

Max Wisniewski , Alexander Steen

Tutor: Sebastian Scherer

Aufgabe 1

Das folgende matlab-Programm rechnet für $k = 1, 2, 6$ die jeweiligen Newton-Cotes-Formeln aus. Für andere Eingaben von k wird das Program mit einem Fehler terminieren. Der Aufwand ist jeweils auf die Berechnung des Programms bezogen, welche Funktionsauswertungen nicht mehrfach benutzt und darum etwas ineffizienter ist, als es sein könnte. Allerdings wurde in der Aufgabe nichts von Effizienz gesagt, also ist das ja kein Problem.

```
function [S,A] = sumnc(ab,f,k,n)
% ab Integrationsintervall
% f zu integrierende Funktion
% k regel der newton-cotes-formel
% n anzahl der teilintervalle

% Bezeichnungen ab hier wie im Skript
% h ist der Abstand zwischen den Stellen
h = (ab(2) - ab(1))/n;
% z sind die Stuetzstellen
z = ab(1) + (1:n).*h;

% Gewichte jeweils aus dem Skript entnommen,
% sowie Berechnungsvorgehen

if (k == 1) %% Trapez
    x = @(k) z(k);
    S = h/2 * (f(ab(1)) + f(ab(2)));
    A = 2; % Bereits 2 Auswertungen
    for k = 2:(n),
        S = S + h*f(x(k)); % Siehe Skript
        A = A+1; % Eine Berechnung mehr
    end
    return % Aufwand n+1
elseif (k == 2) %% Simpson
    % (Algorithmisches) Vorgehen analog zu Trapez
    x = @(k,j) z(k/2) + j*h/2;
    S = 0;
    A = 0;
    for k = 1:(n),
        S = S + h/6 * (f(x(2*k,0)) + 4*f(x(2*k,1)) + f(x(2*k,2)))
        ;
        A = A + 3;
    end
    return % Aufwand 3n
elseif (k == 6) %% Weddle-Regel
    x = @(k,j) z(k/6) + j*h/6;
    S = 0; A = 0;
    for k = 1:n,
        S = S + h/840 * (41*f(x(6*k,0)) + 216*f(x(6*k,1)) + 27*f(x(6*k,2)) + 272*f(x(6*k,3)) + 27*f(x(6*k,4)) + 216*f(x(6*k,5)) + 41*f(x(6*k,6)));
        A = A + 6;
    end
    return %Aufwand 6n
end
```

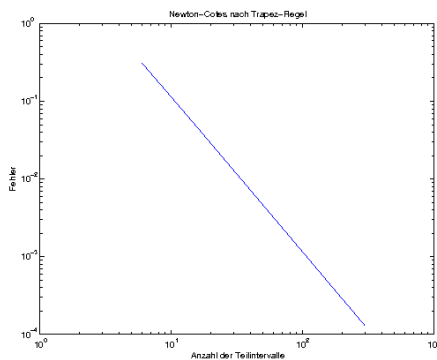
- a) Die Auswertungen wurden sowohl bei a) als auch bei b) mit folgendem Code gemacht (natürlich bei beiden Teilaufgaben mit verschiedenen Funktionen f und Grenzen ab)

```
>> for k = 1:50,
    erg(k) = sumnc(ab,f,1,6*k);
end
>> for k = 1:50,
    erg2(k) = sumnc(ab,f,2,6*k);
end
>> for k = 1:50,
    erg3(k) = sumnc(ab,f,6,6*k);
end
```

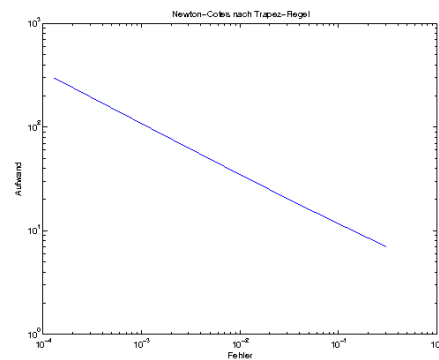
Und dann mit diesem Code geplottet (Achsenbenennung weggelassen):

```
h=loglog(6:6:6*50,abs(2-erg))
h2=loglog(6:6:6*50,abs(2-erg2))
h3=loglog(6:6:6*50,abs(2-erg3))
```

Dies kann man machen, da das exakte Integral $\int_0^\pi \sin(x) dx = 2$.

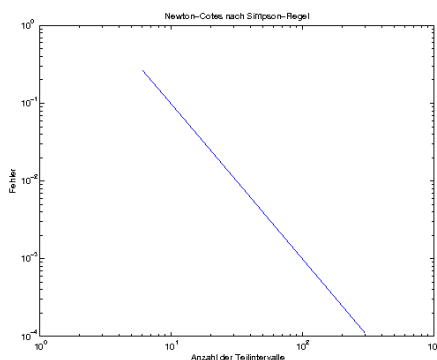


(a) Fehler über Teilintervalle

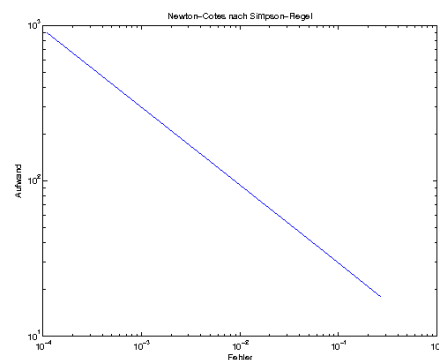


(b) Aufwand über Fehler

Abbildung 1: Newton-Cotes-Formel nach Trapez-Regel

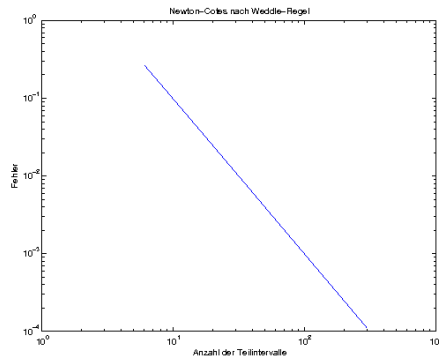


(a) Fehler über Teilintervalle

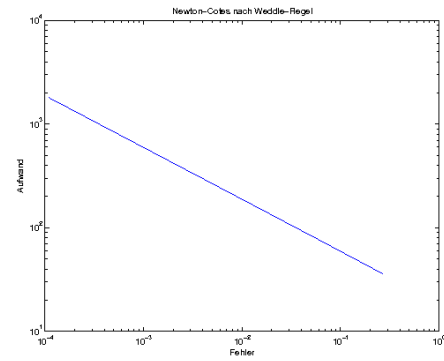


(b) Aufwand über Fehler

Abbildung 2: Newton-Cotes-Formel nach Simpson-Regel



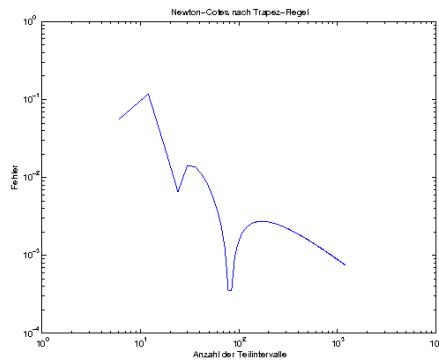
(a) Fehler über Teilintervalle



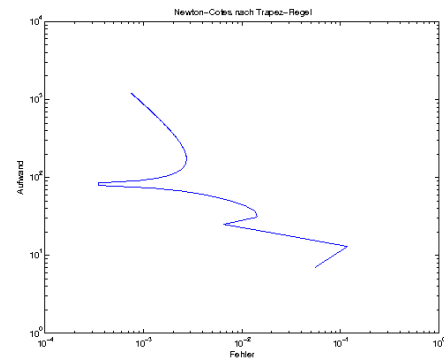
(b) Aufwand über Fehler

Abbildung 3: Newton-Cotes-Formel nach Weddle-Regel

b) Berechnung analog zu a). Hier ist der exakte Wert des Integrals $\int_0^1 \left(\sqrt{x} + \sin(21\pi x) \right) dx = \frac{2}{21} \left(7 + \frac{1}{\pi} \right)$. Dies haben wir so in matlab eingegeben und als Vergleichswert benutzt.

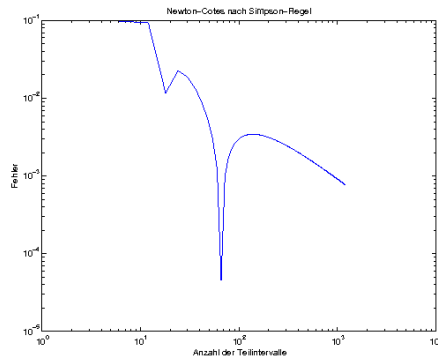


(a) Fehler über Teilintervalle

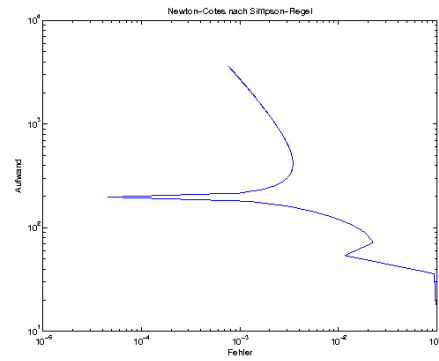


(b) Aufwand über Fehler

Abbildung 4: Newton-Cotes-Formel nach Trapez-Regel

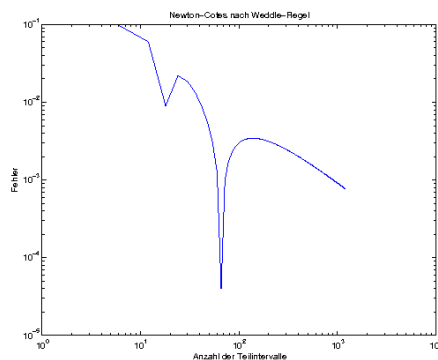


(a) Fehler über Teilintervalle

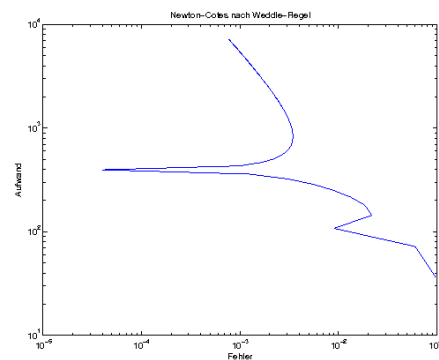


(b) Aufwand über Fehler

Abbildung 5: Newton-Cotes-Formel nach Simpson-Regel



(a) Fehler über Teilintervalle



(b) Aufwand über Fehler

Abbildung 6: Newton-Cotes-Formel nach Weddle-Regel

Schaut man sich die Integrale an, so stellt man fest, dass die erste Funktion einen glatten Bogen beschreibt und die zweite Funktion extrem springt, also viele starke Variationen enthält.

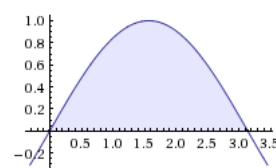
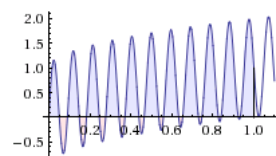

(a) $\int_0^\pi \sin(x) dx$

(b) $\int_0^1 \left(\sqrt{x} + \sin(21\pi x) \right) dx$

Abbildung 7: Graphische Darstellung der beiden Integrale (von: Wolframalpha.com)

Der lineare Verlauf (auf der log-Skala) der Fehler über die Teilintervalle zeigt ein akzeptables Verhältnis und ist bei allen drei Formeln (nahezu) gleich. Dadurch, dass die Variationen des ersten Integrals sehr begrenzt sind, haben wir kleine Fehler. Auch der Aufwand steht in einem guten Verhältnis zum Fehler und ist bei allen drei Verfahren (bis auf die Konstante) gleich.

Bei der zweiten Integration fallen die starken Variationen bzw. die geringe Glattheit eine extreme Rolle bei den entstehenden Fehlern. Je nach Anzahl der Teilintervalle vari-

iert die Genauigkeit der numerischen Integration sehr stark, erreicht aber bei allen drei Formeln bei ungefähr 10^2 Teilintervallen ein relativ gutes Ergebnis. Dabei ist aber auch zu erkennen, dass der Aufwand nicht sehr gleichmäßig auf die Fehler aufgeteilt ist. Hier ähneln sich die drei Formeln wieder bis auf Konstanten.

Zusammenfassend kann man sagen, dass bei einer sehr sprunghaften Funktion (wie der zweiten) bei äquidistanten Stützstellen ein großer Fehler auftreten kann, falls diese Stützstellen ungünstig liegen.