Fickel, Norman

Working Paper

Visualisierung der Volatilität bei der Interpolation von Zeitreihen: Excel-Makro Saffint

Diskussionspapiere // Friedrich-Alexander-Universität Erlangen-Nürnberg, Lehrstuhl für Statistik und Ökonometrie, No. 15/1996

Provided in cooperation with:
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)

Suggested citation: Fickel, Norman (1996) : Visualisierung der Volatilität bei der Interpolation von Zeitreihen: Excel-Makro Saffint, Diskussionspapiere // Friedrich-Alexander-Universität Erlangen-Nürnberg, Lehrstuhl für Statistik und Ökonometrie, No. 15/1996, http://hdl.handle.net/10419/29600

Visualisierung der Volatilität bei der Interpolation von Zeitreihen: Excel-Makro „Saffint“

Norman Fickel

Friedrich-Alexander-Universität Erlangen-Nürnberg
Wirtschafts- und Sozialwissenschaftliche Fakultät
Lehrstuhl für Statistik und empirische Wirtschaftsforschung
Lange Gasse 20
D-90403 Nürnberg

Abstract

Häufig werden in grafischen Darstellungen Werte einer Zeitreihe linear interpoliert, um einen groben Eindruck ihres Verlaufs zu geben. Das Ergebnis ist ein Streckenzug, in dem einzelne Geradenstücke aneinandergehängt sind, wobei die Ausgangsdaten des Streckenzugs die Knickpunkte bilden. Allerdings wird durch eine solche lineare Interpolation der visuelle Eindruck einer Zeitreihe mit großer Volatilität, wie etwa eines Börsenkurses, nur sehr unzureichend wiedergegeben, da auch zwischen den verfügbaren Kurswerten starke Schwankungen vorliegen. Es wäre daher ein Verfahren wünschenswert, das statt Geradenstücken Kurvenabschnitte verwendet, die rauh wirken. Dabei sollte der Grad der Rauheit parametrisierbar sein, so daß sowohl der Eindruck einer sehr starken Volatilität als auch einer schwachen erzeugbar ist.

In diesem Beitrag wird dafür Barnsleys selbstaffine (oder „fraktale“) Interpolation vorgeschlagen, die Ergebnis eines Iterationsprozesses ist: Gestartet wird mit dem Streckenzug der linearen Interpolation. Bei einem Iterationsschritt wird dann jeder Kurvenabschnitt durch ein affin gestauchtes Bild der gesamten Kurve des vorigen Schritts ersetzt. Die Iteration bricht ab, wenn sich in der Auflösung des Bildschirms nichts mehr ändert. Der Grad der vertikalen Stauchung ist dabei in Grenzen frei wählbar und liefert den gewünschten Volatilitätsparameter. Ist dieser im Extremfall gleich null, so wird der Strekkenzug vertikal exakt auf ein Geradenstück gestaucht, womit die selbstaffine Interpolation mit der linearen Interpolation übereinstimmt. Größere Werte des Parameters liefern eine rauh wirkende Interpolation und visualisieren damit einen bestimmten Grad von Volatilität.

Für die praktische Durchführung der selbstaffinen Interpolation existiert ein Algorithmus, der sich in wenigen Zeilen in BASIC als Excel-Makro programmieren ließ und der die direkte Berechnung einzelner Interpolationswerte erlaubt.

Inhalt

Inhalt der Begleitdiskette

DEMO.XLS	Excel 5.0-Arbeitsmappe mit dem Beispiel aus Anhang A
SAFFINT.TXT	BASIC-Quellcode des Excel-Makros „Saffint“ aus Anhang B

1. Einleitung

Manche Zeitreihen, etwa viele Aktienkurse, weisen einen ‘rauhen’ Graphen auf, d. h. die Kurven wirken gezackt. Dieser Eindruck bleibt auch bestehen, wenn man einen kürzeren Zeitraum betrachtet. In der Praxis kennt man häufig nur wenige, weit auseinanderliegende Werte der Zeitreihe. Deshalb stellt sich bei der grafischen Darstellung das Problem, aus ein paar wenigen Stützwerten Zwischenwerte so zu interpolieren, daß die Interpolationsfunktion im geeigneten Sinne rauh oder eben ‘volatil’ aussieht.

Sowohl mit linearen Interpolationsverfahren als auch mit Splineinterpolationen kann man diese Aufgabe nicht lösen, da sie nur weitgehend glatte Verläufe liefern, die für sehr kurze Zeiträume sogar linear oder annähernd linear sind. Der visuelle Eindruck der Volatilität geht im wesentlichen verloren, die rekonstruierte Zeitreihe sieht ganz anders als die ursprüngliche Reihe aus. Insbesondere läßt sich eine zusätzlich vorhandene Information über die vermutete Volatilität gar nicht grafisch darstellen.

In einem stochastischen Modell würde hier das Hinzufügen einer rauhen Komponente Abhilfe schaffen, die etwa aus einem weißen Rauschen bestehen kann. Dazu müßte zu der ‘glatten’ Interpolationsfunktion an jeder Stelle der Zeitachse eine zentrierte, normalverteilte Zufallsvariable addiert werden. Eine Volatilitätsinformation ließe sich dann durch geeignete Wahl der Varianz berücksichtigen. Für praktische Zwecke haben solche stochastischen Vorgehensweisen jedoch einige Nachteile: Zum einen hängt die Qualität der Darstellung vom verwendeten Zufallszahlengenerator ab, wobei verschiedene Startwerte des Zufallszahlengenerators jeweils andere Rekonstruktionen liefern. Zum anderen ist das durch den Zufallszahlengenerator festgelegte Konstruktionsschema an der Grafik kaum nachzuvollziehen.

Ein alternatives, rein deterministisches Vorgehen besteht darin, die geraden Verbindungsstücke einer linearen Interpolation durch rauhe Kurvenabschnitte zu ersetzen. Die Struktur der Rauheit wird dabei aus den gegebenen Interpolationsdaten in einem Iterationsprozeß erzeugt.

2. Beschreibung des Iterationsprozesses

Ausgegangen wird von der Größe nach angeordneten Stützstellen $t_0 < t_1 < ... < t_n$, beliebigen Interpolationswerten x_0, x_1, ..., x_n und einem Parameter α, welcher den Grad der Volatilität anzeigt. Der Iterationsprozeß startet mit dem Streckenzug der linearen Interpolation s, der an jeder Stützstelle t_i die Bedingung $s(t_i) = x_i$ erfüllt und auf allen Teilintervallen $[t_{i-1}, t_i]$ linear ist.

Im ersten Iterationsschritt wird eine Funktion f_1 auf dem Intervall $[t_0, t_n]$ dadurch gebildet, daß jeder Abschnitt des Streckenzug s durch ein passend horizontal und mit dem Faktor α vertikal gestauchtes Abbild von s ersetzt wird. Dies läßt sich in der Form

$$f_1(t) = \alpha\, s(b_i t + c_i) + d_i t + e_i \quad \text{für } t_{i-1} \le t < t_i \tag{1}$$

schreiben. Dabei sind b_i und c_i so gewählt, daß $b_i t_{i-1} + c_i = t_0$ und $b_i t_i + c_i = t_n$ gilt. Die Abbildung $t \mapsto b_i t + c_i$ streckt also das Teilintervall $[t_{i-1}, t_i]$ auf das Basisintervall $[t_0, t_n]$. Die Werte der Hilfsgrößen d_i und e_i ergeben sich für jeden Index i eindeutig daraus, daß auch die Funktion f_1 die gegebenen Werte interpolieren soll: $f_1(t_i) = x_i$. Man erhält nämlich für jedes Paar d_i, e_i gerade die zwei Bestimmungsgleichungen $x_{i-1} = \alpha\, x_0 + d_i t_{i-1} + e_i$ und $x_i = \alpha\, x_n + d_i t_i + e_i$.

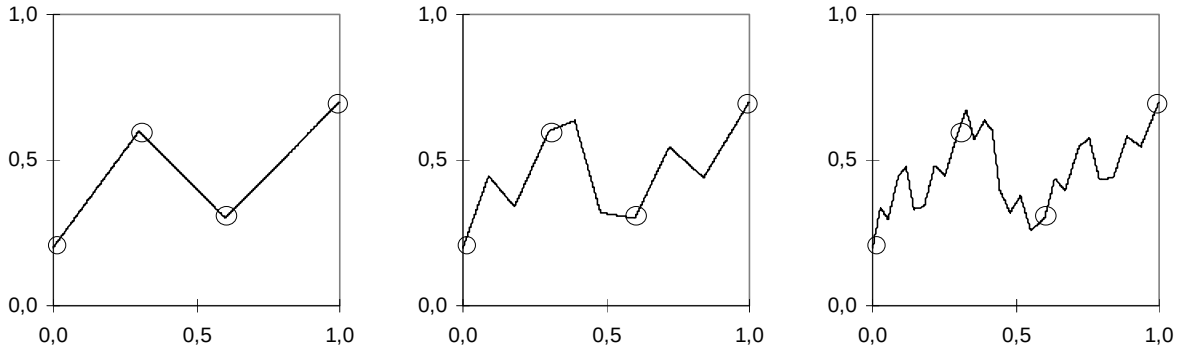


Abb. 1: Ein Beispiel für die ersten Iterationsschritte
bei einem Volatilitätsparameter 0,5

Nun kann eine Folge von Funktionen f_2, f_3, f_4, ... rekursiv definiert werden durch das Gleichungssystem

$$f_k(t) = \alpha\, f_{k-1}(b_i t + c_i) + d_i t + e_i \quad \text{für } t_{i-1} \le t \le t_i\,.$$

Die Funktion wird also abschnittsweise für alle Teilintervalle $[t_0, t_1]$, $[t_1, t_2]$, ..., $[t_{n-1}, t_n]$ und damit auf dem ganzen Basisintervall $[t_0, t_n]$ eindeutig definiert. Dabei gewährleistet die obige Festlegung der Hilfsgrößen gerade, daß auch an den Berührungspunkten zweier aufeinanderfolgender Teilintervalle die jeweiligen Bedingungen widerspruchsfrei sind.

In Abb. 1 sind als Beispiel für Stützstellen $t_0 = 0$; $t_1 = 0{,}3$; $t_2 = 0{,}6$; $t_3 = 1$ und Interpolationswerte $x_0 = 0{,}2$; $x_1 = 0{,}6$; $x_2 = 0{,}3$; $x_3 = 0{,}7$ bei einem Vertikalfaktor $\alpha = 0{,}5$ die Folgenglieder $f_0 = s$, f_1 und f_2 dargestellt.

Vorausgesetzt der Vertikalfaktor ist betragsmäßig kleiner als Eins, d. h. $|\alpha| < 1$, so ändert sich nach einigen Iterationsschritten die Interpolationsfunktion nicht mehr wahrnehmbar. Man erhält also ein stabiles Endergebnis. Ist der Vertikalfaktor dagegen im Absolutbetrag gleich oder sogar größer als Eins, dann werden außerhalb der Interpolationsstellen einige Werte immer größer, der Iterationsprozeß führt also zu keiner brauchbaren Grenzfunktion.

Außerdem ist die Volatilität, also der Grad der Rauheit, um so größer, je näher der Volatilitätsparamter α an 1 bzw. -1 liegt. Ist er null, so wird durch den Iterationsprozeß der ursprüngliche Streckenzug der linearen Interpolation immer nur wieder in sich selbst überführt, es ändert sich also nichts.

Für die Daten aus dem Beispiel von Abb. 1 sind die Interpolationsfunktionen für Parameterwerte von α = 0,2; 0,5 und 0,8 in Abb. 2 zu sehen. Dabei wurden, um die Stabilität des Prozesses zu erreichen, jeweils zehn Iterationsschritte durchgeführt.

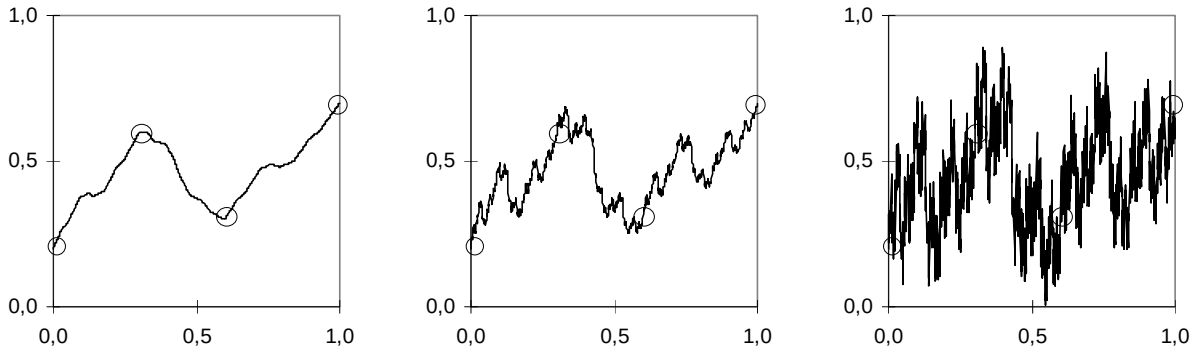


Abb. 2: Beispiele des Iterationsergebnisses
für verschiedene Werte des Volatilitätsparameters

Obige Beschreibung des Iterationsprozesses läßt sich zwar direkt als Computerprogramm implementieren, doch ist dies ziemlich umständlich, da in jedem Iterationsschritt die gesamte Funktion als Streckenzug im Hauptspeicher zu halten ist. Besonders unvorteilhaft ist dies, wenn gezielt ein Wert der Interpolationsfunktion bestimmt werden soll, da dazu zunächst die Berechnung für den gesamten Definitionsbereich erforderlich ist, aus dem dann erst der gesuchte Wert abgegriffen werden kann.

Die Grundlage für effizientere und einfacher zu programmierende Algorithmen wird durch eine mathematische Analyse der Eigenschaften des Iterationsergebnisses geschaffen, den sogenannten selbstaffinen Funktionen.

3. Mathematik der selbstaffinen Funktionen

Wenn sich der Iterationsprozeß stabilisiert hat, also $f_k(t) \approx f_{k-1}(t)$ gilt, so heißt das nach der rekursiven Festlegung der Folgenglieder gerade, daß sich für hinreichend große Indizes k die Näherung

$$f_k(t) \approx \alpha\ f_k(b_i t + c_i) + d_i t + e_i$$

machen läßt. Diese Funktionalgleichung besagt, daß sich der Verlauf von f_k auf jedem Teilintervall $[t_{i-1}, t_i]$ von dem auf dem Basisintervall $[t_0, t_n]$ nur durch eine lineare Komponente unterscheidet. Oder anders gesagt: Die Funktion geht durch eine affine Transformation in sich selbst über. Dies sei hier der Anlaß, Funktionen, für die solch eine Eigenschaft exakt gilt, besonders zu benennen:

Definition: Sei $n \in \mathbb{N}$ und $t_0, t_1, ..., t_n \in \mathbb{R}$ mit $t_0 < t_1 < ... < t_n$, sowie $\alpha \in \mathbb{R}$. Für jedes $i = 1, ..., n$ seien b_i und c_i so gewählt, daß $b_i t_{i-1} + c_i = t_0$ und $b_i t_i + c_i = t_n$ gelten. Dann heißt eine Funktion f: $[t_0, t_n] \to \mathbb{R}$ *selbstaffin mit Stützstellen $t_0, t_1, ..., t_n$ und Vertikalfaktor* α, wenn auf jedem Teilintervall $[t_{i-1}, t_i]$ die Abbildung

$$t \mapsto f(t) - \alpha\ f(b_i t + c_i) \qquad (2)$$

linear ist. Das heißt, falls es Zahlen $\hat{d}_i$ und $\hat{e}_i$ gibt mit

$$f(t) - \alpha\ f(b_i t + c_i) = \hat{d}_i t + \hat{e}_i \quad \text{für } t_{i-1} \le t \le t_i .$$

Offensichtlich sind insbesondere alle linearen Funktionen selbstaffin, da in der Zuordnung (2) rechts eine Verkettung linearer Abbildungen steht, die ja stets wieder eine lineare Abbildung liefert. Außerdem ist jede Summe und jedes skalare Vielfache selbstaffiner Funktionen – zu festen Stützstellen und einem festen Vertikalfaktor – wie-

der selbstaffin. Die selbstaffinen Funktionen bilden also einen Vektorraum, was ihre funktionalanalytische Betrachtung vereinfacht (vgl. Fickel 1988).

Eine selbstaffine Funktion ist nach ihrer Definition die Lösung eines Funktionalgleichungssystems, für das Barnsley (1986) einen Existenz- und Eindeutigkeitssatz gefunden hat:

Satz 1: Seien $t_0 < t_1 < ... < t_n$ reelle Zahlen, seien weiter $x_0, x_1, ..., x_n \in \mathbb{R}$ und $|\alpha| < 1$. Dann existiert genau eine beschränkte Funktion f: $[t_0, t_n] \rightarrow \mathbb{R}$ mit

1. f ist selbstaffin mit Stützstellen $t_0, t_1, ..., t_n$ und Vertikalfaktor α,
2. $f(t_i) = x_i$ für $i = 0, ..., n$.

Zusätzlich ist diese Funktion f stetig.

Beweisskizze: Der Beweis ist eine direkte Anwendung des Fixpunktsatzes von Banach (vgl. auch Hutchinson 1981: 730 f.). Als zugrundeliegender Funktionenraum wird dazu die Menge aller beschränkten Funktionen auf dem Intervall $[t_0, t_n]$ gewählt. Darauf ist ein Operator T definiert, der eine Funktion g dieses Raumes wieder in eine beschränkte Funktion Tg abbildet. Die Zuordungsvorschrift $g \mapsto Tg$ des Operators besteht dabei gerade aus der abschnittsweisen Definition

$$Tg(t) = \alpha\, g(b_i t + c_i) + d_i t + e_i \quad \text{für } t_{i-1} \le t \le t_i .$$

Die Parameter b_i, c_i, d_i und e_i sind hierbei wie bei Gleichung (1) gewählt. Da $|\alpha| < 1$, ist der Operator T bezüglich der Supremumsmetrik eine kontrahierende Abbildung, weshalb nach Banachs Satz der Operator genau einen Fixpunkt f besitzt. Aus der Eigenschaft $Tf = f$ folgt nun unmittelbar die erste Bedingung des Satzes, und aus der speziellen Wahl der Parameter d_i und e_i kann seine zweite Bedingung abgeleitet werden. Da mit g auch stets Tg stetig ist, läßt sich der Operator auf den Raum aller stetigen Funktionen einschränken, weshalb die Fixpunktfunktion f automatisch stetig ist. Q. e. d.

Aus dem Beweis folgt insbesondere, daß der Iterationsprozeß aus Abschnitt 2 bezüglich der Supremumsmetrik, also gleichmäßig, gegen die selbstaffine Grenzfunktion konvergiert. Die Konvergenzgeschwindigkeit ergibt sich dabei aus der Abschätzung

$$\max_t |f_k(t) - f(t)| \le C\alpha^k \quad \text{für } k = 1,2,3,\ldots$$

mit einer Konstanten C.

Satz 1 zeigt auch, daß eine selbstaffine Funktion alleine durch die zugehörige Funktionalgleichung eindeutig bestimmt wird, und es daher gar nicht nötig ist, die Iteration mit dem Streckenzug der linearen Interpolation zu beginnen. Jede beschränkte Startfunktion führt zum selben Ergebnis, wobei sich beim Konvergenzverhalten nur die Konstante C ändert. Dies ermöglicht die Entwicklung sehr unterschiedlicher Interpolationsalgorithmen, die ganz anders als der ursprüngliche Iterationsprozeß arbeiten können.

4. Ein iterativer Interpolationsalgorithmus

Sei für eine bestimmte Stelle t der Interpolationswert $f(t)$ zu berechnen. Dafür ist es zunächst nötig, die Position der Stelle im Basisintervall $[t_0, t_n]$ bezüglich den Ausgangsstellen t_0, t_1, ..., t_n zu bestimmen. Für den ersten Iterationsschritt genügt es zu wissen, in welchem Teilintervall $[t_{i-1}, t_i]$ sie liegt. Sei der zugehörige Index i mit i_1 bezeichnet. Im zweiten Iterationsschritt benötigt man die Position von t in $[t_{i_1-1}, t_{i_1}]$ bezüglich den transformierten Stellen $(t_0 - c_{i_1}) / b_{i_1}$, $(t_1 - c_{i_1}) / b_{i_1}$, ..., $(t_n - c_{i_1}) / b_{i_1}$. Sei der Index des zugehörigen Teil-Teilintervalls dann mit i_2 benannt. Dieses Vorgehen läßt sich wiederholen, bis alle nötigen Indizes i_1, i_2, ..., i_k gefunden sind.

Da die Teilintervalle für hinreichend viele Iterationschritte k beliebig kurz werden, konvergiert also die rechte (und auch die linke) Teilintervallgrenze gegen die Stelle t. Bezeichnet man die Intervallgrenzentransformation mit $L_i(s) = (s - c_i) / b_i$, so gilt also mit wählbarer Genauigkeit

$$t \approx L_{i_1}(L_{i_2}(\ldots L_{i_k}(t_n))).$$

Die Indizes i_1, i_2, ..., i_k sind so die Adresse von t bezüglich der Stützstellen t_1, ..., t_n. Die Idee der Adressen ist eine Verallgemeinerung der Dezimalbruchentwicklung einer reellen Zahl: Für die speziellen Stützstellen $t_i = i / 10$ mit $i = 0, 1, ..., 10$ entspricht die Folge der Indizes gerade den Ziffern hinter dem Komma von t im Intervall $[0; 1]$. Beispielsweise erhält man für $t = 0{,}583$ die Indizes $i_1 = 5$, $i_2 = 8$ und $i_3 = 3$.

Ist nun die Folge der Indizes bereits bestimmt, so kann man den gesuchten Funktionswert als wiederholte Hintereinanderausführung von Hilfsabbildungen unmittelbar angeben. Die Indexwerte steuern dabei die Auswahl der jeweils anzuwendenden Variante:

$$f(t) \approx W_{i_1}(W_{i_2}(\ldots W_{i_k}(t_n, x_n)))_2.$$

Die Hilfsfunktionen W_i, die Paare reeller Zahlen auf Paare abbilden, sind dazu durch

$$W_i(s, y) = \left(\frac{s - c_i}{b_i}, \alpha y + d_i \frac{s - c_i}{b_i} + e_i\right)$$

festgelegt (die tiefgestellte 2 bezeichnet die zweite Komponente des resultierenden Paares). Dies folgt wegen der Stetigkeit selbstaffiner Interpolationsfunktionen aus folgendem Lemma, welches mittels Induktion direkt herleitbar ist:

Lemma: Sei $f:[t_0,t_n]\to I\!R$ eine selbstaffine Interpolationsfunktion mit Stützstellen $t_0 < t_1 < ... < t_n$ und Vertikalfaktor $|\alpha|<1$. Für $x_i = f(t_i)$ seien die Parameter b_i, c_i, d_i, e_i und damit die Hilfsabbildungen W_i wie oben definiert. Dann liefert jede Folge von Indizes $i_1, i_2, ..., i_k$ durch

$$(t,x) = W_{i_1}(W_{i_2}(\ldots W_{i_k}(t_n,x_n)))$$

stets einen Punkt auf dem Graphen von f, d. h. der Interpolationswert an der Stelle t ist $x = f(t)$.

Eine mögliche Implementierung könnte also aus zwei Schritten bestehen: Im ersten Schritt wird die Adresse der Interpolationsstellen mittels einer Intervallschachtelung bestimmt und in einem Vektor $(i_1,i_2,\ldots,i_k)$ abgespeichert. Anschließend steuert der Vektor die Berechnung des Interpolationswertes. Die verwendete Programmiersprache muß dazu nur die Verwaltung von Vektoren unterstützen, nicht jedoch rekursive Funktionsaufrufe.

5. Ein rekursiver Interpolationsalgorithmus

Der hier vorgestellte Algorithmus erfordert eine Programmiersprache, die rekursive Funktionsaufrufe unterstützt, wie etwa PASCAL, C oder neuere Dialekte von BASIC. Denn dann kann die selbstaffine Funktionalgleichung direkt verwendet werden. Dies läßt sich nämlich mit folgender rekursiv definierten Hilfsfunktion $f^{\mathrm{rec}}(t,k)$ bewerkstelligen:

$$f^{\mathrm{rec}}(t,k) = \begin{cases} \alpha\, f^{\mathrm{rec}}(b_i t + c_i, k-1) + d_i t + e_i & \text{falls} \quad k > 0 \\ x_n & \text{falls} \quad k = 0 \end{cases},$$

wobei der Index i so gewählt wird, daß $t_{i-1} \le t \le t_i$ gilt. In Abb. 3 ist die Funktionsdefinition konkret als Struktogramm dargestellt. Der Parameter k gibt dabei die Anzahl der noch durchzuführenden Rekursionsschritte an. Ist er null, so ist der Funktionswert unabhängig vom Argument t stets die Konstante x_n.

Daß sie auch tatsächlich selbstaffin interpoliert, zeigt folgender

Satz 2: Sei $f:[t_0,t_n]\to I\!R$ die selbstaffine Interpolationsfunktion zu den Stützstellen $t_0 < t_1 < ... < t_n$ und Werten $x_i = f(t_i)$ bei einem Vertikalfaktor $|\alpha|<1$. Sei damit die rekursive Hilfsfunktion f^{rec} wie oben definiert. Dann konvergieren mit wachsender Re-

kursionstiefe die Hilfsfunktionswerte gegen die Interpolationswerte, d. h. es gilt für $t_0 \le t \le t_n$ stets

$$f(t) = \lim_{k\to\infty} f^{\text{rec}}(t,k).$$

Beweis: Mit dem Operator T aus dem Beweis von Satz 1 läßt sich die Definition der rekursiven Hilfsfunktion einfach als Anwendung dieses Operators auf die Hilfsfunktion mit einer um Eins verringerten Rekursionstiefe schreiben. Denn es gilt, wenn man $f_k^{\text{rec}}(t) = f^{\text{rec}}(t,k)$ setzt, die Gleichheit $Tf_{k-1}^{\text{rec}} = f_k^{\text{rec}}$. Wählt man dann für $k = 0$ die konstante Startfunktion $f_0^{\text{rec}}(t) = x_n$, so ist $f_k^{\text{rec}} = T^k f_0^{\text{rec}}$. Da T wegen $|\alpha|<1$ kontrahierend ist, konvergiert die Funktionenfolge f_1^{rec}, f_2^{rec}, f_3^{rec}, ... gleichmäßig gegen die Interpolationsfunktion f. Daraus folgt die behauptete punktweise Konvergenz. Q. e. d.

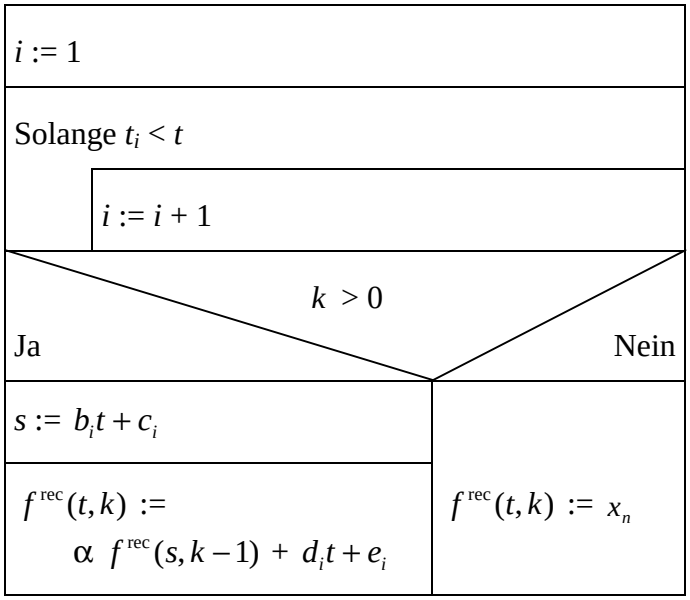


Abb. 3: Struktogramm des rekursiven Prozedurteils

Bei der Programmierung dieses Algorithmus ist es, um eine hohe Effizienz zu erzielen, sinnvoll, die Berechnung der Parameter b_i, c_i, d_i, e_i ganz zu Beginn einmal durchzuführen und anschließend die Hilfsfunktion auf alle gewünschten Interpolationsstellen in einem Durchgang anzuwenden. Die über die Schnittstelle des Programmiersystems bereitgestellte Prozedur hat dann folgende Parameter (vgl. Anhang A):

1. die *Interpolationsstellen* als einen Vektor $(t_0, t_1, \ldots, t_n)$ reeller Zahlen,
2. die *Interpolationswerte* als einen Vektor $(x_0, x_1, \ldots, x_n)$ reeller Zahlen,
3. die *zu interpolierenden Stellen* als einen Vektor $(s_1, s_2, \ldots, s_m)$ reeller Zahlen (für $m = 1$ ist nur ein Funktionswert gewünscht),

4. der *Vertikalfaktor* als reelle Zahl α ,
5. die *Rekursionstiefe* als ganze Zahl k .

Von der Prozedur werden dann die interpolierten Werte $(y_1, y_2, \ldots, y_m)$ berechnet, wie in Abb. 4 als Struktogramm dargestellt ist.

Für $i := 1$ bis n
$b_i := \frac{t_n - t_0}{t_i - t_{i-1}}, \quad c_i := t_n - b_i t_i$
$d_i := \frac{x_i - x_{i-1} - \alpha(x_n - x_0)}{t_i - t_{i-1}}, \quad e_i := x_i - \alpha x_n - d_i t_i$
Für $j := 1$ bis m
$y_j := f^{\text{rec}}(s_j, k)$

Abb. 4: Struktogramm der Rahmenprozedur

Abb. 5 zeigt ein Anwendungsbeispiel für die selbstaffine Interpolation. Links ist der tagesgenaue Kurs der Bayeraktie im Zeitraum vom 16.03.87 bis zum 17.01.96 dargestellt. Rechts sind aus den Informationen über die Kurswerte am 05.10.87, 10.11.87, 02.02.90, 08.11.90, 08.03.91, 10.02.92, 12.10.92 und 27.04.94 (mit den Randtagen also $n+1 = 10$ Stützstellen) und einem Volatilitätsparameter von $\alpha = 0{,}3$ die tagesgenauen fehlenden Werte interpoliert. Der Volatilitätsparameter wurde dabei durch Ausprobieren manuell so gewählt, daß die ursprüngliche Volatilität möglichst gut rekonstruiert wird.

Falls bei der Anwendung keine numerische Volatilitätsinformation vorliegt, so kann als Standardwert $\alpha = 0{,}5$ verwendet werden. Es geht ja hier darum, den visuellen Eindruck zu optimieren. Grundsätzlich ist allerdings einschränkend zu bemerken: Wird die Approximationsgüte durch ein formales Maß, wie etwa die Fehlerquadratsumme beurteilt, so bietet die selbstaffine Interpolation wohl keinen Vorteil. Dann dürfte sogar eine lineare Interpolation mindestens gleich gute Ergebnisse liefern.

Alle Grafiken in diesem Aufsatz wurden mit Visual-BASIC für Microsoft Excel der Version 5 erstellt. Das zugehörige Excel-Makro „Saffint“ ist in Anhang A beschrieben und sein Quellcode in Anhang B aufgelistet.

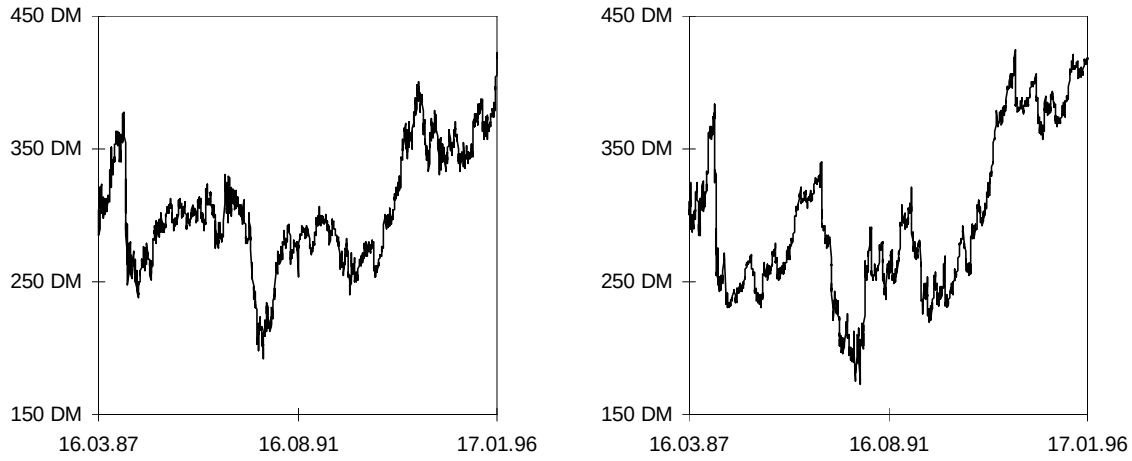


Abb. 5: Tagesgenauer Aktienkurs der Bayer AG (links) und seine selbstaffine Interpolation mit Volatilitätsparameter 0,3 (rechts)

6. Volatilitätsmessung

Ein Vertikalfaktor α kann als Volatilitätsparameter nach dem visuellen Eindruck manuell gefunden werden. Sein konkret gewählter Zahlenwert ist dabei allerdings von den Interpolationsdaten abhängig und daher nicht ohne weiteres auf ähnliche Probleme, wie etwa einer anderen Wahl der Stützstellen, übertragbar. Möchte man dies jedoch erreichen, so muß der Volatilitätsparameter geeignet standardisiert werden. Von verschiedenen möglichen Meßkonzepten für die Volatilität einer Zeitreihe wird hier der Höldersche Exponent verwendet. Er beschreibt in gewisser Weise, wie stark die Zeitreihe von einem glatten – im mathematischen Sinne differenzierbaren – Verlauf abweicht. Der Höldersche Exponent einer Funktion g ist das Supremum aller positiven Zahlen $\delta \leq 1$, für die der Differenzenquotient

$$\frac{g(s)-g(t)}{|s-t|^{\delta}}$$

dem Betrag nach für alle s, t beschränkt bleibt. Es gilt dann stets die Abschätzung

$$|g(s)-g(t)| \leq D|s-t|^{\delta}$$

mit einer Konstanten D. Man beachte, daß jede differenzierbare Funktion auf ihrem Definitionsintervall $[t_0, t_n]$ einen Hölderschen Exponenten von $\delta = 1$ hat. Der Exponent läßt sich auf verschiedene Arten schätzen, unter der Annahme eines Gaußschen Prozesses beispielsweise durch das Abzählen von Niveau-Überquerungen (vgl. Feuerverger u. a. 1994).

Für eine selbstaffine Interpolationsfunktion kann der Höldersche Exponent direkt angegeben werden: Setzt man $b = \max_i b_i$ und gilt $|\alpha| > 1/b$, so beträgt er

$$\delta = \frac{\log(1/|\alpha|)}{\log b}, \tag{3}$$

sofern nicht alle Interpolationspunkte auf einer Geraden liegen. In allen anderen Fällen ist der Exponent gleich 1 (vgl. Bedford 1989, Fickel 1988). Für betragsmäßig kleine Volatilitätsparameter $|\alpha| << 1/b$ ist die Interpolationsfunktion also glatt. Je größer über diese Schranke hinaus der Parameter wird, desto geringer ist dann der Höldersche Exponent.

Ist also für eine Zeitreihe der Höldersche Exponent δ als Volatilitätsmaß identifiziert, so kann die Wahl des Vertikalfaktors α automatisiert werden: Man muß nur Gleichung (3) nach dem Vertikalfaktor auflösen und erhält dann als Wahlmöglichkeit

$$\alpha = \pm e^{-\delta \log b}. \tag{4}$$

Wurde für eine Zeitreihe wie im Beispiel von Abb. 5 der Vertikalfaktor durch Probieren visuell bestimmt, sollen jedoch andere Stützstellen verwendet werden, so kann man wie folgt verfahren: Zunächst wird zu den alten Stützstellen mit Gleichung (3) der Höldersche Exponent berechnet (hier gilt $b = 89{,}7$ und damit $\delta = 0{,}27$), dann wird für die neuen Stützstellen der Vertikalfaktor gemäß Formel (4) festgelegt. Bei diesem Vorgehen ist der Höldersche Exponent nur eine Hilfsgröße, er dient hier nur mittelbar als konkretes Volatilitätsmaß. Weiterführend wäre es dazu interessant, mittels Versuchspersonen in Befragungen empirisch zu überprüfen, welche formalen Maße besonders hoch mit einer visuell empfundenen Volatilität korrelieren.

7. Schlußbemerkung

Die Visualisierung der Volatilität bei der Interpolation ermöglicht zu veranschaulichen, wie „eine Zeitreihe ausgesehen haben dürfte, wenn im zeitlichen Verlauf Beobachtungen kontinuierlich gemacht worden wären.“ (Chatterjee/Yilmaz 1992: 135, übers. N.F.). Dazu hat die Anwendung der selbstaffinen Interpolation bei Verwendung des hier dargestellten Algorithmus folgende Vorteile: Der Grad der Volatilität kann entweder manuell durch Ausprobieren oder durch ein formales Volatilitätsmaß festgelegt werden. Dabei wird der Graph deterministisch konstruiert, d. h. ein Zufallszahlengenerator wird nicht benötigt. So ist die grafische Darstellung unabhängig von der Hardwarekonfiguration, da nur einige Parameter und keine Pixels zu speichern sind. Zudem ist das Konstruktionsschema anschaulich an der Grafik als selbstaffine Struktur nachvollziehbar und es

hängt nur von den vorgegebenen Interpolationspunkten und dem vorgegebenen Volatilitätsgrad ab.

Offen ist allerdings noch die Frage, welche Beziehung zu ARCH- und GARCH-Ansätzen oder ähnlichen Modellierungen der Volatilität besteht (vgl. Lütkepohl 1996). Abschließend sei nochmals bemerkt, daß die selbstaffine Interpolation weniger ein Hilfsmittel ist, um bestimmte fehlende Ausprägungen einer Zeitreihe zu schätzen. Die Gesamtheit aller interpolierten Werte visualisiert jedoch auf anschauliche Art eine vorhandene Volatilitätsinformation. Damit läßt sich beim Betrachten einer so rekonstruierten Kurve – im Unterschied zur linearen Interpolation – wahrnehmen, wie volatil ihr tatsächlicher Verlauf gewesen sein könnte.

An dieser Stelle möchte ich mich bei den Professoren Gerd Hansen und Jürgen Wolters bedanken, die mir die Möglichkeit gegeben haben, obige Ideen auf dem Workshop für Angewandte Ökonometrie in Berlin zu diskutieren. Außerdem danke ich Gunnar Bönte, der mir die Daten für Abb. 5 bereitgestellt hat, und Prof. Siegfried Maaß für wertvolle Hinweise.

Literatur

Barnsley, Michael F. "Fractal functions and interpolation". *Constructive Approximation* 2 (1986): 303-329.

Barnsley, Michael F. *Fraktale: Theorie und Praxis der Deterministischen Geometrie.* Heidelberg: Spektrum, 1995.

Bedford, Tim. "Hölder exponents and box dimension for self-affine fractal functions". *Constructive Approximation* 5 (1989): 33-48.

Chatterjee, Sangit, u. Mustafa R. Yilmaz. "Use of estimated fractal dimension in model identification for time series". *Journal of Statistical Computation and Simulation* 41 (1992): 129-141.

Feuerverger, Andrey, Peter Hall u. Andrew T. A. Wood. "Estimation of fractal index and fractal dimension of a Gaussian process by counting the number of level crossings". *Journal of Time Series Analysis* 15 (1994): 587-606.

Fickel, Norman. *Selbstaffine Interpolation*. Diplomarbeit. Erlangen: Universität Erlangen-Nürnberg, 1988.

Hutchinson, John E. "Fractals and self similarity". *Indiana University Mathematics* Journal 30.5 (1981): 713-747.

Lütkepohl, Helmut. *Statistische Modellierung von Volatilitäten*. Diskussionspapier zur Quantifikation und Simulation ökonomischer Prozesse 70/1996. Berlin: Humboldt-Universität, 1996.

Anhang A: Beschreibung des Excel-Makros „Saffint“

Nachfolgend ist das Excel-Makro „Saffint“ als benutzerdefinierte Tabellenfunktion für Microsoft Excel 5.0 im Stile der zugehörigen Hilfefunktion beschrieben.

SAFFINT

Liefert Werte, die sich durch eine selbstaffine Interpolation ergeben. Diese Funktion interpoliert die in den Vektoren **Y_Werte** und **X_Werte** übergebenen Werte und gibt als Ergebnis die y-Werte, die zu den in **Neue_x_Werte** angegebenen x-Werten gehören, zurück.

Syntax

SAFFINT(**Y_Werte**; **X_Werte**; **Neue_x_Werte**; Vertikalfaktor; Rekursionstiefe)

Y_Werte sind die y-Werte, die Sie zur Interpolation vorgeben.

- Der Bereich **Y_Werte** darf nur aus einer Spalte bestehen. Die maximale Zeilenanzahl beträgt 20.

X_Werte sind die x-Werte, die Sie zur Interpolation vorgeben.

- Der Bereich **X_Werte** darf nur aus einer Spalte bestehen und muß ebenso viele Zeilen wie **Y_Werte** besitzen.

Neue_x_Werte sind die neuen x-Werte, für welche die Funktion SAFFINT die zugehörigen y-Werte liefern soll.

- Der Bereich **Neue_x_Werte** darf nur aus einer Spalte bestehen und kann 1 bis 1.000 Zeilen besitzen.

Vertikalfaktor ist der Streckungsfaktor, mit dem die Interpolationsdaten vertikal gestaucht werden. Der Vertikalfaktor muß ein Wert zwischen -1,0 und 1,0 sein (vom Betrag echt kleiner als 1,0).

- Ist **Vertikalfaktor** nicht angegeben, so wird mit dem Wert 0,5 gerechnet.
- Statt einem einzigen Wert kann **Vertikalfaktor** auch ein Vektor (Bereich mit einer Spalte) sein, der genau ein Element weniger als der Vektor **X_Werte** bzw. **Y_Werte** enthält.
- Bei einem **Vertikalfaktor** gleich 0,0 berechnet SAFFINT eine lineare Interpolation.

Rekursionstiefe ist die Anzahl der Rekursionsschritte, die bei der Berechnung ausgeführt werden.

- Wird die selbstaffine Interpolation grafisch dargestellt, so muß die Rekursionstiefe desto größer sein, je höher die Auflösung des Diagramms ist.
- Ist **Rekursionstiefe** nicht angegeben, so wird der Wert 5 verwendet.

Hinweis

- Formeln, die als Ergebnis eine Matrix liefern, müssen als Matrixformel eingegeben werden. Dies trifft bei SAFFINT zu, falls der Bereich **Neue_x_Werte** mehr als eine Zeile umfaßt.

Beispiel

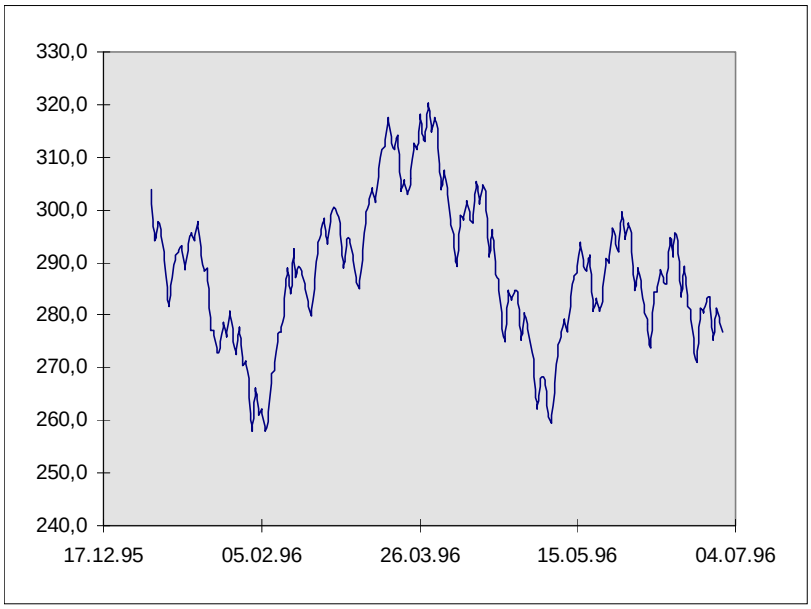
Angenommen, für eine grafische Präsentation soll der Aktienkurs des letzten Halbjahres eines Unternehmens dargestellt werden. Zur Verfügung stehen aber als y-Werte nur die Kurswerte an den Monatsanfängen, die im Bereich B1:B7 abgelegt sind. Der Bereich C1:C182 enthält die Datumszahlen vom 1. Januar bis zum 30. Juni:

	A	B	C	D
1	01.01.96	304 DM	01.01.96	
2	01.02.96	266 DM	02.01.96	
3	01.03.96	297 DM	03.01.96	
4	01.04.96	314 DM	04.01.96	
5	01.05.96	270 DM	05.01.96	
6	01.06.96	294 DM	06.01.96	
7	01.07.96	273 DM	07.01.96	
182			30.06.96	

Wird die folgende Anweisung als einspaltige Matrix in den Bereich D1:D182 eingegeben, liefert sie die selbstaffine Interpolation des Aktienkurses für das erste Halbjahr 1996 mit Vertikalfaktor 0,4:

SAFFINT(B1:B7;A1:A7;C1:C182;0,4) ergibt {304,0; 294,2; ...; 276,7}

Markieren Sie nun den Bereich C1:D182 und starten Sie den Diagramm-Assistenten, so können Sie die erzeugte Zeitreihe grafisch darstellen:



Verwandte Funktionen (in Excel 5.0)

TREND	Berechnet eine Regressionsgerade nach der Methode der kleinsten Quadrate.
VARIATION	Ist mit TREND vergleichbar, paßt den jeweiligen Daten aber eine Exponentialkurve an.

Anhang B: BASIC-Quellcode des Excel-Makros „Saffint“

Der nachfolgende Quellcode ist in Abschnitt 5 mittels Struktogrammen dokumentiert.

```
' Selbstaffine Interpolation

Option Explizit

' Voreinstellungen
Konst vor_Vertikalfaktor = 0,5
Konst vor_Rekursionstiefe = 5

Konst NMAX = 20          ' Obergrenze für n
Konst MMAX = 1000        ' Obergrenze für m

' Zu berechnende Hilfsgrößen
Dim n Als Ganz           ' Anzahl der Stützstellen
                         ' (um Eins verringert)
Dim m Als Ganz           ' Anzahl der neuen x-Werte
                         ' (um Eins verringert)
Dim x(NMAX) Als Doppelt  ' Stützstellen
Dim y(NMAX) Als Doppelt  ' Werte
Dim alpha(NMAX) Als Doppelt ' Vertikalfaktor(en)

Dim b(NMAX) Als Doppelt
Dim c(NMAX) Als Doppelt
Dim d(NMAX) Als Doppelt
Dim e(NMAX) Als Doppelt


' *** Subprozedur zum Berechnen der Hilfsgrößen ***
Privat Sub Berhilf()
  Dim i Als Ganz ' Schleifenindex
  Dim dx Als Doppelt
  Dim dy Als Doppelt

  dx = x(n) - x(0)
  dy = y(n) - y(0)

  Für i = 1 Bis n
    b(i) = dx / (x(i) - x(i - 1))
    c(i) = x(n) - b(i) * x(i)
    d(i) = (y(i) - y(i - 1) - alpha(i) * dy) _
           / (x(i) - x(i - 1))
    e(i) = y(i) - alpha(i) * y(n) - d(i) * x(i)
  Nächste i
Ende Sub
```

```
' *** rekursive Hilfsfunktion ***
Privat Funktion SaffRekursiv(x_ Als Doppelt; _
                  r Als Ganz) Als Doppelt
  Dim i Als Ganz
  Dim x_inv Als Doppelt

  Wenn x_ > x(n) Dann
    i = n
  Sonst
    i = 1
    Solange x(i) < x_
      i = i + 1
    EndeSolange
  Ende Wenn

  Wenn r > 0 Dann
    x_inv = b(i) * x_ + c(i)
    SaffRekursiv = alpha(i) * SaffRekursiv(x_inv; r - 1) _
                   + d(i) * x_ + e(i)
  Sonst
    SaffRekursiv = y(0)
  Ende Wenn
Ende Funktion


' *** Rahmenprozedur ***
Funktion SaffInt(Y_Werte Als Variant; _
                 X_Werte Als Variant; _
                 Neue_x_Werte Als Variant; _
    Optional Vertikalfaktor Als Variant; _
    Optional Rekursionstiefe Als Variant) Als Variant
  Dim i Als Ganz
  Dim j Als Ganz
  Dim Neue_y_Werte(MMAX; 0) Als Doppelt
  Dim r Als Ganz
  Dim Anzahl_Alphas Als Ganz

  n = Anwendung.Anzahl(Y_Werte) - 1
  m = Anwendung.Anzahl(Neue_x_Werte) - 1

  Wenn IstFehlend(Vertikalfaktor) Dann
    Anzahl_Alphas = 1
    alpha(1) = vor_Vertikalfaktor
  SonstWenn IstZahl(Vertikalfaktor) Dann
    Anzahl_Alphas = 1
    alpha(1) = Vertikalfaktor
  Sonst
    Anzahl_Alphas = n
    alpha(1) = Vertikalfaktor(1)
  Ende Wenn
```

```
  Wenn IstFehlend(Rekursionstiefe) Dann
    r = vor_Rekursionstiefe
  Sonst
    r = Rekursionstiefe
  Ende Wenn

  x(0) = X_Werte(1)
  y(0) = Y_Werte(1)
  Für i = 1 Bis n
    x(i) = X_Werte(i + 1)
    y(i) = Y_Werte(i + 1)
    Wenn Anzahl_Alphas = 1 Dann
      alpha(i) = alpha(1)
    Sonst
      alpha(i) = Vertikalfaktor(i)
    Ende Wenn
  Nächste i

  Rufe Berhilf

  Für j = 0 Bis m
    Neue_y_Werte(j; 0) = _
      SaffRekursiv(Neue_x_Werte(j + 1); r)
  Nächste j

  SaffInt = Neue_y_Werte
Ende Funktion
```