Symboliska serier och derivator

1 Inledning

Vi har redan sett att verktygslådan Symbolic Math Toolbox i Matlab kan utföra symbolisk matematik inom analys i en variabel och inom linjär algebra. Nu skall vi se lite hur vi summerar serier och beräknar partiella derivator.

2 Summor och produkter

Vi kan beräkna summor som exempelvis $\sum_{k=1}^{n} k$ eller $\sum_{k=1}^{n} k^2$ med

```
>> syms k n
>> s=symsum(k,1,n)
s =
(n*(n + 1))/2

>> s=symsum(k^2,1,n)
s =
(n*(2*n + 1)*(n + 1))/6
```

Vi ser att

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2} \quad \text{och} \quad \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

En lite mer komplicerad summa som $\sum_{k=1}^{n} r^{k-1}$ beräknar vi med

```
>> syms k n r
>> s=symsum(r^(k-1),k,1,n)
s =
piecewise([r = 1, n], [r <> 1, (r^n - 1)/(r - 1)])
```

och med lite tolkning av utskriften ser vi att

$$\sum_{k=1}^{n} r^{k-1} = \begin{cases} \frac{r^n-1}{r-1}, & r \neq 1 \\ n, & r = 1 \end{cases}$$

Vi beräknar Leibniz formel

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

med

```
>> syms k
>> s=symsum((-1)^k/(2*k+1),k,0,Inf)
s =
pi/4
```

Produkter kan vi också bilda, t.ex. $\prod_{k=2}^{5} k^2$ och $\prod_{k=1}^{n} k$ ges av

```
>> syms k
>> p=symprod(k^2,2,5)
p =
14400

>> syms k n
>> p=symprod(k,1,n)
p =
factorial(n)
```

Vi beräknar Wallis formel

$$\prod_{k=1}^{\infty} \frac{2k}{2k-1}\,\frac{2k}{2k+1}$$

med

```
>> syms k
>> p=symprod(2*k/(2*k-1)*2*k/(2*k+1),k,1,Inf)
p =
pi/2
```

3 Taylorutveckling

Vi kan Taylorutveckla exempelvis $f(x) = \sqrt{x}$ runt $a = 4$, och ta med termer t.o.m. ordning $n = 3$, med funktionen `taylor` enligt

```
>> syms x a n
>> a=4; n=3;
>> t=taylor(sqrt(x),x,a,'Order',n+1)
t =
x/4 - (x - 4)^2/64 + (x - 4)^3/512 + 1
```

Lägg märke till att vi skall ge gradtalet plus 1 som parameter till `taylor`. Resttermen bestämmer vi med

```
>> syms xi
>> r=diff(sqrt(xi),n+1)/factorial(n+1)*(x-a)^(n+1)
r =
-(5*(x - 4)^4)/(128*xi^(7/2))
```

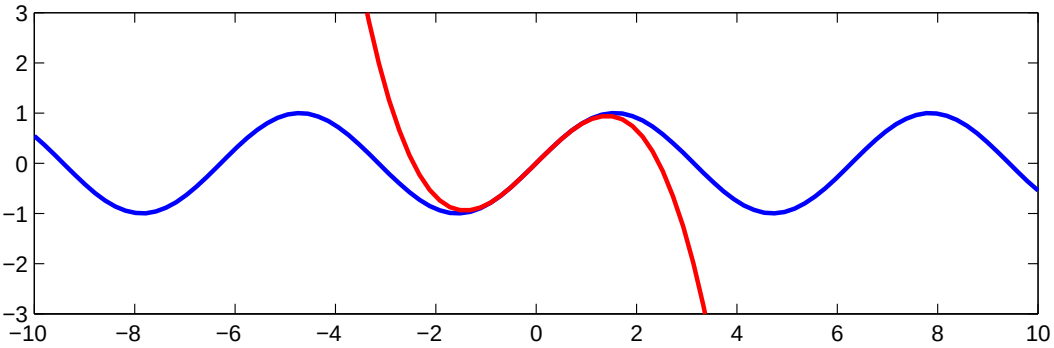
Vi får alltså $f(x) = 2 + \frac{x-4}{4} - \frac{(x-4)^2}{64} + \frac{(x-4)^3}{512} - \frac{5\xi^{-7/2}}{128}(x-4)^4$.

Vi kan Maclaurinutveckla $f(x) = \sin(x)$ med

```
>> syms x n
>> n=3;
>> f=sin(x);
>> t=taylor(f,'Order',n+1)
t =
- x^3/6 + x
```

och rita upp funktionen och Maclaurinpolynomet med

```
>> fn=matlabFunction(f);      % fn blir numerisk motsvarighet till f
>> tn=matlabFunction(t);      % tn blir numerisk motsvarighet till t
>> xn=linspace(-10,10);       % xn numeriska x-värden för uppritning
>> plot(xn,fn(xn),'b',xn,tn(xn),'r','linewidth',2)
>> axis equal, axis([xn(1) xn(end) -3 3])
```



Om du har tid: Bestäm Maclaurinpolynomen till $f(x) = \sin(x)$ av successivt allt högre gradtal ($n = 1, 3, \cdots, 21$ exempelvis). Rita successivt upp Maclaurinpolynomen i en figur där ni redan har ritat $f(x)$. Lägg en liten paus mellan varje uppritning.

4 Partiella derivator

Vi kan beräkna partiella derivator. Låt oss som exempel derivera $f(x,t) = \sin(xt^2)$ med avseende på t. Först definierar vi funktionen

```
>> syms x t
>> f=sin(x*t^2)
f =
sin(t^2*x)
```

sedan beräknar derivatorna med

```
>> dfdt=diff(f,t)
dfdt =
2*t*x*cos(t^2*x)

>> dfdx=diff(f,x)
dfdx =
t^2*cos(t^2*x)
```

Vi kan bilda Jacobimatrisen till

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1^3 + x_2^2 - 1 \\ \exp(x_1 x_2) + x_1 + x_2 - 2 \end{bmatrix}$$

enligt

```
>> syms x1 x2
>> f1=x1^3+x2^2-1;
>> f2=exp(x1*x2)+x1+x2-2;
>> f=[f1;f2]
f =
          x1^3 + x2^2 - 1
 x1 + x2 + exp(x1*x2) - 2

>> x=[x1;x2];
>> Df=jacobian(f,x)
Df =

[            3*x1^2,              2*x2]
[ x2*exp(x1*x2) + 1, x1*exp(x1*x2) + 1]
```

Givetvis fungerar det även med

```
>> Df=[diff(f1,x1) diff(f1,x2)
       diff(f2,x1) diff(f2,x2)]
```

som ger exakt samma svar.

Låt oss bilda gradienten till

$$f(\boldsymbol{x}) = 2x_1^3 - 3x_1^2 - 6x_1x_2(x_1 - x_2 - 1)$$

Vi definierar funktionen

```
>> syms x1 x2
>> f=2*x1^3-3*x1^2-6*x1*x2*(x1-x2-1)
```

och beräknar de partiella derivatorna

```
>> dfdx1=diff(f,x1)
dfdx1 =
6*x2*(x2 - x1 + 1) - 6*x1*x2 - 6*x1 + 6*x1^2
>> dfdx2=diff(f,x2)
dfdx2 =
6*x1*x2 + 6*x1*(x2 - x1 + 1)
```

som vi sedan sätter ihop till gradienten $\boldsymbol{\nabla} f(\boldsymbol{x})$ med

```
>> gradf=[dfdx1;dfdx2]
gradf =
 6*x2*(x2 - x1 + 1) - 6*x1*x2 - 6*x1 + 6*x1^2
                6*x1*x2 + 6*x1*(x2 - x1 + 1)
```

Alternativt använder vi `gradient` enligt

```
>> gradf=gradient(f,[x1 x2])
```

eller

```
>> x=[x1;x2];
>> gradf=gradient(f,x)
```

Hessematrisen bildar vi med `hessian` enligt

```
>> H=hessian(f,[x1 x2])
H =
[ 12*x1 - 12*x2 - 6, 12*x2 - 12*x1 + 6]
[ 12*x2 - 12*x1 + 6,              12*x1]
```

eller

```
>> H=hessian(f,x)
```

Vi passar på att nämna att man kan försöka lösa ickelinjära ekvationssystem med `solve`. Låt oss lösa systemet

$$\begin{cases} 2x_1 - x_1x_2 = 0 \\ x_2 + 0.4x_1x_2 - x_2^2 = 0 \end{cases}$$

Så här gör vi

```
>> syms x1 x2
>> f1=2*x1-x1*x2; f2=x2+0.4*x1*x2-x2^2
>> [x1s,x2s]=solve([f1==0,f2==0],[x1,x2])
x1s =
   0
   0
 5/2

x2s =
 0
 1
 2
```