Facharbeit Mathematik

Wie tief ist der Altwarmbüchener See?

Funktionen von R^2 in R. Darstellungsmöglichkeiten mit dem Computer, Untersuchung der Möglichkeit, Extrema zu bestimmen.

Von Benjamin Reh

März 2002

Inhaltsverzeichnis

Vorwort

Der Facharbeit liegt die Idee zugrunde, die Tiefe des Altwarmbüchener Sees durch eine mathematische Funktion des Typs $R^2 \rightarrow R$ darzustellen, von welcher Minima bestimmt werden, um schließlich die tiefste Stelle zu ermitteln.

Dann werden Darstellungsweisen solcher Funktionen beschrieben, wobei der Schwerpunkt auf die zugrunde liegenden Methoden und den Vor- und Nachteilen für den Betrachter liegt. Für die Veranschaulichung der einfacheren Techniken wurden dafür eigene Programme geschrieben, für komplexere Sachverhalte wurden Darstellungsweisen anderer Programme untersucht.

Im zweiten Teil wird dann ein Verfahren zur Suche von Extrema bei diesen Funktionstypen präsentiert. Auf zusätzliche Spezialfälle des Graphen wird nur eingegangen, soweit dies für die Betrachtung der Extrema relevant ist.

Da selbst eine noch recht ungenaue Funktion zur Beschreibung des Altwarmbüchener Sees wesentlich zu kompliziert wäre, wird im dritten Teil an einer Stellvertreterfunktion für einen beliebigen See noch einmal beispielhaft ein Teil der vorgestellten Verfahren verdeutlicht.

Darstellungsweisen mit dem Computer

Allgemeines zur Darstellung durch den Computer

Die zu behandelnden Funktionen von $R^2 \rightarrow R$ ergeben bei räumlicher Darstellungsweise des Graphen zu jedem Wertepaar (x, y) der Ausgangsmenge R^2 genau einen Wert z der Zielmenge R. Diese drei Werten sind die Koordinaten für einen Punkt P(x, y ,z) im Raum. Angenommen die Funktion sei global stetig, was noch später behandelt wird, so bilden diese Punkte eine Fläche.

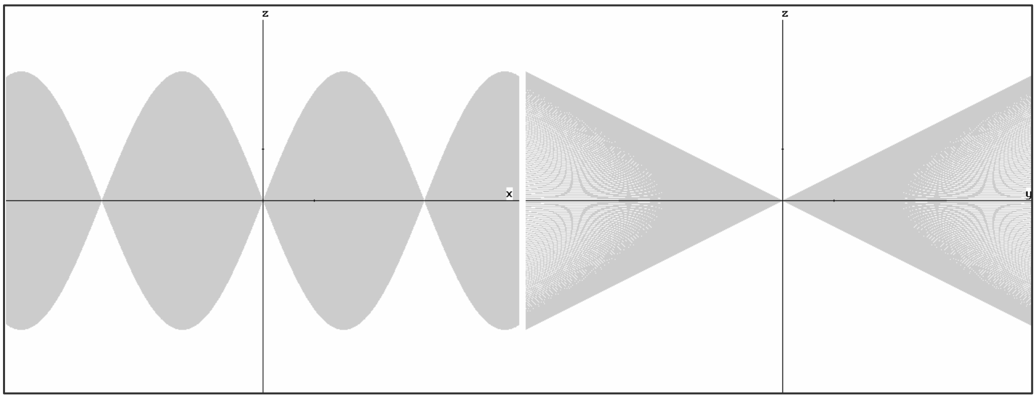
Allgemein spielt bei Darstellungen auch außerhalb des Computer die Genauigkeit eine wesentliche Rolle. Während es bei Zeichnungen möglich ist, mit Hilfe von Linealen und Schablonen relativ genau zu sein, so werden schon alleine bei den Berechnungen der Werte im Computer je nach verwendetem Programm mehr oder weniger starke Rundungsfehler gemacht. Diese ungenauen Werte werden dann bei der Umrechnung der mathematischen Koordinaten auf die Koordinaten der Bildpunkte (Pixel) noch mehr verfälscht.

Bei der noch recht groben Auflösung von 640 x 480 Pixel, die ich aus technischen Gründen in meinen Programmen verwende, sind solche Fehler noch auffallender. Es existieren zwar Methoden solche Ungenauigkeiten zu vermeiden, deren Vorgehensweisen dann aber oft mathematisch inkorrekt sind, wie zum Beispiel dem Anti-Aliasing, bei dem durch Unschärfe die entstandenen Punkte etwas mit der Umgebung verschwimmen. Oft werden auch ungenaue Maßstäbe an die Achsen gelegt, die nur beim Benutzen einer Vergrößerungensfunktion genauer werden. Zu Gunsten der Übersichtlichkeit und mathematische Korrektheit habe ich in meinen Programmen auf solche Methoden verzichtet. Im Übrigen sollen die Programme nur zur Veranschaulichung des Prinzips dienen und nicht zu einer quantitativen Betrachtung von Graphen herangezogen werden.

Übertragung des Raums auf eine Fläche

Das Besondere der Darstellungsweise am Computer ist die Übertragung des virtuellen dreidimensionalen Raumes auf den zweidimensionalen Bildschirm. Da gibt es grundsätzlich einmal die Möglichkeit, die dritte Dimension zu vernachlässigen und sich einzig auf die Projektionen der Graphen zu beschränken. Sinnvoll sind aber nur die Projektionen auf eine parallele Ebene zur x-z bzw. y-z-

Ebene. Für genau diesen Zweck habe ich das Programm fach_a01 geschrieben. Es ist in der Programmiersprache BASIC bzw. genauer im QuickBASIC-Dialekt abgefaßt. Die Benutzungshinweise befinden sich im Programmtext, so daß hier nur auf die Ausgabe eingegangen wird.



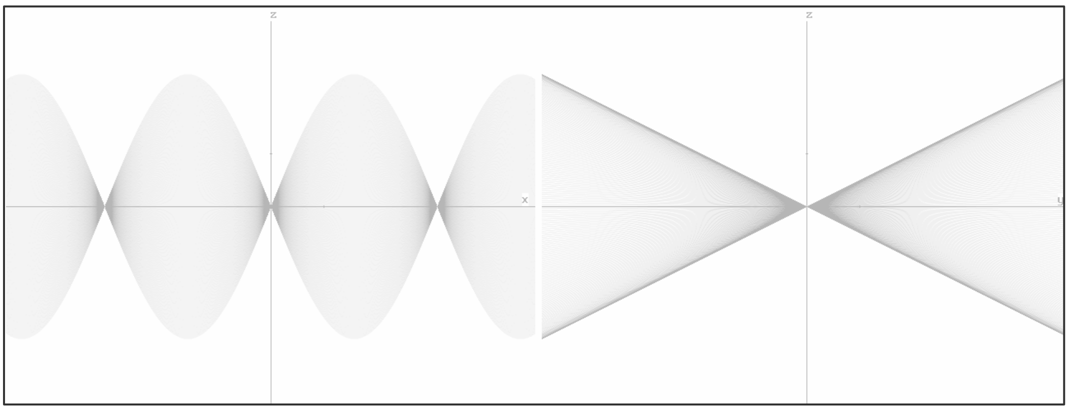
Dargestellt wurde die Funktion $f(x,y)=\frac{\sin x}{2}y$ für x, y Intervall [-5;5]

Erkennbar sind besonders bei der Projektion auf eine y-z-Ebene die Maserung, die sich aus den oben erwähnten Rundungs- und anderen Ungenauigkeit ergeben.

Mit Hilfe dieses ersten Programms kann man sich nun einen ungefähren Überblick über das Aussehen des Graphen verschaffen, besonders wenn die Betrachtung von Extrema in einem bestimmten Intervall gefordert ist.

Soll nun die dritte Dimension mit einbezogen werden, ist es möglich, die gezeichneten Bildpunkte auf der Projektionsebene intensiver darzustellen, falls häufiger Graphenpunkte darauf projiziert werden.

Für diese Aufgabe existiert das vom vorigen Programm abgewandelte fach_a02. Es liefert zur gleichen Bedingung folgende Ausgabe:

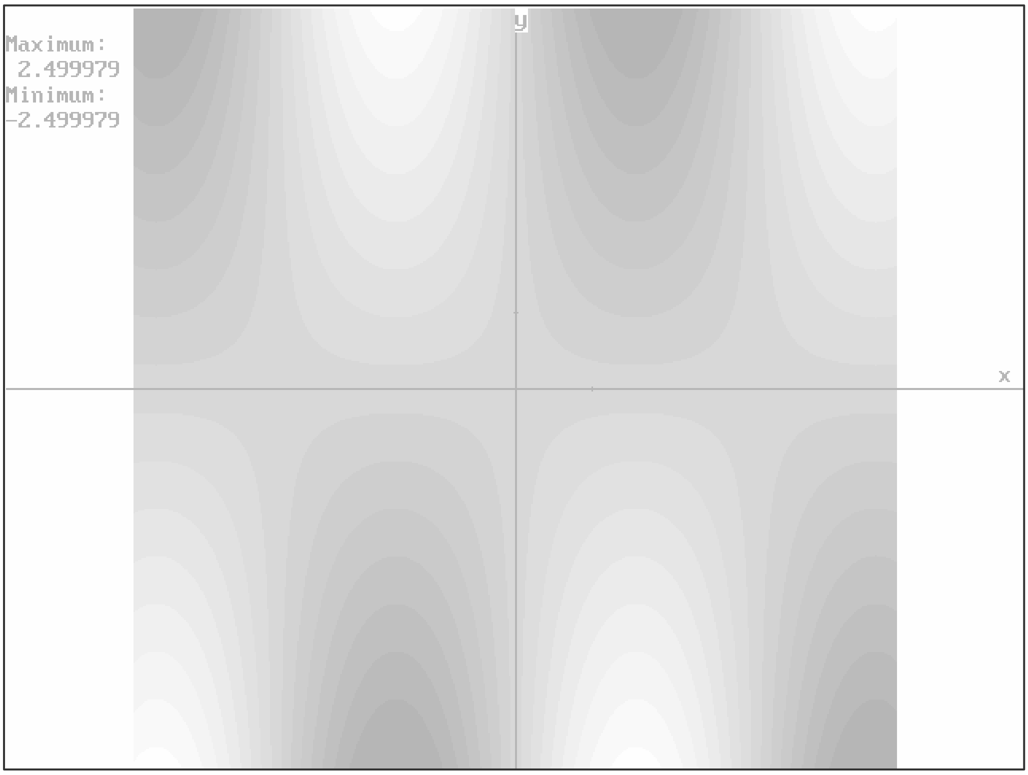


Es existiert noch eine hier nur kurz erwähnte Methode, Schnitte durch einen Graphen darzustellen. Dazu wird lediglich die Funktion

$(x, y) \rightarrow f(x, y) = z$

mit konstantem x in einem zweidimensionalen y-z-System dargestellt bzw. dementsprechend mit konstantem y in einem x-z-System. Diese Schnitte liefert das Programm fach_a03

An den bisher genannten Darstellungsweisen lassen sich aber immer noch keine Verläufe des Graphen erkennen, lediglich dessen Extrema und Tendenzen. Es scheint also eine Art Aufsicht auf die x-y-Ebene notwendig zu sein. Die Projektion an einer Parallelen zur x-y-Ebene ist nur für Funktionen interessant, die Definitionslücken besitzen, denn ansonsten liefern diese Projektionen eine mathematische Ebene bzw. eine Fläche am Computer. Wie oben ist es möglich zur Visualisierung der dritten Dimension das Mittel der Schattierung einzusetzen. Das Programm fach_a04 liefert genau solche Ausgaben, bei denen größere z durch intensivere Punkte gezeigt werden. Hier wieder mit der gleichen Funktion *f*(x, y) = 0,5sin(x)y im Intervall [-5,5] für x und y:

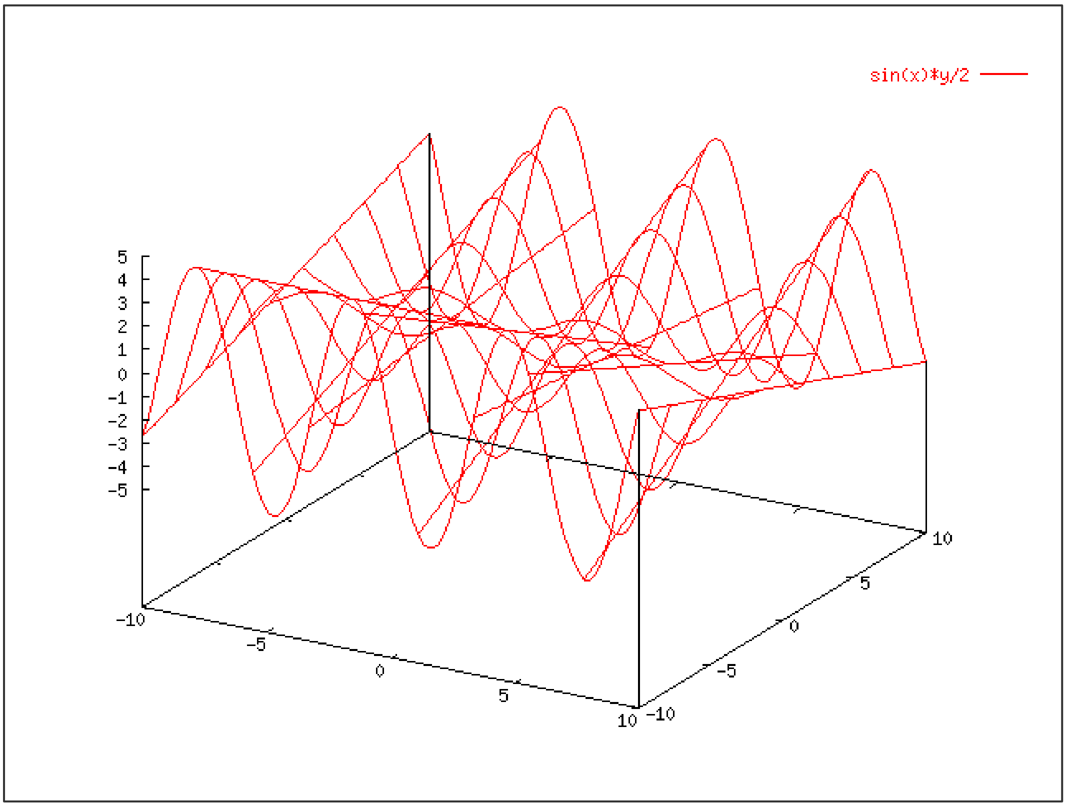


Zusätzlich liefert das Programm Maximum und Minimum der ihm begegneten Funktionswerte, weil es diese zur Berechnung der Schattierungsstufen benötigt.

Räumliche Abbilder

Durch die bis jetzt genannten Darstellungsarten wäre es nun möglich, die Form des dreidimensionalen Graphen qualitativ zu rekonstruieren. Tatsächlich fehlt es aber weiterhin an Anschaulichkeit.

Programme wie Derive von Texas Instruments oder das freie gnu-plot bieten die Möglichkeit, zweidimensionale Bilder der dreidimensionalen Graphen für beliebige Blickwinkel, Ausschnitte und Vergrößerungen zu liefern.



gnu-plot liefert bei Standardparametern nur einzelne Schnitte des Graphen auf folgender mathematischer Grundlage:

f sei eine beliebige zu zeichnende Funktion und *d* die Funktion des dazugehörigen Gittermodells.

Angenommen

$$graphf = \{(x, y, f(x, y)) \mid x, y \in R\}$$

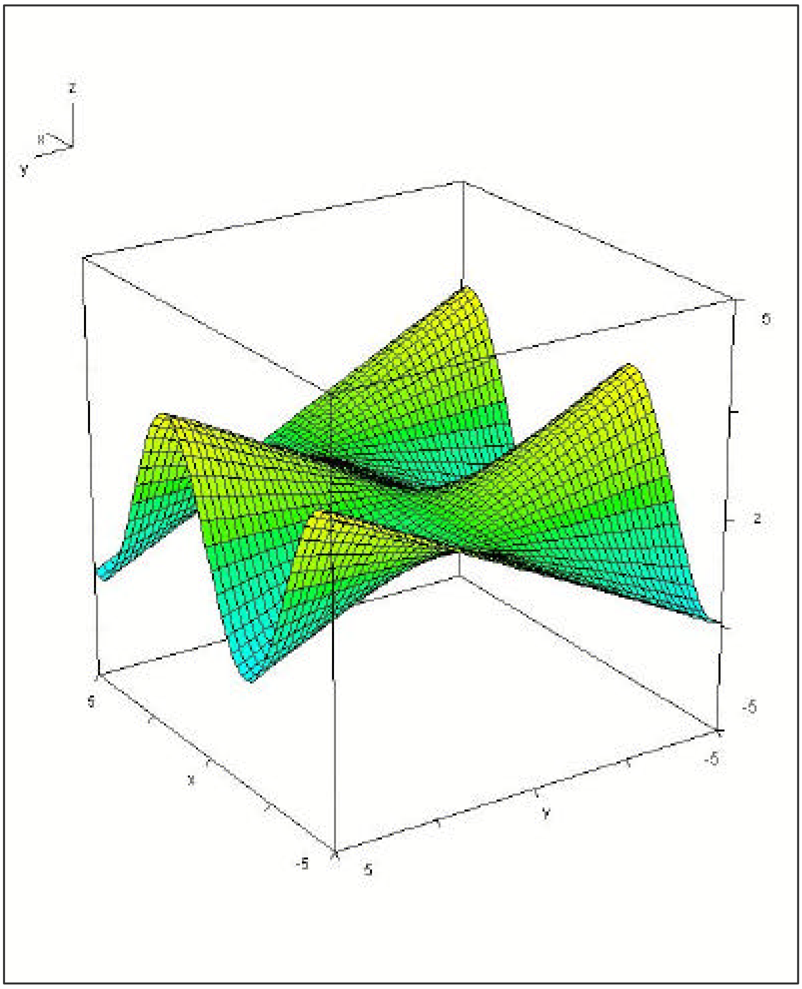
dann gilt

$$graphd = \{(x, y, f(x, y)) \mid (x = c_1 \cdot n \wedge y \in R) \vee (x \in R \wedge y = c_2 \cdot n)\}\}$$
$$n \in Z$$

dabei sind c_1 und c_2 beliebige Konstanten, die den Abstand zwischen zwei Gitterlinien jeweils in x und y-Richtung bestimmen.

Derive hingegen benutzt nur die Schnittpunkte der Gitterlinien des gnu-plot-Modells, hier angegeben als die Menge S aller Schnittpunkte

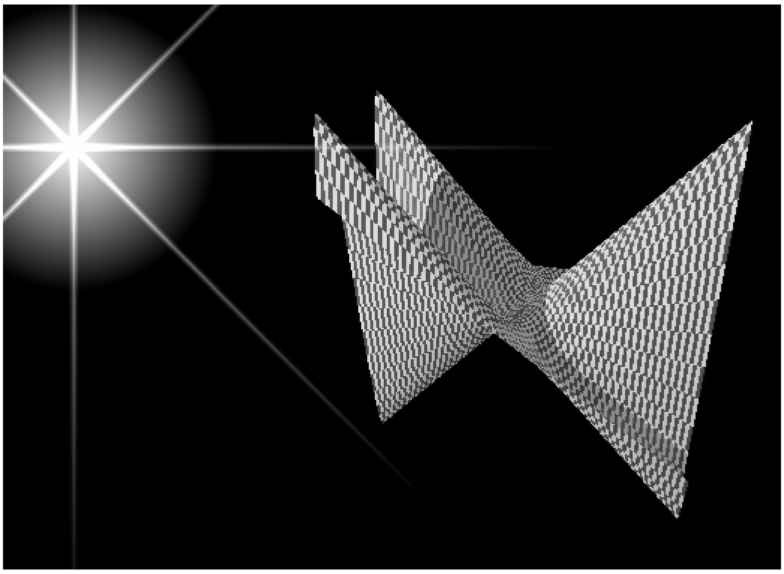
$$S = \{(x, y, f(x, y)) \mid x = c_1 \cdot n \wedge y = c_2 \cdot n\}$$
$$n \in Z$$



Die Graphenfläche an sich wird durch eine Vielzahl kleinerer flacher Teilflächen dargestellt, die jeweils durch vier dieser Schnittpunkte begrenzt werden.

Für ein fotorealistischeres Abbild ist der Einsatz eines sogenannten Raytracing-Programms sinnvoll. Raytracing bedeutet, daß als Grundlage ein mit Flächen genähertes Modell des zeigenden Objekts angelegt wird. Im Gegensatz zu Derive sind es aber meist Dreiecke, weil diese universeller einsetzbar sind als Vierecke. Nun werden Lichtstrahlen simuliert, die von einer Lichtquelle ausgehen, von Flächen reflektiert werden und schließlich in der Kamera enden. (nach Wardley, A. und Dilger, A., 1996) Diese „Strahlenverfolgung“ erfolgt auch oft in entgegengesetzter Richtung, also von Kamera zur Lichtquelle.

Das hier verwendete Reflections 4.1 in der Light-Version (kurz: REF 4.1l) ist auf die Darstellung fotorealistischer Einzelbilder und auch Filmszenen unterschiedlichster Arten von Körpern entwickelt worden. Aus diesem Grund ist es auch nicht ohne weiteres möglich, eine mathematische Funktion korrekt darzustellen was Skalierung und Zeichenintervall betrifft. Der Einsatz einer solchen Software ist also nur dann sinnvoll, wenn eine optisch stark aufbereitete Darstellung gewünscht ist und die mathematische Grundlage in den Hintergrund rückt. Einsatzgebiete könnten dabei zum Beispiel die Präsentation von Maschinenteilen, die als Basis eine mathematische Funktion haben, speziell die im nächsten Kapitel folgende Stellvertreterfunktion für den Altwarmbüchener See oder wie hier in der vorherigen Abbildung der Schattenwurf durch eine Lichtquelle. Auf der beigelegten CD-ROM befindet sich eine Datei namens „film01.avi“, die die Drehung des Graphen um die z-Achsen in Form eines Videos darstellt.



Untersuchung von Extrema

Allgemeines zur Untersuchung von Extrema

Zur Feststellung von Extrema scheint der Weg über die Ableitung, der bei Funktionen des Typs R   R ein sehr hilfreiches Mittel ist, wieder einer Untersuchung wert. Zuerst muß aber darüber diskutiert werden, wann eine Funktion differenzierbar ist und wie dabei dann zu verfahren ist.

Stetigkeit

Die aus den Funktionen mit einer Variablen bekannte Definition der Stetigkeit in einer etwas anderen Form lautet (nach Schweizer, W., et al., 1976):

$$\lim_{\substack{x \to x_0 \\ x < x_0}} f(x) = f(x_0)$$

$$\lim_{\substack{x \to x_0 \\ x > x_0}} f(x) = f(x_0)$$

Sind beide Aussagen wahr, dann existiert der rechtsseitige und der linksseitige Grenzwert und entspricht dieser dem Funktionswert an der Stelle x_0 so ist die Funktion in x_0 stetig. Ist die Aussage für alle x_0 0 D wahr so ist die Funktion global stetig.

Wenn dies nun auf Funktionen mit zwei Variablen übertragen wird, so ergibt sich logischerweise, daß gilt:

$$\lim_{\substack{x \to x_0 \\ y \to y_0}} f(x, y) = f(x_0, y_0)$$

Ist diese Aussage wahr für alle vier Möglichkeiten wie sich x und y zu x_0 und y_0 verhalten, so ist die Funktion an dieser Stelle stetig. Die globale Stetigkeit wäre dann analog zu oben, für alle (x, y) 0 D

Daß die Betrachtung nur einer Variable nicht genügt, zeigt folgendes Beispiel

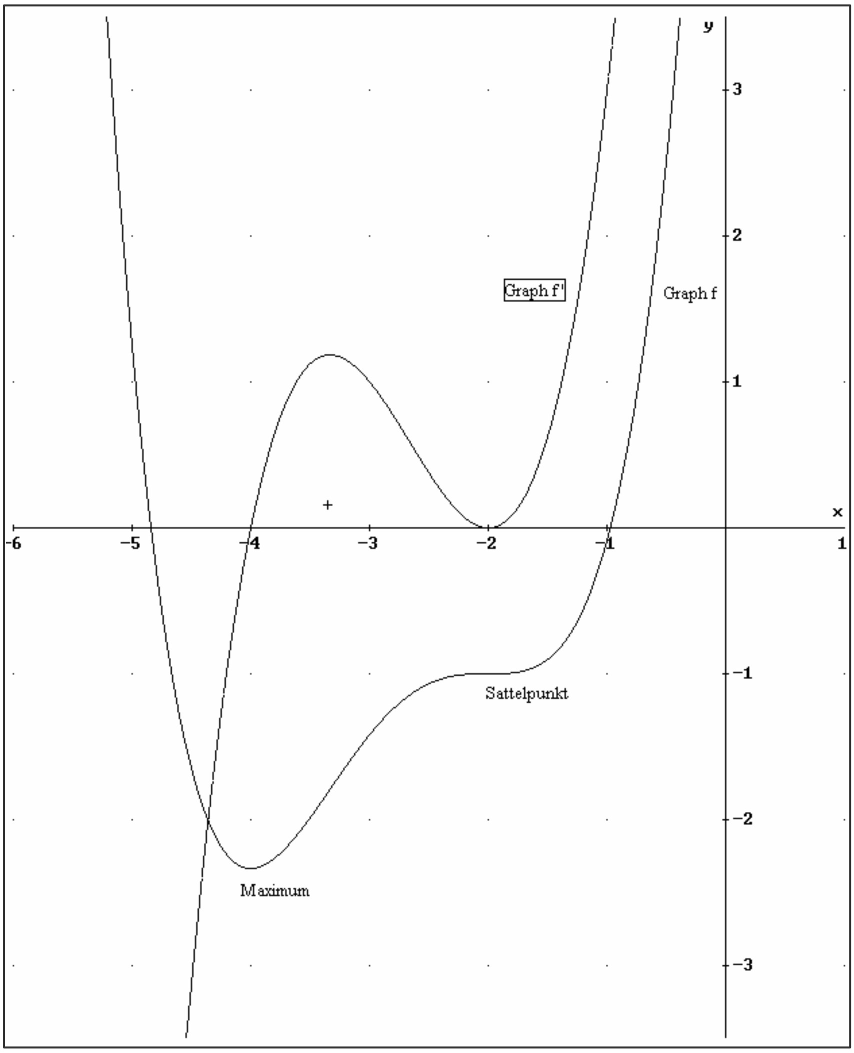
$$f(x, y) = \frac{x^2}{y}$$

$$\lim_{y \to 0} f(x, y) \neq f(0,0)$$

Das Beispiel erfüllt zwar für das Argument (0,0) bei der Betrachtung für x alle Bedingungen der Stetigkeit, für y allerdings keine: Der linksseitige Grenzwert ist - , der rechtsseitige + und der Funktionswert ist für y = 0 nicht einmal definiert; d.h. selbst wenn die Definitionslücke geschlossen wird, indem *f*(0,0) = 0 definiert wird, so liegt aufgrund der unterschiedlichen Grenzwerte immer noch keine Stetigkeit vor.

Überlegungen zur Bestimmung von Extrema

Über Funktionen mit einer Variablen ist bekannt, daß ein Extremum nur denn vorliegt, wenn die Ableitung an dieser Stelle einen Nulldurchgang aufweist. Andersherum bedeutet dies für die Suche nach Extrema, daß diese durch die Suche der Nulldurchgänge der ersten Ableitung zu finden sind. Zu beachten ist, daß es wahre Durchgänge sein müssen und nicht einfach nur Nullstellen.



Die Überlegung, die dabei zugrunde liegt, ist die, daß an Extrema die Steigung (1. Ableitung) des Graphen 0 ist. Da dies bei Sattelpunkten auch der Fall ist, wird ein weiteres Kriterium nötig: Nur an Extrema wechselt das Vorzeichen der Steigung. Anschaulich bedeutet dies, daß der Graph nach einem Maximum nicht mehr weiter steigen bzw. nach einem Minimum nicht weiter fallen kann, nach einem Sattelpunkt aber schon.

Die Überlegungen müssen nun sinnvoll auf beliebige Graphen stetiger Funktion von R^2 R übertragen werden. Angenommen, der Graph habe im Punkt $P(x_0, y_0, z_0)$ ein Minimum. Erkennbar ist dies rein anschaulich daran, daß bei Schnitten nach dem im ersten Kapitel genannten Verfahren an parallelen Ebenen zur x-z- und y-z-Ebene durch P die nun zweidimensionale Graphen ebenfalls Minima haben. Auch ebenfalls aus dem ersten Kapitel ist bekannt, daß sich diese Graphen

$$graphxz\{(x, y_0, f(x, y_0))\} \Rightarrow z = f(x, y_0)$$
$$graphyz\{(x_0, y, f(x_0, y))\} \Rightarrow z = f(x_0, y)$$

wie folgt ergeben:

Wenn diese Schnitte also jeweils ein Minimum an der gleichen Stelle haben, so hat auch die Originalfunktion ein Minimum. Es steht fest, daß dies natürlich auch für Maxima gelten muß.

Die Suche nach Extrema

$$f(x, y) = (x-1)^2 + (y+1)^2 + xy$$

Ableitung :

$$\frac{\partial f}{\partial x} = 2x - 2 + y$$

Nulldurchgänge :

$$2x + y - 2 = 0$$
$$\Leftrightarrow x = 1 - \frac{y}{2}$$

Ableitung :

$$\frac{\partial f}{\partial y} = 2y + 2 + x$$

Nulldurchgänge

$$x + 2y + 2 = 0$$
$$\Leftrightarrow y = -1 - \frac{x}{2}$$

Lösen :

$$x = 1 - \frac{y}{2} \wedge y = -1 - \frac{x}{2}$$
$$\Leftrightarrow x = 2 \wedge y = -2$$
$$f(2,-2) = -2$$

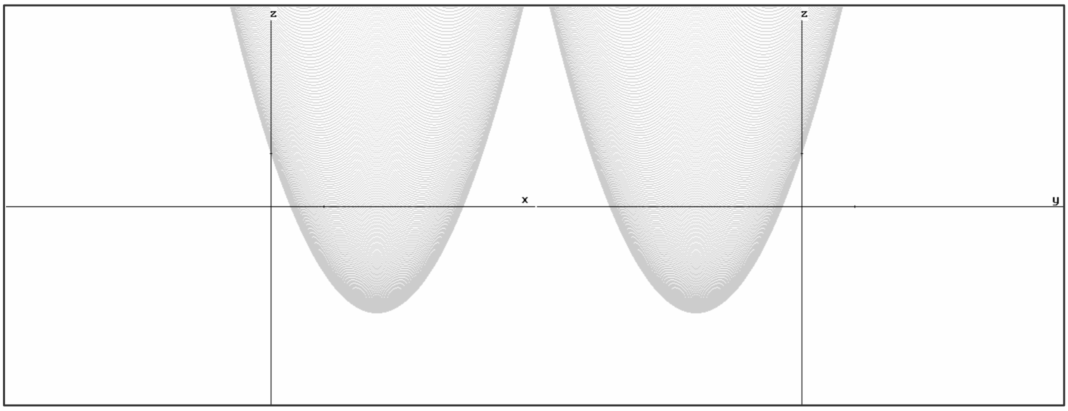
Sollen nun Extrema einer Funktion unbekannten Aussehens bestimmt werden, so bietet sich aus den vorhergegangen Überlegungen über Schnitte folgende Vorgehensweise an, die an einem Beispiel verdeutlicht wird.

Die Funktion wird nach x und y abgeleitet und die Nulldurchgänge dieser sogenannten partiellen Ableitungen werden gesucht. Die jeweils andere Funktionsvariable wird als Konstante behandelt, wie es zum Beispiel auch bei Funktionsscharen getan wird. (nach Marsden, J. E. und Tromba, A. J., 1995)

Dadurch wurden die Extrema beliebiger Schnitte für x in Abhängigkeit von y bzw. für y in Abhängigkeit von x gefunden.

Da beide partielle Ableitungen ein Vorzeichenwechsel von negativ nach positiv

haben, so handelt es sich beide Male um ein Minimum. Zu suchen sind also die x und y, für die beide Aussagen wahr werden.
Demnach liegt das Minimum im Punkt T(2,-2,-2)
Anschaulich wird dieses Ergebnis durch die Ausgabe von fach_a01 bestätigt:



Einem Sonderfall soll noch Beachtung geschenkt werden, es dürfen nur durch die partiellen Ableitungen gefundenen Extrema der gleichen Art verknüpft werden.
Gegenbeispiel:

$$f(x,y) = x^2 - y^2$$

$$\frac{\partial f}{\partial x} = 2x \qquad \frac{\partial f}{\partial y} = -2y$$

$$2x = 0 \Leftrightarrow x = 0 \qquad -2y = 0 \Leftrightarrow y = 0$$

Daraus könnte man fälschlicherweise schließen, P(0,0,0) sei ein Extremum. Da aber die partiellen Ableitungen unterschiedliche Vorzeichenwechsel aufweisen, ist dieser Punkt <u>kein</u> Extremum der Ausgangsfunktion.

Rückblickend war eine detaillierte Betrachtung der Stetigkeit notwendig, in der eigentlich die Differenzierbarkeit der Funktionen überprüft wurde, aus denen dann die partiellen Ableitungen gebildet wurden.

Beispiel Altwarmbüchener See

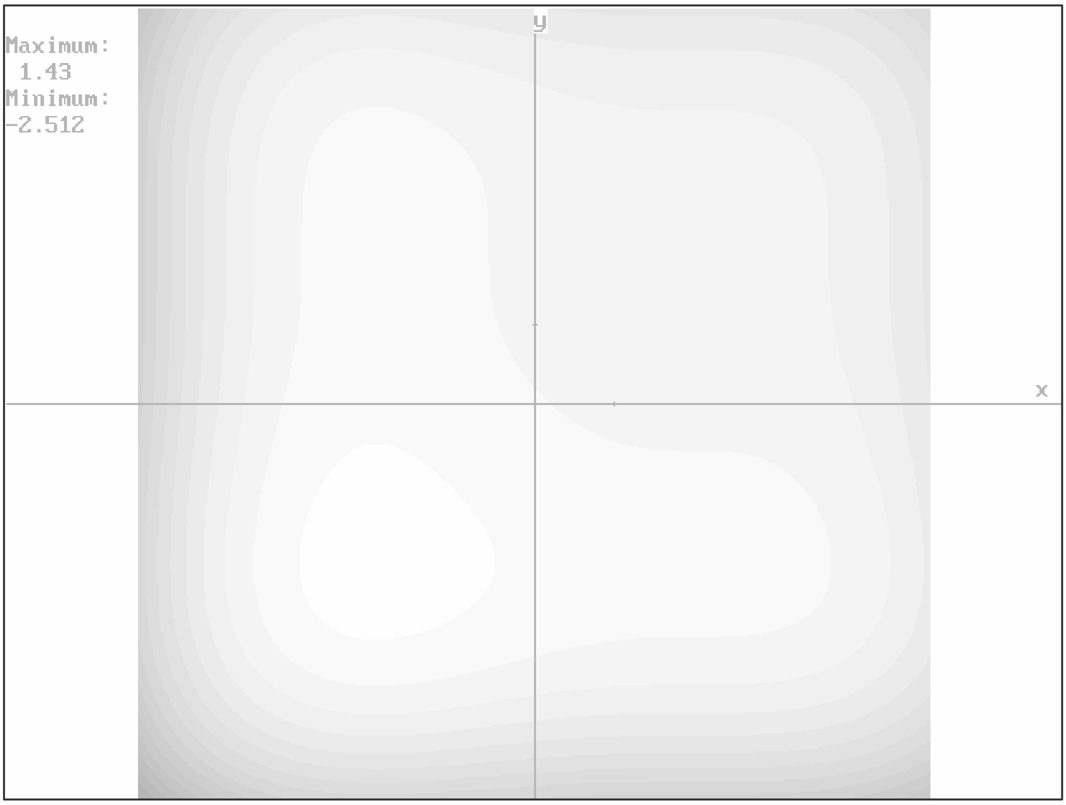
Da das Finden selbst einer sehr ungenauen Funktion zur Beschreibung des Altwarmbüchener Sees zu einem viel zu komplexen Funktionsterm führen würde und nicht zuletzt auch den Rahmen dieser Arbeit wesentlich sprengen würde, sei hier eine Funktion gewählt, die einen beliebigen See darstelle und zur Veranschaulichung der in den vorherigen Kapiteln erwähnten Verfahren dienen soll.

Die folgende Funktion *f* gibt zu jeder Position (x, y) auf der Seeoberfläche die entsprechende Höhe z des Grundes zur Wasseroberfläche an.
Die Funktion ist aber nur für Höhen unter dem Wasserspiegel definiert

$$f(x,y) = \frac{(x-2)^3 \cdot (3x+10) + (y-2)^3 \cdot (3y+10)}{1000} - 2$$

$$D = \{(x,y) \in R^2 \mid f(x,y) < 0\}$$

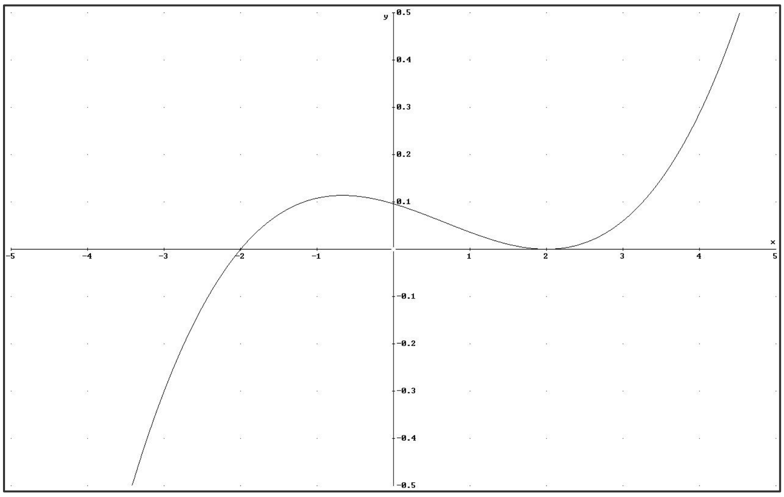
fach_a04 liefert zu dieser Funktion die folgende Ausgabe:



Die Funktion wird nun nach x und nach y abgeleitet und deren Nulldurchgänge bestimmt:

$$\textit{Ableiten}:$$
$$\frac{\partial f}{\partial x} = \frac{3 \cdot (x^3 - 2x^2 - 4x + 8)}{250}$$
$$\textit{Nullstellen}$$
$$0 = \frac{3 \cdot (x^3 - 2x^2 - 4x + 8)}{250} \Leftrightarrow x = -2 \vee x = 2$$

Der Graph der Ableitung zeigt, daß nur bei x = –2 ein Vorzeichenwechsel und zwar von negativ nach positiv vorliegt.



Aufgrund der einfach gewählten Funktion entsprechen sich diese Ableitung und die nach y bis auf das Argument:

$$\textit{Ableiten}:$$
$$\frac{\partial f}{\partial y} = \frac{3 \cdot (y^3 - 2y^2 - 4y + 8)}{250}$$
$$\textit{Nullstellen}$$
$$0 = \frac{3 \cdot (y^3 - 2y^2 - 4y + 8)}{250} \Leftrightarrow y = -2 \vee y = 2$$

Dementsprechend ist auch nur bei y = –2 ein Vorzeichenwechsel von negativ nach positiv.

Beide Vorzeichenwechsel deuten auf ein Minimum hin, so daß feststeht, daß der See im Punkt T(-2;-2;*f*(-2,-2)) die tiefste Stelle mit etwa 2,5 Längeneinheiten hat; die Ausgabe des Programms fach_a04 wird also bestätigt.

Dokumentation zu den eigenen Programmen

Die Programme bestehen alle aus zwei Teilen. Im ersten Teil werden Konstanten wie das Zeichnungsintervall und die Funktion an sich definiert. Außerdem enthält er einen kurzen Benutzungshinweis.

Eine Möglichkeit durch den Benutzer eingegebene Terme in eine für den Computer verständliche Zahl umzuwandeln existiert in der Sprache BASIC nicht. Daher muß auf die Fähigkeit des Interpreters zurückgegriffen werden und die Funktionen müssen direkt im Quelltext eingegeben werden.

Der zweite Programmteil besteht im Wesentlichen aus mehreren verschachtelten Schleifen für die Argumente und Benutzereingaben.

Kurzbeschreibung zu dem Funktionsumfang der eigenen Programme

fach_a01

Dieses Programm liefert die Projektionen der Graphen wahlweise auf die x-z- bzw. y-z-Ebene. Die Ansicht kann mit der Leertaste gewechselt werden. Das Intervall kann durch die Konstante *i* als [-*i*; *i*] bestimmt werden. Der Graph darf keine Definitionslücken besitzen.

fach_02

Das Programm entspricht fach_a01, nur daß mittels unterschiedlicher Schattierungen die Häufigkeit der sich räumlich „dahinter“ befindenden Graphenpunkte dargestellt wird. Außerdem kann mit + und – die Schattierungsintensität verändert werden, falls der Graph zu hell oder zu dunkel erscheinen sollte.

fach_a03

Dieses Programm liefert Schnitte durch den Graphen an Parallelen zur x-z und –y-z-Ebene. Mittels Leertaste wird wieder die Ansicht im definierten Intervall [-*i*; *i*] geändert und mit + und – wird die Schnittebene räumlich verschoben.

fach_a04

Dieses Programm zeichnet eine schattierte Aufsicht auf die x-y-Ebene, größere *z* werden dabei intensiver dargestellt.

Anhang

Quellenverzeichnis

Schweizer, W., et al.	1976	Analysis. Ernst Klett Verlag Stuttgart, 3. Auflage, S. 64
Wardley, A., Dilger, A.	1996	comp.graphics.rendering.raytracing FAQ (part 1/2) – What is Ray Tracing? http://www.povray.org/documents/rayfaq/part1/faq-doc-2.html
Marsden, Jerrold E., Tromba, Anthony J.	1995	Vektoranalysis. Spektrum Akad. Vlg., Heidelberg, S. 114 –115

Quelltexte

Quelltext zu fach_a01

```
'Programm zum Zeichnen von Silhouetten (Projektionen)von
'R² -> R Funktionsgraphen

'Form: f(x,y)=z         für x,y im Intervall [-i;i]
'ACHTUNG: Der Graph darf in diesem Intervall keine Definitionslücken haben!

'Bitte geben Sie den Funktionsterm in folgender Form ein:
'DEF FNz(x,y) = sin(x+y), wobei sin(x+y) der Funktionsterm ist

DEF FNz (x, y) = ((x - 2) ^ 3 * (3 * x + 10) + (y - 2) ^ 3 * (3 * y + 10)) / 1000 - 2

'Bitte geben Sie das Zeichenintervall [-i;i] an
CONST i = 7

'ALLGEMEINE BEDIENUNG:  [ESC] beendet das Programm
'                       [LEERTASTE] wechselt die Ansicht
'                       jede andere Taste führt zu einer erneuten Zeichnung

'Ab hier folgt der normale Programmtext
ON ERROR GOTO fehlerbehandlung:
DIM x AS SINGLE, y AS SINGLE, argument AS SINGLE' Variablen als "einfach genau" definieren
SCREEN 12      'Gafik mit 640 x 480 Pixel initialisieren
intervallfaktor = i / 320' Umrechnungskonstante von math.- auf
                              ' Bildschirmeinheiten (Pixel)
WHILE NOT (Benutzereingabe$ = CHR$(27))  ' warten bis Benutzer [ESC] drückt
     CLS ' Bildschirm zurücksetzen
     FOR x = -i TO i STEP intervallfaktor              'Schleife für x
          FOR y = -i TO i STEP intervallfaktor         'Schleife für y
          IF anzeigemodus = 0 THEN argument = x ELSE argument = y 'Anzeige Modus beachten
               PSET (320 + argument / intervallfaktor, 240 - FNz(x, y) / intervallfaktor), 4
               ' Punkt setzen (PointSET)
          NEXT y
     NEXT x
     LINE (0, 240)-(640, 240): LINE (320, 0)-(320, 480)      'Achsenkreuz zeichen
     LOCATE 1, 41: PRINT "z": LOCATE 15, 79                  'und beschriften
     LINE (320 + 1 / intervallfaktor, 239)-(320 + 1 / intervallfaktor, 241)
     LINE (319, 240 - 1 / intervallfaktor)-(321, 240 - 1 / intervallfaktor)
     IF anzeigemodus = 0 THEN PRINT "x" ELSE PRINT "y"
     Benutzereingabe$ = INPUT$(1)                      ' Benutzereingabe einlesen
     IF Benutzereingabe$ = " " THEN anzeigemodus = NOT (anzeigemodus)' Anz.-Mod. wechseln
WEND
SYSTEM  'Programm beenden

fehlerbehandlung: 'Fehlerbehandlungsroutine
IF ERR = 11 THEN PRINT "Der Graph hat möglicherweise eine Definitionslücke ('Division by zero')"
PRINT "Fehler: "; ERR'sonstiger Fehler
SYSTEM
```

Quelltext zu fach_a02

```
'Programm zum Zeichnen von SCHATTIERTE Silhouetten (Projektionen) von
'R² -> R Funktionsgraphen
'SCHATTIERUNG:  häufig gezeichnete Punkte werden heller rot,
'               als selten gezeichnete

'Form: f(x,y)=z        für x,y im Intervall [-i;i]
'ACHTUNG: Der Graph darf in diesem Intervall keine Definitionslücken haben!

'Bitte geben Sie den Funktionsterm in folgender Form ein:
'DEF FNz(x,y) = sin(x+y), wobei sin(x+y) der Funktionsterm ist

DEF FNz (x, y) = ((x - 2) ^ 3 * (3 * x + 10) + (y - 2) ^ 3 * (3 * y + 10)) / 1000 - 2

'Bitte geben Sie das Zeichenintervall [-i;i] an
CONST i = 7

'ALLGEMEINE BEDIENUNG:  [ESC] beendet das Programm
'                       [LEERTASTE] wechselt die Ansicht
'                       [+] und [-] verändert die Schattierung
'                       jede andere Taste führt zu einer erneuten Zeichnung

'Ab hier folgt der normale Programmtext
'ON ERROR GOTO fehlerbehandlung:
DIM x AS SINGLE, y AS SINGLE, argument AS SINGLE' Variablen als "einfach genau" definieren
SCREEN 12       'Gafik  mit 640 x 480 Pixel initialisieren
FOR temp = 0 TO 15                              'Farbpalette auf
       PALETTE temp, (temp * 4) '* (1 + 256 + 65536)  'Rotabstufungen
NEXT                                            'abändern
intervallfaktor = i / 320' Umrechnungskonstante von math.- auf
                               ' Bildschirmeinheiten (Pixel)
farbfaktor = i / 8
shattkorr = 0
WHILE NOT (Benutzereingabe$ = CHR$(27))  ' warten bis Benutzer [ESC] drückt
       CLS ' Bildschirm zurücksetzen
       FOR x = -i TO i STEP intervallfaktor                'Schleife für x
              FOR y = -i TO i STEP intervallfaktor         'Schleife für y
              IF anzeigemodus = 0 THEN argument = x ELSE argument = y 'Anzeige Modus beachten
                     farbwert = POINT(320 + argument / intervallfaktor, 240 - FNz(x, y) / intervallfaktor)
                     PSET (320 + argument / intervallfaktor, 240 - FNz(x, y) / intervallfaktor), farbwert + farbfaktor + schattkorr
                     ' Punkt setzen (PointSET)
              NEXT y
       NEXT x
       LINE (0, 240)-(640, 240): LINE (320, 0)-(320, 480)      'Achsenkreuz zeichnen
       LINE (320 + 1 / intervallfaktor, 239)-(320 + 1 / intervallfaktor, 241)
       LINE (319, 240 - 1 / intervallfaktor)-(321, 240 - 1 / intervallfaktor)
       LOCATE 1, 41: PRINT "z": LOCATE 15, 79                 'und beschriften
       IF anzeigemodus = 0 THEN PRINT "x" ELSE PRINT "y"
       Benutzereingabe$ = INPUT$(1)                          ' Benutzereingabe einlesen
       IF Benutzereingabe$ = " " THEN anzeigemodus = NOT (anzeigemodus)' Anz.-Mod. wechseln
       IF Benutzereingabe$ = "+" THEN schattkorr = schattkorr + .5
       IF Benutzereingabe$ = "-" THEN schattkorr = schattkorr - .5
WEND
SYSTEM  'Programm beenden

fehlerbehandlung: 'Fehlerbehandlungsroutine
IF ERR = 11 THEN PRINT "Der Graph hat möglicherweise eine Definitionslücke ('Division by zero')"
PRINT "Fehler: "; ERR'sonstiger Fehler
SYSTEM
```

Quelltext zu fach_a03

```
'Programm zum Zeichnen von Schnitten durch
'R² -> R Funktionsgraphen

'Form: f(x,y)=z        für x,y im Intervall [-i;i]
'ACHTUNG: Der Graph darf in diesem Intervall keine Definitionslücken haben!

'Bitte geben Sie den Funktionsterm in folgender Form ein:
'DEF FNz(x,y) = sin(x+y), wobei sin(x+y) der Funktionsterm ist

DEF FNz (x, y) = ((x - 2) ^ 3 * (3 * x + 10) + (y - 2) ^ 3 * (3 * y + 10)) / 1000 - 2

'Bitte geben Sie das Zeichenintervall [-i;i] an
CONST i = 5

'Schrittweite für Schnittabstand
CONST schritt = .1

'ALLGEMEINE BEDIENUNG:  [ESC] beendet das Programm
'                  [LEERTASTE] wechselt die Ansicht
'                  mit [+] und [-] kann die Schnittfläche
'                   um die Konstante schritt verändert werden
'                  jede andere Taste führt zu einer erneuten Zeichnung


'Ab hier folgt der normale Programmtext
ON ERROR GOTO fehlerbehandlung:
DIM x AS SINGLE, y AS SINGLE, argument AS SINGLE' Variablen als "einfach genau" definieren
SCREEN 12      'Gafik mit 640 x 480 Pixel initialisieren
intervallfaktor = i / 320' Umrechnungskonstante von math.- auf
                           ' Bildschirmeinheiten (Pixel)
WHILE NOT (Benutzereingabe$ = CHR$(27))  ' warten bis Benutzer [ESC] drückt
      CLS ' Bildschirm zurücksetzen
      LINE (0, 240)-(640, 240): LINE (320, 0)-(320, 480)      'Achsenkreuz zeichnen
      LINE (320 + 1 / intervallfaktor, 239)-(320 + 1 / intervallfaktor, 241)
      LINE (319, 240 - 1 / intervallfaktor)-(321, 240 - 1 / intervallfaktor)
      FOR argument1 = -i TO i STEP intervallfaktor / 4              'Schleife für argument1
            x = argument1: y = argument2
            IF NOT (anzeigemodus = 0) THEN SWAP x, y'Anzeige Modus beachten
            PSET (320 + argument1 / intervallfaktor, 240 - FNz(x, y) / intervallfaktor), 4
                  ' Punkt setzen (PointSET)
      NEXT argument1
      LOCATE 1, 41: PRINT "z": LOCATE 15, 79                'und beschriften
      IF anzeigemodus = 0 THEN PRINT "x" ELSE PRINT "y"
      LOCATE 30, 35: PRINT "Schnitt: ";
      IF anzeigemodus = 0 THEN PRINT "y";  ELSE PRINT "x";
      PRINT "="; argument2;
      Benutzereingabe$ = INPUT$(1)                         ' Benutzereingabe einlesen
      SELECT CASE Benutzereingabe$
      CASE " "
            anzeigemodus = NOT (anzeigemodus)' Anz.-Mod. wechseln
      CASE "+"
            argument2 = argument2 + schritt
      CASE "-"
            argument2 = argument2 - schritt
      CASE ELSE
      END SELECT

WEND
SYSTEM  'Programm beenden

fehlerbehandlung: 'Fehlerbehandlungsroutine
IF ERR = 11 THEN PRINT "Der Graph hat möglicherweise eine Definitionslücke ('Division by zero')"
IF ERR = 6 THEN PRINT "Die erreichnete Zahl ist möglicherweise zu groß ('Overflow')"
PRINT "Fehler: "; ERR'sonstiger Fehler
SYSTEM
```

Quelltext zu fach_a04

```
'Programm zum Zeichnen von schattierten Aufsichten von
'R² -> R Funktionsgraphen
'SCHATTIERUNG:  höhere Punkte (kleine z) sind heller als Tiefere (große z)

'Form: f(x,y)=z         für x,y im Intervall [-i;i]
'ACHTUNG: Der Graph darf in diesem Intervall keine Definitionslücken haben!

'Bitte geben Sie den Funktionsterm in folgender Form ein:
'DEF FNz(x,y) = sin(x+y), wobei sin(x+y) der Funktionsterm ist

DEF FNz (x, y) = ((x - 2) ^ 3 * (3 * x + 10) + (y - 2) ^ 3 * (3 * y + 10)) / 1000 - 2

'Bitte geben Sie das Zeichenintervall [-i;i] an
CONST i = 5

'ALLGEMEINE BEDIENUNG:  [ESC] beendet das Programm
'                       jede andere Taste führt zu einer erneuten Zeichnung

'Ab hier folgt der normale Programmtext
'ON ERROR GOTO fehlerbehandlung:

DIM x AS SINGLE, y AS SINGLE, argument AS SINGLE' Variablen als "einfach genau" definieren
SCREEN 12       'Gafik mit 640 x 480 Pixel initialisieren
FOR temp = 0 TO 15                              'Farbpalette auf
    PALETTE temp, (temp * 4) '* (1 + 256 + 65536)  'Rotabstufungen
NEXT                                    'abändern
intervallfaktor = i / 240' Umrechnungskonstante von math.- auf
                                ' Bildschirmeinheiten (Pixel)
WHILE NOT (Benutzereingabe$ = CHR$(27))  ' warten bis Benutzer [ESC] drückt
'Erster Schleifendurchlauf um die absoluten Extrema zu suchen
'  zur Anpassung der 16 (von 0 bis 15) möglichen Schattierungen
    LOCATE 12, 35: PRINT "Einen Moment bitte..."
    FOR x = -i TO i STEP intervallfaktor                'Schleife für x
        FOR y = -i TO i STEP intervallfaktor            'Schleife für y
            z = FNz(x, y)   'nur einmal berechnen, spart Rechenzeit
            IF maximum < z THEN maximum = z
            IF minimum > z THEN minimum = z
        NEXT y
    NEXT x
    farbfaktor = 16 / (maximum - minimum)
    CLS ' Bildschirm zurücksetzen
'Zweite Schleife zum Zeichnen
    FOR x = -i TO i STEP intervallfaktor                'Schleife für x
        FOR y = -i TO i STEP intervallfaktor            'Schleife für y
            PSET (320 + x / intervallfaktor, 240 - y / intervallfaktor), (FNz(x, y) - minimum) * farbfaktor
            ' Punkt setzen (PointSET)
        NEXT y
    NEXT x
    LINE (0, 240)-(640, 240): LINE (320, 0)-(320, 480)      'Achsenkreuz zeichnen
    LINE (320 + 1 / intervallfaktor, 239)-(320 + 1 / intervallfaktor, 241)
    LINE (319, 240 - 1 / intervallfaktor)-(321, 240 - 1 / intervallfaktor)
    LOCATE 1, 41: PRINT "y": LOCATE 15, 79: PRINT "x"       'und beschriften
    LOCATE 2: PRINT "Maximum:"
    LOCATE 3: PRINT maximum
    LOCATE 4: PRINT "Minimum:"
    LOCATE 5: PRINT minimum
    Benutzereingabe$ = INPUT$(1)                        ' Benutzereingabe einlesen
WEND
SYSTEM  'Programm beenden

fehlerbehandlung: 'Fehlerbehandlungsroutine
IF ERR = 11 THEN PRINT "Der Graph hat möglicherweise eine Definitionslücke ('Division by zero')"
PRINT "Fehler: "; ERR'sonstiger Fehler
SYSTEM
```

What is Ray Tracing?

```
Ray Tracing, in a one-line description, is a method that allows
you to
create stunning photo-realistic images on a computer.  All you
need is
a computer, some ray tracing software, a little imagination and
some
patience.

The first stage of creating this masterpiece is to "describe" what
it
is that you want to depict in your picture.  You may do this using
an
interactive modelling system, like a CAD package, or by creating a
text
file that has a programming language-like syntax to describe the
elements.  Either way, you will be specifying what objects are in
your
imaginary world, what shape they are, where they are, what colour
and
texture they have and where the light sources are to illuminate
them.
Having done all of this, you feed it into your ray tracer, sit
back and
wait.

And wait...

That's the main drawback of ray tracing - it's not fast.  The
software
actually mathematically models the light rays as they bounce
around
this virtual world, reflecting, refracting and generally having a
good
time until they end up in the lense of your imaginary camera.
This can
quite literally involve thousands and millions of floating-point
calculations and this takes time.  Tracing images can take
anything
from a few minutes to many days.  It's a long process, I know, but
the
results can make it all worth while.

Ray tracing isn't the only method for creating photo-realistic
pictures.  There are packages like 3D Studio which uses scanline
rendering, Radiance, which uses radiosity, and so on.  Although
these
don't count as ray tracing, the methods you use from one system to
the
next are often sufficiently similar to warrant their discussion in
this
group.  So if you think it's relevant, feel free to bring it up.
These
systems will be mentioned in a little more detail later on.
```

Parent document is "Ray Tracing FAQ"
Previous document is "Table of Contents"
Next document is "1 - Ray Tracing Software"

Beigelegte CD-ROM

Die CD enthält unter Anderem alle unbearbeiteten Originalausgaben der Programme sowie alle in der Facharbeit erwähnten eigenen Programme.