Volker Traxler

Mathematica, eine Hilfe im Unterricht

Die Einsatzmöglichkeiten des Computeralgebrasystems Mathematica im MU werden an Hand konkreter Unterrichtsbeispiele demonstriert und besprochen.

Anschrift des Autors:

smail:

Volker Traxler
TGM
Abteilung Wirtschaftsingenieurwesen

Wexstrasse 19 -23
1200 Wien

e-mail:

traxler@email.tgm.ac.at

Volker Traxler, TGM Wien XX

Näherung der Binomialverteilung durch die Normalverteilung

Mathematische Inhalte:

- Darstellung von Wahrscheinlichkeitsfunktionen
- Standardisieren
- Modellbildung (Übergang diskret - stetig)
- Summe versus Integral, Differenzen-, Differentialgleichung, Grenzwert, uneigentliches Integral

Anwendung:

Statistische Grundlagen der Qualitätssicherung.

Kurzzusammenfassung:

Visualisierung der Näherung der Binomialverteilung durch die Normalverteilung mit Hilfe von Animationen, Herleitung der Wahrscheinlichkeitsdichtefunktion der Normalverteilung.

Lehrplanbezug:

4. Jahrgang HTL - Statistik und Wahrscheinlichkeitsrechnung, (8. Klasse AHS).

Zeitaufwand:

Je nach „Besprechungstiefe" 15 Minuten bis 4 Stunden.

Mediales Umfeld:

Medien: PC, Overhead-LCD-Display, Software: Mathematica Version 2.2

Anmerkungen:

Ich habe versucht, dieses Thema (außer Anwendungen auf konkrete Beispiele und dem dabei auftauchenden Problem der Stetigkeitskorrektur) vollständig auszuleuchten. Ihnen bleibt es nun überlassen, ob Sie nur z.B. die beiden Animationen zwecks Visualisierung der Problemstellung vorführen wollen oder ob Sie gar bis zur Herleitung der Normalverteilung (und sei es nur an der Tafel) mit Ihrer Klasse vordringen.

Wichtig:

Mit Hilfe der Software *MathReader* Version 2.2 (die an Schüler und Lehrer weitergegeben werden darf) *können Sie den Inhalt meines Mathematica-Files, ohne im Besitz von Mathematica selbst zu sein, am PC betrachten und die Animationen vorführen !!!!.* Allerdings können Sie keine Eingaben tätigen oder Berechnungen vornehmen !!!!

Bitte verwechseln Sie nicht MathReader (stark eingeschränkte Oberfläche, keine Eingabemöglichkeit etc.) mit dem CAS Mathematica !!!!

Da in Winword Graphik-Animationen nicht möglich sind, erscheinen statt einer Graphik, die animiert werden kann, alle 10 Einzelbilder untereinander.

Der Ausdruck auf Papier gibt nicht die übliche Struktur eines Mathematica-Files in Form einer Inhaltsübersicht und dem jeweiligen „Blättern" in einem „elektronischen" Buch wieder.

Der nachfolgende Ausdruck gibt nur einen Teil des Inhalts des Mathematica-Files wieder.

Näherung der Bi durch die NV

☐ Einleitung

Bespricht man die Binomialverteilung, so gelangt man über kurz oder lang zu ähnlichen Fragestellungen:

1. Wie groß ist die Wahrscheinlichkeit bei 1000-maligem Werfen einer "idealen" Münze zwischen 468- und 532-mal Wappen zu erhalten ?
2. Die Wahrscheinlichkeit einer Knabengeburt beträgt p = 0.51. Bei einer Untersuchung von 10000 Geburten stellte sich die Frage, mit welcher Wahrscheinlichkeit zwischen 5000 und 5200 Geburten beobachtet werden können?

Oder der "Praxis" näherstehende Fragestellungen:

3. Eine Zuverlässigkeitsanalyse hat ergeben, daß von einem bestimmten Teil 75% nach 5 Jahren Einsatz ausgefallen sein werden. In einer Anlage sind 100 derartige Teile eingebaut. Mit wie vielen Ausfällen muß man bei zehn Anlagen maximal rechnen (Irrtumswahrscheinlichkeit 10 %) ?
4. Von den ersten 1000 gefertigten Einheiten waren 35 fehlerhaft. Mit welchem Fehleranteil in der Fertigung muß man rechnen (Irrtumswahrscheinlichkeit 5 %) ?

Es erhebt sich nun die Frage, wie man den Rechenaufwand (übliche Tabellen und Larson-Nomogramm versagen) verringern kann, ohne die technisch erforderliche Genauigkeit zu vernachlässigen.

☐ Sammeln von Information

Betrachten wir verschiedene Arten der Darstellung der Wahrscheinlichkeitsfunktion einer Binomialverteilung, wobei eigene "Funktionen" definiert werden.

■ Darstellung der Wahrscheinlichkeitsfunktion g(x) als Punktdiagramm:

☐ Die Funktion "punktdiagramm".

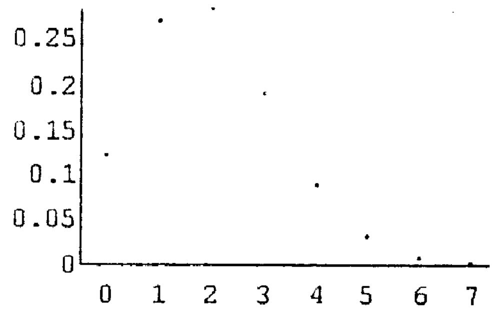
Bemerkung: mittels Eingabe von ? Befehlsname erhält man Hilfe, wie z. B.

In[12]:=

```
? punktdiagramm
Darstellung der Wahrscheinlichkeitsfunktion g(x) der Binomialverteilung
   als Punktdiagramm. Aufruf mit

 punktdiagramm[n,p,optionen]

 n ............... Stichprobenumfang n
 p ............... Grundwahrscheinlichkeit der Binomialverteilung
 optionen ........ Optionen von ListPlot zulässig
```

In[13]:=

```
punktdiagramm[20,0.1];
```



Bemerkung: Sollten Sie bei obiger Graphik keine Punkte am Bildschirm entdecken, ist im Menü Graph "Make Lines Thin" eingestellt. Deaktivieren dieser Einstellung löst dieses Probleml

Darstellung der Wahrscheinlichkeitsfunktion g(x) als Stabdiagramm:

Die Funktion "histogramm".

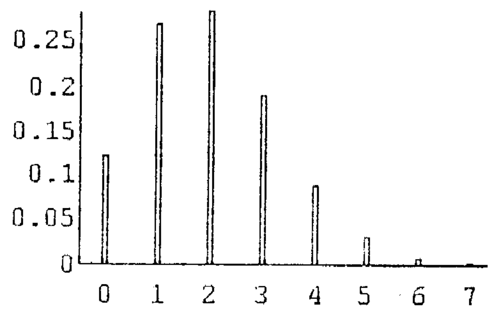
In[14]:=

```
? histogramm
```

```
Darstellung der Wahrscheinlichkeits funktion g(x) der Binomialverteilung
   als Stabdiagramm. Aufruf mit

 histogramm[n,p,breite,optionen]

 n ............... Stichprobenumfang n
 p ............... Grundwahrscheinlichkeit der Binomialverteilung
 breite .......... gibt die Breite des Histogramms an
 optionen ........ Optionen von GeneralizedBarChart zulässig
```

In[15]:=

```
histogramm[20,0.1,0.1];
```



Sie sind eingeladen, die Werte für n und p zu verändern (Sollte sich der Plotbereich (PlotRange) oder die Wahl des Achsenkreuzungspunkt (AxesOrigin) nicht als sinnvoll erweisen, so bitte ich Sie, dies zu ändern).

Das Galtonbrett

Als Modell für den Sonderfall der Binomialverteilung mit p = 0.5, sowie später, als Modell für die Überlagerung von Störungen bei Fertigungsprozessen, dient das Galtonbrett. Das Fächergestell des Galtonbretts weist in Richtung Histogramm.

■ Summe versus Integral

Hier kann die Tatsache in Erinnerung gerufen werden, daß Integrale das stetige Analogon zu Summen sind. ("Integrale sind im gewissen Sinne leichter zu handhaben als eine Summe vieler Summanden!").

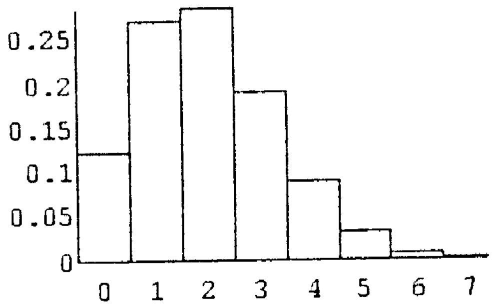
Beispiel für Summen: Verteilungsfunktion G(x)

□ Auswertung der Information

Obiges Kapitel legt nahe, die Wahrscheinlichkeit g(x) nicht als Punkt, sondern durch den Flächeninhalt eines Rechtecks mit der Breite 1 und der Höhe g(x) darzustellen.

In[18]:=

```
histogramm[20,0.1,1];
```



□ Das Experiment

Der Computer soll nun Binomialverteilungen mit verschiedenen n, jedoch gleichem p, nacheinander darstellen. Betrachten wir folgende Animation:

□ Die Funktion animatio

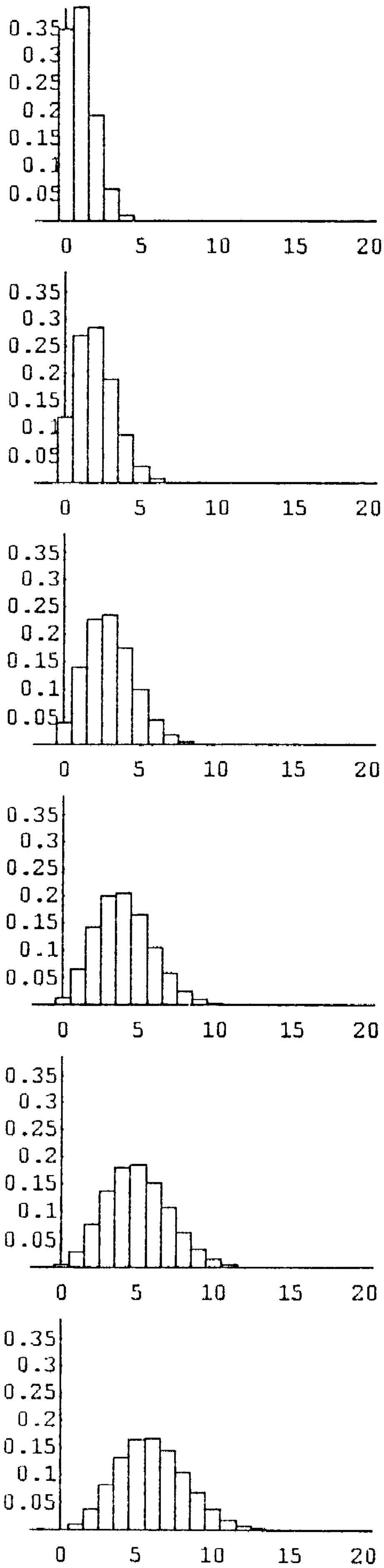
In[19]:=

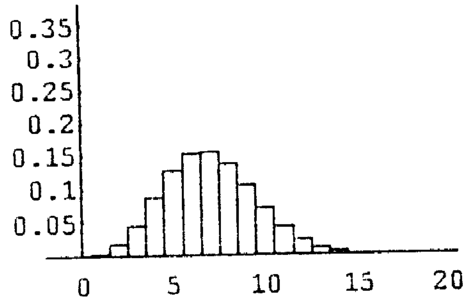
```
? animatio
```

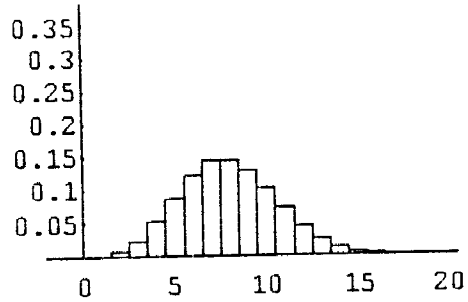
```
Der Computer soll nun Binomialverteilungen mit verschiedenen n, jedoch
  gleichem p, nacheinander darstellen. (Animation!)
Aufruf mit

 animatio[n,p,anfangswert,schrittweite,optionen]

n ............... Stichprobenumfang n
p ............... Grundwahrscheinlichkeit der Binomialverteilung
anfangswert ..... Anfangswert von p
schrittweite .... Schrittweite von p
optionen ........ Optionen des Befehls GeneralizedBarChart
```

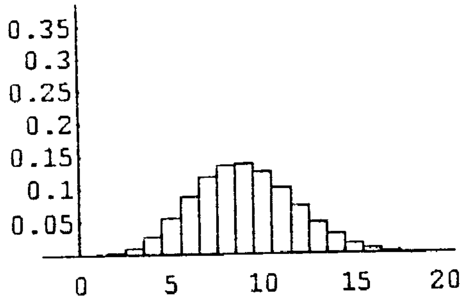
In[20]:=

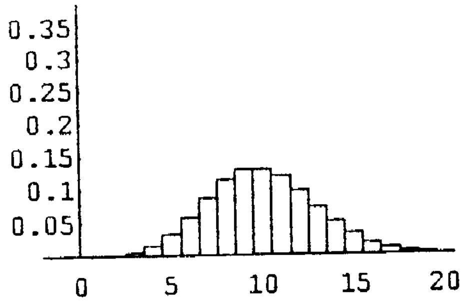
```
animatio[100,0.1,10,10]
```











Die Kurven werden i.a. immer breiter und sie wandern nach rechts, je größer n wird. Um nun die Binomialverteilungen miteinander vergleichen zu können, führen wir eine "Standardisierung" durch.

Ziel: "Festnageln" der Lage (my=0).
"Festnageln" der Streuung (ungefähr "gleiche Ausdehnung" der Histogramme).
Gesamtfläche des Histogramms muß 1 bleiben.

Transformation: u = (x-my)/sigma.

Ergebnis: Der Nullpunkt fällt mit Mittelwert zusammen.
Die neue Breite der Rechtecke beträgt das 1/sigma-fache der alten.
Die neue Höhe der Rechtecke beträgt das sigma-fache der alten.

□ Die Funktion animatio1

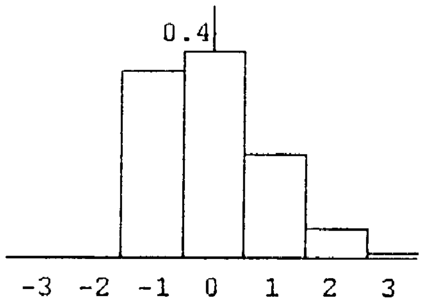
In[21]:=

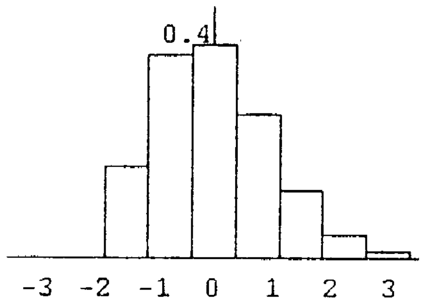
```
? animatio1
```

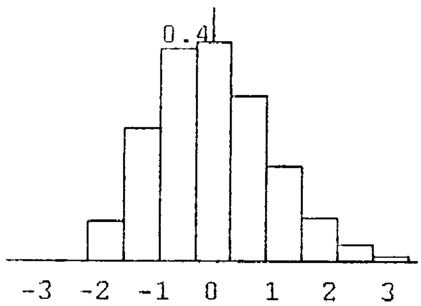
```
Der Computer soll standardisierte Binomialverteilungen mit verschiedenen
  n, jedoch gleichem p, nacheinander darstellen. (Animation!)
Aufruf mit

 animatio[n,p,anfangswert,schrittweite,optionen]

n ............... Stichprobenumfang n
p ............... Grundwahrscheinlichkeit der Binomialverteilung
anfangswert ..... Anfangswert von p
schrittweite .... Schrittweite von p
optionen ........ Optionen des Befehls GeneralizedBarChart
```

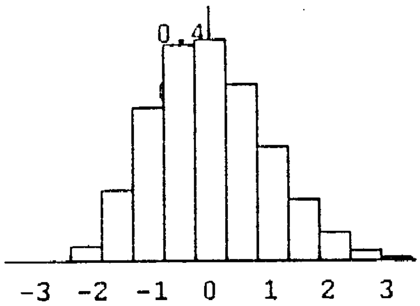
In[22]:=

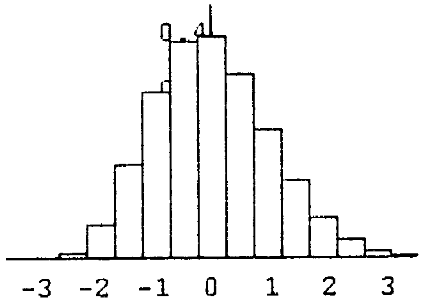
```
animatio1[100,0.1,10,10];
```

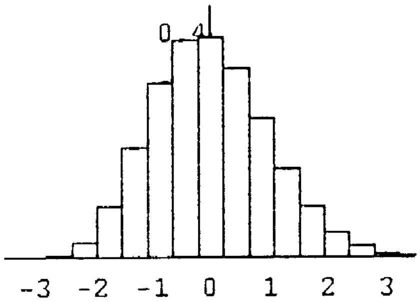


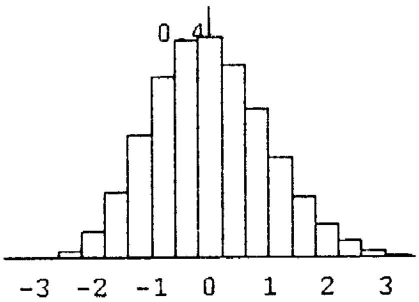


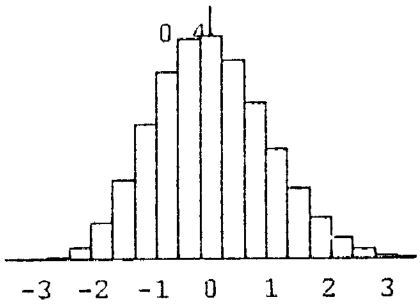


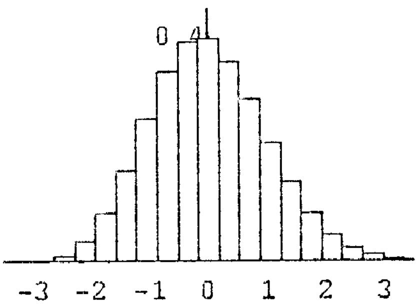

0,4
-3 -2 -1 0 1 2 3

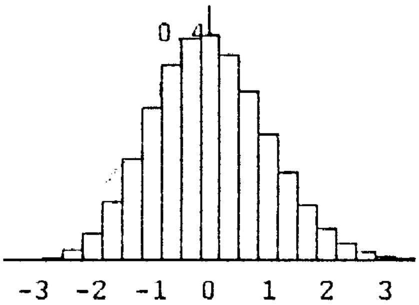

0,4
-3 -2 -1 0 1 2 3


0,4
-3 -2 -1 0 1 2 3


0,4
-3 -2 -1 0 1 2 3


0,4
-3 -2 -1 0 1 2 3


0,4
-3 -2 -1 0 1 2 3



Wie man sieht, lassen sich die Histogramme der standardisierten Binomialverteilung bei n-> unendlich durch eine Glockenkurve approximieren.

Bemerkung: Um diese "Standardisierung" durchzuführen, ändern wir ein klein wenig den Code des Befehls animatio und wir erhalten den Code von animatio1.

Ausschnitt aus dem Code von animatio:

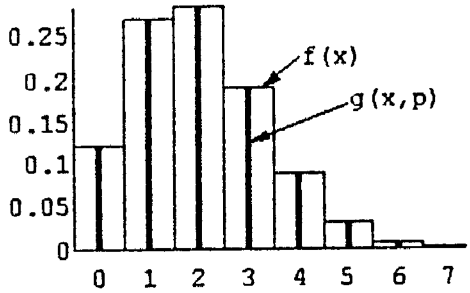
```
.....................
     Table[x, {x,0,n}],


     (* Höhe der Rechtecke *)
     Table[Evaluate[PDF[BinomialDistribution[n,p],x]],
                       {x,0,n}],


     (* Breite der Rechtecke *)
     Table[1,{x,0,n}]
...............
```

Ausschnitt aus dem Code von animatio:

```
..............................
(* "Festnageln" der Lage (my=0),Transf.:(x-my)/sigma     *)

     Table[(x-my)/sigma, {x,0,n}],

(* Die neue Höhe der Rechtecke beträgt das sigma-fache
   der alten                               *)

  sigma*Table[Evaluate[PDF[BinomialDistribution[n,p],x]],
                       {x,0,n}],

(* Die neue Breite der Rechtecke beträgt das
   1/sigma-fache der alten                    *)

     Table[1/sigma,{x,0,n}]
.............................
```

Herleitung der Normalverteilung

Wie in der obigen Animation veranschaulicht, verteilt man die Wahrscheinlichkeit g(x,p) der diskreten Verteilung an der Stelle x gleichmäßig über den Bereich x-0.5 bis x+0.5.



und erhält eine stetige Verteilung f(x) (kurz f bezeichnet) mit dx = 1.

Bilden wir mit Hilfe der Rekursionsformel für die Binomialverteilung die Differenz zweier benachbarter Werte

In[26]:=

```
df = f((n-x)/(x+1) p/q - 1) dx
```

Out[26]:=

$$dx\ f\ \left(-1 + \frac{p\ (n - x)}{q\ (1 + x)}\right)$$

und bringen den Klammerausdruck auf gemeinsamen Nenner

In[27]:=

```
gl1 = Together[df]
```

Out[27]:=

$$\frac{dx\ f\ (n\ p - q - p\ x - q\ x)}{q\ (1 + x)}$$

Ersetzen wir p x + q x durch x

In[28]:=

```
gl2 = gl1 /. -p x-q x-> -x
```

Out[28]:=

$$\frac{dx\ f\ (n\ p - q - x)}{q\ (1 + x)}$$

und führen wir die Standardabweichung

In[29]:=

```
s = Sqrt[n p q];
```

ein.

Standardisieren des Merkmals x (siehe Standardisierung der Binomialverteilung) und Bildung der Differenzengleichung

In[30]:=

```
gl3 = gl2/du //. {x -> u s + n p , dx -> du s }
```

Out[30]:=

```
f Sqrt[n p q] (-q - Sqrt[n p q] u)
----------------------------------
   q (1 + n p + Sqrt[n p q] u)
```

ergibt für n->unendlich die rechte Seite der Differentialgleichung

In[31]:=

```
Limit[gl3,n->Infinity]
```

Out[31]:=

```
-(f u)
```

Die Lösung der Differentialgleichung

In[32]:=

```
lsg1 = DSolve[f'[u] == -f[u] u, f[u],u]
```

Out[32]:=

```
               C[1]
{{f[u] ->      ----}}
                2
               u /2
              E
```

ergibt folgende Funktion:

In[33]:=

```
f[u_] = f[u] /. lsg1[[1]]
```

Out[33]:=

```
C[1]
----
  2
 u /2
E
```

Als Wert für die Fläche unterhalb des Graphen erhalten wir:

In[34]:=

```
lsg2 = Integrate[f[u],{u,-Infinity,Infinity}]
```

Out[34]:=

```
Sqrt[2 Pi] C[1]
```

Um den Wert der Konstante C[1] festzulegen, wählen wir als Wert für die Fläche sinnvollerweise 1:

In[35]:=

```
lsg3 = Solve[lsg2 == 1, C[1]]
```

Out[35]:=

```
                 1
{{C[1] -> ----------}}
          Sqrt[2 Pi]
```

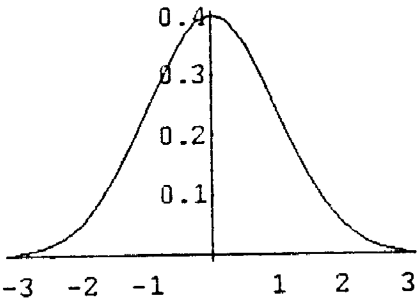
In[36]:=
```
lsg4 = C[1] /. lsg3[[1]]
```
Out[36]:=
```
      1
-----------
Sqrt[2 Pi]
```

Somit lautet unsere Approximationsfunktion:

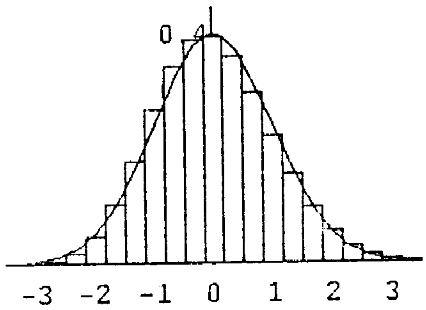
In[37]:=
```
f[u_] = f[u] /. C[1]->lsg4
```
Out[37]:=
```
         1
---------------
  2
 u /2
E     Sqrt[2 Pi]
```

Der Graph der standardisierten Normalverteilung ergibt sich zu

In[38]:=
```
nv1 = Plot[f[u],{u,-3,3}];
```



Betrachten wir gemeinsam das Histogramm der Binomialverteilung für n = 100 und p=0,1 und deren Näherung für n-> unendlich:



Bemerkung: Die Anwendung auf die zu Beginn aufgezählten Beispiele wird hier nicht weiter ausgeführt.

Buchliste:

□ Reichel, Hans Christian (Hrsg.): "Wahrscheinlichkeitsrechnung und Statistik", Mathematik für Schule und Praxis, Verlag Hölder-Pichler-Tempsky, 1987
ISBN 3-209-00736-5

□ Stange, Kurt : "Angewandte Statistik", Erster Teil, eindimensionale Probleme, Springer-Verlag, Berlin-Heidelberg-New York 1970

□ Wolfram, Stephen: "Mathematica, Ein System für Mathematik auf dem Computer", Zweite Auflage,Addison-Wesley Verlag (Deutschland) GmbH

□ Kofler, Michael: "Mathematica", Einführung und Leitfaden für den Praktiker, 1. Auflage, Addison-Wesley Verlag (Deutschland) GmbH
ISBN 3-89319-485-1

Bemerkung: Folgende Befehle sind vom Autor entwickelt und nicht im Standard-Befehlsumfang von Mathematica enthalten: punktdiagramm, histogramm, animatio und animatio1.

Volker Traxler, TGM Wien XX

Newton-Raphson

Mathematische Inhalte:

Numerische Verfahren, Iteration

Anwendung:

Newton-Verfahren

Kurzzusammenfassung:

An Hand von Graphiken sollen Faustregeln für die Wahl eines Startpunktes bzw. Bedingungen für die Konvergenz des Newton-Verfahrens plausibel erläutert werden.

Lehrplanbezug:

3. Jahrgang HTL; 7.Klasse AHS.

Zeitaufwand:

Je nach „Besprechungstiefe" 20 Minuten bis 3 Stunden.

Mediales Umfeld:

Medien: PCs, Overhead-LCD-Display.
Software: Mathematica Version 2.2 für Windows.

Anmerkungen:

Mit Hilfe der Software *MathReader* Version 2.2 (die an Schüler und Lehrer weitergegeben werden darf) *können Sie den Inhalt meines Mathematica-Files, ohne im Besitz von Mathematica selbst zu sein, am PC betrachten und die Animationen vorführen !!!!.* Allerdings können Sie keine Eingaben tätigen oder Berechnungen vornehmen !!!!
Bitte verwechseln Sie nicht MathReader (stark eingeschränkte Oberfläche, keine Eingabemöglichkeit etc.) mit dem CAS Mathematica !!!!
Der Ausdruck auf Papier gibt nicht die übliche Struktur eines Mathematica-Files in Form einer Inhaltsübersicht und dem jeweiligen „Blättern" in einem „elektronischen" Buch wieder.
Da in Winword Graphik-Animationen nicht möglich sind, erscheint statt einer Graphik, die animiert werden kann, jeweils das letzte Einzelbild.
Der nachfolgende Ausdruck gibt nur einen Teil des Inhalts des Mathematica-Files wieder und berücksichtigt nicht die Farbgebung der Graphiken !!!!

Newton-Raphson

Einführung

Für die folgenden Ausführungen wird vorausgesetzt, daß die Formel des Newton-Verfahrens bekannt bzw. abgeleitet und vielleicht schon an Hand einiger Beispiele (ohne auf das Problem der Wahl eines Startwertes ~~einzugehen)~~ der Algorithmus demonstriert wurde. Weiters kann angenommen werden, daß durch geschickte Auswahl von Beispielen bei der Selbsttätigkeit der Schüler Probleme durch ihre Auswahl der Startwerte auftraten.

Ziel der folgenden Ausführungen ist es, an Hand von Graphiken Faustregeln für die Wahl eines Startpunktes bzw. Bedingungen für die Konvergenz des Newton-Verfahrens (siehe Kapitel Zusammenfassung) plausibel zu machen.

Diskussion

Wahl der Funktion f(x)

Für die folgenden Ausführungen bestimmen wir eine Funktion f(x), sowie die dazugehörige Newton´sche Näherungsformel g(x).

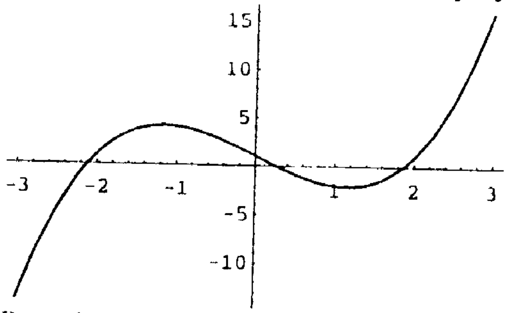
In[13]:=

```
f[x_]:= x^3-4x+1; g[x_]:= x-f[x]/f'[x]
```

Stellen wir den Graphen der Funktion f(x) dar und speichern diese Graphik mittels der Variablen gr1.

In[14]:=

```
gr1 = Plot[f[x], {x,-3,3}];
```



Bemerkung: Der Vorteil eines "elektronischen" Notebooks besteht u.a. darin, daß der Schüler eine andere Funktion bzw. andere Startwerte wählen und die folgenden Befehle analog mit seinen Eingaben aufrufen kann. Er schafft sich damit seine eigene persönliche Mitschrift!

Extremwerte der Funktion f(x)

Bei der Suche nach der Ursache von Problemen bei der Wahl eines geeigneten Startwertes wird meistens bald herausgefunden, daß f'(x) ungleich Null sein muß (Wiederholung: Definitionsmenge !!!). Die geometrische Interpretation der 1.Ableitung mußte öfters von den Schülern eingefordert werden.

Es folgt die Berechnung jener x-Werte, für die f'(x) = 0 gilt:

In[15]:=

```
Solve[f'[x] == 0,x]
```

Out[15]:=

```
          -2                  2
{{x -> -------}, {x -> -------}}
        Sqrt[3]            Sqrt[3]
```

Die Funktion PlotNewton

Um einen Eindruck weiterer Problemstellen zu bekommen, betrachten wir einige Animationen.

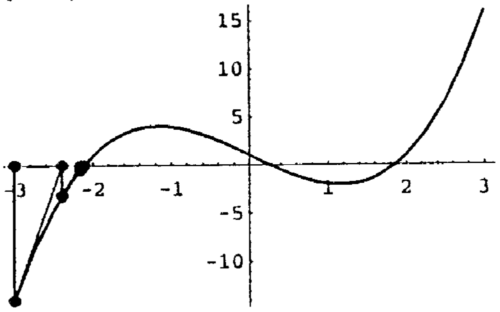
Bemerkung: mittels Eingabe von ? Befehlsname erhält man Hilfe, wie z. B.

In[16]:=

```
? PlotNewton
PlotNewton[f[x],{x,xmin,xmax},InitialPoint->x0,Iterations->m] plots f[x]
   together with m Newton approximations starting at the initial point x0.
   The sequence of approximations is also printed.  The default value of
   Iterations is 3.  Graph->False suppresses the graph.
   TangentStyle->style1 applies style1 to the tangent lines and points.
   PointStyle->PointSize[n] sets the point size.  Together->False produces
   a series of plots suitable for animation.  Other options are passed to
   Plot.
```

In[17]:=

```
PlotNewton[f[x], {x,-3,3}, InitialPoint->-3,
                                        Together->False];
{-3., -2.391304348, -2.154963544, -2.115945495}
```



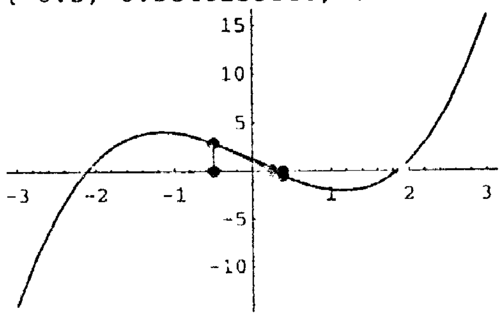
In[18]:=

```
PlotNewton[f[x], {x,-3,3}, InitialPoint->-2/Sqrt[3],
                                        Together->False];
                                       1
Power::infy: Infinite expression -- encountered.
                                       0.

Infinity::indet:
   Indeterminate expression 1 + ComplexInfinity + ComplexInfinity
     encountered.
{-1.154700538, ComplexInfinity, Indeterminate, Indeterminate}
```

Mathematica weist uns mit einer Flut von Fehlermeldungen auf den Umstand hin, daß bei der obigen Rechnung der Nenner f '(x) Null und daher nicht definiert ist.

In[19]:=

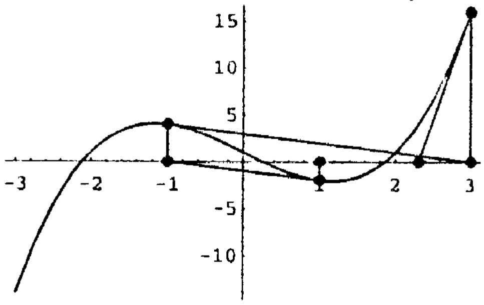
```
PlotNewton[f[x], {x,-3,3}, InitialPoint->-.5,
                                        Together->False];
{-0.5, 0.3846153846, 0.2492000512, 0.2540969477}
```



In[20]:=

```
PlotNewton[f[x], {x,-3,3}, InitialPoint->1,
                                    Together->False];
```

```
{1., -1., 3., 2.304347826}
```

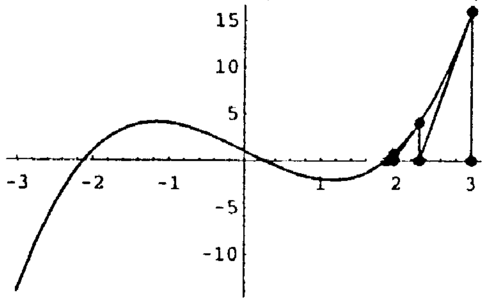


Aus obiger Animation entnimmt man: in der Nähe eines Extremwertes der Funktion f(x) kann es zu "Bocksprüngen" kommen.

In[21]:=

```
PlotNewton[f[x], {x,-3,3}, InitialPoint->3,
                                    Together->False];
```

```
{3., 2.304347826, 1.967489477, 1.869470471}
```



Fazit: die Bereiche von -unendlich bis ca. -2. bzw. von ca. 1.8 bis + unendlich scheinen problemlos zu sein.

Die Funktion xngraph

Um nun einen besseren Überblick zu bekommen, tragen wir in einem Diagramm eine Auswahl möglicher Startwerte x0 auf der x-Achse und als y-Wert den dazugehörigen z.B. 10-mal iterierten Wert ein, so erhalten wir folgendes Bild:

Diese Graphik kann problemlos mit folgendem Befehl xngraph (Name willkürlich, kann geändert werden) erzeugt werden.

Anforderung von Hilfe zu diesem selbstgestrickten Befehl erfolgt mit :

In[22]:=

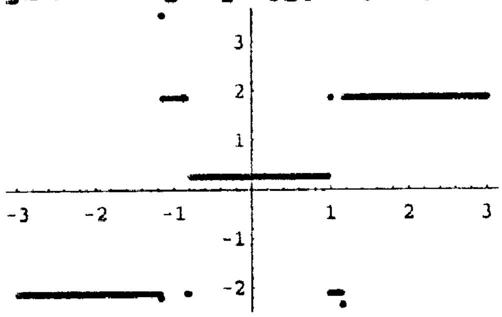
```
? xngraph
```

```
Zeichnet eine Menge von Punkten mit den Koordinaten (Startwert/n-ter
  iterierter Wert)

xngraph[g, xn, xmin, xmax, schrittweite, optionen]

g ..... eine Funktion (z.B g, nicht g[x] !!!)
xn .... Anzahl der Iterationen
xmin... linke Begrenzung der Graphik
xmax... rechte Begrenzung der Graphik
schrittweite
optionen ... Alle Optionen von ListPlot sind erlaubt!
```

Die so erzeugte Graphik wird unter den Variablennamen gr2 abgelegt:

In[23]:=

```
gr2 = xngraph[g,10,-3,3,1/60];
```



Schon diese Graphik gibt Anlaß zu Diskussionen. Trotz 10-maliger Iterationen "erreichen" einige Startwerte nicht eine der drei reellen Wurzeln.

Die Funktion xigraph

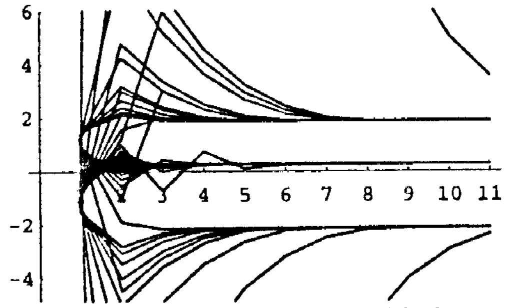
Tragen wir für verschiedene Startwerte wiederholt folgende Zahlenpaare (1, x0), (2, g(x0)), (3, g(g(x0))) in einem Koordinatensystem auf und legen die entstandene Graphik unter dem Variablennamen gr3 ab.

In[24]:=

```
? xigraph
Zeichnet eine Menge von Polygonzuegen
 mit den Punkten (1,x0), (2,x1), (3,x2) .......(n+1,xn)

 Eingabe analog xngraph!!!!
```

In[25]:=

```
gr3 = xigraph[g,10,-1.5,1.5,1/20];
```



Wie man sieht, spielt die Wahl eines geeigneten Startwertes eine entscheidende Rolle. Ein unglücklich gewählter Startwert führt entweder zu keinem Ergebnis oder zumindest zu einem gewaltigem Rechenaufwand.

Fixpunkte von g(x)

Wir haben vorher bei der Betrachtung von g(x) bemerkt, daß f '(x) ungleich Null sein muß. Erinnern wir uns an das Ziel unserer Betrachtungen: wir wollen jene x-Werte von f(x) bestimmen, für welche f(x) = 0 gilt.
Setzen wir dies in g(x) ein, so erhalten wir g(x) = x. Dies bedeutet, daß der x-Wert gleich dem y-Wert ist. (Wiederholung: 1. Mediane !!!!!!). Wir erhalten also einen Fixpunkt.

Die Funktion graphik

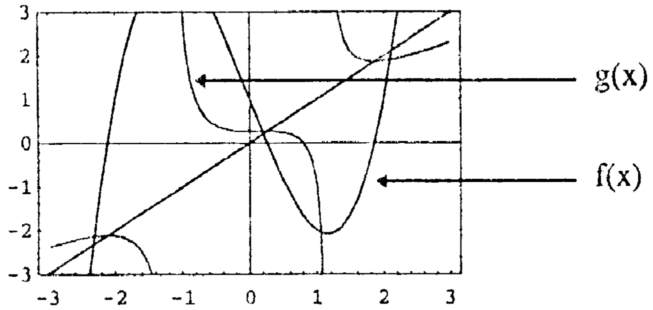
Um diesen Sachverhalt zu demonstrieren, benutzen wir den Befehl graphik

In[26]:=

```
? graphik
Zeichnet den Graph der Funktion f(x), g(x) und der ersten Mediane.
 f(x) erscheint schwarz, g(x) erscheint gruen, 1. Mediane blau

 graphik[f,,xmin,xmax,{Pole von g(x)}, {Pole von f(x)}]

 f ..... Funktion f (z.B f, nicht f[x] !!!)
 xmin... linke Begrenzung,  xmax... rechte Begrenzung
 Pole von g(x) und f(x) in Form einer Liste, also in { } setzen !!!
```

In[27]:=

```
gr4 = graphik[f,-3,3, {-2/Sqrt[3],2/Sqrt[3]}];
                                               1
Power::infy: Infinite expression - encountered.
                                               0
                                               1
Power::infy: Infinite expression - encountered.
                                               0
```



Didaktischer Hinweis:Erzeugen Sie jeweils eine Folie mit dem Graph f(x), g(x) und der 1. Mediane und legen Sie diese nacheinander übereinander auf und verbinden Sie mittels Overheadstift die Punkte f(x) = 0 und die darüber liegenden Punkte g(x) = x. Die Verbindungslinien sind in Graphik gr4 nicht eingezeichnet.

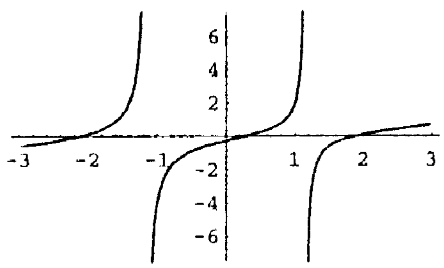
f(x) /f '(x)

Da wir ja i.a. keine exakten Lösungen der Gleichung f(x) = 0 angeben können, müssen wir erreichen, daß g(x) ungefähr gleich x wird, d.h. der Term |f(x) / f '(x)| soll möglichst klein sein.

Lassen wir uns f(x)/f '(x) graphisch darstellen.

In[28]:=

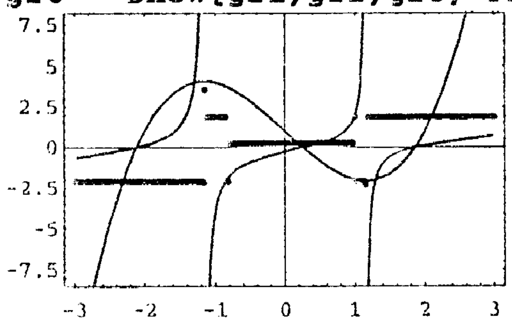
```
gr5 = PlotJump[f[x]/f'[x], {x,-3,3},
                    Jumps->{-2/Sqrt[3], 2/Sqrt[3]},
                    PlotStyle->RGBColor[.3,0,0]];
                                               1
Power::infy: Infinite expression - encountered.
                                               0
                                               1
Power::infy: Infinite expression - encountered.
                                               0
```



Überlagern wir die Graphiken gr1, gr2 und gr5:

In[29]:=

```
gr6 = Show[gr1,gr2,gr5, Frame->True];
```



Man sieht, daß in einer Umgebung der jeweiligen Nullstelle der Wert von |f(x) / f '(x)| tatsächlich klein ist, in der Nähe der Pole von f(x) / f '(x) jedoch ein "Sprung" zu einer anderen Nullstelle erfolgt oder ein Erreichen der Nullstelle nach 10 Iterationsschritten noch immer nicht erfolgt ist.
Bemerkung: f(x) ungefähr gleich Null reicht i.a. nicht aus!!!!!!

■ Die Funktion phasen

Wie näheren wir uns g(x) = x?
Wiederholen wir: wir nehmen einen (hoffentlich vernünftigen) Startwert x0, bilden g(x0), g(g(x0)), ... u.s.w.
Diese (hoffentlich konvergierende) Zahlenfolge kann mit dem Mathematica-Befehl NestList erzeugt werden.

In[30]:=

```
? NestList
NestList[f, expr, n] gives a list of the results of applying f to expr 0
   through n times.
```

In[31]:=

```
liste1 = NestList[g,3,3] // N
```

Out[31]:=

```
{3., 2.30435, 1.96749, 1.86947}
```

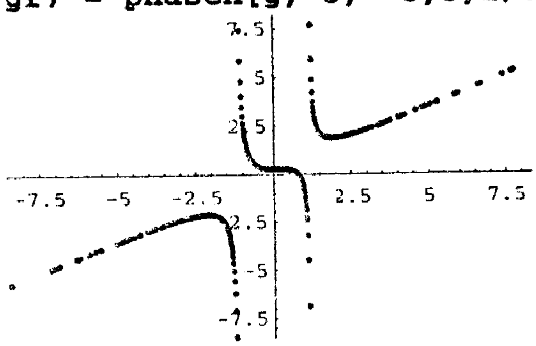
Vergleiche mit der Ausgabe von PlotNewton mit Startwert 3 weiter oben!
Wie weiter oben bereits angerissen, wird das Finden einer Nullstelle einer Funktion f(x) zurückgeführt auf das Finden eines Fixpunktes einer Funktion g(x). Geometrisch interpretiert ist dies der Schnittpunkt der ersten Mediane mit g(x) (siehe Graphik gr4).
Mit dieser, sogenannten Fixpunktiteration, bilden wir eine Folge von Punkten, (x0, g(x0)), (g(x0), g(g(x0))) u.s.w.
Mit Startwert 3 erhalten wir für oben definierter Funktion g(x) die Punktfolge
(3, 2.30435), (2.30435, 1.96749), (1.96749, 1.86947)
Versuchen wir diesen Sachverhalt für eine Reihe von verschiedenen Startwerten graphisch darzustellen:

In[32]:=

```
? phasen
Zeichnet das Phasendiagramm der  Iterationsfunktion g(x)

phasen[g, xn, x0min, x0max, schrittweite, optionen]

g  ..... Funktion g (z.B g, nicht g[x] eingeben !!!)
xn ..... Anzahl der Iterationen
x0min .. kleinster Startwert
x0max .. größter Startwert
schrittweite
optionen siehe Optionen von ListPlot:
```

In[33]:=

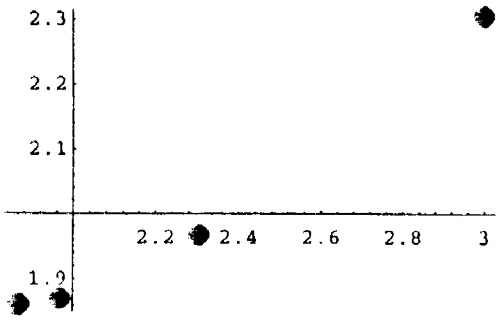
```
gr7 = phasen[g, 5, -3,3,1/60];
```



Diese Graphik kommt uns bekannt vor. Sie entspricht der Graphik der Funktion g(x) (siehe gr4). Dabei beobachtet man die Anhäufung der einzelnen Iterationspunkte bei den Fixpunkten der Funktion g(x). (Stichwort: Grenzwert !!!)
Stellen wir die Folge der Zahlenwerte (3, 2.30435), (2.30435, 1.96749), (1.96749, 1.86947) dar:

In[34]:=

```
gr8 = phasen[g,10,3,3.5,3, PlotStyle-> {RGBColor[0,.5,.5],
                                        PointSize[.03]}];
```



■ Die Funktion iteration

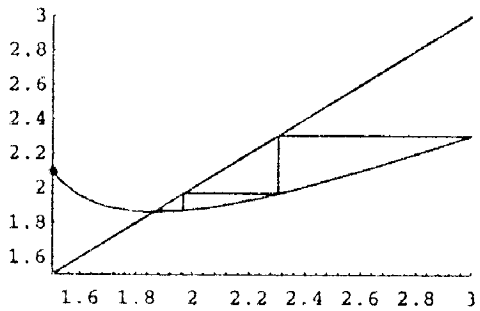
Gibt es eine graphische Möglichkeit, rasch diese Punktfolge zu zeichnen ?
Man kann die einzelnen Punkte mittels eines "Zick-Zack Bandes" zwischen dem Graphen von g(x) und der ersten Mediane verbinden. Dazu kann man den Befehl iteration benutzen:

In[35]:=

```
? iteration
 Geometrische Darstellung der Iteration einer Funktion g als Zick-Zack-
Band
   zwischen dem Graphen der Funktion g und der 1. Mediane!

 iteration[g, x0, xi, xn, xmin, xmax, pole]

 g .... zu iterierende Funktion
 x0 .... Startwert
 xi .... Zwischenwert, ab dem die Iterationen dargestellt werden sollen
 xn .... Endwert
 xmin .. untere Grenze des Zeichenbereichs
 xmax .. obere Grenze des Zeichenbereichs
 pole .. Liste der Pole der Funktion
         Achtung: Pole in Form einer Liste eingeben, z.B.: {-7, 0.1, 3}
```

In[36]:=

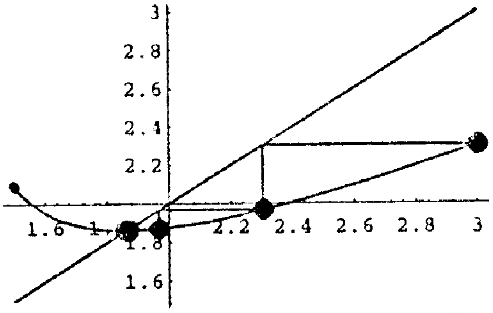
```
gr9 = iteration[g,3,0,10,1.5,3];
```



Überlagerung der Graphiken gr8 und gr9 ergibt:

In[37]:=

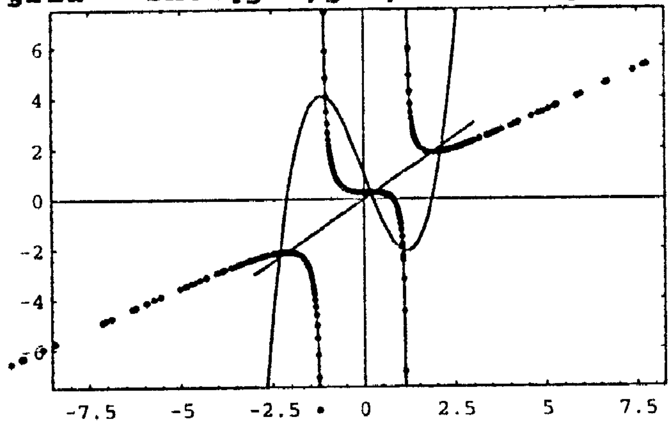
```
gr10 = Show[gr8, gr9, PlotRange->All];
```



Daß die Iterationspunkte auf g(x) liegen, zeigt die Graphik gr11:

In[38]:=

```
gr11 = Show[gr4,gr7, PlotRange->{-7.5, 7.5}];
```



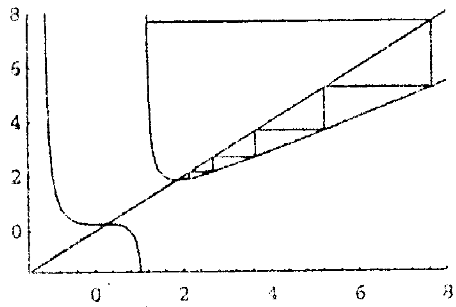
Wie sieht es nun mit der Konvergenz der Iteration für einzelne Startpunkte aus? Weiter oben (siehe PlotNewton) wurde bereits vermutet, daß die Bereiche von -unendlich bis ca. -2 bzw. von ca. 1,8 bis + unendlich problemlos zu sein scheinen.

Leicht können andere Startwerte ausprobiert werden.

■ **|g ' (x)| < 1**

In[39]:=

```
gr12 = iteration[g,1.2,0,10,-1.5,8, {-2/Sqrt[3],2/Sqrt[3]}];
                                            1
Power::infy: Infinite expression - encountered.
                                            0
                                            1
Power::infy: Infinite expression - encountered.
                                            0
```



Wie man sieht, ist der rechte obere Zweig von g(x) eine konvexe Funktion (Stichwort: Kurvendiskussion). Offensichtlich gilt in einer Umgebung des Fixpunktes (1.36, 1.86), daß g'(x) < 1 ist. Genauere Untersuchungen zeigen, daß -salopp ausgedrückt- in einem Intervall um den Fixpunkt |g'(x)| < 1 gelten muß.

Definieren wir in Analogie zu f(x) und g(x) zwei Funktionen u(x) und v(x), um die Definitionen für f(x) und g(x) nicht zu überschreiben:

In[40]:=

```
v[x_] := x-u[x]/u'[x]
```

Die erste Ableitung ergibt:

In[41]:=

```
v'[x]
```

Out[41]:=

```
u[x] u''[x]
-----------
        2
  u'[x]
```

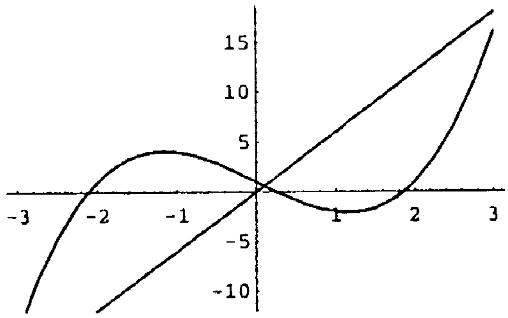
Offensichtlich muß f(x) mindestens zweimal differenzierbar sein.

■ f ''(x)

Untersuchen wir die Rolle der zweiten Ableitung von f(x):

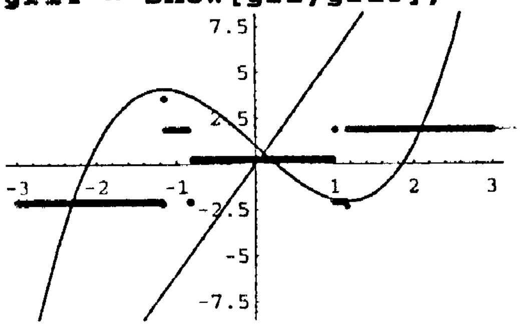
In[42]:=

```
gr13 = Plot[Evaluate[{f[x], f''[x]}, {x,-3,3},
                    PlotStyle->{{},{RGBColor[0,0.5,.5]}}]];
```



In[43]:=

```
gr14 = Show[gr2,gr13];
```



Man erkennt: wenn sgn f(x) = sgn f '' (x) gilt, konvergiert die Iterationsfolge mit einem Startwert aus diesem Bereich sicher zu der diesem Bereich nächstgelegenen Nullstelle.

Zusammenfassung

Es ist also unbedingt notwendig, den ungefähren Bereich der Nullstelle graphisch zu bestimmen.

Für die Randpunkte a und b des Bereichs sollte gelten: f(a) · f(b) < 0

Die Funktion sollte in einer Umgebung der Nullstelle zweimal stetig differenzierbar sein

Die erste Ableitung darf für alle Werte aus dieser Umgebung nicht Null ergeben

Der Startwert muß hinreichend "nahe" bei der Nullstelle liegen

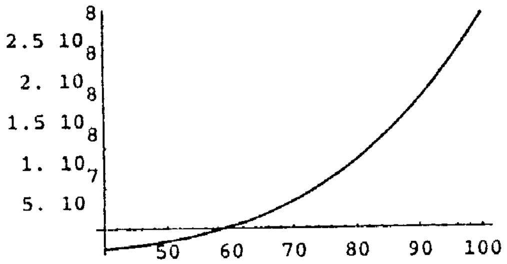
Praxis: sgn f(x) = sgn f''(x)

❑ Beispiel

Gegeben ist der Innendurchmesser eines Rohres di = 50 mm. Das Rohr soll ein Drehmoment M = 1 kNm aufnehmen. Die zulässige Torsionsspannung Tzul = 50 N/mm^2. Berechnen Sie den erforderlichen Außendurchmesser da.
Es ergibt sich untenstehende Formel als Bedingung für den Außendurchmesser da:

In[44]:=

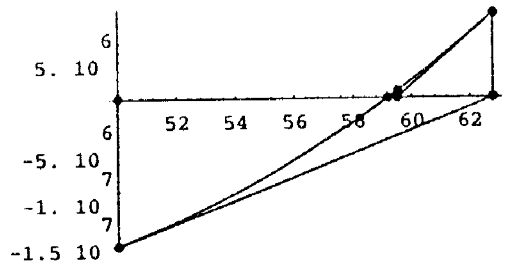
```
Plot[Pi da^4-Pi 50^4-16 20000 da, {da,40,100}];
```



Nehmen wir nun als Startwert den Innendurchmesser, so erhalten wir:

In[45]:=

```
PlotNewton[ Pi da^4-Pi 50^4-16 20000 da, {da,50,61},
                              InitialPoint->50];
{50., 62.79185081, 59.52786986, 59.19978184}
```



Zur Kontrolle die Lösung mit einen in Mathematica eingebauten Befehl:

In[46]:=

```
NSolve[Pi da^4-Pi 50^4-16 20000 da == 0,da]
```

Out[47]:=

```
{{da -> -38.8937}, {da -> -10.1515 - 51.1032 I},
  {da -> -10.1515 + 51.1032 I}, {da -> 59.1966}}
```

Für den Techniker reicht meistens bereits die erste Iteration, da wegen der Verwendung von Normen sowieso nur bestimmte Außendurchmesser angenommen werden können!!!!

Hinweis:

Folgende Befehle sind vom Autor selbst entwickelt worden und sind nicht im Standardpaket Mathematica enthalten: xngraph, xigraph, graphik, phasen, iteration.

Die Befehle PlotNewton und PlotJump wurden aus einem ebenfalls nicht im Standardpaket enthaltenen Package („KnoxPackage", siehe Anweisungen auf Diskette) eingelesen.

Bei Aufruf des Programms durch Mathematica werden durch Anklicken der Ja-Button all diese Befehle (sofern KnoxPackage ordnungsgemäß installiert) automatisch eingelesen.