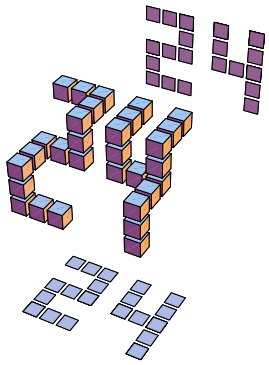
Kapitel 24

Reihenentwicklungen



Bei vielen Berechnungen treten Funktionen auf, die mathematisch nur mit sehr hohem Aufwand (oder überhaupt nicht) weiterbearbeitet werden können. Vor allem in technischen Anwendungen können solche Funktionen oft durch besser handhabbare Reihenentwicklungen ersetzt werden.

`series`
führt eine Taylor-, Laurent- oder eine verallgemeinerte Potenzreihenentwicklung durch. Eine mögliche Anwendung von `series` besteht in der Lösung von Differentialgleichungen.

`mtaylor` und `poisson`
führen multivariable Taylor-Reihenentwicklungen durch.

`powseries`
bezeichnet ein Package, das verschiedene Kommandos zur Bearbeitung von formalen Potenzreihen enthält.

`numapprox`
stellt ein Package dar, das Kommandos zur numerischen Berechnung von (zumeist rationalen) Näherungsfunktionen enthält.

Verweis: Bereits in Kapitel 15 wurde das Kommando `eulermac` behandelt, das eine Euler-Maclaurin-Reihe für eine allgemeine Summenformel erstellt. Informationen zur Fourier-Reihenentwicklung für periodische Funktionen finden Sie im nächsten Kapitel.

Taylor-, Laurent- und allgemeine Potenzreihenentwicklung

Das Standardkommando zum Aufstellen von Reihenentwicklungen lautet `series`. Es produziert je nach der Form der zu entwickelnden Funktion eine Taylor-, eine Laurent- oder eine allgemeine Potenzreihe (jeweils auch komplex). Dem Kommando wird im ersten Parameter die zu entwickelnde Funktion und im zweiten Parameter der Entwicklungspunkt übergeben. Im dritten Parameter kann die maximale Ordnung der Entwicklung angegeben werden. Wird darauf verzichtet, greift Maple dafür auf die globale Variable `Order` zurück (Voreinstellung 6). Für die Entwicklung von Funktionen in Reihen und vor allem für das Weiterarbeiten mit diesen Reihen ist es wichtig, sich mit den verschiedenen Typen vertraut zu machen.

```
typen:=x->[type(x,taylor),type(x,laurent),type(x,series),whattype(x)];
series(sin(x),x,10); typen(%);
```

$$x - \frac{1}{6}\,x^3 + \frac{1}{120}\,x^5 - \frac{1}{5040}\,x^7 + \frac{1}{362880}\,x^9 + \mathrm{O}(x^{10})$$

$$[true,\ true,\ true,\ series]$$

```
series(z/((z-(1+I))^2*(z-2)),z=1+I,2); typen(%);
```

$$-I\,(z-1-I)^{-2} - I\,(z-1-I)^{-1} + \frac{1}{2} - \frac{1}{2}\,I + \frac{1}{2}\,(z-1-I) + \mathrm{O}((z-1-I)^2)$$

$$[false,\ true,\ true,\ series]$$

```
series(sin(sqrt(x)),x,4); typen(%);
```

$$\sqrt{x} - 1/6\,x^{3/2} + \frac{1}{120}\,x^{5/2} - \frac{1}{5040}\,x^{7/2} + \frac{1}{362880}\,x^{9/2} + O(x^5)$$

$$[false, false, false, \text{`+`}]$$

Weitere Beispiele finden Sie im Worksheet. Es gibt auch den Entwicklungspunkt ∞ (in diesem Fall ruft `series` das Kommando `asympt` auf, um die Reihenentwicklung durchzuführen).

```
series(arctan(x), x=infinity);
```

$$\frac{1}{2}\,\pi - \frac{1}{x} + \frac{1}{3}\,\frac{1}{x^3} - \frac{1}{5}\,\frac{1}{x^5} + \mathrm{O}\left(\frac{1}{x^6}\right)$$

Weiterverarbeitung von Reihen

Mit `series` erhält man normalerweise ein Ergebnis im Datentyp `series`. Dabei wird eine besonders effiziente Speicherung der Reihe (in Form des Entwicklungsorts und der Größe

der Koeffizienten) verwendet. Aus diesem Grund kann mit den Ergebnissen von `series` nicht unmittelbar weitergearbeitet werden.

```
f:=series(sin(x), x=Pi);
```

$$f := -x + \pi + \frac{1}{6}(x-\pi)^3 - \frac{1}{120}(x-\pi)^5 + O\left((x-\pi)^6\right)$$

```
evalf(subs(x=2, f));
```

$$0.9097898165 + O((2-\pi)^6)$$

```
whattype(f);
```

$$series$$

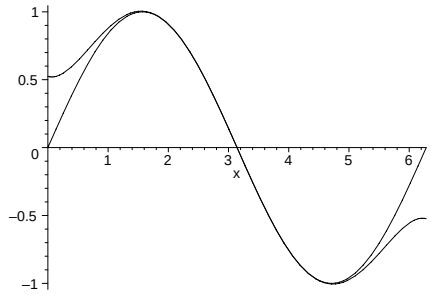
Mit dem Kommando `convert/polynom` kann eine Umwandlung in ein normales Polynom ohne Ordnungsterm durchgeführt werden. Dieses Polynom kann in der Folge zum Rechnen, Vereinfachen, Differenzieren, Zeichnen etc. verwendet werden.

```
p:=convert(f, polynom);
```

$$p := -x + \pi + \frac{(x-\pi)^3}{6} - \frac{(x-\pi)^5}{120}$$

Die Abbildung zeigt die Reihenentwicklung von $\sin(x)$ um den Punkt $x = \pi$ und die Originalfunktion.

```
plot({p, sin(x)}, x=0..2*Pi);
```



Ein weiterer Sonderfall tritt ein, wenn `series` bei einer Reihenentwicklung zu einem exakten Ergebnis kommt. Das kann beim Rechnen mit Reihen vorkommen oder einfach so erzeugt werden:

```
f:=series(x^2, x=3);
```

$$f := 9 + 6\ (x-3) + (x-3)^2$$

```
simplify(f);
```

$$9 + 6\ (x-3) + (x-3)^2$$

Da das Ergebnis exakt ist, entfällt der Ordnungsterm. Das Ergebnis sieht deswegen aus wie ein normales Polynom. Dennoch wird es intern durch den `series`-Datentyp dargestellt und muss mit `convert` in ein Polynom verwandelt werden, bevor die Vereinfachung zur Ausgangsfunktion x^2 gelingt.

```
whattype(f);
```

$$series$$

```
convert(f,polynom);
```

$$-9+6\,x+(x-3)^2$$

```
simplify(%);
```

$$x^2$$

Innerhalb von `series` können Sie mit Reihen ähnlich wie mit anderen mathematischen Formeln arbeiten: Erlaubt sind nicht nur Operationen mit skalaren Größen (z.B. die Multiplikation der Reihe mit einem Faktor), sondern auch die Verknüpfung von mehreren Reihen (Addition, Multiplikation ...). In den folgenden Beispielen werden unter anderem die Reihen zu $\sin(x)$ und $\cos(x)$ miteinander multipliziert. Zum selben Ergebnis käme auch das Kommando `series(sin(x)*cos(x), x=Pi/2)`.

```
s1:=series(sin(x),x=Pi/2);
```

$$s1 := 1-\frac{1}{2}\left(x-\frac{1}{2}\,\pi\right)^2+\frac{1}{24}\left(x-\frac{1}{2}\,\pi\right)^4+\mathrm{O}\left(\left(x-\frac{1}{2}\,\pi\right)^6\right)$$

```
s2:=series(cos(x), x=Pi/2);
```

$$s2 := -\left(x-\frac{1}{2}\,\pi\right)+\frac{1}{6}\left(x-\frac{1}{2}\,\pi\right)^3-\frac{1}{120}\left(x-\frac{1}{2}\,\pi\right)^5+\mathrm{O}\left(\left(x-\frac{1}{2}\,\pi\right)^6\right)$$

```
series(1/s1, x=Pi/2);  #Reihe zu 1/sin(x)
```

$$1+\frac{1}{2}\left(x-\frac{1}{2}\,\pi\right)^2+\frac{5}{24}\left(x-\frac{1}{2}\,\pi\right)^4+\mathrm{O}\left(\left(x-\frac{1}{2}\,\pi\right)^6\right)$$

```
series(s1+x^2, x=Pi/2);  #Reihe zu sin(x)+x^2
```

$$\left(1+\frac{1}{4}\,\pi^2\right)+\pi\left(x-\frac{1}{2}\,\pi\right)+\frac{1}{2}\left(x-\frac{1}{2}\,\pi\right)^2+\frac{1}{24}\left(x-\frac{1}{2}\,\pi\right)^4+\mathrm{O}\left(\left(x-\frac{1}{2}\,\pi\right)^6\right)$$

```
series(s1*s2, x=Pi/2);  #Reihe zu sin(x)*cos(x)
```

$$-\left(x-\frac{1}{2}\,\pi\right)+\frac{2}{3}\left(x-\frac{1}{2}\,\pi\right)^3-\frac{2}{15}\left(x-\frac{1}{2}\,\pi\right)^5+\mathrm{O}\left(\left(x-\frac{1}{2}\,\pi\right)^6\right)$$

```
s1:=series(sin(x),x=0);
series(arcsin(s1), x=0); # Umkehrfunktion
simplify(convert(%, polynom));
```

$$x+\mathrm{O}(x^6)$$

$$x$$

Differentialgleichungen mit Reihenentwicklungen lösen

Eine praktische Anwendung von `series` besteht in der Lösung von Differentialgleichungen durch einen Reihenansatz. Wir wollen das mit Maple an einer etwas ungewöhnlichen DGL versuchen.

```
de:=y(x)+diff(y(x),x)=sin(sqrt(x));
dsolve({de,y(0)=0},y(x));
```

$$de := \mathrm{y}(x) + \frac{\partial}{\partial x}\,\mathrm{y}(x) = \sin(\sqrt{x})$$

$$\mathrm{y}(x) = e^{(-x)} \int_0^x \sin(\sqrt{u})\, e^u \, du$$

Es gibt keine Stammfunktion

```
integral:=int(sin(sqrt(x))*exp(x),x);
```

$$integral := \int \sin(\sqrt{x})\, e^x \, dx \quad ,$$

und deshalb auch keine Reihe:

```
series(integral,x);
Error, (in series/int) unable to compute series
```

Nun gibt es mehrere Möglichkeiten:

1. Die rechte Seite (Inhomogenität) in eine Reihe entwickeln.

```
sr:=convert(series(sin(sqrt(x)),x,9),polynom);
```

$$sr := \sqrt{x} - \frac{1}{6}\,x^{(3/2)} + \frac{1}{120}\,x^{(5/2)} - \frac{1}{5040}\,x^{(7/2)} + \frac{1}{362880}\,x^{(9/2)} - \frac{1}{39916800}\,x^{(11/2)} +$$

$$\frac{1}{6227020800}\,x^{(13/2)} - \frac{1}{1307674368000}\,x^{(15/2)} + \frac{1}{355687428096000}\,x^{(17/2)}$$

```
de:=y(x)+diff(y(x),x)=sr:
sol:=rhs(dsolve({de,y(0)=0},y(x)));
```

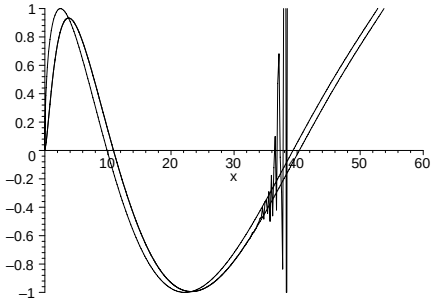
$$sol := \frac{3392923553}{5284823040}\,I\,e^{(-x)}\,\sqrt{\pi}\,\mathrm{erf}(I\,\sqrt{x}) + \frac{3392923553}{2642411520}\,\sqrt{x} - \frac{7333793}{34681651200}\,x^{(7/2)} +$$

$$\frac{89909153}{9909043200}\,x^{(5/2)} - \frac{750512033}{3963617280}\,x^{(3/2)} + \frac{929}{5579410636800}\,x^{(13/2)} - \frac{22433}{858370867200}\,x^{(11/2)} +$$

$$\frac{452513}{156067430400}\,x^{(9/2)} + \frac{1}{355687428096000}\,x^{(17/2)} - \frac{1}{1268047872000}\,x^{(15/2)}$$

Die Fehlerfunktion `erf` stört vielleicht etwas, also können wir nochmals in eine Reihe entwickeln (siehe Worksheet) und erhalten folgende Lösung:

```
plot([sr,sol,psol],x=0..60,-1..1);
```



Dabei ist `sr` die rechte Seite der DGL, die Lösung `sol` (Fehlerfunktion ohne Entwicklung) folgt ihr verzögert und die Entwicklung der Fehlerfunktion (`psol`) verursacht starke Oszillationen. Die Lösung mit der nicht entwickelten Fehlerfunktion ist dagegen (je nach Rechengenauigkeit) auch in einem zehnfach größeren Bereich stabil.

Zwei weitere Möglichkeiten finden Sie im Worksheet: 2. Entwicklung des Integranden, der in dem von Maple nicht ausgewerteten Integral vorkommt und 3. Aufstellen einer benutzerdefinierten Reihe für die linke Seite der DGL (und Koeffizientenvergleich mit `match`). Für welche Methode man sich entscheidet, wird letzten Endes immer eine Frage der geforderten Genauigkeit und des Rechenaufwands sein.

Multivariable Taylor-Reihenentwicklung

Das Kommando `mtaylor` führt Taylorentwicklungen für multivariable Funktionen durch (während `series` nur für Funktionen einer Variablen ausgelegt ist). Der Aufruf des Kommandos erfolgt prinzipiell wie bei `series`, im zweiten Parameter muss allerdings eine Liste der Entwicklungspunkte der betroffenen Variablen angegeben werden. Im optionalen dritten Parameter wird wiederum die Ordnung der Reihenentwicklung angegeben. Beachten Sie, dass bei der Defaulteinstellung 6 bereits sehr umfangreiche Ausdrücke mit bis zu 21 Termen entstehen. Das Kommando liefert das Ergebnis als normales Polynom ohne Ordnungsterme.

```
mtaylor(log(x+y), [x=1, y=Pi], 3);
```

$$\ln(1+\pi)+\frac{x-1}{1+\pi}+\frac{y-\pi}{1+\pi}-\frac{(x-1)^2}{2\,(1+\pi)^2}-\frac{(y-\pi)\,(x-1)}{(1+\pi)^2}-\frac{(y-\pi)^2}{2\,(1+\pi)^2}$$

Das Kommando `poisson` stellt eine Variante zu `mtaylor` dar. Der wesentliche Unterschied besteht darin, dass bei trigonometrischen Funktionen die Koeffizienten nach der kanonischen Fourier-Form angeordnet werden. `poisson` ist im Gegensatz zu `mtaylor` nicht in der Lage, Reihenentwicklungen um einen beliebigen Punkt durchzuführen. Außerdem ist die Auswahl der Funktionen, für die überhaupt eine Entwicklung durchgeführt werden kann, im Vergleich `mtaylor` stark eingeschränkt.

Das folgende Beispiel führt eine Reihenentwicklung für die Funktion $\sin(a+x)*\cos(b+y)$ sowohl mit `poisson` als auch mit `mtaylor` durch. Mit `combine/trig` lässt sich zeigen, dass die beiden Ergebnisse identisch sind.

```
f := sin(a+x)*cos(b+y);
f1:=poisson(f, [x,y], 3 );
```

$$f1 := \frac{\sin(a+b)}{2} + \frac{\sin(a-b)}{2} + \left(-\frac{\cos(a-b)}{2} + \frac{\cos(a+b)}{2}\right) y + \left(-\frac{\sin(a+b)}{2} + \frac{\sin(a-b)}{2}\right) yx + \left(\frac{\cos(a-b)}{2} + \frac{\cos(a+b)}{2}\right) x + \left(-\frac{\sin(a+b)}{4} - \frac{\sin(a-b)}{4}\right) x^2 + \left(-\frac{\sin(a+b)}{4} - \frac{\sin(a-b)}{4}\right) y^2$$

```
f2:=mtaylor(f, [x,y], 3);
```

$$f2 := \sin(a)\cos(b) - \sin(a)\sin(b)y + \cos(a)x\cos(b) - \frac{\sin(a)\cos(b)y^2}{2} - \cos(a)x\sin(b)y - \frac{\sin(a)x^2\cos(b)}{2}$$

```
combine(f1-f2,trig);
```

$$0$$

Formale Reihen

Das Package `powseries` enthält Kommandos zur Bearbeitung formaler Reihen. Damit sind Reihen gemeint, die durch eine allgemeine Entwicklungsformel definiert werden. Das Package `powseries` ermöglicht es, mit solchen Reihen zu rechnen. Erlaubte Rechenschritte sind unter anderem Addition, Subtraktion, Multiplikation und Division zweier Reihen, das Logarithmieren von Reihen, das Bilden von Ableitung und Integral, das Bilden der inversen Reihe. Der Vorteil von formalen Reihen besteht darin, dass sie nach Ende der Berechnung in beliebiger (!) Ordnung ausgewertet werden können.

```
with(powseries);
```

$[compose, evalpow, inverse, multconst, multiply, negative, powadd, powcos, powcreate, powdiff, powexp, powint, powlog, powpoly, powsin, powsolve, powsqrt, quotient, reversion, subtract, template, tpsform]$

Die beiden elementarsten Kommandos des Package lauten `powcreate` und `tpsform`. Mit `powcreate` wird eine Prozedur erzeugt, die die berechneten Koeffizienten und das Bildungsgesetz in der *remember table* ablegt.

```
powcreate( p(n)=1/n!);
seq(p(i),i=0..6);
```

$$1,\ 1,\ \frac{1}{2},\ \frac{1}{6},\ \frac{1}{24},\ \frac{1}{120},\ \frac{1}{720}$$

```
p(100);
```

$$1/933262154439441526816992388562667004907159682643816214685929638\backslash$$
$$952175999932299156089414639761565182862536979208272237582511\backslash$$
$$852109168640000000000000000000000000$$

Hier steht das Bildungsgesetz:

```
p(_k);
```

$$\frac{1}{\_k!}$$

Und die *remember table* kann so eingesehen werden:

```
op(4,op(p));
```

$$\mathrm{table}([0=1,\ 1=1,\ 2=\frac{1}{2},\ 3=\frac{1}{6},\ 4=\frac{1}{24},\ 5=\frac{1}{120},\ 6=\frac{1}{720},\ \_k=\frac{1}{\_k!},$$

$$100=1/933262154439441526816992388562667004907159682643816214685929638\backslash$$
$$952175999932299156089414639761565182862536979208272237582511\backslash$$
$$852109168640000000000000000000000000])$$

So kann man sich auch die ganze Prozedur ausgeben lassen (Ausgabe hier nicht gedruckt):

```
interface(verboseproc=3): print(p); interface(verboseproc=1):
```

Die so gebildeten Koeffizienten können in der 'truncated powerseries form' dargestellt werden:

```
tpsform(p, x, 5);
```

$$1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4+\mathrm{O}(x^5)$$

Bevor eine neue Reihe mit dem alten Namen gebildet werden soll, sollte man den Namen wieder freigeben. Die Koeffizienten können auch rekursiv und mit Anfangsbedingungen angegeben werden:

```
p:='p':  powcreate( p(n)=p(n-1)*p(n-2), p(0)=1, p(1)=1/2);
tpsform(p, x, 5);
```

$$1+\frac{1}{2}x+\frac{1}{2}x^2+\frac{1}{4}x^3+\frac{1}{8}x^4+\mathrm{O}(x^5)$$

Zur Übung können wir nun unsere eigene Taylorreihe aufstellen. Wenn man nicht um $x = 0$ entwickelt, benötigt man allerdings einen kleinen Trick, um dies `tpsform` mitzuteilen.

```
powcreate(mytayl(n)=diff(sin(a),a$n)/n!,mytayl(0)=0):
seq(eval(mytayl(i),a=0),i=1..10);
```

$$1,\ 0,\ \frac{-1}{6},\ 0,\ \frac{1}{120},\ 0,\ \frac{-1}{5040},\ 0,\ \frac{1}{362880},\ 0$$

```
tpsform(mytayl,x);
```

$$\cos(a)\,x - \frac{1}{2}\sin(a)\,x^2 - \frac{1}{6}\cos(a)\,x^3 + \frac{1}{24}\sin(a)\,x^4 + \frac{1}{120}\cos(a)\,x^5 + \mathrm{O}(x^6)$$

```
eval(tpsform(mytayl,x),x=x-a);
```

$$\cos(a)\,(x-a) - \frac{1}{2}\sin(a)\,(x-a)^2 - \frac{1}{6}\cos(a)\,(x-a)^3 + \frac{1}{24}\sin(a)\,(x-a)^4 +$$
$$\frac{1}{120}\cos(a)(x-a)^5 + \mathrm{O}((x-a)^6)$$

Wenn man die Taylorformel nicht verwendet, muss man unter Umständen dafür sorgen, dass Koeffizienten Null werden:

```
powcreate(p(k)=1/k!*signum(sin(Pi/2*k)));
tpsform(p,x,9);  #series(sin(x),x);
```

$$x - \frac{1}{6}\,x^3 + \frac{1}{120}\,x^5 - \frac{1}{5040}\,x^7 + \mathrm{O}(x^9)$$

```
powcreate(q(k)=1/k!*signum(cos(Pi/2*k)));
```

$$1 - \frac{1}{2}\,x^2 + \frac{1}{24}\,x^4 - \frac{1}{720}\,x^6 + \frac{1}{40320}\,x^8 + \mathrm{O}(x^9)$$

Nun kann mit den Reihen gerechnet werden. Für algebraische Operationen verwendet man `evalpow`, für die Umkehrung `reversion`, für die Ableitung `powdiff` usw., wobei diese Befehle jeweils wieder eine kleine Prozedur erzeugen:

```
r:=evalpow(p*q);
```

$r :=$ **proc**$(powparm)$ ... **end proc**

```
tpsform(r,x,9);
```

$$x - \frac{2}{3}\,x^3 + \frac{2}{15}\,x^5 - \frac{4}{315}\,x^7 + \mathrm{O}(x^9)$$

```
series( sin(x)*cos(x), x, 9);
```

$$x - \frac{2}{3}\,x^3 + \frac{2}{15}\,x^5 - \frac{4}{315}\,x^7 + \mathrm{O}(x^9)$$

r:=reversion(p);

$$x + \frac{1}{6}\,x^3 + \frac{3}{40}\,x^5 + \frac{5}{112}\,x^7 + \mathrm{O}(x^9)$$

```
series( arcsin(x), x, 9);
```

$$x + \frac{1}{6}\,x^3 + \frac{3}{40}\,x^5 + \frac{5}{112}\,x^7 + \mathrm{O}(x^9)$$

```
r:=powdiff(p); tpsform(r,x,9);
```

$$1 - \frac{1}{2}\,x^2 + \frac{1}{24}\,x^4 - \frac{1}{720}\,x^6 + \frac{1}{40320}\,x^8 + \mathrm{O}(x^9)$$

Die Koeffizienten kann man sich auch anders besorgen: Die Übernahme von Reihen, die mit `series` gebildet wurden, erfolgt mit `powpoly`. Dabei kann man leicht 'beliebig viele' Koeffizienten zur Verfügung stellen:

```
s:=series( tan(x), x=0, 100):
r:=powpoly( convert(s,polynom), x ):
t:=evalpow(p+r):  t(81);
```

$$417527403333151351098563350293660679577328543206131760201204876021\backslash$$
$$606216432104998754655524723365930416039/\,252048957423798\backslash$$
$$608081727575286004743713580749770940259075907176074625659981\backslash$$
$$669095896766019714129330176000000000000000000000$$

```
tpsform(t,x,15);
```

$$2\,x + \frac{1}{6}\,x^3 + \frac{17}{120}\,x^5 + \frac{271}{5040}\,x^7 + \frac{7937}{362880}\,x^9 + \frac{353791}{39916800}\,x^{11} + \frac{22368257}{6227020800}\,x^{13} + \mathrm{O}(x^{15})$$

```
series(tan(x)+sin(x),x,15);
```

$$2\,x + \frac{1}{6}\,x^3 + \frac{17}{120}\,x^5 + \frac{271}{5040}\,x^7 + \frac{7937}{362880}\,x^9 + \frac{353791}{39916800}\,x^{11} + \frac{22368257}{6227020800}\,x^{13} + \mathrm{O}(x^{15})$$

Numerische Berechnung von Näherungsfunktionen

Das Package `numapprox` fasst verschiedene Funktionen zur Berechnung von Näherungsfunktionen zusammen:

```
with(numapprox); with(orthopoly):
```

[*chebdeg*, *chebmult*, *chebpade*, *chebsort*, *chebyshev*, *confracform*, *hermite_pade*, *hornerform*, *infnorm*, *laurent*, *minimax*, *pade*, *remez*]

- `minimax` erstellt eine rationale Näherungsfunktion, die die gegebene Funktion in einem Bereich ($x0..x1$) annähert.

- `pade` ist die Pade-Approximation zur Berechnung einer rationalen Näherungsfunktion. Dabei wird wie bei `series` um einen Punkt $x0$ entwickelt.
- `chebpade` berechnet ebenfalls eine rationale Näherungsfunktion für den Bereich ($x0..x1$). Die Näherungsfunktion ist aus Tschebyscheff-Polynomen zusammengesetzt.
- `chebyshev` berechnet eine Tschebyscheffsche Reihe für die angegebene Funktion im Bereich $x0..x1$.

Daneben enthält das Package einige weitere Kommandos: `taylor` und `laurent` stellen eingeschränkte Varianten des `series`-Kommandos dar. `hornerform` und `confracform` führen Polynome bzw. rationale Funktionen in die Hornerform bzw. in einen Kettenbruch über – siehe `convert/horner` bzw. `/confrac` in Kapitel 10. `remez` wird von `minimax` zur Optimierung der rationalen Funktion verwendet und wird nur in Sonderfällen direkt vom Anwender aufgerufen.

Anmerkungen: Die Rechenzeit der Kommandos ist fallweise ziemlich hoch. Eine Reduzierung von `Digits` kann erhebliche Verbesserungen bringen. Wenn die Kommandos statt eines Ergebnisses eine Fehlermeldung liefern, können Sie versuchen, eine kleine Veränderung der Ausgangsparameter (inklusive der Rechengenauigkeit) vorzunehmen.

In den folgenden Beispielen werden die verschiedenen Algorithmen verglichen. Als 'Härtetest' können wir z.B. die Funktion $\tan(x^2)$ mit der Unstetigkeit (Pol) im Punkt $x0 = \sqrt{\pi/2}$ nehmen. Wie üblich können Sie im Worksheet andere Funktionen einsetzen und alle Parameter ändern.

Wir definieren die Funktion, die Unstetigkeitsstelle und Bereichsangaben.

```
f:=x->tan(x^2);
```

$$f := x \rightarrow \tan(x^2)$$

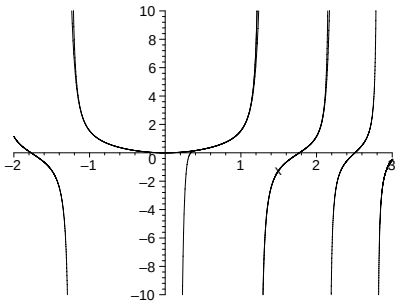
```
x0:=sqrt(Pi/2): a:=0: b:=1:
```

Nun kann mit `series` eine Laurentreihe um $x0$ und eine Taylorreihe um den Ursprung entwickelt werden.

```
sx0:=series(f(x),x=x0,40):
s:=series(f(x),x=a,40):
```

```
plot([f(x),convert(s,polynom),convert(sx0,polynom)],x=-2..3,-10..10,);
```

Die Laurentreihe ist am Pol nicht von der Funktion zu unterscheiden, hat aber für kleine x eine sehr große Abweichung von f(x). Die Taylorreihe approximiert die Funktion für kleine x sehr gut, gibt aber jenseits der Pole die Funktion nicht wieder.



Nun können wir untersuchen, inwiefern die Befehle aus dem Paket `numapprox` eine Verbesserung der Annäherung bringen. Die Prozedur `minimax` approximiert nach dem Remes-Algorithmus. Die erste Variante *mini0* ist ein Polynom fünften Grades. Bei der zweiten Variante *mini25* hat der Zähler den Grad 2 und der Nenner den Grad 5.

```
mini0:=minimax(f(x),x=a..x0-0.1,5,1,'fehler'); expand(mini0); fehler;
```

$$mini0 := -.100575884 + (4.867599482$$

$$+ (-36.71069950 + (102.4890774 + (-114.2815642 + 45.21405146\ x)\ x)\ x)\ x)x$$

$$-.100575884 + 4.867599482\ x - 36.71069950\ x^2 + 102.4890774\ x^3 - 114.2815642\ x^4$$

$$+ 45.21405146\ x^5$$

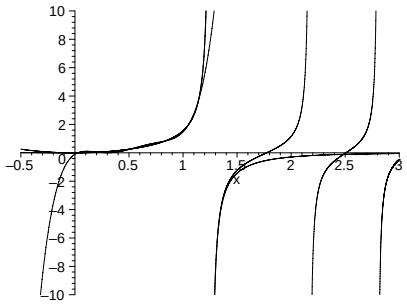
$$fehler = .1006673773$$

Mit der gebrochen rationalen Funktion (Ausgabe hier unterdrückt) erhält man eine wesentlich bessere Näherung:

```
mini25:=minimax(f(x),x=a..x0-0.05,[2,5],1,'maxerror'); maxerror;
```

$$maxerror = .0005507722888$$

```
plot([f(x),mini0,mini25],x=-1/2..3,-10..10);
```



Die Annäherung ist bei den hier verwendeten Parametern schlechter als mit der normalen Reihenentwicklung. Wenn man mit der rechten Bereichsgrenze zu nahe an $x0$ geht, beginnt das Näherungspolynom zu oszillieren.

Als Nächstes können wir die beiden Sorten der Pade-Approximation untersuchen. Zunächst eine Enwicklung um den Ursprung:

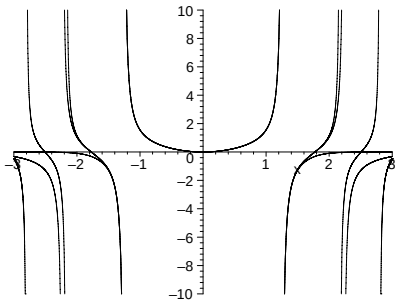
```
padef:=pade(f(x),x=a,[6,17]);
```

$$padef := \frac{-\frac{10}{99}\,x^6 + x^2}{1 - \frac{43}{99}\,x^4 + \frac{17}{1485}\,x^8 + \frac{4}{31185}\,x^{12} + \frac{1}{467775}\,x^{16}}$$

Und dann eine Tschebyscheff-Pade-Approximation von Pol zu Pol (Ausgabe im Worksheet):

```
chebpadef1:=chebpade(f(x),x=-x0+0.1..x0-0.1,[6,17]):
plot([f(x),padef,chebpadef1],x=-3..3,-10..10);
```

Die Pade-Approximation liefert hier mit dem geringsten Aufwand das beste Ergebnis, indem sie sowohl die Symmetrie als auch noch den zweiten Pol wiedergibt. Insofern ist also hier der Einsatz von Tschebyscheff-Polynomen überflüssig.



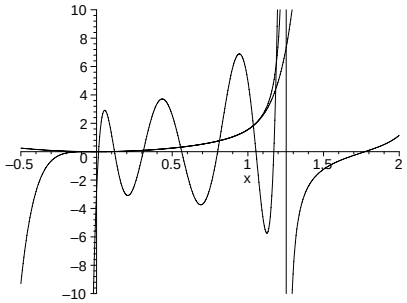
Schließlich folgt noch eine Demonstration, wie man es nicht machen sollte:

```
chebpadef2:=chebpade(f(x),x=a..1.25,[7,0]);

cheby:=chebyshev(f(x),x=a..1,0.001);

plot([f(x),chebpadef2,cheby],
      x=-1/2..2,-10..10);
```

Die mit `chebpade` gebildete Reihe beginnt sehr stark zu oszillieren, wenn man sich mit der Bereichsgrenze dem Pol nähert. Die mit `chebyshev` gebildete Reihe ist dagegen zwar stabil, aber eben nur im Intervall von 0 bis 1 eine gute Näherung.



Syntaxzusammenfassung

`series(f,x);` `series(f,x=x0,n);`
führt eine Laurent-, Taylor- oder allgemeine Potenzreihenentwicklung für f um $x = 0$ oder um $x = x0$ bis zur Ordnung `Order` oder bis zur im dritten Parameter angegebenen Ordnung n durch.

`asympt(f,x);` `asympt(f,x,n);`
führt eine Reihenentwicklung für $x \to \infty$ bis zur Ordnung n durch.

`whattype(%);`
stellt den Datentyp des letzten Ergebnisses fest. Nach den Kommandos `series` und `asympt` kommen `series` oder `+` in Frage.

`convert(reihe, polynom);`
konvertiert eine Reihe vom Datentyp `series` in ein Polynom.

`Order:=n;`
bestimmt die Defaulteinstellung für die Ordnung, bis zu der Reihen entwickelt werden sollen.

`mtaylor(f,[x=x0,y=y0,...],n);`
führt eine multivariable Taylor-Reihenentwicklung für $f = f(x, y, ...)$ durch.

`poisson(f,[x=x0,y=y0,...],n);`
wie oben, allerdings werden die Koeffizienten in der kanonischen Fourier-Form angeordnet.

Formale Reihen

`powcreate(f(n)=...);`
definiert die formale Reihe f durch eine Funktion für die Koeffizienten zu $x^0, x, x^2, x^3, ...$. Die Koeffizienten können mit f(0), f(1), ... abgerufen werden.

`tpsform(f, x, n);`
gibt die formale Reihe f als Reihe (Datentyp `series`) der Variable x bis zur Ordnung n zurück.

`powpoly(polynom, x);`
verwandelt ein Polynom in eine formale Reihe.

`evalpow(f +-*/ g);` `evalpow(f */^ s);` `evalpow(f(g));`
führt elementare Berechnungen mit den Reihen f und g und dem Skalar s durch (Addition, Subtraktion, Multiplikation, Division etc.).

`powdiff(f);` `powint(f);`
differenziert bzw. integriert die Reihe.

`powlog(f);` `powexp(f);`
bildet den Logarithmus und die Exponentialfunktion der Reihe.

`reversion(f);`
bildet die Reihe zur Umkehrfunktion.

Numerische Berechnung von Näherungsfunktionen

`minimax(f, x=a..b, [z,n]);`
stellt eine rationale Funktion mit dem Zählergrad z und dem Nennergrad n auf, die die Funktion f im Bereich zwischen a und b annähert.

`pade(f, x=x0, [z,n]);`
stellt eine rationale Näherungsfunktion auf, die f im Bereich um $x = x0$ annähert.

`chebpade(f, x=a..b, [z,n]);`
wie oben, allerdings wird die Funktion aus Tschebyscheff-Polynomen $T(n, x - x0)$ zusammengesetzt.

`chebyshev( f, x=a..b, eps);`
berechnet eine Reihenentwicklung aus Tschebyscheff-Polynomen, die f im Bereich zwischen a und b mit einer maximalen Abweichung von eps annähert.

`T(n,x);`
gibt das Tschebyscheff-Polynom n-ter Ordnung zurück. Zum Arbeiten mit diesen Polynomen kann das Paket `orthopoly` aufgerufen werden. *Achtung*: Dadurch werden Variablen mit den Namen G, H, L, P, T, U überschrieben.