Table of Contents

1(a) 微分

In [18]:
```
from sympy import *
init_session()

x,y = symbols('x y')
```

```
IPython console for SymPy 1.0 (Python 3.6.1-64-bit) (ground types: python)

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at http://docs.sympy.org/1.0/
```

In [19]:
```
simplify(diff(acos((4+5*cos(x))/(5+4*cos(x))),x))
```

Out[19]:

$$\frac{3 \sin(x)}{\sqrt{\frac{\sin^2(x)}{(4\cos(x)+5)^2}}(4\cos(x)+5)^2}$$

1(b) パラメータ微分

In [20]:
```
from sympy import *
init_session()

x,y, t = symbols('x y t')
```

```
IPython console for SymPy 1.0 (Python 3.6.1-64-bit) (ground types: python)

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at http://docs.sympy.org/1.0/
```

In [21]:
```
x = 3*t/(1+t**3)
x
```

Out[21]:

$$\frac{3t}{t^3+1}$$

In [22]:
```
y = 3*t**2/(1+t**3)
y
```

Out[22]:

$$\frac{3t^2}{t^3+1}$$

In [23]:
```
simplify(diff(y,t)/diff(x,t))
```

Out[23]:

$$\frac{t\left(t^3-2\right)}{2t^3-1}$$

2(a)

In [25]:
```
from sympy import *
init_session()

x,y, t = symbols('x y t')
```

```
IPython console for SymPy 1.0 (Python 3.6.1-64-bit) (ground types:
python)

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at http://docs.sympy.org/1.0/
```

Out[25]:
$$\frac{-2\cos(x)+1}{-4\cos(x)+5}$$

In [29]:
```
f = (1-2*cos(x))/(5-4*cos(x))
```

In [45]:
```
integrate(f,x)
```

Out[45]:
$$\frac{x}{2}+\frac{i}{2}\log\left(\tan\left(\frac{x}{2}\right)-\frac{i}{3}\right)-\frac{i}{2}\log\left(\tan\left(\frac{x}{2}\right)+\frac{i}{3}\right)$$

複素数が出たままで気持ち悪いんですが，これも正解なようです．ここらがsympyの限界なんで．．．

2(b)はパスね．

3(a)

In [47]:
```
from sympy import *
init_session()
```

```
IPython console for SymPy 1.0 (Python 3.6.1-64-bit) (ground types:
python)

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at http://docs.sympy.org/1.0/
```

In [50]:
```
A = Matrix([[2,1,1],[1,2,1],[0,0,1]])
A
```

Out[50]:
$$\begin{bmatrix}2 & 1 & 1\\1 & 2 & 1\\0 & 0 & 1\end{bmatrix}$$

In [62]:
```
P, D = A.diagonalize()
P
```

Out[62]:
$$\begin{bmatrix}-1 & -1 & 1\\1 & 0 & 1\\0 & 1 & 0\end{bmatrix}$$

In [61]:
```
P.inv()*A*P
```

Out[61]:
$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 3\end{bmatrix}$$

3(b)

In [64]:
```
from sympy import *
init_session()
a,b,c = symbols('a b c')
```

```
IPython console for SymPy 1.0 (Python 3.6.1-64-bit) (ground types:
python)

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at http://docs.sympy.org/1.0/
```

In [66]:
```
A=Matrix([[0,c,b],[c,0,a],[b,a,0]])
A
```

Out[66]:
$$\begin{bmatrix} 0 & c & b \\ c & 0 & a \\ b & a & 0 \end{bmatrix}$$

In [67]:
```
B=Matrix([[-1,1,1],[1,-1,1],[1,1,-1]])
B
```

Out[67]:
$$\begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

In [70]:
```
det(A*B)
```

Out[70]: $8abc$

4 センター試験原本

In [89]:
```
from sympy import *
init_session()
a,b,c,x = symbols('a b c x')
```


```
IPython console for SymPy 1.0 (Python 3.6.1-64-bit) (ground types:
python)

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at http://docs.sympy.org/1.0/
```


In [90]:
```
f = a*x**2+b*x+c
```

In [91]:
```
eq1=f.subs({x:-1})
eq1
```

Out[91]: $a - b + c$

In [92]:
```
eq2=f.subs({x:2})
eq2
```

Out[92]: $4a + 2b + c$

In [109]:
```
s1=solve({eq1-4,eq2-7},{b,c})
s1
```

Out[109]: $\{b : -a + 1, \quad c : -2a + 5\}$

In [114]:
```
p = solve(diff(f,x),x)[0].subs(s1)
p
```

Out[114]:
$$-\frac{-a+1}{2a}$$

In [129]:
```
f.subs({x:p}).subs(s1)
```

Out[129]:
$$-2a + 5 - \frac{(-a+1)^2}{4a}$$

In [132]:
```
qq = simplify(f.subs({x:p}).subs(s1)*4*a)
qq
```

Out[132]: $-9a^2 + 22a - 1$

In [145]:
```
q = qq/4/a
```

4(a) a=2の時の頂点移動，セソタチツテ

In [138]:
```
-p.subs({a:2})
```

Out[138]: $-\frac{1}{4}$

In [146]:
```
-q.subs({a:2})
```

Out[146]: $-\frac{7}{8}$

4(b) ト

In [141]:
```
solve(p,a)
```

Out[141]: [1]

In [147]:
```
q.subs({a:1})
```

Out[147]: 3

4(c) 最小値　ナニヌネノ

In [148]:
```
solve(q, a)
```

Out[148]: $\left[-\frac{4\sqrt{7}}{9} + \frac{11}{9}, \quad \frac{4\sqrt{7}}{9} + \frac{11}{9} \right]$

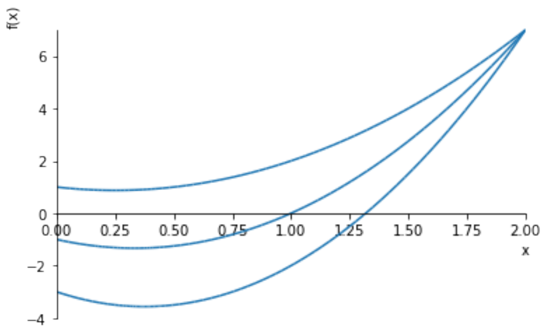
In [169]:
```
f.subs(s1).subs({x:1})
```

Out[169]: $-2a + 6$

In [170]:
```
solve(f.subs(s1).subs({x:1}),a)
```

Out[170]: [3]

In [171]:
```
%matplotlib inline
plot(f.subs(s1).subs({a:2}),
     f.subs(s1).subs({a:3}),
     f.subs(s1).subs({a:4}),
     (x,0,2))
```



Out[171]: <sympy.plotting.plot.Plot at 0x11af5e048>

5

In [172]:
```
from sympy import *
init_session()
a,b,c,x = symbols('a b c x')
```

```
IPython console for SymPy 1.0 (Python 3.6.1-64-bit) (ground types:
python)

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at http://docs.sympy.org/1.0/
```

In [173]:
```
f = a*x**2+b*x+c
```

In [174]:
```
eq1=f.subs({x:-1})
eq1
```

Out[174]: $a - b + c$

In [175]:
```
eq2=f.subs({x:2})
eq2
```
Out[175]: $4a + 2b + c$

In [176]:
```
s1=solve({eq1-4,eq2-6.5},{b,c})
s1
```
Out[176]: $\{b : -a + 0.833333333333333, \quad c : -2.0a + 4.83333333333333\}$

In [177]:
```
p = solve(diff(f,x),x)[0].subs(s1)
p
```
Out[177]: $-\frac{-a + 0.833333333333333}{2a}$

In [179]:
```
q = f.subs({x:p}).subs(s1)
q
```
Out[179]: $-2.0a + 4.83333333333333 - \frac{(-a + 0.833333333333333)^2}{4a}$

In [180]:
```
-p.subs({a:2})
```
Out[180]: -0.291666666666667

In [181]:
```
-q.subs({a:2})
```
Out[181]: -0.663194444444444

In [183]:
```
aa= solve(p,a)
aa
```
Out[183]: $[0.833333333333333]$

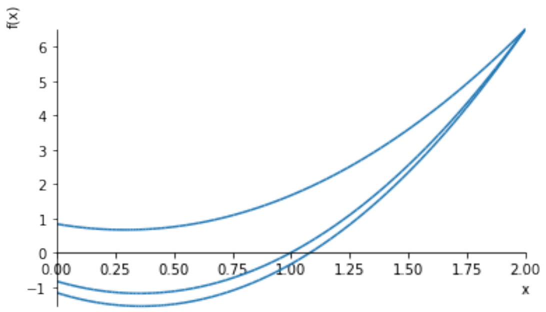
In [185]:
```
q.subs({a:aa[0]})
```
Out[185]: 3.16666666666667

In [186]:
```
solve(q, a)
```
Out[186]: $[0.0335512192016034, \quad 2.29978211413173]$

In [187]:
```
f.subs(s1).subs({x:1})
```
Out[187]: $-2.0a + 5.66666666666667$

In [188]:
```
solve(f.subs(s1).subs({x:1}),a)
```
Out[188]: $[2.83333333333333]$

In [190]:
```
%matplotlib inline
plot(f.subs(s1).subs({a:2}),
     f.subs(s1).subs({a:2.83333333}),
     f.subs(s1).subs({a:3}),
     (x,0,2))
```



Out[190]: <sympy.plotting.plot.Plot at 0x11b26dc18>

In [ ]: