LaTeX–Anhang: GnuPlot

10.11 Plot

`help plot` liefert als Syntaxbeschreibung

```
plot {<ranges>}
    {<iteration>}
    {<function> | {"<datafile>" {datafile-modifiers}}}
    {axes <axes>} {<title-spec>} {with <style>}
    {, {definitions{,}} <function> ...}
```

10.12 Ranges – Definitions– und Wertebereich

- `<ranges>` steht für eine Angabe `[xa:xb]` oder `[xa:xb][ya:yb]`, Koordinate $*$ steht für automatisch.
- Keine Angabe bedeutet `xrange [-10:10]`, `yrange` automatisch bzw. bei Datenreihen beides automatisch.
- Können global gesetzt werden mit `set xrange [xa:xb]` etc.

10.13 Variable und benutzerdefinierte Funktionen

Gnuplot hat, in schmalem Umfang, die Fähigkeiten einer Programmiersprache. Es können Variable (im Plot Konstante) und Funktionen definiert werden.

```
f(a,x)=sin(a*x)+sin((a+1)*x)
b=10
plot [-5:15] f(b,x)
```

10.14 Daten(reihen)dateien

Datendateien sind Textdateien mit einer konstanten Zahl von Zahlen pro Zeile, getrennt durch Leerzeichen, Tabulatoren, Komma, Semikolon. Zeilen, die mit # beginnen, werden ignoriert.

```
# t      x           y
0.0     1.0          0.0
0.2     1.0         -0.2
0.4     0.96        -0.4
0.6     0.88        -0.592
0.8     0.7616      -0.768
1.0     0.608       -0.92032
1.2     0.423936    -1.04192
```

10.15 Auswahl der Spalten

```
plot 'e0200.dat'
plot 'e0200.dat' using 1:3
plot 'e0200.dat' using 1:3
plot 'e0200.dat' using 1:3 with lines
plot 'e0200.dat' using 1:3 with linespoints
plot 'e0200.dat' using 2:3
plot 'e0200.dat' using 1:(sqrt($2**2+$3**2))
```

Ein Feld kann in Formeln mit Präfix `$` verwendet werden, zur Unterscheidung von numerischen Konstanten.

10.16 Erzeugen von Grafikdateien

eps – Encapsulated PostScript – für druckbare Artikel
pdf – aus eps per epstopdf – erhält Vektorgrafik
svg – Scalable Vector Graphics – für HTML–Seiten

```
set term push
set term post port enh color lw 2 32 size 10,10

set output "euler/e0200_ty.eps"

plot 'euler/e0200.dat' using 1:3 with linespoints lw 5 ps 3

unset output
set term pop
```

```
set term push
set term svg enh lw 2 fsize 17

set output "euler/e0200_ty.svg"

plot 'euler/e0200.dat' using 1:3 with linespoints lw 3 ps 1.5

unset output
set term pop
```

10.17 Kurvenanpassung

```
h=0.2

f(a,b,x)=cos((1-a)*x)*exp(b*x)

a=0.01, b=0.05
fit f(a,b,x) './euler/e0200.dat' via a,b

plot './euler/e0200.dat', f(a,b,x), f(h*h/3,h/2,x)
```

11 Algorithmen und Komplexität

11.1 Polynommultiplikation

$$\sum_{j=0}^{n} a_j x^j \cdot \sum_{k=0}^{n} b_k x^k = \sum_{m=0}^{2n} \sum_{j} a_j b_{m-j}$$

erfordert n^2 Multiplikationen und etwa genausoviele Additionen

Karatsuba (1960)

$$(a_0 + a_1 x)(b_0 + b_1 x) = (a_0 b_0) + (a_0 b_1 + a_1 b_0)x + a_1 b_1 x^2 = c_0 + c_1 x + c_2 x^2$$

und für $x = 1$

$$(a_0 + a_1)(b_0 + b_1) = (a_0 b_0) + (a_0 b_1 + a_1 b_0) + a_1 b_1$$

Berechne also $c_0 = a_0 b_0$, $c_2 = a_1 b_1$ sowie $c_1 = (a_0 + a_1)(b_0 + b_1) - c_0 - c_2$

Vorher also 4 Multiplikationen und eine Addition, nun 3 Multiplikationen und 4 Additionen.

Längere Polynome spaltet man in der Mitte, ist deren Grad $deg(a), \deg(b) \leq n - 1 = 2d - 1$, d.h. sie haben maximal $2d$ Koeffizienten, dann

$$a(x) = a^0(x) + x^d a^1(x), \quad a^j(x) = sum_{k=0}^{d-1} a_{jd+k} x^k$$

$$b(x) = b^0(x) + x^d b^1(x), \quad b^j(x) = sum_{k=0}^{d-1} b_{jd+k} x^k$$

und wendet auf diese Zerlegung den Karatsuba–Trick an.

Berechne also $c_0(x) = a_0(x) b_0(x)$, $c_2(x) = a_1(x) b_1(x)$ sowie $c_1(x) = (a_0(x) + a_1(x))(b_0(x) + b_1(x)) - c_0(x) - c_2(x)$. Die Auftretenden drei Polynomprodukte mit Graden kleinergleich $d - 1$ werden rekursiv wieder mit dem Karatsuba–Trick bestimmt.

11.2 Laufzeitanalyse

Annahme: In Gleitkommaarithmetik kostet eine Addition $c < 1$ Anteile einer Multiplikation.

Betrachtet man die Aufspaltung des naiven Algorithmus, so wird eine Polynommultiplikation der Länge $2d$ in vier Multiplikationen der Länge d und 3 Additionen der Länge $2d$ aufgespalten.

$$M(2d) = 4M(d) + 6cd$$

$$m_k = M(2^k)/4^k \implies m_{k+1} = m_k + \frac{3c}{2^{k+1}} = \cdots = m_0 + 3c(2^{-1} + 2^{-2} + \cdots + 2^{-k-1}$$

$$\implies m_k \leq m_0 + 3c \implies M(2^k) \leq 4^k(1 + 3c) \implies M(d) \leq d^2(1 + 3c)$$

Die Aufspaltung im Karatsuba-Verfahren erfordert 2 Additionen der Länge d, 3 Multiplikationen in Länge d und eine Addition der Länge $2d$. Denn im Ergebnis überlappen sich das vordere Produkt $a_0 b_0$ vom Grad $2d - 1$ und das hintere Produkt $x^{2d} a_1 b_1$ nicht. Es muss also nur das mittlere Produkt mit Koeffizienten der Grade d bis $3d - 2$ hinzuaddiert werden.

$$M(2d) = 3M(d) + 4cd$$

$$m_k = M(2^k)/3^k \implies m_{k+1} = m_k + 2c\left(\frac{2}{3}\right)^{k+1} = \cdots = m_0 + 3c(2/3 + (2/3)^2 + \cdots + (2/3)^{-k-1}$$

$$\implies m_k \le m_0 + 6c \implies M(2^k) \le 3^k(1+6c) \implies M(d) \le d^{\log_2(3)}(1+6c)$$

Wegen $1+6c \le 2(1+3c)$ ist Karatsuba spätestens dann schneller, wenn $4^k > 2 \cdot 3^k$. Das ist bei $k = 3$ der Fall, d.h. ab Grad $7 = 2^3 - 1$ ist Karatsuba schneller als die naive Multplikation.

11.3 Determinante

nach Laplace: 2^n Zwischenergebnisse, $(n-1)(n-1)!$ Multiplikationen

Leverrier–Faddejew: $n^4 + O(n^3)$ Multiplikationen

Die Determinante einer quadratischen Matrix $A \in \mathbb{R}^{n\times n}$ ist ist ein Polynom in den Einträgen von A vom Grad n.

Die Adjunkte $A^\#$ einer quadratischen Matrix $A \in \mathbb{R}^{n\times n}$ ist die Transponierte der Kofaktormatrix, d.h. diejenige Matrix, die

$$A^\# \cdot A = \det(A)\, I \tag{1}$$

erfüllt. Die Einträge von $A^\#$ sind Polynome in den Einträgen von A vom Grad $n-1$.

Die Spur (en: trace) $\operatorname{tr}(A)$ einer quadratischen Matrix $A \in \mathbb{R}^{n\times n}$ ist die Summe der Diagonaleinträge. Insbesondere gilt für die Einheitsmatrix $\operatorname{tr}(I) = n$.

Ableitung der Determinante: Für differenzierbare matrixwertige Funktionen $H : (a,b) \to \mathbb{R}^{n\times n}$ gilt

$$\frac{d}{dt}\det(H(t)) = \operatorname{tr}\left(H(t)^\# \cdot H'(t)\right). \tag{2}$$

Für die Matrixfunktion $H(t) = tI - A$ gilt also:

- $(tI-A)^\#$ ist ein Polynom in t mit Matrixkoeffizienten,
$$(tI-A)^\# = B_0 + B_1 t + \cdots + B_{n-1}t^{n-1}, \tag{3}$$
- $\det(tI-A)$ ist ein (skalares) Polynom vom Grad n in t mit gradhöchstem Koeffizienten 1,
$$\det(tI-A) = c_0 + c_1 t + \cdots + c_{n-1}t^{n-1} + t^n \tag{4}$$

Auswerten von (1) ergibt für die Koordinaten aus (3)

$$\begin{aligned}\det(tI-A)I &= (tI-A)^\# \cdot (tI-A)\\ &= -B_0A + (B_0 - B_1A)\lambda + (B_1 - B_2A)\lambda^2 + \cdots + (B_{n-2} - B_{n-1}A)\lambda^{n-1} + B_{n-1}\lambda^n \qquad (5)\\ &\qquad (6)\end{aligned}$$

Im Vergleich mit (4) also

$$B_{n-1} = I, \qquad B_k = c_{k-1}I + B_{k-1}A,\ k = n-2,\ldots,1,0,\ \text{und}\ 0 = c_0 I + B_0 A \tag{7}$$

Die Ableitung der Determinante nach (2) kann, nach Einsetzen der Koeffizientendarstellung (3), mit der Ableitung der Polynomdarstellung (4) verglichen werden:

$$\begin{aligned}t\tfrac{d}{dt}\det(tI-A) &= \operatorname{tr}\left((tI-A)^\# \cdot (tI-A+A)\right) = \det(tI-A)\operatorname{tr}(I) + \operatorname{tr}\left((tI-A)^\# A\right)\\ \left(n - t\tfrac{d}{dt}\right)\det(tI-A) &= c_{n-1}t^{n-1} + 2c_{n-2}t^{n-2}\cdots + (n-1)c_1 t + nc_0 \qquad (8)\\ &= -\operatorname{tr}\left((tI-A)^\# A\right) = -\operatorname{tr}(B_{n-1}A))t^{n-1} - \cdots - \operatorname{tr}(B_1A)t - \operatorname{tr}(B_0A) \qquad (9)\end{aligned}$$

nach Koeffizientenvergleich also

$$c_{n-1} = -\operatorname{tr}(B_{n-1}A) = -\operatorname{tr}(A),\ 2c_{n-2} = -\operatorname{tr}(B_{n-2}A),\ \ldots,\ (n-1)c_1 = -\operatorname{tr}(B_1A),\ nc_0 = -\operatorname{tr}(B_0A) \tag{10}$$

Methode von Leverrier–Faddejew: Starte mit $B = I$. Für $k = n-1$ absteigend bis 0 führe aus:

$$C = BA,\ c_k = -\tfrac{1}{n-k}\operatorname{tr}(C),\ B = C + c_k I$$

Am Ende kann man auf $B = 0$ prüfen. Läßt man die letzte Berechnung von B weg, so ist $-B/c_0$ die Inverse von A.