j} B(j) \quad \text{for} \quad j \geq 1$$

Design a recursive procedure `B(n)` which computes $B(n)$ for given $\mathtt{n} \in \mathbb{N}$.

Can you evaluate $B(1000)$?

b) Let the functions $\phi_j(\cdot)$ be recursively defined by

$$\phi_0(z) = e^z, \quad \text{and} \quad \phi_j(z) = \frac{\phi_{j-1}(z) - \frac{1}{(j-1)!}}{z} \quad \text{for} \quad j \geq 1$$

Design a recursive procedure `generate_phi(j::nonnegint)`[4] returns the 'ready-cooked' Maple function $\phi_j(\cdot)$. Use `unapply` and `normal`. Then the resulting functions should be[5]

$$\phi_1(z) = \frac{e^z - 1}{z}, \quad \phi_2(z) = \frac{e^z - 1 - z}{z^2}, \quad \ldots$$

Exercise 7.5. *Two further recursive procedures: nothing special, just to train recursion.*

a) Design a *recursive* procedure `p(n)` which produces the following output (using `print(...)`, up to the value `n` specified on call):

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

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) A list `L` is *palindromic* if `L[`i`]=L[`n+1−i`]` for $i = 1 \ldots n$, where n denotes the length of `L`.

Design a *recursive* procedure[6] `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.

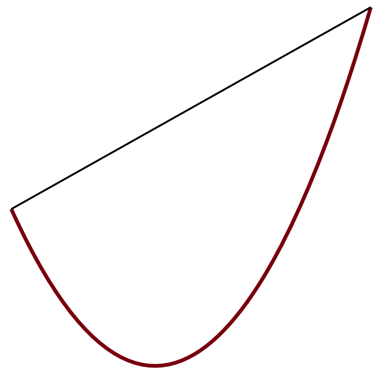
Exercise 7.6. *Convex minimization: a numerical bisection algorithm.*

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 described $\to$ below. `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.

[4] `nonnegint` is the type representing nonnegative integers $\in \mathbb{N}_0$, a subtype of the numeric type `integer` representing $\mathbb{Z}$. The type `posint` represents $\mathbb{N}$.

[5] Without using `normal` you would generate, e.g., $\phi_2(z)$ in the form $\frac{\frac{e^z-1}{z}-1}{z}$.

[6] Of course, this can also be easily realized using a `for` loop.



→ 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. Note that, 'by chance', x_{min} may be found exactly (see (i)). In this case the algorithm immediately returns this value.

Exercise 7.7. *Formatted output.*

a) Design a procedure `print_sudoku(S)` which produces a formatted output of a Sudoku (see **7.2)**) to the screen.

```
-------------------------
|   8   | 4 2 3 |     3 |
| 8 2 3 |   2 7 | 1   5 |
| 9   2 | 1 5 3 |       |
-------------------------
| 1 2   |   2 7 | 5   3 |
|     3 | 7   3 | 1 2   |
| 1   3 | 5 2 8 | 1   2 |
-------------------------
| 1 2 8 | 4 2 3 | 7   9 |
| 7   3 | 4   3 |       |
|   2 5 |   2 9 | 1   3 |
-------------------------
```

Use an auxiliary function which converts 0 to the string `" "` and integers $n > 0$ to the string "n".

Hint: Use `sprintf` and `printf`.

b) Design a procedure `print_sudoku(S,filename)` which prints a Sudoku to a textfile (the filename is specified as a string).

Hint: Use `fprintf`.

Exercise 7.8. 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.

There are many different packages. If you have no other special preference, you may take a closer look at the `plots` package. The package `geometry` is also very nice.