Matlab-Funktionen zum Stoff des 2. Semesters Mathematik bei ET

Voraussetzungen: MATLAB mit Symbolic Math Toolbox

Erhältlich als Studentenversion mit den Bestandteilen:

MATLAB
Simulink
Control System Toolbox
Signal Processing Toolbox
Signal Processing Blockset
Statistics Toolbox
Optimization Toolbox
Image Processing Toolbox
Symbolic Math Toolbox.

1. Anwendungen der Integration

1.1. Analytisch

Voraussetzung: syms var
int(f)
int(f,var) Integration der Funktion f bezüglich der Variable var
int(f,var,a,b) Integration der Funktion f bezüglich der Variable var in den Grenzen von a bis b

Bei Nichtberechenbarkeit des Integrals kommt als Matlab-Antwort die Eingabe zurück oder ein Term mit der Fehlerfunktion erf oder anderen nicht berechenbaren Termen

```
syms x
int(sin(x))
        ans =-cos(x)
int (sin(x),0,pi)
        ans =2

syms a b t
 g=cos(a*t+b);
 int(g,t)
        ans=1/a*sin(a*t+b)

int(1./(x.^3-2*x-5))
        symsum(b*log(b*(3*x - b*(12*x + 45))), b in RootOf(c^3 + (6*c)/643 - 1/643, c))
int(1./(x.^3-2*x-5),x,10,20)
        ans =log(120/(643*(717788808900^(1/2)/1063390828 -…
       double(ans)=0.0038
```

1.2. Numerisch

Nur für bestimmte Integrale!! Zwei Proceduren: quad und quadl sind möglich

quad(f,a,b,tol): adaptive Simpsonmethode zur Integration der Funktion f in den Grenzen von a bis b, bei wahlfrei angebbarer Genauigkeit tol.

quadl(f,a,b,tol) Quadraturformel vom Gauß-Lobatto-Typ für glatte Funktionen, Parameter wie oben, zusätzliche Zwischenwerte, Anzahl der Schritte... verfügbar, rechenaufwendiger als quad.

Der Funktionsname muss als Zeichenkette oder mit führendem @ angegeben werden, damit die Funktion im aktuellen Directory als m-File gefunden wird.
Bei Fehlen von tol wird standardmäßig tol = 1e-6 angenommen

```
function y=f3(x)
y=1./(x.^3-2*x-5);

function y=f1(x);
y=0.5-x-0.2*sin(x);
```

```
quad(@f1,0,2)
        ans =   -1.2832
quad('f1',0,2)
        ans =   -1.2832
quad(@f3,10,20)
        ans =    0.0038
```

2. Zahlenreihen und Konvergenzbereiche von Potenzreihen

In einfachen Fällen liefert die symbolische Summation symsum(f, k, a, e) die Summe einer Reihe, wobei f das allgemeine Glied der Reihe und k der Summationsindex, der von a bis e läuft, sind.

```
syms x k
s1 = symsum(1/k^2,1,inf)
s2 = symsum(x^k,k,0,inf)

        s1 = 1/6*pi^2
        s2 = piecewise([1 <= x, Inf], [abs(x) < 1, -1/(x - 1)])
```

Weiterhin kann Matlab als Rechenhilfe eingesetzt werden, z.B. zur Vereinfachung von Termen und zur Grenzwertbildung.

Beispiel: Bestimmen Sie den Konvergenzbereich der Reihe $\sum_{k=1}^{\infty} \frac{x^k}{3^k (k+1)}$

```
syms l k
l=x^k/3^k/(k+1)
l1=subs(l,k,k+1)/l
l2=simplify(l1)
limit(l2,k,inf)
```

(limit(l^(1/k),k,inf) kann von Matlab nicht vereinfacht werden! ➔ Quotientenkriterium verwenden)

```
symsum(x^k/3^k/(k+1),k,1,inf)
        piecewise([3 <= x, Inf], [abs(x) <= 3 and x ~= 3, - (3*log(1 - x/3))/x - 1])
```

Taylorreihen

taylor (f,v,a,'Order',n) erstellt die ersten n Terme der Taylorreihe der Funktion f an der Stelle a, bezüglich der Variablen v, d.h. eine Entwicklung bis zur Ordnung n-1. Standard : n=6, v=x

```
syms x; f=1/(5+4*cos(x));
T=taylor(f,'Order',8)
        T = (49*x^6)/131220 + (5*x^4)/1458 + (2*x^2)/81 + 1/9

syms t; f=1/(5+4*cos(t));
T=taylor(f,t,'Order',9)
        T = (443*t^8)/13226976 + (49*t^6)/131220 + (5*t^4)/1458 + (2*t^2)/81 + 1/9

T=taylor(exp(x),x,1)
        T = exp(1) + exp(1)*(x - 1) + (exp(1)*(x - 1)^2)/2 + (exp(1)*(x - 1)^3)/6 +
            + (exp(1)*(x - 1)^4)/24 + (exp(1)*(x - 1)^5)/120
          = (exp(1)*(x^5 + 10*x^3 + 20*x^2 + 45*x + 44))/120
```

3. Fourierreihen

Die Fourierreihenentwicklung kann mittels der Integration und der Summationsfunktion in MATLAB nachgestellt werden:

Bsp.: Fourierentwicklung von $f(x)=1+x$ für $0 \le x \le 2$, periodisch mit $T = 4$, gerade

```
syms x  y k
c0=2/4*int(1+x,0,2)              %=2
ck=2/4*int((1+x)*cos(k*2*pi*x/4),0,2); simplify(ck)
        ck = -(4*sin((pi*k)/2)^2 - 3*pi*k*sin(pi*k))/(pi^2*k^2)
           = -4/k^2/pi^2        für ungerade k
```

Die symbolische Summation symsum versagt jedoch meist im Fall der Fourierreihe, so dass die Reihe per Hand aufgeschrieben werden sollte, einschließlich der angebbaren Wertbelegungen, für z.B. cos(k*pi).

Auswertung:

Berechnung einiger Koeffizienten aus Wertbelegungen für ck:

```
for i=1:1:5;koeff(i)=subs(ck,k,i); end; koeff
koeffreell=2*koeff   %  Umrechnung für die reelle Reihendarstellung
```

Funktion zur wertmäßigen Berechnung der ersten ke Glieder der Reihe:

```
function y=fourier(x,ke)
k=1;y=2;
while  k<ke
% k ist Ende der Summation; hochsetzen, um Genauigkeit zu verbessern
   y=y-8/k/k/pi/pi*cos(1/2*pi*k*x);
   k=k+2;
end;

fourier(1.5,2), fourier(1.5,7), fourier(1.5,10), fourier(1.5,16),fourier(1.5,20)
```

Zeichnung des Anfangs der Reihe und der zu approximierenden Funktion

Funktion zum wertmäßigen Berechnen der gegebenen Funktion:

```
function y=gegkurv(x)
        while x<-2
                x=x+4;
        end
        while x>2
                x=x-4;
        end
        if x>=-2 & x<0
                y=1-x;
        else
                y=1+x;
        end
```

```
xa=linspace(-10,10,201);
        for i=1:201
        ya(i)=gegkurv(xa(i));
        end;
yb=fourier(xa,16);
plot(xa,ya,'r',xa,yb,'b')
```

4. Laplacetransformation

L=laplace(F)	: berechnet die Laplacetransformierte L(s) von F(t)
L=laplace(F,w,z)	: berechnet die Laplacetransformierte (L(z) von F(w)
F = ilaplace(L)	: berechnet die Originalfunktion F(t) zu L(s)
F = ilaplace(L,y,x)	: berechnet die Originalfunktion F(x) zu L(y)

Die diskrete Version zur Laplacetransformation ist die z-Transformation, zu finden unter ztrans bzw. iztrans mit analogen Parametern.

```
syms x u; syms a real;
g=1/(u^2-a^2);
ilaplace(g,x)
simplify(ans)
```

```
syms p t
g=17*p/(7*p^2-4);F=ilaplace(g)
        F=(17*cosh((2*7^(1/2)*t)/7))/7
         ( =17/14*exp(-2/7*7^(1/2)*t)+17/14*exp(2/7*7^(1/2)*t))
```

5. Fouriertransformation

F = fourier(f)	: berechnet die Fouriertransformierte F(w) der Funktion f(x)
F = fourier(f,u,v)	: berechnet die Fouriertransformierte F(v) der Funktion f(u)

```
syms x w u;
```

```
f=exp(-x^2);g=exp(-w^2);
fourier(f)
fourier(g)
fourier(g,u)
```

Bei stückweise definierten Funktionen über die direkte Berechnung des Integrals gehen!

Die numerische Realisierung der Zerlegung eines Signals führt bei Diskretisierung des Signals bzw. numerischer Integration der Fourierintegrale zu ein und demselben Algorithmus, der schnellen Fouriertransformation (FFT). Die Fourierentwicklung wird im Funktionensystem der komplexwertigen e-Funktionen durchgeführt. Damit ist das Ergebnis, die Fourierkoeffizienten, ebenfalls komplex (c_k). Die FFT ist in Matlab mit unterschiedlichen Vorfaktoren realisiert.
Wichtig: Anzahl der Stützstellen sollte eine Zweierpotenz sein!

Syntax: fft(vector,n), ifft(vector,n), n: Anzahl der in Betracht gezogenen Punkte, mehrere Parameter sind möglich

Beispiel: Signalsynthese und –analyse

1. Kreation eines Beispielsignals:

```
t = 0:.001:.255;
x = cos(2*pi*30*t) + sin(2*pi*100*t);
plot(x(1:256))
y = x + 2*randn(size(t));
plot(y(1:256))
title('Verrauschtes Ausgangssignal')
```

2. Signalanalyse

```
Y = fft(y,256);
Pyy = Y.*conj(Y);
% Beträge der Koeffizienten/Energieäqivalent
f = 1000/256*(0:127);
plot(f,Pyy(1:128))
title('Spektraldichte')
xlabel('Frequenz (Hz)')
```

(3. Zoom)

```
plot(f(1:50),Pyy(1:50))
title('Spektraldichte')
xlabel('Frequenz (Hz)')
```

4. Synthese

```
z=ifft(Y);
t1=0:255;plot(t1,y,'r',t1,z,'b')
subplot(1,3,1),plot(t1,y),subplot(1,3,2),plot(t1,z),subplot(1,3,3),plot(t1,y-z)
```

5. Denoising

```
for k=1:256; if abs(Y(k))<65; Y(k)=0;end;abs(Y(k)), end  %70,75
z1=ifft(Y);
plot(t1,z1)
```

```
plot(t1,y,'r',t1,z,'b',t1,z1,'g')
%subplot(2,1,1),plot(t,x),subplot(2,1,2),plot(t,z1,'g')
%figure, plot(t,x),figure,plot(t,z1) %FourierEt2.m
```