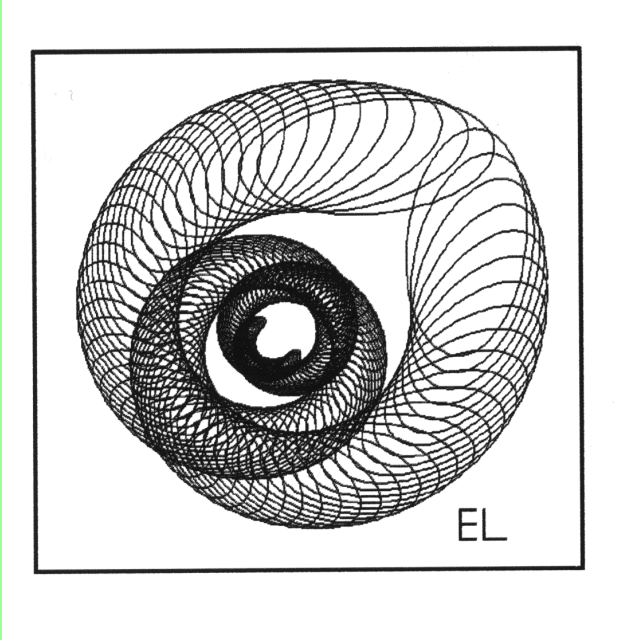
Visualisieren einiger elementarer Informatik-Algorithmen

mit Hilfe von Computeralgebrasystemen (CAS)

Königstein /Sächsische Schweiz - 2005

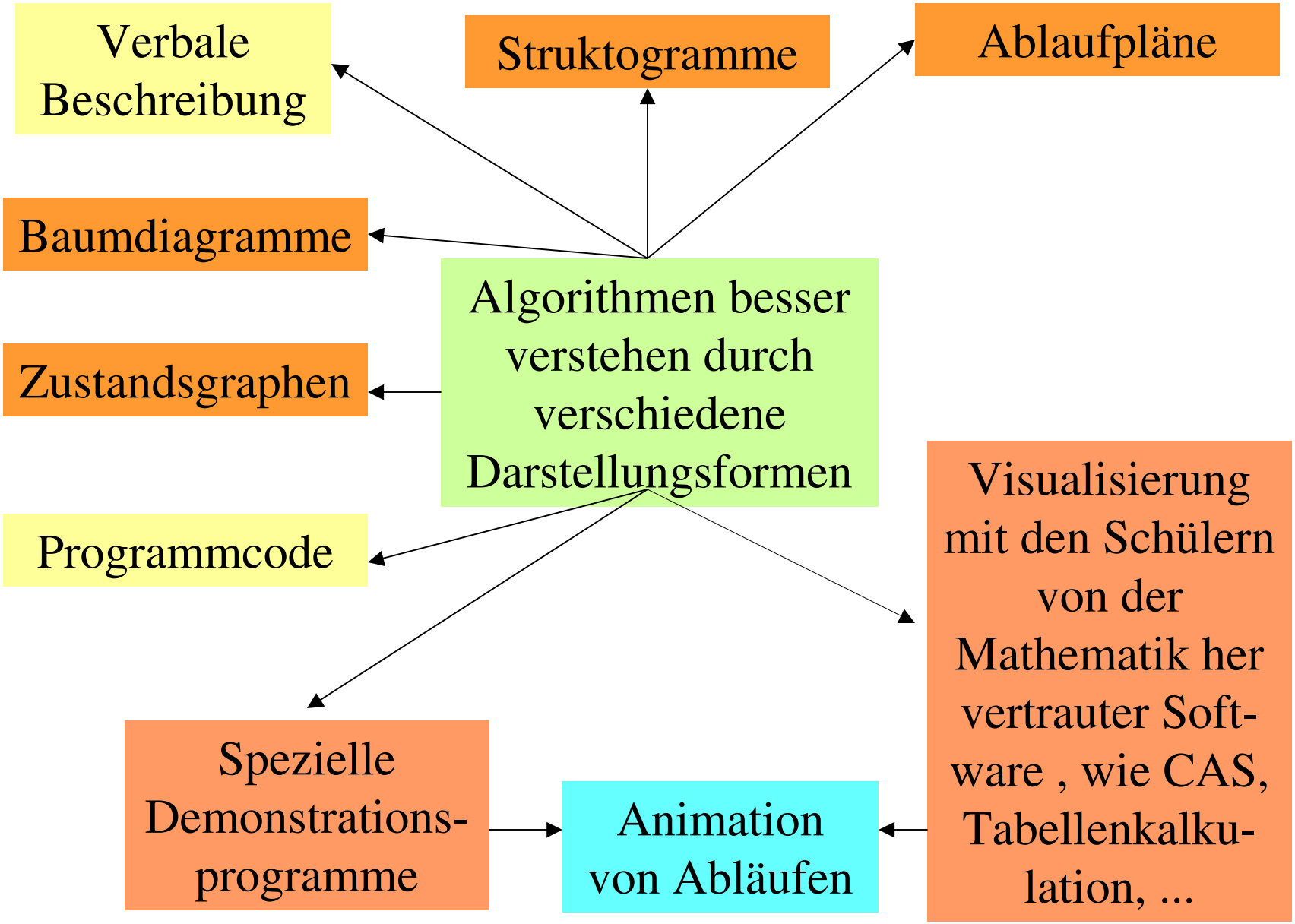


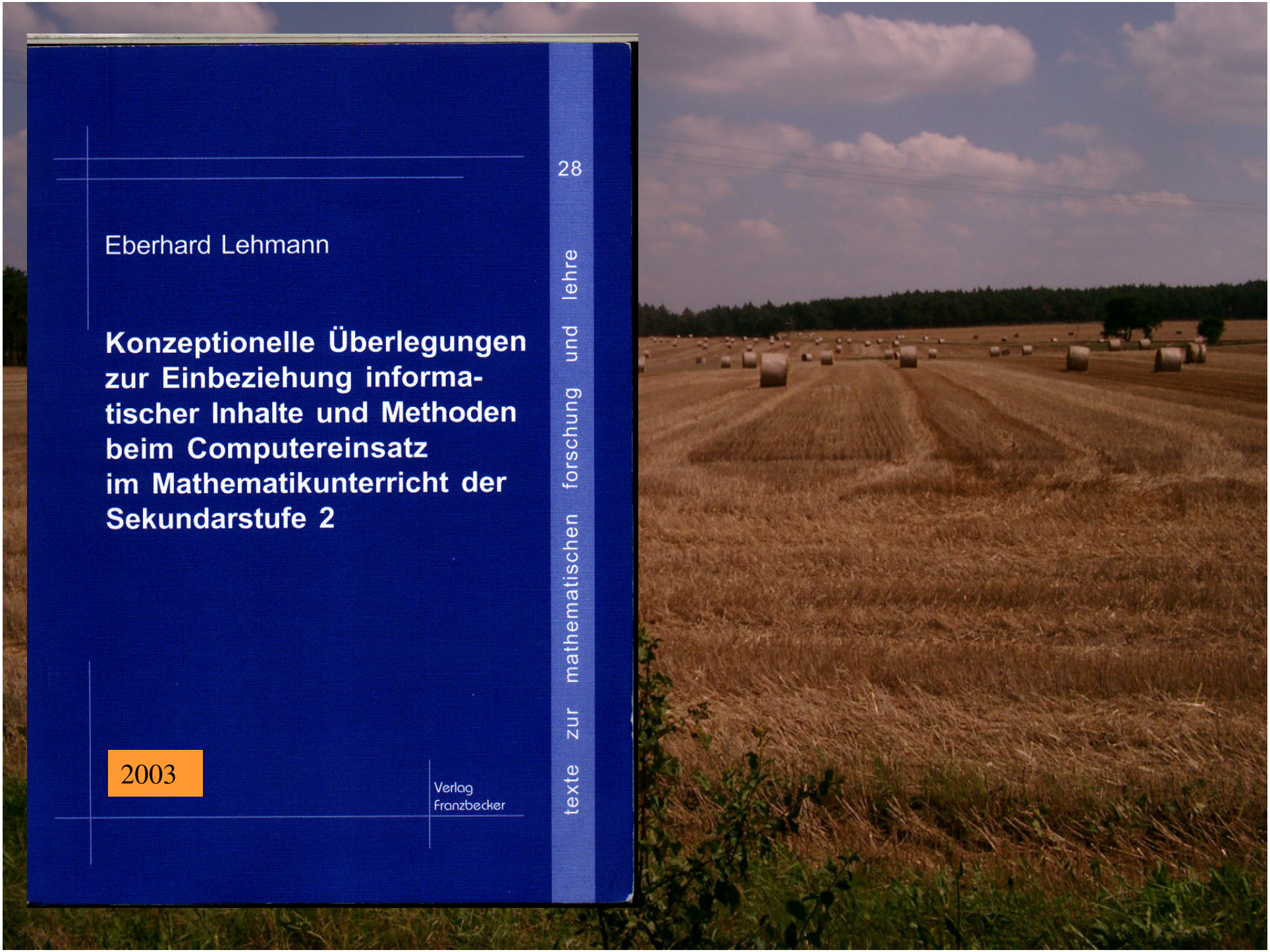
Lehmanns-Logo

Plot2-oetz2.exe

Dr. Eberhard Lehmann, Berlin

mirza@snafu.de --- www.snafu.de/~mirza


Verbale Beschreibung
Struktogramme
Ablaufpläne
Baumdiagramme
Algorithmen besser verstehen durch verschiedene Darstellungsformen
Zustandsgraphen
Programmcode
Visualisierung mit den Schülern von der Mathematik her vertrauter Soft-ware , wie CAS, Tabellenkalku-lation, ...
Spezielle Demonstrations-programme
Animation von Abläufen

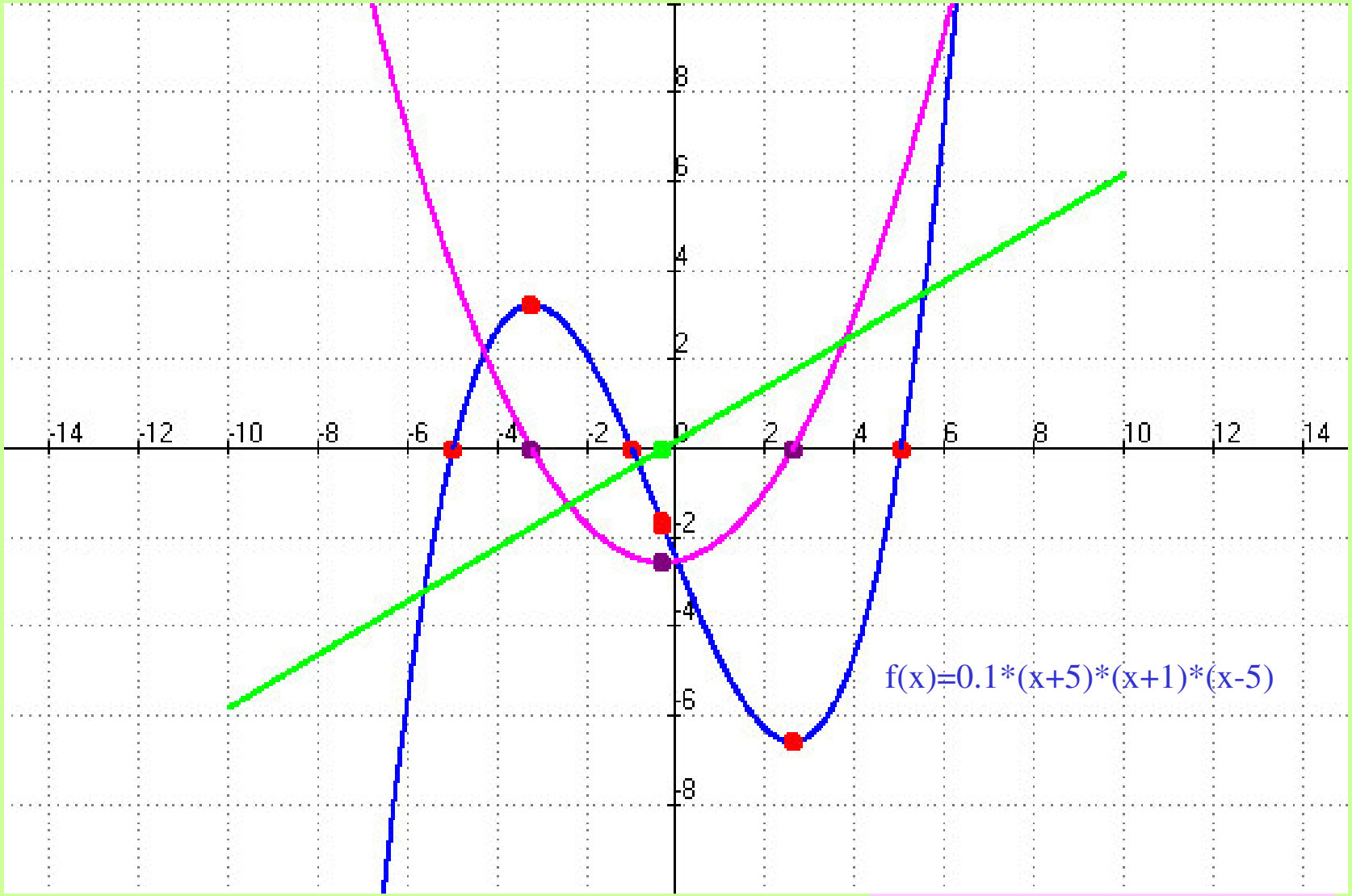

Eberhard Lehmann
Konzeptionelle Überlegungen zur Einbeziehung informatischer Inhalte und Methoden beim Computereinsatz im Mathematikunterricht der Sekundarstufe 2
2003
Verlag Franzbecker
texte zur mathematischen forschung und lehre 28

Vorhandene Beispiele

- Suchen in einer Datenmenge (in einer Wertetabelle)
- **Automatische Kurvendiskussion (Nullstellen suchen)**
- **Bresenham Algorithmus (Geraden zeichnen)**
- **Transformation von Zufallszahlen (für Simulationen)**
- **Güte von Zufallszahlen-Generatoren**
- **Das 3a+1-Problem (auf dem Weg zum Halteproblem)**
- Beasy Beaver für 3 Zustände (Animation)
- Zählen (Visualisierung s:=s+1)

Software: ANIMATO

Nullstellen suchen: Automatische Kurvendiskussion



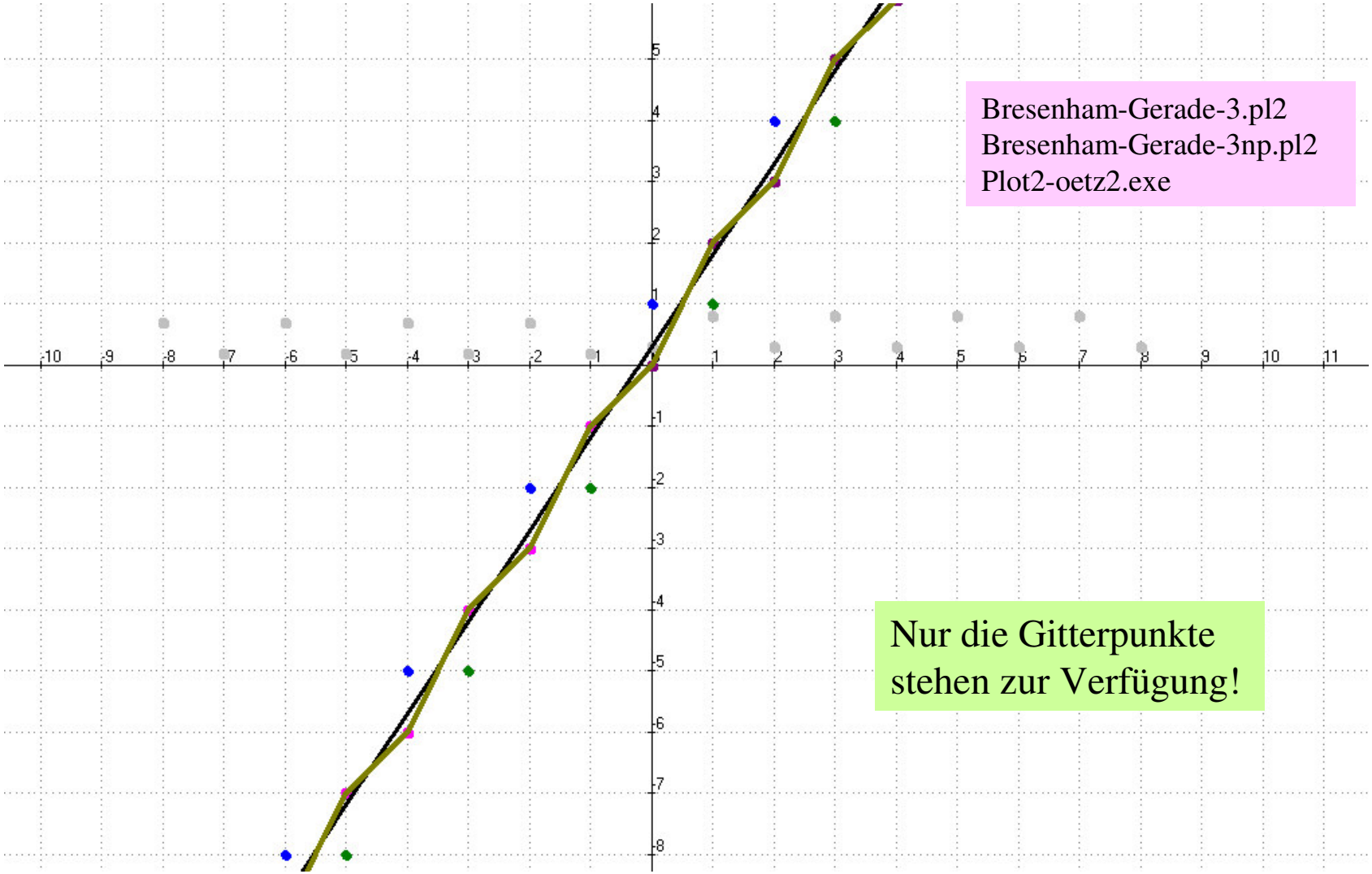
Inf-Suchen-Kurvendiskussion1.pl2
Plot2-oetz2.exe

f1: 0.1*(a+5)*(a+1)*(a-5) **// Ausgangsfunktion**
f2: {abs(f1(a))<0.01:a} // Nullstellen suchen
f3: f2(a),f1(a) // Nullstellen markieren

f5: (f1(a+0.001)-f1(a))/0.001 **// 1.Ableitung (angenähert!)**
f6: {abs(f5(a))<0.01:a} // Nullstellen von f5 suchen
(Extremwerte von f1)
f7: f6(a),f5(a) // Nullstellen von f1´= f5 markieren
f8: f6(a),f1(a) // Extremwerte von f1 markieren
f9: x,0
f10: (f5(a+0.001)-f5(a))/0.001 **// 2.Ableitung (angenähert!)**
f11: {abs(f10(a))<0.01:a} // Nullstellen von f10 suchen
f12: f11(a),f10(a) // Nullstellen von f ´´
f13: f11(a),f5(a) // Extremwerte von f ´
f14: f11(a),f1(a) // Wendepunkte von f

Bresenham-Algorithmus –

Wichtige Grundlage der Computergrafik



Bresenham-Algorithmus ---- Wir wollen die Geraden y = 1.5x+0.3 zeichnen lassen.

Die Auflösung des Bildschirms möge nur die gezeigten Gitterpunkt ermöglichen.

Hier nur für x>=0 und m>=0!

f1: 1.5x+0.3 // Geradengleichung in der Form y = m*x+n

f2: f1 // die exakte Gerade bei höherer Auflösung

f3: int(f1) // ganzzahlig, Nachkommastellen abgeschnitten, immer abgerundet

f4: int(f1+1) // ganzzahlig, immer aufgerundet

f5: {abs(f1-int(f1))<0.5:int(f1):int(f1+1)} // die Bresenham-Gerade

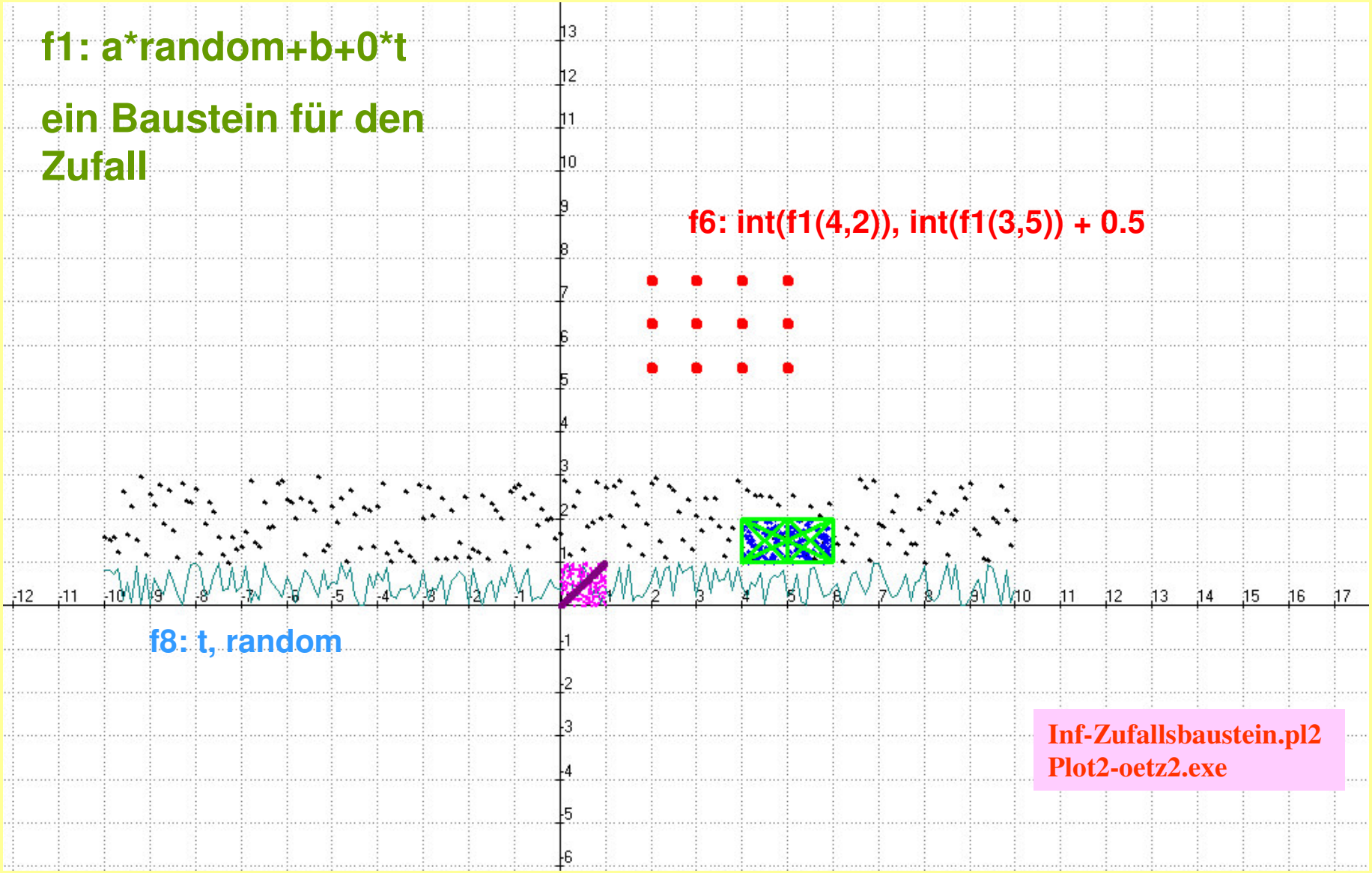
Für negative Steigungen funktioniert das soo nicht, Nachweis z.B. mit y = -1.5x+0.3. Warum nicht?

Und für negativen y-Abschnitt auch nicht. Warum nicht?

--> Algorithmus verbessern, mit Fallunterscheidungen

Und für eine Parabel? Zum Beispiel 0.5x^2+0.3.

Visualisierung der Transformation von Zufallszahlen



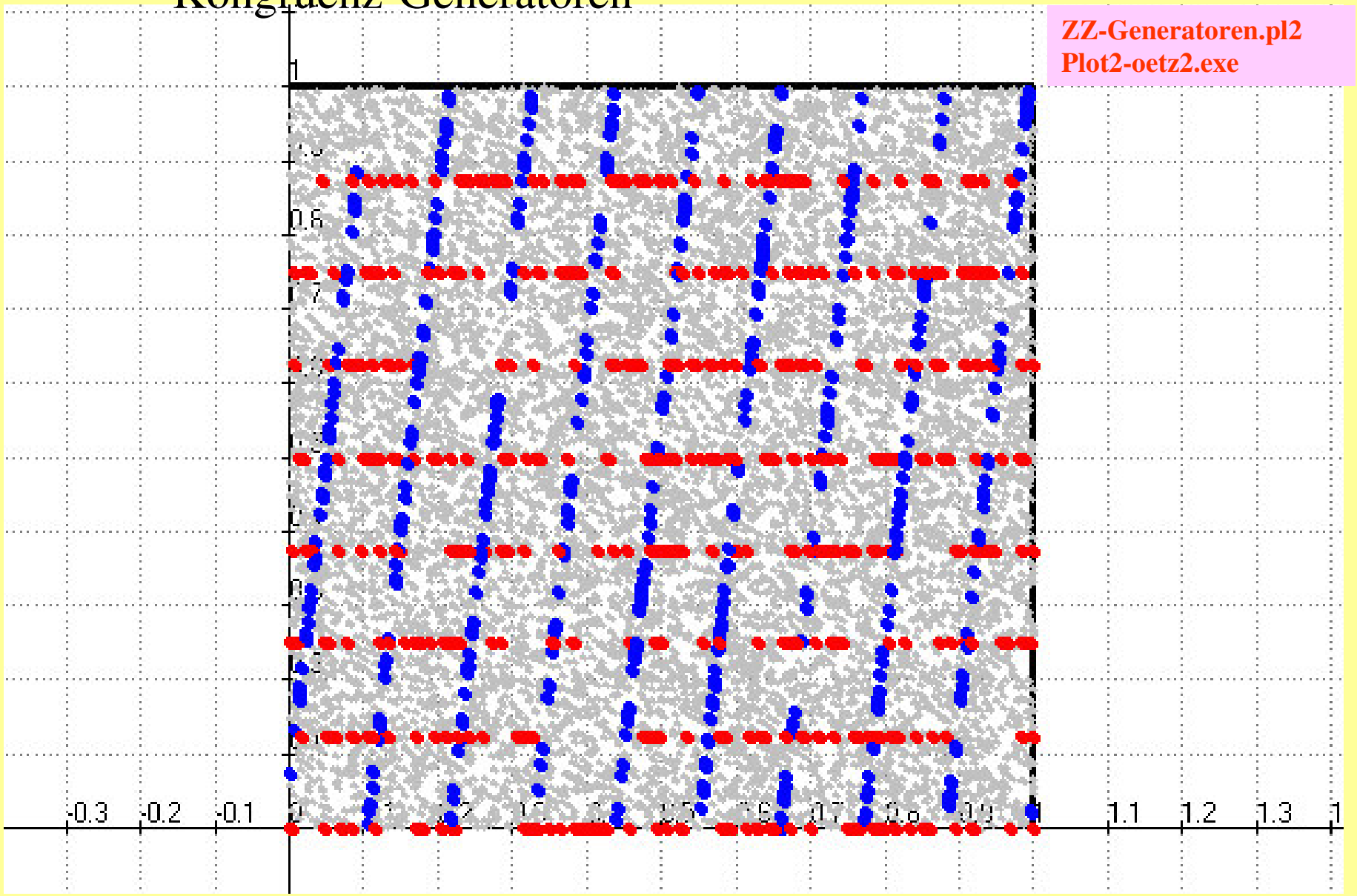
f1: a*random+b+0*t, ein Baustein für den Zufall

mit den Parametern a und b

Er wird benutzt zur Transformation von Zufallszahlen.

```
f2: f1(2,1)                      // entspricht 2*random+1, Werte zwischen 1 und 3
f3: f1(-2,6),f1(1,1)             // x-Bereich zwischen 4 und 6, y-Bereich zwischen 1 und 2
f4: f1(1,0), f1(0,random)        // Werte im Einheitsquadrat
f5: f1(1,0),f1(1,0)              // Winkelhalbierende, random liefert innerhalb einer
                                    Funktion, hier f5, den gleichen Wert
f6: int(f1(4,2)),int(f1(3,5))+0.5  // x aus {2,3,4,5}, y aus {5.5,6.5,7.5}
f7: int(f1(3,4)),int(f1(2,1))    // x-Bereich aus {4,5,6}, y-Bereich aus {1,2}
f8: t, random                    // Zufallszahlen zwischen 0 und 1
```

Über die Güte von Zufallszahlen-Generatoren – hier Kongruenz-Generatoren



Kongruenz-Zufallsgeneratoren der Form z(n) := b*z(n-1)+c (mod.d),

(n spielt bei ANIMATO die Rolle von a)

f1 zeichnet das Einheitsquadrat

ZZ-Baustein

f2 {n=0:71124:frac((b*f3(n-1)+c)/d)} setzt einen willkürlich gewählten Anfangswert und berechnet dann die Zufallszahl aus dem Intervall [0,1[

f3 {n=0:f2(0,b,c,d):f2(n,b,c,d)*d} berechnet die Zufallszahl * Modul d

f4 f2(n-1,3^9,7,10^6+1),f2(n,3^9,7,10^6+1) ein Bausteinaufruf mit den Werten b=3^9, c = 7 und d = 10^6+1. Diese Werte liefern erfahrungsgemäß einen guten Zufallsgenerator mit langer Periode. z(n) = 3^9*z(n-1) + 7 (mod. 1000001), Gerade y=19683*x+7 (mod. 1000001)

f5, f6, f7 testen den Aufruf ohne Bausteinbenutzung

f9, f10 sind Bausteinaufrufe die einen schlechten Zufallsgenerator liefern

f9: f2(n-1,3^2,7,10^2+1),f2(n,3^2,7,10^2+1) // schlecht, Gerade y= 9*x+7 (mod.101)

f10: f2(n-1,3^2,7,10+1),f2(n,3^2,7,8) // schlecht, Gerade y= 9*x+7, mod.8

x,t,n läuft z.B. von 0 bis 10000

Informatik und Mathematik - Ein leicht verständlicher Algorithmus, der es in sich hat!

Visualisierung des 3a+1-Problems, eine Vorstufe zum Halteproblem (theoretische Informatik) - Die Visualisierung erfolgt graphisch oder mit der Wertetafel.

Das 3a+1-Problem, siehe A.Engel, Elementarmathematik vom algorithmischen Standpunkt Klett-Verlag, 19xx (Werte verglichen mit Buchwerten, sind ok)

Allgemeine Lösung

Startwert wählen, wenn Zahl gerade, dann zahl:=zahl / 2, sonst zahl:=3*zahl +1.

Der Algorithmus für ANIMATO:

f1: u Startwert

f2: {n = 1 : f1 : {(f2(n-1)/2 = int(f2(n-1)/2)) : f2(n-1)/2 : 3*f2(n-1)+1}}, Folgenwerte

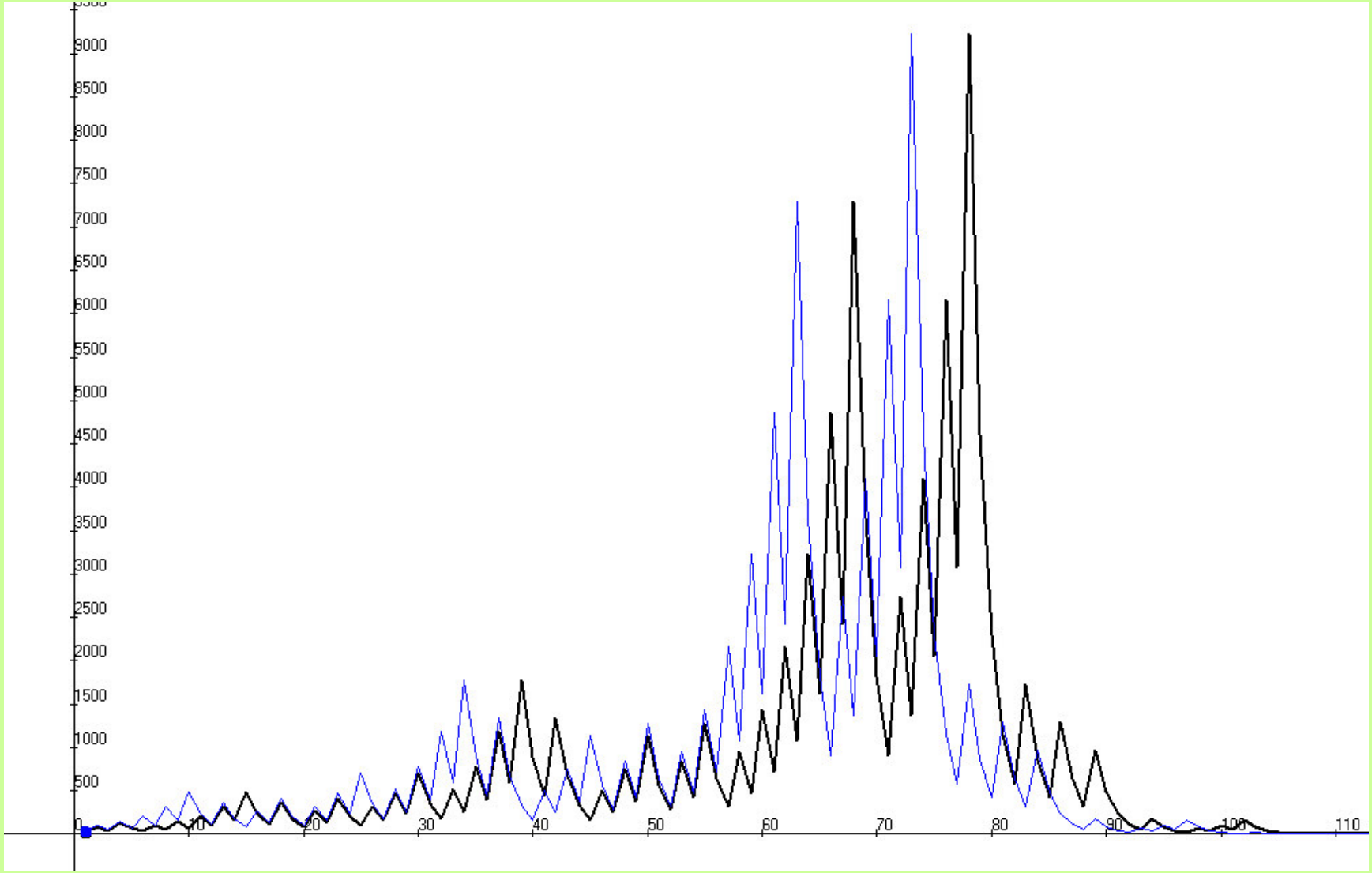
Zahlenbeispiel: 15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 20, 5, 16, 8, 4, 2, 1, 4, 2, 1

Beispiele: Startwerte sind 27 bzw. 31 - hier zum Vergleich der Laufzeit bis zum Zyklus.

f3: 27 f4: {n=1:f3:{(f4(n-1)/2 = int(f4(n-1)/2)) : f4(n-1)/2 : 3*f4(n-1)+1}}

f6: 31 f7: {n=1:f6:{(f7(n-1)/2 = int(f7(n-1)/2)) : f7(n-1)/2 : 3*f7(n-1)+1}}

Informatik und Mathematik: Auf dem Weg zum Halteproblem - das (3a+1)-Problem



Inf-3a+1-Halteproblem-Baustein.pl2

[PDF] Elementarer Lösungsansatz zum Collatz-**Ulam-Problem** CUP oder das " **...**
Dateiformat: PDF/Adobe Acrobat - HTML-Version
... November 2000 1 Elementarer Lösungsansatz zum Collatz-**Ulam-Problem** CUP oder das
"3a + 1"-Problem >>>>> Entwurf - Draft <<<<< Immo O. Kerner **...**
www.inf.hs-zigr.de/~wagenkn/CUP3AP1-neu.pdf - Ähnliche Seiten

www.mathematik.de | Pressestimmen
Pressestimmen, Das **Ulam-Problem**. 13.06.2004 Neue Zürcher Zeitung
www.mathematik.
de wurde bekannt, dass ein Artikel in der Neuen Zürcher **...**
www.mathematik.de/mde/presse/ pressestimmen/ulamproblem.html - 14k - Im Cache - Ähnliche Seiten

The 3x+1 problem and its generalizations - [Diese Seite übersetzen]
... The 3x+1 problem, also known as the Collatz problem, the Syracuse problem, Kakutani's
problem, Hasse's algorithm, and **Ulam's problem**, concerns the behavior of **...**
www.cecm.sfu.ca/organics/papers/lagarias/ - 4k - Im Cache - Ähnliche Seiten

Ulam's Problem -- from MathWorld - [Diese Seite übersetzen]
... U. **Ulam's Problem**. Collatz Problem. **...** Eric W. Weisstein. "**Ulam's Problem**." From MathWorld--A Wolfram Web Resource.
http://mathworld.wolfram.com/UlamsProblem.html. **...**
mathworld.wolfram.com/UlamsProblem.html - 15k - Im Cache - Ähnliche Seiten

Das Halteproblem fragt nach einem Algorithmus, mit dem man bei Programmen, Automaten oder Computern feststellen kann, ob sie für gewisse oder für alle Eingaben anhalten oder nicht.

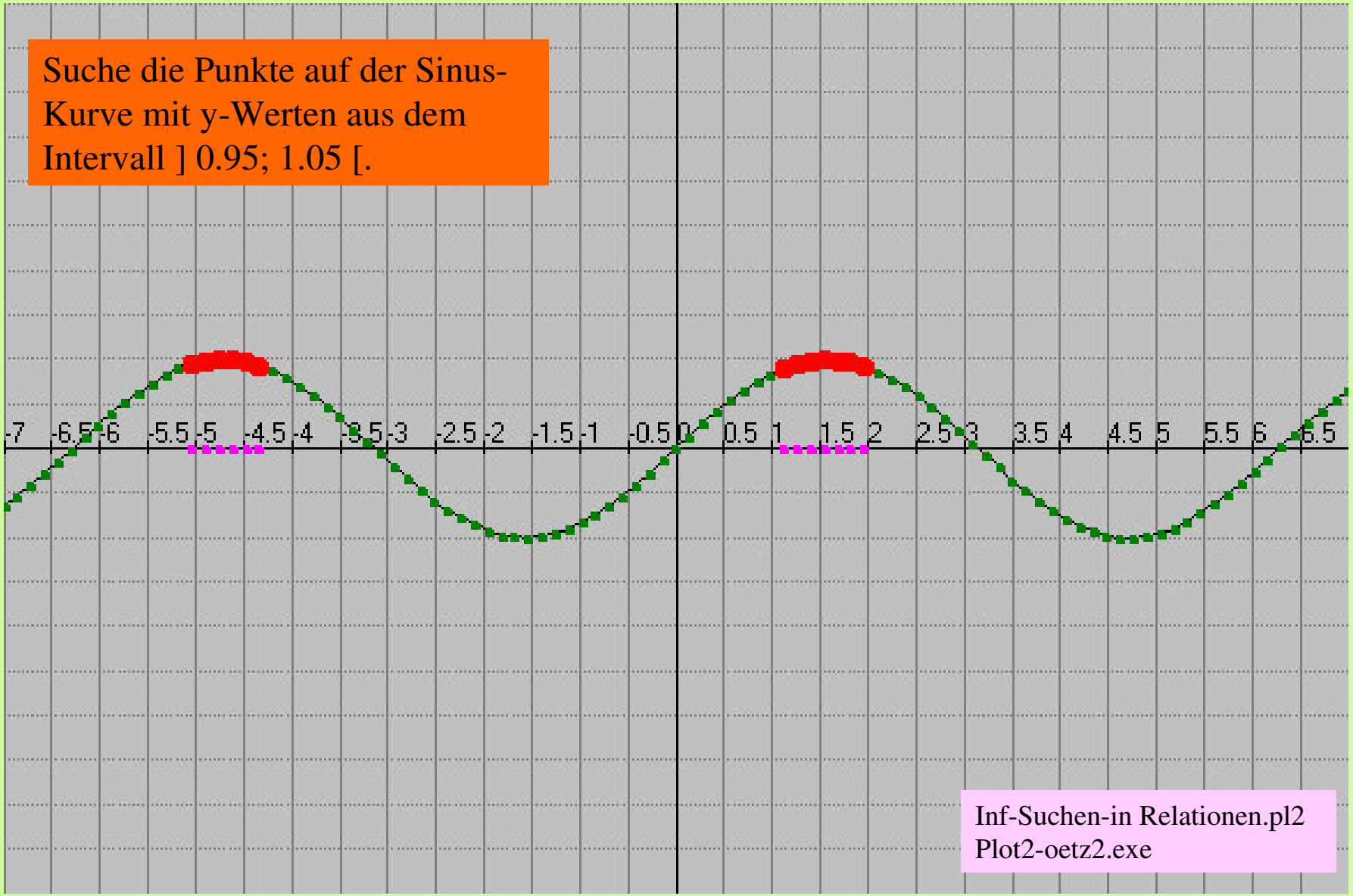
Unentscheidbarkeit des Halteproblems für Turingmaschinen:
Es gibt keinen Algorithmus, der für alle Turingmaschinen und alle Eingaben feststellt, ob die TM mit der entsprechenden Eingabe anhält oder nicht.

Es gibt keinen Algorithmus, der für jedes Programm feststellt, ob es eine Endlosschleife enthält oder nicht.

HALT

Suchen in einer Datenmenge - hier in einer Wertetabelle

Suche die Punkte auf der Sinus-Kurve mit y-Werten aus dem Intervall] 0.95; 1.05 [.



Inf-Suchen-in Relationen.pl2
Plot2-oetz2.exe

Informatik und Mathematik

Informatik: Lineares Suchen (hier in einer Tabelle nach einem y-Intervall)
Mathematik: Anwendung z.B. bei automatischer Kurvendiskussion, siehe Extraprogramm

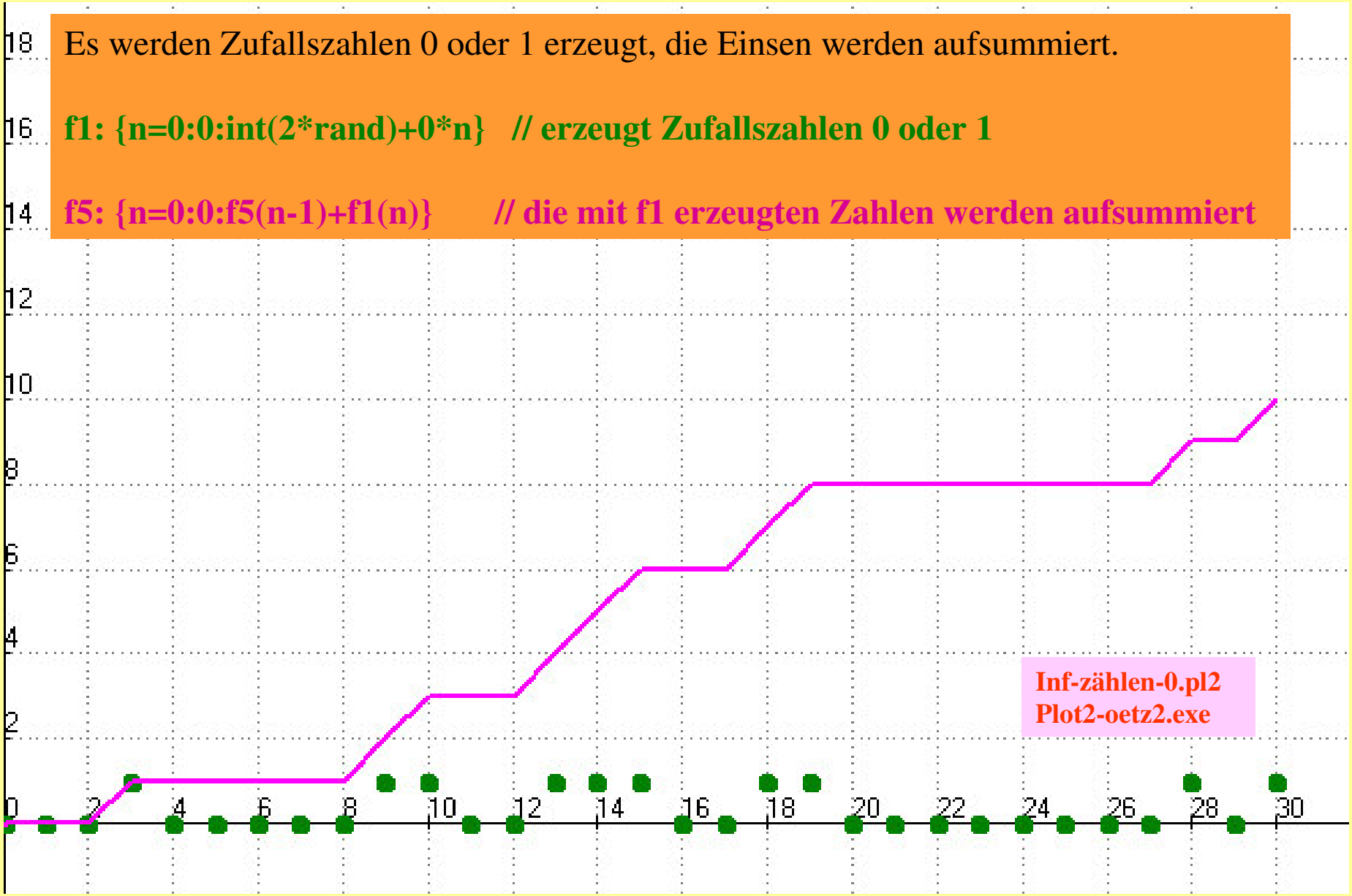
f1: sin(x) // zu untersuchende Relation, leicht zu variieren
f3: 0.95   // untere Grenze für die y-Werte, leicht zu variieren
f4: 1.05 // obere Grenze, leicht zu variieren
Weitere Variationsmöglichkeiten liegen in der Schrittweite, Berechnungsintervall usw.
f5: {f1<f4 & f1>f3:f1:undef}   // diese y-Werte erfüllen die Bedingung oder nicht
f7: {f1<f4 & f1>f3:0:undef}   // die dazugehörigen x-Werte

Informatik: Zählen - Visualisierung von s:=s+1

Es werden Zufallszahlen 0 oder 1 erzeugt, die Einsen werden aufsummiert.

f1: {n=0:0:int(2*rand)+0*n} // erzeugt Zufallszahlen 0 oder 1

f5: {n=0:0:f5(n-1)+f1(n)} // die mit f1 erzeugten Zahlen werden aufsummiert



Busy-Beaver und das Halteproblem

Die Fortsetzung des Ansatzes kann mit dem

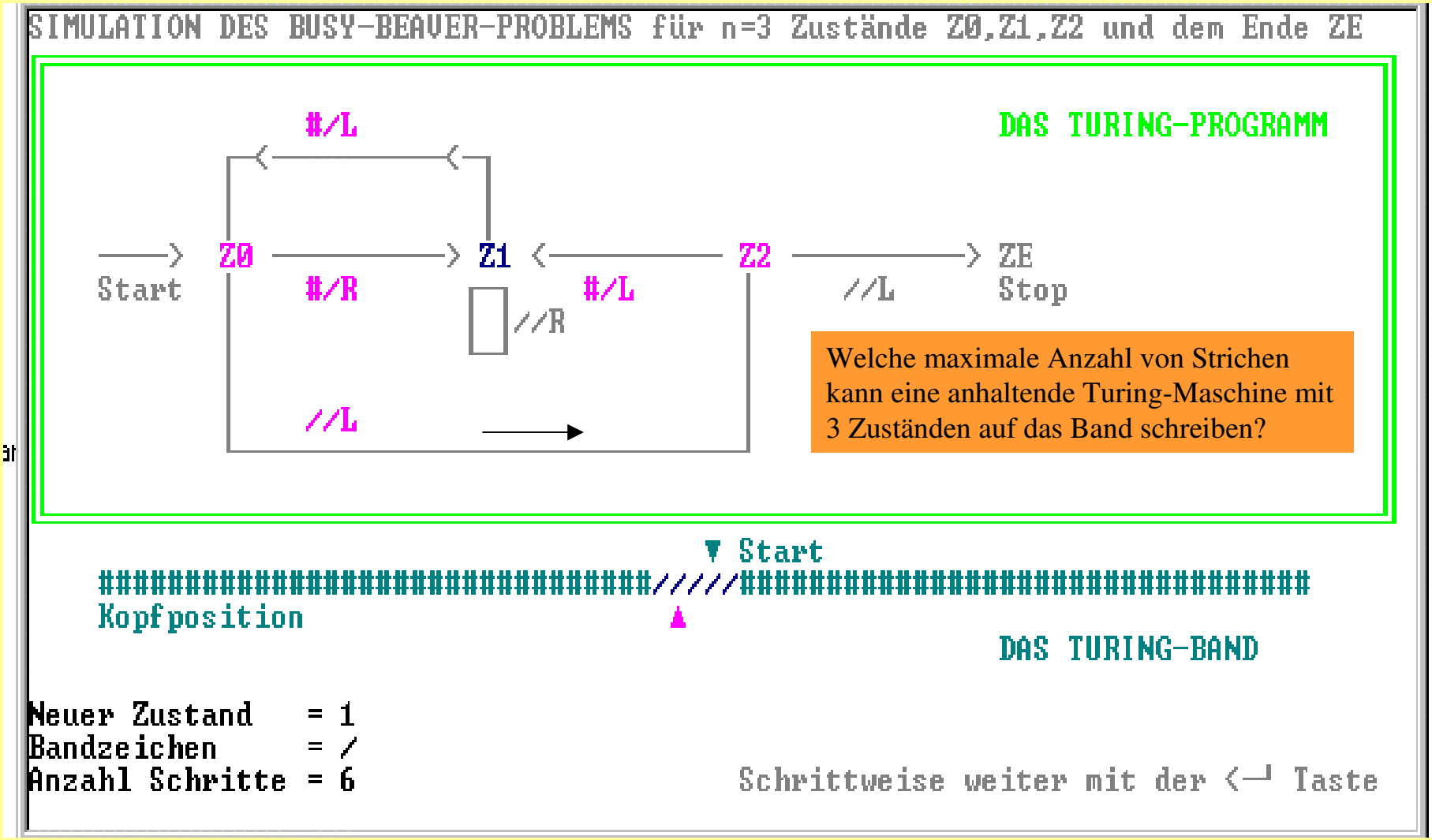
Busy-Beaver-Problem erfolgen.

Man nehme ein nach beiden Seiten **unendliches Band**, das **Bandalphabet** { #, / } und eine **Turingmaschine** mit n Zuständen Z0, Z1, Z2, Z(n-1) und einem zusätzlichen Haltezustand ZE.

In jedem Schritt soll die Turingmaschine eines der Symbole # oder / schreiben und eine Bewegung nach links oder rechts machen oder aber anhalten.

Die Turingmaschine mit n Zuständen, die auf dem anfangs leeren Band (anfangs lauter #-Zeichen) die meisten Striche schreibt, erhält den Titel **fleißiger Biber** (die Strichfolge darf Lücken enthalten).

Simulationsprogramm - Busy Beaver für 3 Zustände



Beaver3.exe

Anzahl der Zustände (ohne Endzustand) n	Anzahl der möglichen Turingtafeln $T(n)=(4(n+1))^{2n}$	Busy Beaver Anzahl der Striche $\sum(n)$	Busy Beaver Anzahl der Schritte S(n)
1	64	1	
2	20 736	4	
3	16 777 216	6	13
4	25 600 000 000	13	
5	63 403 380 965 376	$\geq$ 4098	
6	28^{12}		
7	32^{14}		
8	36^{16}		
k	berechenbare Funktion	nicht berechenbare Funktion	nicht berechenbare Funktion

Abb. 3.3.2-k

Die maximale Anzahl der Striche bei n Zuständen bis der Busy-Beaver ermittelt ist, ist eine nicht berechenbare Funktion.

Zum Fall n = 5 schreibt *A. K. Dewdney*

„Der Sprung von $\sum = 13$ bei n=4 auf $\sum \geq 4098$ bei n=5 ist symptomatisch für die nichtberechenbare Natur von $\sum$. Die Zahl 4098 hat eine interessante Geschichte hinter sich.

In der Augustausgabe 1984 des *Scientific American* erschien ein Artikel über den damals fleißigsten 5-Zustands-Biber, der 1984 von Uwe Schult, einem deutschen Informatiker, gefunden wurde. Schults fleißiger Biber erzeugt 501 Einsen, bevor er anhielt. Als Antwort auf den Artikel führte George Uhing, ein amerikanischer Programmierer, eine Computersuche nach fleißigen Bibern mit fünf Zuständen durch und fand einen, der 1915 Einsen [*Anm.: Striche*] ausgab, bevor er anhielt. Später, 1989, wurde Uhings Rekordbiber durch einen neuen, fleißigeren Biber verdrängt, der von Jürgen Buntrock und Heiner Marxen in Deutschland entdeckt wurde. Der Buntrock-Marxen-Biber, der bei einer dreitägigen Suche auf einem Hochgeschwindigkeitsrechner gefunden wurde, schreibt 4098 Einsen, bevor er anhält!" *[A. K. Dewdney: Der Turing-Omnibus – eine Reise durch die Informatik mit 66 Stationen, Springer-Verlag, Berlin-Heidelberg, 1995, S. 287]*