Winfried Auzinger
Dirk Praetorius

Sommersemester 2018
16. Mai 2018

Übungsaufgaben zur VU Computermathematik
Serie 7

Exercise 7.1: ***Recursive computation of parameter-dependent integrals.***

a) *Use integration by parts (manually) to derive a recursion w.r.t. n for the integrals*

$$I_n := \int x^n\, e^{\lambda x}\, dx \qquad (\lambda \neq 0,\ n \in \mathbb{N}_0),$$

and implement this recursion in form of a recursive procedure `IR(x,n)`. *Compare your results with the results delivered by* `int`.

Remark: Maple knows an explicit expression for I_n for general n (*check*).

b) *Does* **a)** *provide the correct answer when taking the limit $\lambda \to 0$?*

c) *Same as in* **a)**, for

$$\int \frac{dx}{(1+x^2)^n}\,.$$

Exercise 7.2: ***Solution of linear two-step recursions.***

a) Consider the two-step recursion

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

with given $a, b \in \mathbb{R}$. We wish to find the general form of the solution. To this end we use the ansatz

$$x_n = \lambda^n$$

with some (unknown) parameter λ and plug it into (R). Now it is easy to see that there are two possible values $\lambda = \lambda_1$ and $\lambda = \lambda_2$ such that the ansatz works (*check*).

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 (R) is given by

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

with arbitrary constants c_1, c_2.

However, there is a special case where the solution looks different. You may find out how the general solution looks like in this special case. (If not, don't worry.)

b) *Design a procedure* `twostep(a,b,x0,x1,n)` *which delivers an expression depending on n ($n = 0,1,2,3,4,\ldots$) for the solution x_n for given starting values x0 and x1.*

Hint: Use `solve` to determine the respective constants c_1 and c_2. What happens in the special case mentioned in **a)** ?

c) *Generate an explicit formula for the* Fibonacci numbers F_n *defined by $F_0 = 0$, $F_1 = 1$, and*

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

Exercise 7.3: ***Facing the devil.***

a) Consider the sequence of continuous functions[16] $D_n\colon [0,1] \to [0,1]$, recursively defined by $D_1(x) := x$ and

$$D_n(x) := \begin{cases} \frac{1}{2}\, D_{n-1}(3\,x), & 0 \le x < \frac{1}{3}, \\ \frac{1}{2}, & \frac{1}{3} \le x \le \frac{2}{3}, \\ \frac{1}{2}\,\big(1 + D_{n-1}(3\,x-2)\big), & \frac{2}{3} < x \le 1. \end{cases}$$

for $n > 1$.

Implement these functions in form of a recursive procedure `Devil(x,n)` *and produce plots for several values of n.*

Note that it makes no sense to call `Devil(x,n)` with a numerical value `n` but unspecified `x`. *Why?* As a consequence, you must not pass `Devil(x,n)` to the `plot` command, but `'Devil(x,n)'`. *Explain.*

b) Include `option remember` to your procedure from **a)** and compare execution times (use `time()`). Do you observe a difference?

Exercise 7.4: ***A special class of matrices.***

a) A quadratic matrix A is called circulant if it is of the form

$$A = \begin{pmatrix} a_1 & a_n & a_{n-1} & a_{n-2} & \cdots & a_2 \\ a_2 & a_1 & a_n & a_{n-1} & \cdots & a_3 \\ a_3 & a_2 & a_1 & a_n & \cdots & a_4 \\ a_4 & a_3 & a_2 & a_1 & \cdots & a_5 \\ & \ddots & \ddots & \ddots & \ddots & \\ a_n & a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_1 \end{pmatrix}$$

Design a procedure `iscirculant(A)` *which expects a quadratic matrix as its argument and which returns* `true` *if it is circulant and* `false` *otherwise.*

b) Circulant matrices are examples of 'data-sparse' matrices: A circulant matrix is uniquely defined by its first column.

Assume that a vector c represents a circulant matrix A, namely via its first column. Design a procedure which expects c and another vector x as its arguments and which computes the matrix-vector product $A \cdot x$ in an efficient way, without explicitly building the matrix A.

Exercise 7.5: ***Divided differences.***

For a function $f(x)$ and a given set of pairwise distinct values (nodes) $\{x_1, \ldots, x_n\}$, the so-called *divided differences* of f with respect to the x_j are defined recursively as

$$f[x_j, \ldots, x_k] := \begin{cases} f(x_j) & \text{for } j = k, \\ \dfrac{f[x_{j+1}, \ldots, x_k] - f[x_j, \ldots, x_{k-1}]}{x_k - x_j} & \text{for } j < k. \end{cases}$$

a) *Implement the evaluation of the divided differences by means of a recursive Maple procedure realizing the mapping $(j,k) \mapsto f[x_j, \ldots, x_k]$ for a function f and a list of nodes x_j :*

```
dd := proc(j,k,f,nodes)
```

Example: The call `dd(1,3,sin,[x[1],x[2],x[3],x[4]])` returns

$$\frac{\dfrac{\sin(x_3) - \sin(x_2)}{x_3 - x_2} - \dfrac{\sin(x_2) - \sin(x_1)}{x_2 - x_1}}{x_3 - x_1}$$

b) *Use* **a)** *to verify by examples the product formula*

$$(f \cdot g)[x_j, \ldots, x_k] = \sum_{\ell=j}^{k} f[x_j, \ldots, x_\ell] \cdot g[x_\ell, \ldots, x_k]$$

c) *Verify for $n = 1,2,3,4,\ldots$ that for an arbitrary polynomial $p(x)$ of degree n and arbitrary nodes $x_1, \ldots, x_{n+1}$ we have*[17]

[16] Remark: The limiting function $f(x) = \lim_{n\to\infty} fn(x)$ is continuous and it is differentiable almost everywhere, with derivative 0. The graph of the function f is called Devil's staircase.

[17] Note that an empty product is 1.

$$p(x) \equiv \sum_{k=1}^{n+1} p[x_1, \ldots, x_k] \cdot (x - x_1) \cdots (x - x_{k-1})$$

Remark: This is called the Newton representation of the polynomial. It is used in interpolation algorithms.

Exercise 7.6: ***Confluent divided differences.***

Within the setting of Exercise **7.5**, we now drop the assumption that the nodes x_j are pairwise distinct. We define

$$f[x_j, \ldots, x_k] := \begin{cases} f(x_j) & \text{for } j = k, \\ \dfrac{f[x_{j+1}, \ldots, x_k] - f[x_j, \ldots, x_{k-1}]}{x_k - x_j} & \text{for } j < k \text{ and } x_j \neq x_k \quad \text{(non-confluent case)}, \\ \lim\limits_{\varepsilon \to 0} \dfrac{f[x_{j+1}, \ldots, x_k + \varepsilon] - f[x_j, \ldots, x_{k-1}]}{\varepsilon} & \text{for } j < k \text{ and } x_j = x_k \quad \text{(confluent case)}. \end{cases}$$

This is well-defined if f has a sufficient high degree of differentiability (depending on the 'amount of confluence').

a) *Implement the evaluation of the confluent divided differences by means of a recursive Maple procedure realizing the mapping* $(j, k) \mapsto f[x_j, \ldots, x_k]$ *for a function* f *and a list of nodes* x_j *:*

```
cdd := proc(j,k,f,nodes)
```

Example: The call `cdd(1,3,f,[x[1],x[2],x[2]])` returns

$$\frac{D(f)(x_2) - \dfrac{f(x_2) - f(x_1)}{x_2 - x_1}}{x_2 - x_1}$$

b) *Verify by examples that the product formula*

$$(f \cdot g)[x_j, \ldots, x_k] = \sum_{\ell=j}^{k} f[x_j, \ldots, x_\ell] \cdot g[x_\ell, \ldots, x_k]$$

remains valid.

c) *What is* $f[\underbrace{x, \ldots, x}_{n \text{ times}}]$ *?*

d) Also identity **7.5 c)** remains valid. *What does it mean for* $x_1 = x_2 = \ldots = x_{n+1}$ *?*

e) *Verify by examples that the value of* $f[x_j, \ldots, x_k]$ *is invariant under any permutation of the nodes* x_ℓ*.*

Exercise 7.7: ***A two-dimensional integral.***

Let an arbitrary triangle $\Delta = \overline{P_1\, P_2\, P_3} \subseteq \mathbb{R}^2$ be given, with vertices $P_j = (x_j, y_j)$. To compute the integral of a real-valued function $f(x, y)$ defined over Δ, we represent points the $(x, y) \in \Delta$ in the form

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x(\xi, \eta) \\ y(\xi, \eta) \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \xi \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix} + \eta \begin{bmatrix} x_3 - x_1 \\ y_3 - y_1 \end{bmatrix}, \quad 0 \leq \xi + \eta \leq 1$$

with coordinates $(\xi, \eta) \in \Delta_{\text{ref}}$, where Δ_{ref} is the simple 'reference triangle' with vertices $(0, 0)$, $(1, 0)$, and $(0, 1)$. Note that $(x(0, 0), y(0, 0)) = (x_1, y_1)$, $(x(1, 0), y(1, 0)) = (x_2, y_2)$, and $(x(0, 1), y(0, 1)) = (x_3, y_3)$.

Applying the 2-dimensional substitution formula for integrals, we can now express the integral of f over Δ by an integral over Δ_{ref}:

$$\iint_{\Delta} f(x, y)\, dy\, dx = \iint_{\Delta_{\text{ref}}} |\delta(\xi, \eta)|\, f(x(\xi, \eta), y(\xi, \eta))\, d\eta\, d\xi$$

with the Jacobian determinant $\delta(\xi, \eta)$ of the coordinate transformation $(\xi, \eta) \mapsto (x, y)$. In our case, $\delta(\xi, \eta)$ is constant:

$$\delta(\xi, \eta) \equiv \delta = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1),$$

which corresponds to the ratio of the areas of the two triangles. Thus,

$$\iint_{\Delta} f(x, y)\, dy\, dx = |\delta| \cdot \int_{\xi=0}^{1} \int_{\eta=0}^{1-\xi} f(x(\xi, \eta), y(\xi, \eta))\, d\eta\, d\xi\,.$$

a) *Design a procedure* `triangleint`(*Delta,f*) *which computes the integral in this way. Specify the vertices of the triangle in form of a list,* `[[`x_1, y_1`],[`x_2, y_2`],[`x_3, y_3`]]`.

b) If the integral cannot be computed exactly, one approximates it by replacing f by a simpler function. A very basic variant is to replace f by a an affine interpolant of the form

$$p(x, y) = a + b\,x + c\,y$$

chosen in such a way that $p(x_j, y_j) = f(x_j, y_j),\ j = 1, 2, 3$. We claim that the integral over p can be written in the form

$$\iint_\Delta p(x, y)\,dy\,dx = Q(p) := \omega_1\,p(P_1) + \omega_2\,p(P_2) + \omega_3\,p(P_3)\,. \tag{Q}$$

Determine the parameters $\omega_1, \omega_2, \omega_3$ *such that* (Q) *is indeed valid, first for* $\Delta = \Delta_{ref}$ *and then for an arbitrary* Δ.

Hint: Using (Q) as an ansatz, consider the functions $p(x, y) = 1$, $p(x, y) = x$, and $p(x, y) = y$. This gives you 3 linear equations for the coefficients ω_j.

Exercise 7.8: ***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` or `geometry`. Aficionados of combinatorics may look at `combinat` (see also `combstruct`). And there are many, many more.