CosWin

—

Approximation durch Sliding Windows

Jörg Rädler

22. Juni 2001

Zusammenfassung

Bei der Verarbeitung von Zahlenreihen in Simulationsprogrammen besteht meist das Problem, daß Funktionswerte benötigt werden, die zwischen den zur Verfügung stehenden Daten liegen. Lineare Interpolation ist einfach und schnell, die Ergebnisse sind jedoch meist nicht stetig differenzierbar. Kubische Spline-Interpolation ergibt einen sinnvolleren Funktionsverlauf, benötigt aber viel Rechenzeit und Speicher. Die CosWin-Approximation stellt eine einfache Methode dar, durch eine wichtende Fensterfunktion (ein sliding window) diskrete Zahlenreihen in stetige und stetig differenzierbare Funktionen zu wandeln. Dabei kann durch einen einzelnen Parameter der Grad der Glättung bestimmt werden.

1 Grundlagen

Um eine Glättung (Beseitigung von Knicken) einer Funktion zu erzielen, soll der Funktionswert in Abhängigkeit seiner Umgebung (Fenster) bestimmt werden. Das betrachtete Fenster $x_0 \cdots x_1$ um den Wert x_s soll endlich begrenzt sein, damit zur Bestimmung eines Funktionswertes nicht der gesamte Datensatz ausgewertet werden muß. Die Bestimmungsgleichung eines Wertes $y_s = g(x_s)$ nimmt folgende Form an:

$$y_s = \frac{\int_{x_0}^{x_1} f(x) \cdot w(x) dx}{\int_{x_0}^{x_1} w(x) dx} \tag{1}$$

Darin ist $f(x)$ die ursprüngliche Funktion und $w(x)$ die wichtende Fensterfunktion. Der Nenner dient der Normierung des Integrals auf 1.

Für $f(x)$ setzen wir einfach die lineare Interpolation der x, y-Wertepaare an, um einen stetigen Verlauf zu erhalten. Die Funktion nimmt damit die Form

$$f(x) = a \cdot x + b \tag{2}$$

an, mit intervallabhängigen Parametern a und b.

Bleibt nur noch die Wahl einer günstigen Fensterfunktion. Es soll der direkten Nachbarschaft mehr Gewicht eingeräumt werden als den entfernteren Werten. Der Funktionswert und die Ableitung der Funktion an den Fenstergrenzen müssen 0 betragen, damit der Einfluß der in das Fenster eintretenden oder aus dem Fenster austretenden Stützstellen nicht zu Sprüngen oder Knicken führt. Außerdem ist eine einfach zu formulierende Funktion Bedingung für kurze Rechenzeiten. Stetigkeit und Differenzierbarkeit werden natürlich vorausgesetzt.

Alle diese Bedingungen erfüllt z.B. die Funktion

$$w(x) = 0.5 + 0.5 \cdot \cos(x) \tag{3}$$

im Intervall $x_0 = -\pi$ und $x_1 = \pi$. Den Verlauf zeigt Bild 1.

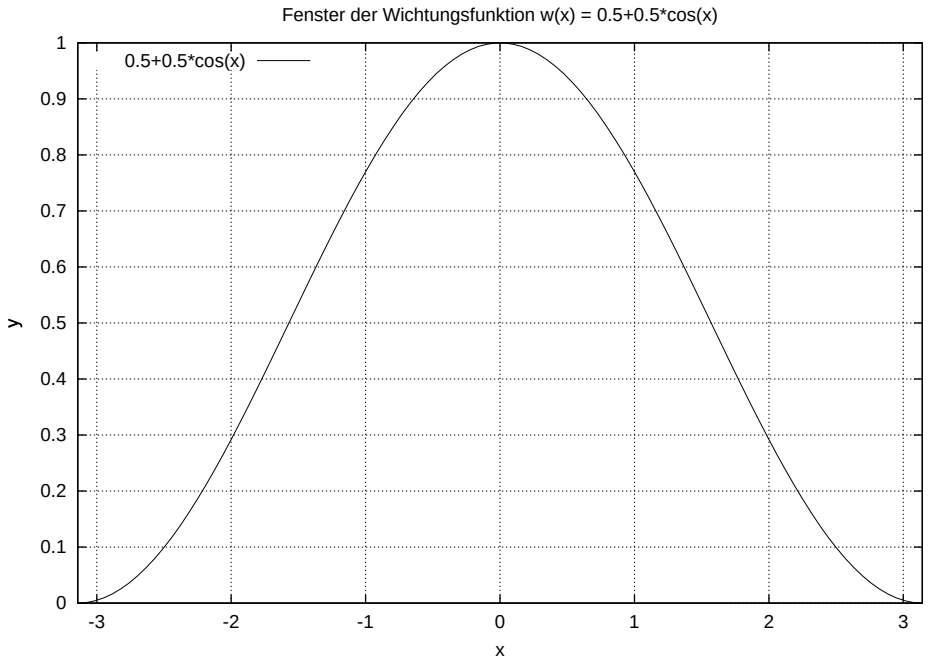

Abbildung 1: Wichtende Fensterfunktion

Der Nenner in Gleichung 1 ergibt sich zu π. Das bestimmte Integral des Produktes beider Funktionen (Zähler) kann einfach formuliert werden. Eine Stammfunktion $U(x)$ des Produktes $f(x) \cdot w(x)$ lautet

$$U(x) = 0.5 \cdot (\sin(x) \cdot (b + a \cdot x) + a \cdot \cos(x) + 0.5 \cdot a \cdot x^2 + b \cdot x). \tag{4}$$

Es ist noch notwendig, einige Transformationen vorzunehmen. Entweder das Fenster muß über den gewünschten Bereich geschoben werden, oder $f(x)$ unter das Fenster, so daß $x_s = 0$ gilt. Außerdem soll die Fenstergröße in Bezug auf unsere Wertepaare vorgegeben werden können.

2 Implementierung

Ziel war, eine C-Funktion zu erstellen, die direkt einen Funktionswert zurückgibt. Diese Funktion steht als

```
double coswin(size_t num, double x, double ws,
              double *xi, double *yi);
```

zur Verfügung. Als Parameter benötigt sie neben der Anzahl der übergebenen Stützstellen num den gewünschten Wert von x, die Fenstergröße ws und einige xi- und yi- Werte der Stützstellen. Der übergebene Bereich der Stützstellen muß groß genug sein, daß das gesamte Fenster um x betrachtet werden kann!

Zuerst werden von der Funktion die Intervalle gesucht, die vom Fenster überdeckt werden[1]. Für jedes dieser Intervalle werden nun a und b bestimmt und passend zum Fenster transformiert, die entsprechenden Anteile an Gleichung 1 unter Beachtung der Fenster- und Intervallgrenzen durch die Stammfunktion errechnet und aufsummiert. Nach der Normierung (Division durch π) kann das Ergebnis zurückgegeben werden.

3 Beispiel

Als Grundlage dient die Funktion $y = \sin(x)$. Sie wird in Intervallen von 1 als diskrete x, y-Wertepaare errechnet. Diese Wertepaare sollen nun durch die CosWin-Approximation ausgewertet und ein Verlauf errechnet werden. Abbildung 2 zeigt das Ergebnis.

Bei einer Fenstergröße von 0.8 nimmt die Ergebnisfunktion stückweise den Verlauf der linearen Interpolation an. Lediglich in der Nähe der Stützstellen tritt eine Glättung auf. Da die Fenstergröße kleiner ist als der Abstand der Stützstellen, liegt das Fenster teilweise nur innerhalb eines Intervalls mit einer einzelnen linearen Funktion!

Erweitert man das Fenster auf 1.8, entfällt dieser Effekt. Es werden nun immer mehrere Intervalle mit (in diesem Falle) unterschiedlichen linearen Funktionen innerhalb des Fensters ausgewertet. Ein glatterer Verlauf ist die Folge, aber auch eine größere Abweichung von der ursprünglichen Funktion!

[1] Also bitte nicht allzuviele Wertepaare übergeben, sonst dauert die Suche viel länger als die eigentliche Berechnung!

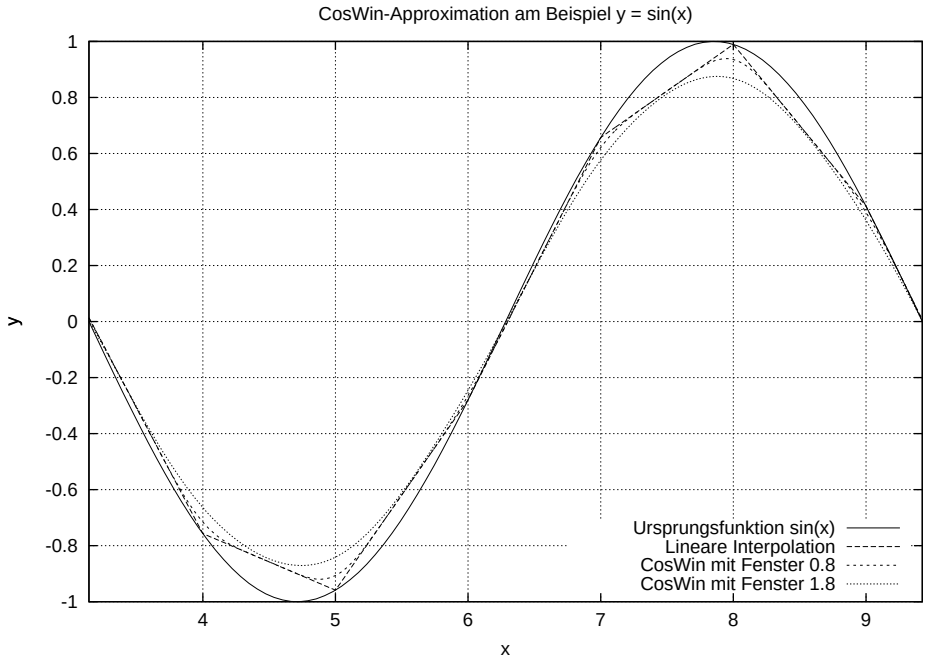

Abbildung 2: Ergebnisse der Anwendung im Vergleich mit der Ursprunsfunktion und einer linearen Interpolation

4 Fazit

Die CosWin-Approximation stellt sicherlich keine so genaue Abschätzung zur Verfügung, wie sie z.B. mit kubischer Spline-Interpolation erreicht wird. Sie ist aber einfach handhabbar, deutlich weniger rechenintensiv[2] und kann auf unterschiedliche Bedürfnisse angepaßt werden. Teilweise lineare Verläufe, die bei kubischen Splines oft zu einer Schwingung führen, stellen kein Problem dar.

Wird lediglich eine Alternative zur stückweise linearen Interpolation mit stetig differenzierbarem Ergebnis gesucht, liefert die CosWin-Approximation mit geringer Fenstergröße gute Ergebnisse.

Der Nachteil der Ungenauigkeit (Glattbügeln der Extrema) bei größeren Fenstern im Verhältnis zum Stützstellenabstand sollte nicht so stark ins Gewicht fallen, da bei stark wechselnden Signalen auch Datensätze mit hoher Auflösung zur Verfügung stehen sollten!

Der Glättungseffekt kann natürlich auch bewußt eingesetzt werden, um ein z.B. durch Quantisierungsfehler entstehendes “Rauschen” von Meßwerten zu unterdrücken.

[2]Erste Abschätzungen ergeben ein Rechenzeitverhältnis von ca. 10. . . 15!