Beispiel 20

December 15, 2009

1 Computermathematik (für Informatik)

4. Übungsblatt (Musterlösung)
9. 12. 2009
Die heutigen Übungen sollen mit dem Computeralgebrasystem **Sage** gelöst werden.
Die Lösung der Beispiele soll auf möglichst kompakte Weise erfolgen. Wenn zum Beispiel eine Funktion für mehrere Werte berechnet werden soll, soll das mittels einer geeigneten Schleifen oder Listen Operation erfolgen, und **nicht** alle Werte einzeln eingetippt werden.
Zwischenergebnisse welche in einem weiteren Berechnungsschritt benötigt werden, sollen in eine Variable gespeichert und weiterverwendet werden (**nicht** neu eintippen).

Beispiel 20

Die *Legendre-Polynome* sind definiert durch die Rekursionsformel:
$L_0(x) = 1$, $L_1(x) = x$ und für $n > 1$:

$$(n+1)L_{n+1}(x) = (2n+1)xL_n(x) - nL_{n-1}(x)$$

Schreiben Sie eine Funktion **legendre(n)** die das n -te Legendre-Polynom berechnet.

(a) Berechnen Sie die ersten 10 Legendre-Polynome.

(b) Schreiben Sie eine Tabelle der ersten 10 Legendre-Polynome in die Datei **tabelle.tex**. Schreiben Sie ein LaTeX-Dokument legendre.tex welches die Datei tabelle.tex inkludiert und die Tabelle ausgibt. Das Ergebnis soll ungefähr folgendermassen aussehen:

$$\begin{aligned} L_0(x) &= 1 \\ L_1(x) &= x \\ &\vdots \end{aligned}$$

(c) Erzeugen Sie den Graphen der Legendre-Polynome $L_0(x), \ldots, L_5(x)$ (alle Polynome sollen gemeinsam in ein Bild gezeichnet werden).

(d) Finden Sie heraus wie man aus Sage Grafiken in eine PDF-Datei exportieren kann. Exportieren Sie den Graphen aus Punkt (c), und binden Sie in das Dokument legendre.tex ein.

(e) Berechnen Sie legendre(200).

Iterative Lösung

Sage code

```
def legendre(n):
    if n == 0:
        return 1

    l_old, l_new = 1, x

    for i in xrange(1, n):
        l_old, l_new = l_new, (2*i+1)/(i+1) * x * l_new - i/(i+1) * l_old

    return l_new
```

Sage code

```
time p = legendre(200)
```

```
Time: CPU 0.42 s, Wall: 0.43 s
```

Rekursive Lösung (sehr langsam)

Sage code

```
P.<x> = QQ[]

def legendre(n):
  if n == 0:
    return 1
  if n == 1:
    return x
  else:
    return 1/n*((2*n - 1) * x * legendre(n - 1) - (n-1)*legendre(n - 2))
```

Sage code

```
time p = legendre(10)
```

```
Time: CPU 0.03 s, Wall: 0.03 s
```

Sage code

```
time p = legendre(20)
```

```
Time: CPU 3.40 s, Wall: 3.40 s
```

Sage code

```
time p = legendre(22)
```

```
Time: CPU 8.87 s, Wall: 8.87 s
```

Eine direkte rekursive Implementation hat exponentielle Laufzeit, und ist somit unbrauchbar.

Rekursive Lösung mit Memoization:

Sage code

```
P.<x> = QQ[]

legendre_cache = {}

def legendre(n):

    def legendre_local(n):
        if n not in legendre_cache:
            legendre_cache[n] = legendre(n)
        return legendre_cache[n]
```

```
    if n == 0:
        return P(1)
    if n == 1:
        return x
    else:
        return 1/n*((2*n-1)*x*legendre_local(n-1) - (n-1)*legendre_local(n-2))
```

Sage code

```
time p = legendre(30)
```

```
Time: CPU 0.01 s, Wall: 0.01 s
```

Rekursive Lösung mit Memoization Decorator

Sage code

```
def remember(f, cache = {}):
    def g(*args):
        key = (f, tuple(args))
        if key not in cache:
            cache[key] = f(*args)
        return cache[key]
    return g
```

Sage code

```
P.<x> = QQ[]

@remember
def legendre(n):
    if n == 0:
        return P(1)
    if n == 1:
        return x
    else:
        return 1/n*((2*n - 1) * x * legendre(n - 1) - (n-1)*legendre(n - 2))
```

Sage code

```
time p = legendre(200)
```

```
Time: CPU 0.13 s, Wall: 0.13 s
```

Es gibt einen Memoization decorator auch vordefiniert in Sage.

Sage code

```
P.<x> = QQ[]

from sage.misc.all import cached_function

@cached_function
def legendre(n):
  if n == 0:
    return P(1)
  if n == 1:
    return x
  else:
    return 1/n*((2*n - 1) * x * legendre(n - 1) - (n-1)*legendre(n - 2))
```

Sage code

```
html.table([legendre(i)] for i in range(10))
```

$$1$$

$$x$$

$$\frac{3}{2}x^2 - \frac{1}{2}$$

$$\frac{5}{2}x^3 - \frac{3}{2}x$$

$$\frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}$$

$$\frac{63}{8}x^5 - \frac{35}{4}x^3 + \frac{15}{8}x$$

$$\frac{231}{16}x^6 - \frac{315}{16}x^4 + \frac{105}{16}x^2 - \frac{5}{16}$$

$$\frac{429}{16}x^7 - \frac{693}{16}x^5 + \frac{315}{16}x^3 - \frac{35}{16}x$$

$$\frac{6435}{128}x^8 - \frac{3003}{32}x^6 + \frac{3465}{64}x^4 - \frac{315}{32}x^2 + \frac{35}{128}$$

$$\frac{12155}{128}x^9 - \frac{6435}{32}x^7 + \frac{9009}{64}x^5 - \frac{1155}{32}x^3 + \frac{315}{128}x$$

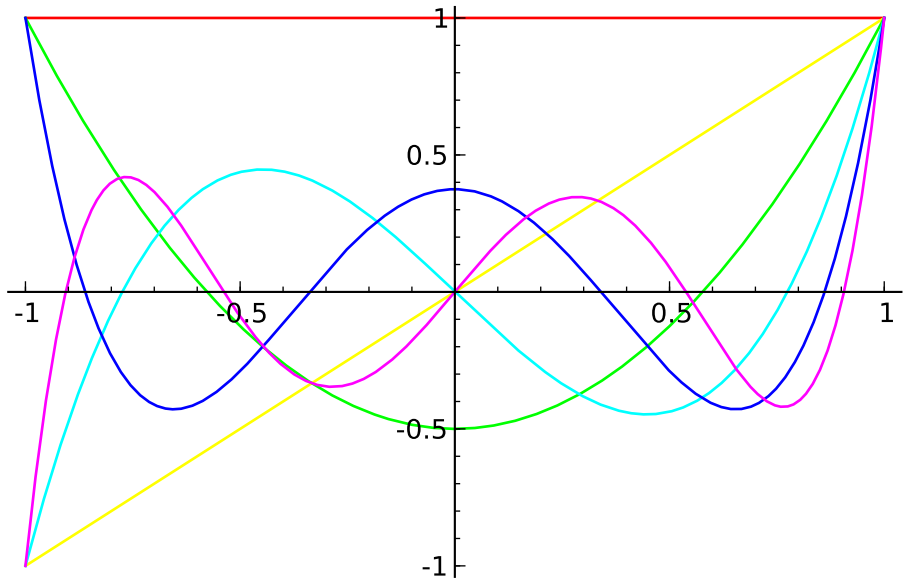
Sage code

```
datei = open("tabelle.tex", 'w')
datei.write("\\begin{align*}\n")
for i in [0..10]:
  datei.write("T_{%d}(x) &= %s \\\\\n" % (i, latex(legendre(i))))
datei.write("\\end{align*}\n")
datei.close()
```

Sage code

```
g = Graphics()
for i in [0..5]:
  g = g + legendre(i).plot(-1, 1, hue = i/6)

g.show()
```



Sage code

```
g.save('legendre.pdf')
```