Table of Contents

2019ペア試験解答例

微積分

偏微分，saddle point

関数$f(x,y)=x^2-y^2$の2次偏導関数$f_{xx},f_{xy},f_{yx},f_{yy}$を求めよ． また，$(x,y)=(0,0)$での判別式

$$D=f_{xy}(0,0)^2-f_{xx}(0,0)f_{yy}(0,0)>0$$

を確かめよ．さらに，関数$f(x,y)$をplot3dして鞍点(saddle point)の意味を確認せよ． (15点)

In [13]:
```
from sympy import *
x,y = symbols('x y')

fx = diff(x**2-y**2,x)
fx
```

Out[13]: 2*x

In [18]:
```
fxx = diff(fx,x)
fxy = diff(fx,y)
print(fxx, ',', fxy)
```

2 , 0

In [6]:
```
fy = diff(x**2-y**2,y)
fy
```

Out[6]: -2*y

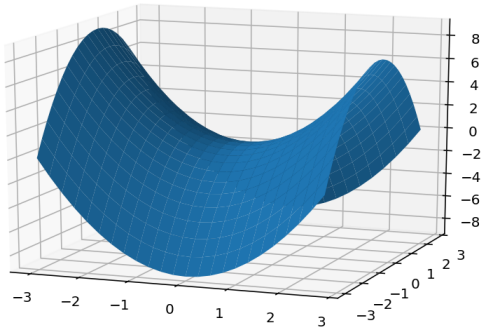
In [19]:
```
fyx = diff(fy,x)
fyy = diff(fy,y)
print(fyx, ',', fyy)
```

0 , -2

In [21]:
```
D = fxy**2 - fxx*fyy
print(D)
```

4

In [11]:
```
%matplotlib notebook
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np

def f(x,y):
    return x**2-y**2 #4*x+2*y-6*x*y

x = np.arange(-3, 3, 0.25)
y = np.arange(-3, 3, 0.25)
X, Y = np.meshgrid(x, y)
Z1 = f(X,Y)

fig = plt.figure()
plot3d = Axes3D(fig)
plot3d.plot_surface(X,Y,Z1)

plt.show()
```



Dが正であることが確認できる．plot3dによって3次元表示すると，鞍のような表面となり，極値でないことが確認できる．すなわち，fxxの曲率が正，fyyの曲率は負であり，2方向によって曲率が違い鞍点(saddle point)となる．

(フーリエ積分)

関数$f(x) = \sin(x)\sin(2x)$の不定積分を求めよ．$f(x)$を$x = -\pi..\pi$でプロットし，この区間での積分値を求めよ．結果についてコメントせよ． (15点)

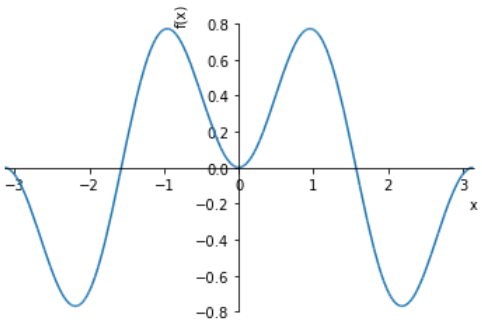
In [33]:
```
# kernel->Restart
from sympy import *
init_session()
```

```
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 [38]:
```
integrate(sin(x)*sin(2*x), x)
```

Out[38]: $$-\frac{2}{3}\sin(x)\cos(2x) + \frac{1}{3}\sin(2x)\cos(x)$$

In [39]:
```
%matplotlib inline


x = Symbol('x')
plot(sin(x)*sin(2*x), (x, -pi, pi))
```



Out[39]: <sympy.plotting.plot.Plot at 0x114866ba8>

In [40]:
```
integrate(sin(x)*sin(2*x), (x, -pi, pi))
```

Out[40]: 0

不定積分の結果は，ややこしいが，プロットからわかるとおり，関数は正の領域と負の領域が対称に出て来ている．したがって，積分すれば正負で相殺して0となる．

線形代数

ヌルスペース

行列$A = \begin{pmatrix} 4 & -1 & -1 & 1 \\ 1 & 2 & -1 & -2 \end{pmatrix}$を表現行列とする$R^4 \to R^2$の線形写像$f$のKer($f$)の1組の基底を求めよ．(15点)

In [42]:
```
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 [45]:
```
A=Matrix([[4,-1,1,1],[1,2,-1,-2]])
A.nullspace()
```

Out[45]:
$$\left[\begin{bmatrix} -\frac{1}{9} \\ \frac{5}{9} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}\right]$$

対角化

行列$A = \begin{pmatrix} 4 & -1 & -1 \\ 1 & 2 & -1 \\ 3 & -1 & 0 \end{pmatrix}$の固有値と固有ベクトルを求めよ． また，対角化行列Pを求めて， $P^{-1}AP$とP^tAPを求め，違いを確かめよ．(15点)

In [46]:
```
A = Matrix([[4,-1,-1],[1,2,-1],[3,-1,0]])
A.eigenvects()
```

Out[46]:
$$\left[\left(1, \quad 1, \quad \left[\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{bmatrix}\right]\right), \left(2, \quad 1, \quad \left[\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right]\right), \left(3, \quad 1, \quad \left[\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}\right]\right)\right]$$

In [47]:
```
P,D = A.diagonalize()
```

In [48]:
```
P.inv()*A*P
```

Out[48]:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

In [50]:
```
P.transpose()*A*P
```

Out[50]:
$$\begin{bmatrix} 6 & 8 & 9 \\ 4 & 6 & 6 \\ 3 & 4 & 6 \end{bmatrix}$$

2015年度大学入試センター試験 本試験 数学II・B第2問

以下のセンター試験問題をpythonでcode化せよ．ただし，関数$f(x)$は

```
f =Rational(1,2)*x**2
f.subs({x:a})
```

などとするべし．

微分係数

関数$f(x)=\frac{1}{2}x^2$の$x=a$における微分係数$f'(a)$を 求めよう．hが0でないとき， xがaから$a+h$まで変化するときの$f(x)$の 平均変化率は $\boxed{ア}+\frac{h}{\boxed{イ}}$ である． したがって，求める微分係数は

$$f'(a)=\lim_{h\to\boxed{ウ}}\left(\boxed{ア}+\frac{h}{\boxed{イ}}\right)=\boxed{エ}$$

である．

In [1]:
```
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 [2]:
```
a,b,h,x = symbols('a b h x')
f =Rational(1,2)*x**2
f
```

Out[2]: $\frac{x^2}{2}$

In [3]:
```
f.subs({x:a+h})
```

Out[3]: $\frac{1}{2}(a+h)^2$

In [4]:
```
f.subs({x:a+h}) - f.subs({x:a})
```

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

In [5]:
```
(f.subs({x:a+h}) - f.subs({x:a}))/h
```

Out[5]: $\frac{1}{h}\left(-\frac{a^2}{2}+\frac{1}{2}(a+h)^2\right)$

In [6]:
```
expand( (f.subs({x:a+h}) - f.subs({x:a}))/h )
```

Out[6]: $a+\frac{h}{2}$

In [7]:
```
limit(expand( (f.subs({x:a+h}) - f.subs({x:a}))/h ), h, 0)
```

Out[7]: a

少し冗長ですが，ゆっくりと数式処理のステップを示しています．

- 1/2だと0.5になるので明示的に有理数(Rational)と宣言
- 関数の代入はsubs(tituion) で実行
- a+hとaとの差をとる
- さらに展開(expand)を実行
- limitを取る．これはテキストでは微分のsectionで紹介している

放物線，接線

放物線$y = \frac{1}{2}x^2$をCとし，C上に点P$\left(a, \frac{1}{2}a^2\right)$をとる． ただし，$a > 0$とする．点Pにおける$C$の接線$l$の方程式は

$$y = \boxed{オ}x - \frac{1}{\boxed{カ}}a^2$$

である． 直線lとx軸との交点Qの座標は$\left(\frac{\boxed{キ}}{\boxed{ク}}, 0\right)$である． 点Qを通り$l$に垂直な直線を$m$とすると，

mの方程式は

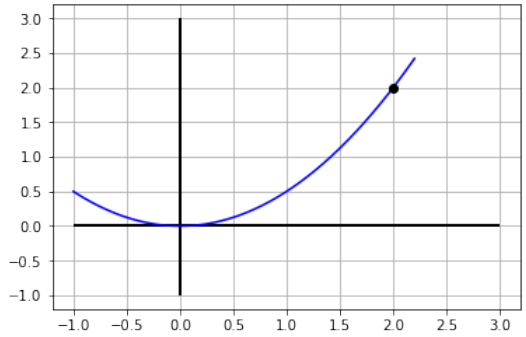
$$y = \frac{\boxed{ケコ}}{\boxed{サ}}x + \frac{\boxed{シ}}{\boxed{ス}}$$

である．

プロット

まずは関数をplotしてさらに，aを適当に仮定して図を書いています． こいつが面倒なんですが，pythonを許してあげてください． コードを覚え必要はなくて，ここにサンプルがあることを覚えておいてください．

In [8]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

from sympy import *

def func(x):
  return 1/2*x**2

xx = np.linspace(-1, 2.2, 100)  #0から2πまでの範囲を100分割した
yy = func(xx)
plt.plot(xx, yy, color = 'b')

plt.plot(2, func(2), "o", color = 'k')
plt.hlines(0, -1, 3, color='k', linestyle='-', linewidth=2)
plt.vlines(0, -1, 3, color='k', linestyle='-', linewidth=2)
plt.grid()
plt.show()
```



In [9]:
```
ll = aa*(x-x0)+y0
```

```
---------------------------------------------------------------------
---------
NameError                                 Traceback (most recent c
all last)
<ipython-input-9-c204fe214fb7> in <module>
----> 1 ll = aa*(x-x0)+y0

NameError: name 'aa' is not defined
```

In [10]:
```
diff(f,x)
```

Out[10]: x

In [11]:
```
diff(f,x).subs({x:a})
```

Out[11]: a

In [12]:
```
aa = diff(f,x).subs({x:a})
```

In [13]:
```
x0 = a
y0 = Rational(1,2)*a**2
ll = aa*(x-x0)+y0
expand(ll)
```

Out[13]: $-\frac{a^2}{2} + ax$

In [14]:
```
solve(ll,x)
```

Out[14]: $\left[\frac{a}{2}\right]$

In [15]:
```
mm = bb*(x-x0)+y0
```

```
---------------------------------------------------------------------
---------
NameError                                 Traceback (most recent c
all last)
<ipython-input-15-871c2a4e54a6> in <module>
----> 1 mm = bb*(x-x0)+y0

NameError: name 'bb' is not defined
```

In [16]:
```
bb = solve(a*b+1,b)[0]
bb
```

Out[16]: $-\frac{1}{a}$

In [17]:
```
mm = bb*(x-a/2)
expand(mm)
```

Out[17]: $\frac{1}{2} - \frac{x}{a}$

In [18]:
```
print(mm.subs({a:2}))
ll.subs({a:2})
```

```
-x/2 + 1/2
```

Out[18]: $2x - 2$

In [36]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

from sympy import *

def func(x):
  return 1/2*x**2

def l_func(x):
  return 2*x-2

def m_func(x):
  return -x/2+1/2


g = plt.subplot()
g.set_ylim([-3,3])
g.set_xlim([-3,3])
g.set_aspect('equal')


xx = np.linspace(-1, 2.2, 100)  #0から2πまでの範囲を100分割した
yy = func(xx)
g.plot(xx, yy, color = 'b')
yy = l_func(xx)
g.plot(xx, yy, color = 'r')
yy = m_func(xx)
g.plot(xx, yy, color = 'r')

plt.plot(2, func(2), "o", color = 'k')
plt.text(2+0.5, func(2), "P(a,1/2a^2)", color = 'k')
plt.plot(1,0, "o", color = 'k')
plt.text(1.2, 0.2, "Q(a/2, 0)", color = 'k')
plt.plot(0,1/2, "o", color = 'k')
plt.text(0.2, 1/2+0.2, "A(0, 1/2)", color = 'k')
plt.plot(0,0, "o", color = 'k')
plt.text(0.2, -0.3, "O", color = 'k')
plt.plot(2,0, "o", color = 'k')
plt.text(2.2,-0.3, "R", color = 'k')


plt.hlines(0, -1, 3, color='k', linestyle='-', linewidth=2)
plt.vlines(0, -1, 3, color='k', linestyle='-', linewidth=2)
plt.grid()
plt.show()
```

aspect比が1のplotは少しコツがいりそう．上の例を参照してください． subplotを使わないとエラーが出たんですが，よくわかりません．

実はここで間違いをしていました．曲線mの傾きを求める式が間違ってたんですが， aspect比が1のplotをして，間違いに気がつきました． 視覚化はいつも大事ですね．

直線の方程式ですが，エラーを出しています． これはわざとです．直線の方程式の公式

$$y - f(a) = f'(a)(x - a)$$

をpythonでアレンジして

```
mm = bb*(x-x0)+y0
```

と書いておくと，どの値が足りないかがわかります． 傾き(bb)，x0, y0と値(あるいはパラメータ)を 指定していって直線が出来上がります．

グラフには以下の問題を解くために，点O,Rを足しています． また，後の計算で間違いが起こりにくいように，座標を書き加えています．

面積

直線mとy軸との交点をAとする．三角形APQの面積をSとおくと

$$S = \frac{a\left(a^2 + \boxed{セ}\right)}{\boxed{ソ}}$$

となる．また，y軸と線分APおよび曲線Cによって囲まれた図形の面積をTとおくと

$$T = \frac{a\left(a^2 + \boxed{タ}\right)}{\boxed{チツ}}$$

となる．

$a > 0$の範囲における$S - T$の値について調べよう．

$$S - T = \frac{a\left(a^2 - \boxed{テ}\right)}{\boxed{トナ}}$$

である． $a > 0$であるから，$S - T > 0$となるようなa のとり得る値の範囲は $a > \sqrt{\boxed{ニ}}$ である． また，$a > 0$のときの$S - T$の増減を調べると， $S - T$は$a = \boxed{ヌ}$ で最小値 $\frac{\boxed{ネノ}}{\boxed{ハヒ}}$ をとることがわかる．

In [20]:
```
mm.subs({x:0})
```

Out[20]: $\frac{1}{2}$

In [55]:
```
OAPR = (Rational(1,2)+f.subs({x:a}))*a/2
OAQ = Rational(1,2)*Rational(1,2)*a/2
QPR = (a-a/2)*1/2*a**2/2
OAPR
```

Out[55]: $\frac{a}{2}\left(\frac{a^2}{2} + \frac{1}{2}\right)$

In [57]:
```
SS = factor(OAPR - OAQ - QPR)
SS
```

Out[57]: $\frac{a}{8}\left(a^2 + 1\right)$

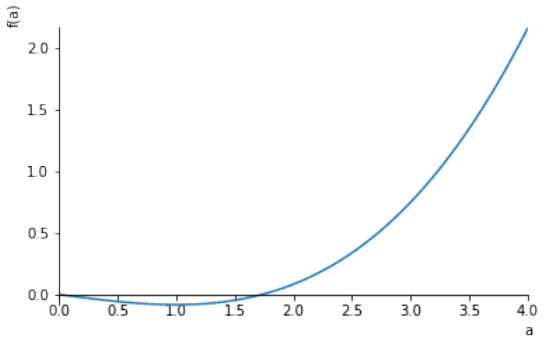
In [61]:
```
TT = factor(OAPR-integrate(f,(x,0,a)))
TT
```

Out[61]: $\frac{a}{12}\left(a^2 + 3\right)$

In [64]:
```
ST = factor(SS-TT)
ST
```

Out[64]: $\frac{a}{24}\left(a^2 - 3\right)$

In [65]:
```
plot(ST, (a,0,4))
```



Out[65]: <sympy.plotting.plot.Plot at 0x116b58fd0>

In [66]:
```
solve(ST,a)
```

Out[66]: $\left[0, \quad -\sqrt{3}, \quad \sqrt{3}\right]$

In [67]:
```
diff(ST,a)
```

Out[67]: $\frac{a^2}{8} - \frac{1}{8}$

In [68]:
```
solve(diff(ST,a),a)
```

Out[68]: $[-1, \quad 1]$

In [69]:
```
ST.subs({a:1})
```

Out[69]: $-\frac{1}{12}$

これでセンター試験の問題は解けました． pythonで解いたら綺麗にセンター試験の解答ボックスを 埋めていくことができるはずです． 出題者は検算をpythonにやらしているのではと訝るほどです． でも，次の問題をこのままでは解けません．

センター試験数値計算化

前問の関数を$f(x) = 0.49x^2$および放物線Cの方程式を$y = 0.49x^2$として問題を解け． 数値解となるので，答えはかっこによらず小数点となる．最後の最小値は-0.08676940ぐらい．(30点)

解法の指針 代数計算でできた問題を数値計算にするという作業はよくあります． 研究や実装においては日常的な作業です． コツは，できるだけ自動化しておくことです． 上の解法では，出て来た結果をそのままベタ打ちしてるところがあります． そうすると変数が変わった時に「もれ」が出て来ます． まずは答えを取り出す時に，手打ちではなく変数として取り出す工夫をします． 例えば，P点の座標を$(a, 1/2 * a^2)$ではなく$(a, f(a))$とするだけで大分自動化が進みます． 他のもやっていきます．

In [70]:
```
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 [99]:
```
a,b,h,x = symbols('a b h x')
#f =Rational(1,2)*x**2
f =0.49*x**2
f
```

Out[99]: $0.49x^2$

In [100]:
```
df = expand( (f.subs({x:a+h}) - f.subs({x:a}))/h )
df
```

Out[100]: $0.98a + 0.49h$

In [101]:
```
limit( df, h, 0)
```

Out[101]: $0.98a$

In [103]:
```
aa = diff(f,x).subs({x:a})
aa
```

Out[103]: $0.98a$

In [118]:
```
x0 = a
y0 = f.subs({x:a})
ll = aa*(x-x0)+y0
expand(ll)
```

Out[118]: $-0.49a^2 + 0.98ax$

In [119]:
```
Q_x = solve(ll,x)[0]
Q_x
```

Out[119]: $0.5a$

In [120]:
```
bb = solve(aa*b+1,b)[0]
bb
```

Out[120]: $-\frac{1.02040816326531}{a}$

In [121]:
```
mm = bb*(x-Q_x)
expand(mm)
```

Out[121]: $0.510204081632653 - \frac{1.02040816326531x}{a}$

In [122]:
```
A_y = mm.subs({x:0})
A_y
```

Out[122]: 0.510204081632653

In [123]:
```
OAPR = (A_y+f.subs({x:a}))*a/2
OAQ = A_y*Q_x/2
QPR = (a-Q_x)*f.subs({x:a})/2
OAPR
```

Out[123]: $\frac{a}{2}\left(0.49a^2 + 0.510204081632653\right)$

In [124]:
```
SS = factor(OAPR - OAQ - QPR)
SS
```

Out[124]: $0.5a\left(0.245a^2 + 0.255102040816327\right)$

In [125]:
```
TT = factor(OAPR-integrate(f,(x,0,a)))
TT
```

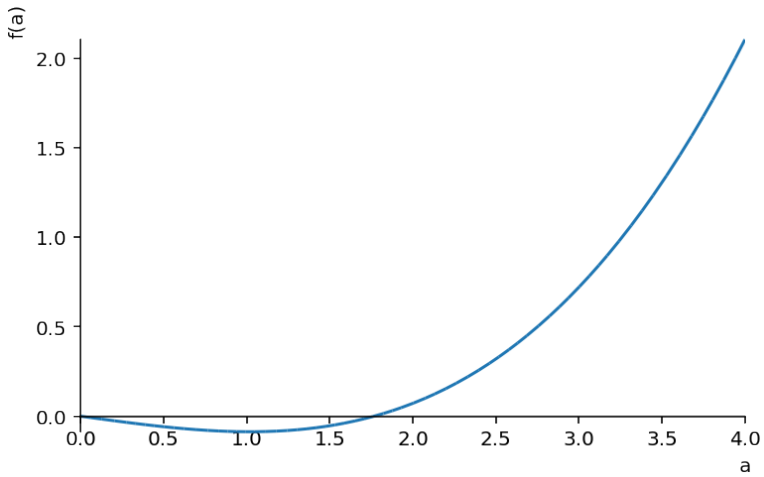
Out[125]: $0.5a\left(0.163333333333333a^2 + 0.510204081632653\right)$

In [126]:
```
ST = factor(SS-TT)
ST
```

Out[126]: $0.5a\left(0.0816666666666667a^2 - 0.255102040816327\right)$

In [127]:
```
plot(ST, (a,0,4))
```

Out[127]: <sympy.plotting.plot.Plot at 0x1169a3b00>



In [128]:
```
solve(ST,a)
```

Out[128]: $[-1.76739878323355,\quad 0.0,\quad 1.76739878323355]$

In [132]:
```
ST_min = solve(diff(ST,a),a)
ST_min[1]
```

Out[132]: 1.02040816326531

In [133]:
```
ST.subs({a:ST_min[1]})
```

Out[133]: -0.0867694016382063

どうです．最初は$f = 1/2x^2$ で結果を確認しながらコードを整理していきます． 最後に係数を0.49に変えて実行します． 最後の結果が違う人はどこかで変数への変換が不十分なのだと思います．チェックしてみてください．

「重複」あるいは冗長な部分を省くとみやすくなっていると思います． 普通のテキストやネットのサンプルは綺麗なコードを書きたがります． でも，ここで示したように， 初めから綺麗な答えがあると思わないでください． プロもグタグタのcodeを書いて，その後でこそっと直しています． 初心者は，特に試行錯誤が必要です． この辺りが，「programmingは数学の問題を解くのに似ている」 と言われる所以(ゆえん)です．

数式処理のコツがわかったでしょうか？ プログラミングの極意に似ています． 「綺麗な動くコード」です． まずは動くコードを書いて，その後重複を省いて綺麗にしてください． この過程がREPLと表現されています．

In [ ]: