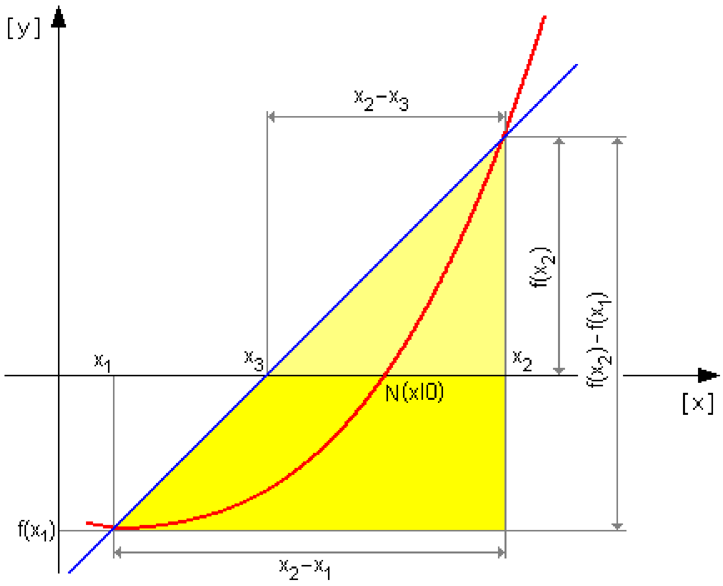
BspNr: J0010

Themenbereich	
Näherungsverfahren zur Nullstellenberechnung (Regula falsi)	
Ziele	**vorhandene Ausarbeitungen**
• Probleme, die bei der näherungsweisen Nullstellenberechnung auftreten können, erkennen.	TI-92 (J0010a)
Analoge Aufgabenstellungen – Übungsbeispiele	J0011
Lehrplanbezug (Österreich):	7. Klasse
Quelle: Karl Weinstich	

Regula Falsi

Die folgende Abbildung beschreibt das Näherungsverfahren „regula falsi“ zur Berechnung von Nullstellen:



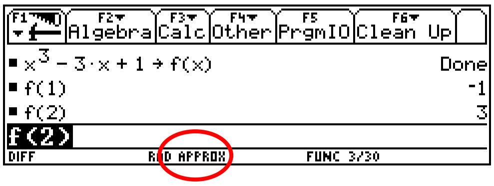
Angabe und Fragen:

a) Beschreibe die prinzipielle Vorgangsweise dieser Methode zur Nullstellenberechnung. Welche Voraussetzungen muss die Funktion erfüllen, damit diese Methode angewendet werden kann?

b) Leite eine Formel her, aus der sich das x_3 berechnen lässt, wenn man $f(x)$, x_1 und x_2 kennt.

c) Die folgende Tabelle zeigt die Berechnung einer Nullstelle der Funktion $f(x) = x^3 - 3x + 1$ mit Hilfe der Regula falsi auf 2 Dezimalen genau. Ermittle die anderen Nullstellen auf analoge Weise.
(Bemerkung: Um das Ergebnis auf n Stellen genau anzugeben, ist so lange zu rechnen, bis sich die n+1-te Dezimalstelle nicht mehr ändert; dann ist auf n Dezimalstellen zu runden)

Suche nach einem Intervall in dem ein Vorzeichenwechsel stattfindet:



⇒ es gibt eine Nullstelle zwischen 1 und 2

n	x1	x2	f(x1)	f(x2)	x3	f(x3)
1	1,000	2	-1,000	3	1,250	-0,7969
2	1,250	2	-0,797	3	1,407	-0,4344
3	1,407	2	-0,434	3	1,482	-0,1897
4	1,482	2	-0,190	3	1,513	-0,0749
5	1,513	2	-0,075	3	1,525	-0,0284
6	1,525	2	-0,028	3	1,529	-0,0106
7	1,529	2	-0,011	3	1,531	-0,0039
8	1,531	2	-0,004	3	1,532	-0,0015
9	1,532	2	-0,001	3	1,532	-0,0005

⇒ Nullstelle $N(1{,}53 | 0)$

d) Erstelle ein Ablaufdiagramm (bzw. eine verbale Beschreibung), das die Berechnung einer Nullstelle mit Hilfe der Regula falsi beschreibt, wobei die Krümmung der Funktion zu berücksichtigen ist.

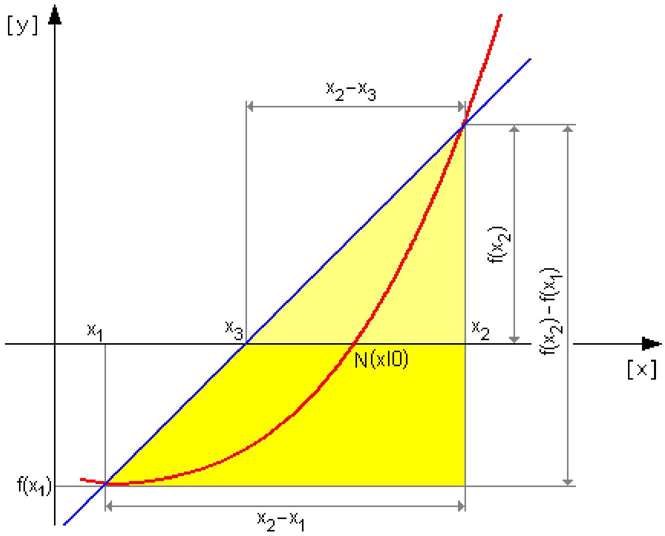
e) Erstelle ein Programm für den TI-92, das die Nullstellen einer Funktion nach der Regula falsi berechnet. Dem Programm sollen die Intervallgrenzen innerhalb welcher sich die Nullstelle befindet und die gewünschte Genauigkeit übergeben werden können. Die Funktion soll nicht dem Programm übergeben werden, sondern vor dem Start des Programms auf die Variable f gespeichert werden.

f) Berechne mit Hilfe deines Programms die Nullstellen der Funktionen
$f_1(x) = x^3 - 3x + 1$, $f_2(x) = x^{11} + 4$ und $f_3(x) = \dfrac{8x^3 - 10x^2 + 6x - 1}{1000}$.
Lass dir dabei die Näherungswerte für die Nullstelle in jedem Schleifendurchlauf anzeigen. Was passiert, wenn man Startwerte verwendet, die von der Nullstelle weiter entfernt liegen? Wurde von deinem Programm das Ergebnis immer mit der gewünschten Genauigkeit berechnet?

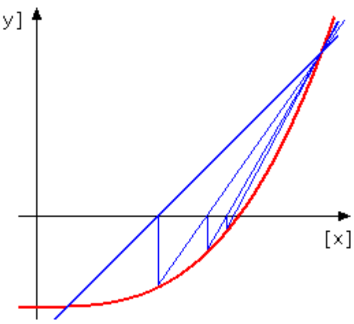
BspNr: J0010a

Ausarbeitung (System: TI-92)

ad a)



Bei diesem Näherungsverfahren zur Nullstellenberechnung wird die Funktion in unmittelbarer Nähe der Nullstelle durch eine Sekante, die durch die Punkte $(x_1 \mid f(x_1))$ und $(x_2 \mid f(x_2))$ geht, ersetzt. Der Schnittpunkt x_3 der Sekante mit der x-Achse liegt immer näher an der gesuchten Nullstelle als die ursprünglichen Werte x_1 und x_2.
Wendet man dieses Verfahren fortwährend an, erhält man eine immer bessere Annäherung der Nullstelle, wie folgende Graphik illustriert:



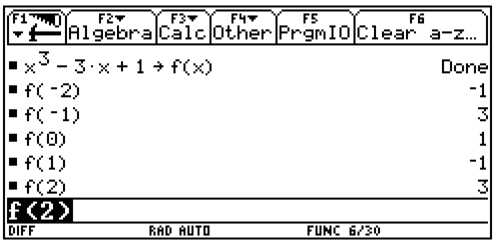
$x_3 = x_1 - \dfrac{f(x_1)(x_2 - x_1)}{f(x_2) - f(x_1)}$ bzw. $x_3 = x_2 - \dfrac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$

ad b)

Aus der Zeichnung erkennt man nach dem Strahlensatz: $(x_2 - x_3):(x_2 - x_1) = f(x_2):[f(x_2) - f(x_1)]$ woraus sich die zweite der oben genannten Näherungsformeln ergibt. Für die andere Formel erfolgt die Herleitung analog. Damit dieses Verfahren Erfolg hat, muss die Funktion auf dem Intervall $[x_1, x_2]$ stetig sein.

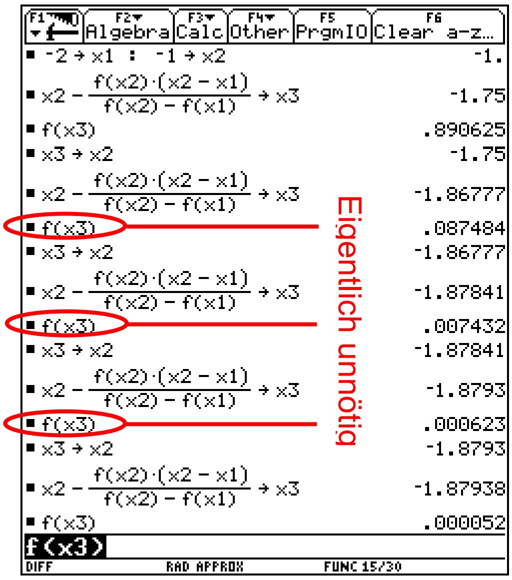
ad c)

Funktionswerte berechnen



⇒ die Nullstellen liegen in folgenden Intervallen:
(-2;-1); (0;1); (1;2)

Nullstellen berechnen



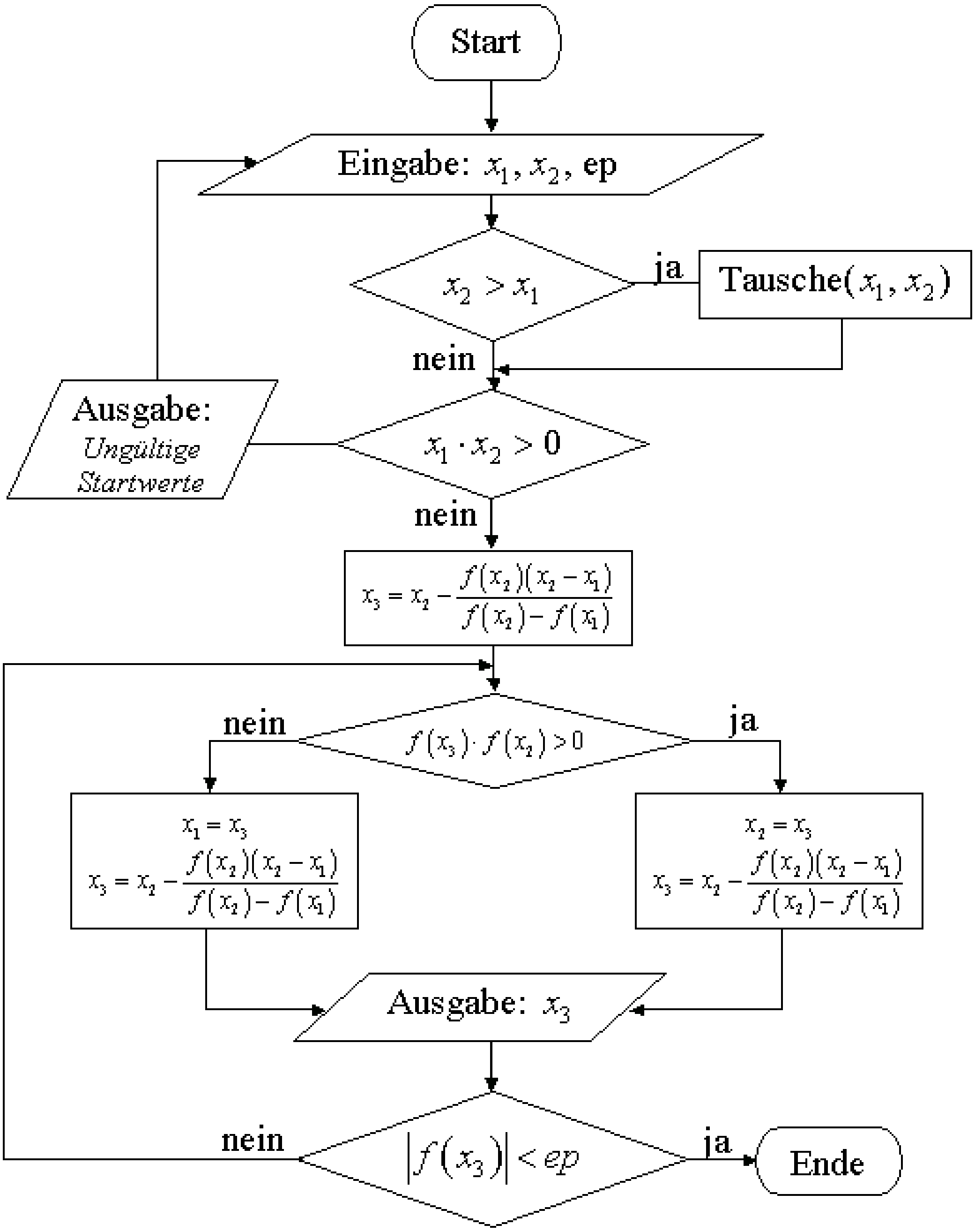
$\Rightarrow N(-1,88 \mid 0)$

Für die andere Nullstelle analog.

n	x1	x2	f(x1)	f(x2)	x3	f(x3)
1	-2,000	-1	-1,000	3	-1,750	0,8906
2	-2,000	-1,750	-1,000	0,89	-1,868	0,0875
3	-2,000	-1,868	-1,000	0,09	-1,878	0,0074
4	-2,000	-1,878	-1,000	0,01	-1,879	0,0006
5	-2,000	-1,879	-1,000	0	-1,879	0,0001

n	x1	x2	f(x1)	f(x2)	x3	f(x3)
1	0,000	1	1,000	-1	0,5000	-0,3750
2	0,000	0,5000	1,000	-0,38	0,3636	-0,0428
3	0,000	0,3636	1,000	-0,04	0,3487	-0,0037
4	0,000	0,3487	1,000	-0	0,3474	-0,0003
5	0,000	0,3474	1,000	-0	0,3473	0,0000
6	0,000	0,3473	1,000	-0	0,3473	0,0000

ad d)

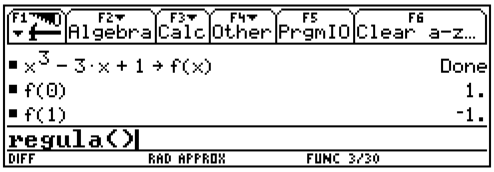


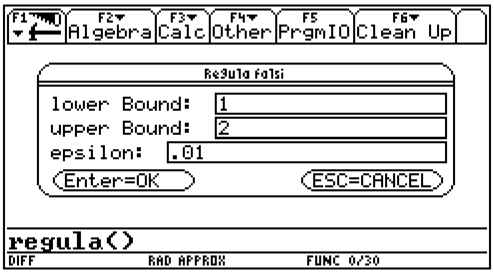
Start
Eingabe: x_1, x_2, ep
$x_2 > x_1$
ja
Tausche(x_1, x_2)
nein
Ausgabe:
Ungültige Startwerte
$x_1 \cdot x_2 > 0$
nein
$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$
nein
$f(x_3) \cdot f(x_2) > 0$
ja
$x_1 = x_3$
$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$
$x_2 = x_3$
$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$
Ausgabe: x_3
nein
$|f(x_3)| < ep$
ja
Ende

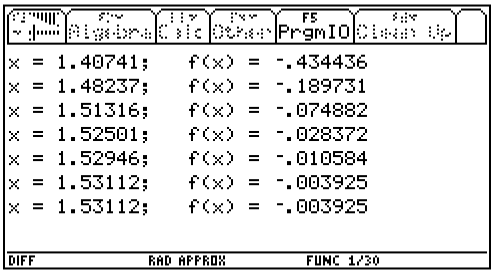
ad e)

```
:regula()
:Prgm
:Local x1,x2,x3,hlp
:Loop
:ClrIO
:  Dialog
:    Title "Regula falsi"
:    Request "lower Bound",x1
:    Request "upper Bound",x2
:    Request "epsilon",ep
:  EndDlog
:  expr(x1)→x1:expr(x2)→x2:expr(ep)→ep
:  If x2<x1 Then
:    x1→hlp:x2→x1:hlp→x2
:  EndIf
:  If sign(f(x1))=sign(f(x2)) Then
:    Disp "Ungültige Startwerte"
:    Pause
:  Else
:    Exit
:  EndIf
:EndLoop
:x2-f(x2)*(x2-x1)/(f(x2)-f(x1))→x3
:Loop
:  If f(x3)*f(x2)>0 Then
:    x3→x2
:  Else
:    x3→x1
:  EndIf
:  x2-f(x2)*(x2-x1)/(f(x2)-f(x1))→x3
:  Disp "x = "&string(x3)&";    f(x) = "&string(f(x3))
:  If abs(f(x3))<ep
:    Exit
:EndLoop
:
:Disp "x = "&string(x3)&";    f(x) = "&string(f(x3))
:EndPrgm
```

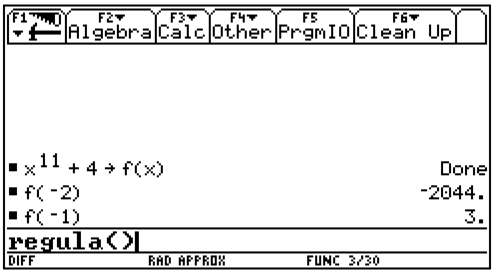
Vor dem Ausführen des Programms muss die Funktionsgleichung auf die Variable *f* gespeichert werden. Weiters müssen geeignete Startwerte ermittelt werden:

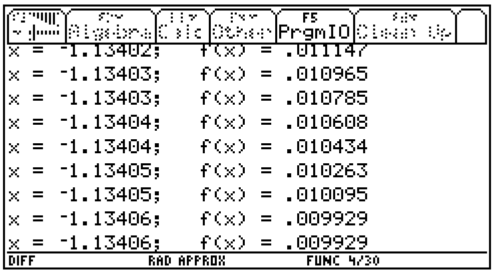






ad f)
Der Vorteil dieses Verfahrens liegt darin, dass das Verfahren immer konvergiert solange die Funktion stetig ist. Leider konvergiert das Verfahren bei ungünstigen Startwerten nur sehr langsam, wie folgendes Beispiel zeigt:





Außerdem ist die Formulierung der Abbruchbedingung für die gewünschte Genauigkeit nicht ganz einfach zu formulieren. Für die im obigen Programm verwendete Abbruchbedingung ***if abs(f(x3)<ep)*** kann es zu fehlerhaften Berechnungen kommen, wie das folgende Beispiel zeigt: