lsg08

Aufgabe 1

```
reset()
v1 = vector([5,-3,4])
v2 = vector([7,6,8])
v3 = vector([9,2,-13])
```

```
norm( (v2-v1).cross_product(v3-v1) )/2
```

```
1/2*sqrt(33105)
```

```
n(norm( (v2-v1).cross_product(v3-v1) )/2)
```

```
90.9738973552304
```

```
polygon3d([v1,v2,v3])
```

Aufgabe 2

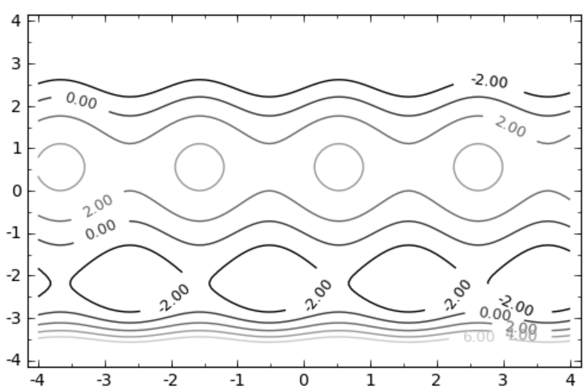
```
reset()
var('x,y')
```

```
(x, y)
```

```
f(x,y) = 0.1*y^4 - 0.3*y^3 - 1.5*y^2 + sin(3*x) + 1.9*y + 3
```

```
plot3d(f(x,y), (x,-4,4), (y,-4,4))
```

```
contour_plot(f(x,y), (x,-4,4), (y,-4,4), contours=[-2,0,2,4,6], fill=False, labels=True)
```



Aufgabe 3

```
reset();forget()
```

```
var('x')
f = 1/(1-x)
g = 1/(1+x+x^2+x^3+x^4+x^5)
```

```
t = 0
for n in [0..3]:
   t = t + diff(f,x,n)(x=3) / n.factorial() * (x-3)^n
t
```

```
1/16*(x - 3)^3 - 1/8*(x - 3)^2 + 1/4*x - 5/4
```

```
# oder:
sum([diff(f,x,n)(x=3)/n.factorial()*(x-3)^n for n in [0..3]])
```

```
1/16*(x - 3)^3 - 1/8*(x - 3)^2 + 1/4*x - 5/4
```

```
#Vorsicht! das geht nicht! Faustregel: diese sum nur bei symbolischen Ausdrücken benutzen
```

```
_=var('n');sum( (diff(f,x,n))(x=3)/n.factorial()*(x-3)^n,n,0,3)
```

```
0
```

```
taylor(f,x,3,3)
```

```
1/16*(x - 3)^3 - 1/8*(x - 3)^2 + 1/4*x - 5/4
```

```
sum([diff(g,x,n)(x=3)/n.factorial()*(x-3)^n for n in [0..3]])
```

```
-44310267/17555190016*(x - 3)^3 + 177633/48228544*(x - 3)^2 -
547/132496*x + 2005/132496
```

```
taylor(g,x,3,3)
```

```
-44310267/17555190016*(x - 3)^3 + 177633/48228544*(x - 3)^2 -
547/132496*x + 2005/132496
```

Aufgabe 4

```
reset();forget()
var('x,n')
f(x) = x*exp(-n*x)
```

```
solve(diff(f,x)==0, x)
```

```
[x == (1/n)]
```

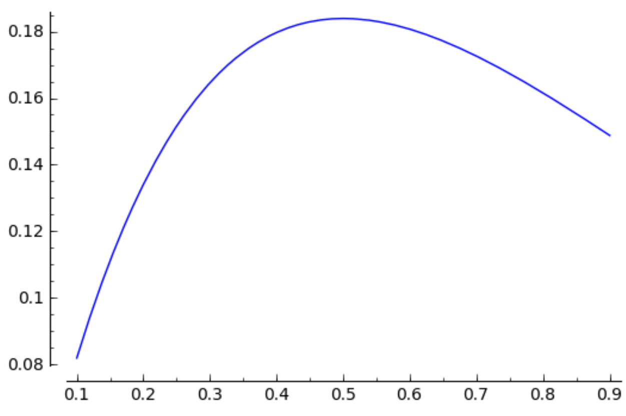
```
forget()
assume(n>0)
print diff(f,x,2)(1/n)
print bool(diff(f,x,2)(1/n) < 0)
```

```
-n*e^(-1)
True
```

```
f(0), f(1/n), f.limit(x=oo)
```

```
(0, e^(-1)/n, x |--> 0)
```

```
plot(f(x,n=2),(x,0.1,0.9))
```



Aufgabe 5

```
reset()
var('x')
```

```
x
```

```
(cos(x)).limit(x=pi/2), (pi/2-x).limit(x=pi/2)
```

```
(0, 0)
```

```
( diff(cos(x),x) / diff(pi/2-x,x) ).limit(x=pi/2)
```

```
1
```

```
(sin(x)-x).limit(x=0), (x*sin(x)).limit(x=0)
```

    (0, 0)

```
( diff(sin(x)-x,x) / diff(x*sin(x),x) ).limit(x=0)
```

    0

Aufgabe 6

```
reset()
var('x')
```

    x

```
limit(sqrt(x), x=1, dir='plus'), limit(x^3+2.5*x-2, x=1, dir='minus')
```

    (1, 1.5)

```
limit(x^2, x=0, dir='plus'), limit(-x, x=0, dir='minus')
```

    (0, 0)

```
limit(diff(x^2,x), x=0, dir='plus'), limit(diff(-x,x), x=0, dir='minus')
```

    (0, -1)

Aufgabe 7

```
reset()
var('x')
```

    x

```
show( diff(exp(sin(2*x)/sin(x)/( (ln(8)/(2+2*cos(x)))^(-1) - 8/ln(x^4) )), x) )
```

$$\frac{1}{2}\left(\frac{\left(\frac{\sin(x)}{\log(8)} - 16\,\frac{1}{x\log\left(x^{4}\right)^{2}}\right)\sin\left(2\,x\right)}{\left(\frac{(\cos(x)+1)}{\log(8)} - 4\,\frac{1}{\log\left(x^{4}\right)}\right)^{2}\sin\left(x\right)} - \frac{\sin\left(2\,x\right)\cos\left(x\right)}{\left(\frac{(\cos(x)+1)}{\log(8)} - 4\,\frac{1}{\log\left(x^{4}\right)}\right)\sin\left(x\right)^{2}} + 2\,\frac{\cos\left(2\,x\right)}{\left(\frac{(\cos(x)+1)}{\log(8)} - 4\,\frac{1}{\log\left(x^{4}\right)}\right)\sin\left(x\right)}\right)e^{\left(\frac{1}{2}\ \frac{\sin(2\,x)}{\left(\frac{(\cos(x)+1)}{\log(8)} - 4\ \frac{}{\log}\right.}\right.}$$

```
show( diff(sin(2^(cos(3*x))), x) )
```

$$-3\,2^{\cos(3\,x)}\log\left(2\right)\sin\left(3\,x\right)\cos\left(2^{\cos(3\,x)}\right)$$

```
4*integral(1/(1+x^2), x,0,1)
```

    pi

```
integral(sin(x)*exp(x), x,0,pi)
```

    1/2*e^pi + 1/2

```
integral(1/(exp(x)+exp(-x)), x)
```

    arctan(e^x)

```
diff(sin(2*x)*exp(x), x,20)
```

    -9653287*e^x*sin(2*x) - 1476984*e^x*cos(2*x)

Aufgabe 8

```
reset()
var('x')
```

    x

```
def nullstelle(f, a0, b0, TOL):
   k = 0
   ak = a0
   bk = b0
```

```
    # Die Bedingung abs(b0-a0) > 2^(k+1)*TOL ist äquivalent
    # zu abs(bk-ak) > 2*TOL

    while abs(b0-a0) > 2^(k+1)*TOL:
        if f(ak)*f((ak+bk)/2) < 0:
            bk = (ak+bk)/2
        else:
            ak = (ak+bk)/2
        k += 1
        if k>1000:
            print "Maximale Anzahl an Iterationen erreicht."
            return False

    # Wenn die Schleife abbricht ist die Nullstelle weniger
    # als TOL vom Intervall-Mittelpunkt entfernt.
    return (ak+bk)/2
```

```
f(x) = x^3 - 2
n(2^(1/3))
```

```
1.25992104989487
```

```
n(nullstelle(f, 0,3, 10^(-4)))
```

```
1.25985717773438
```

```
n(nullstelle(f, 0,3, 10^(-6)) )
```

```
1.25992083549500
```

```
# Alternative - Rekursive Funktion
def nullstelle_rek(f, a, b, TOL, k=0):
    if k>=1000:
        return False
    if abs(b-a) > 2*TOL:
        if f(a)*f((a+b)/2) < 0:
            return nullstelle_rek(f,a,(a+b)/2,TOL,k+1)
        else:
            return nullstelle_rek(f,(a+b)/2,b,TOL,k+1)
    return (a+b)/2
```

```
n(nullstelle_rek(f, 0,3, 10^(-4)))
```

```
1.25985717773438
```