

10 月 6 日配付資料.....	2
配布文書「数学ソフト SAGE のインストール」.....	3
10 月 6 日の講義で使⽤したプログラム.....	5
10 月 20 日の講義で使⽤したプログラムと宿題.....	9
10 月 27 日の講義で使⽤したプログラムその他.....	11
10 月 20 日の宿題について.....	13
11 月 10 日の講義で使⽤したプログラムその他.....	16
11 月 17 日の講義の要約と宿題.....	20
11 月 17 日の宿題について.....	23
12 月 1 日の講義で使⽤したプログラムと宿題.....	26
12 月 15 日の講義の要約.....	28
12 月 22 日の講義の要約.....	30
12 月 24 日「TeX の使⽤方」.....	34
1 月 5 日の講義の要約 (修正版).....	36
12 月 22 日の宿題の解答例.....	40
1 月 19 日の講義の要約.....	43
1 月 26 日の講義の要約.....	47

この⽂書はほとんどは講義の配付資料ですが、一部修正した部分や書き足した部分があります。(2009/7/9)

## 計算機数学 A 配布資料 (2008 年 10 月 6 日) 石田 正典

教科書は指定しません。おもに無料の数学ソフト SAGE を使った演習などをする予定です。ただし川内のマルチメディア棟の計算機にこのソフトは入っていません。いまのところ、この数学棟 3 階の研究資料室の Macintosh 2 台とその書庫の奥にある SONY VAIO 4 台で SAGE が使えます。自宅にパソコンがある人はインターネットを通じてインストールすることを勧めます。

まず、大学の情報教育でもらっているアドレスか、個人のメールアドレスから私宛の電子メールを送り、逆に授業に関する私からのメールを受け取れるように各自メールアドレスを決めてください。アドレスの後からの変更も可能です。

そのために、最初の宿題として 10 月 14 日までに私 (石田 正典) 宛に「計算機数学 A を受講します。」という内容のメールを送ってください。私のメールアドレスは

ishida@math.tohoku.ac.jp

です。また私宛のメールは分類の都合上 Subject の先頭に必ず keisan をつけてください。例えば「Subject: keisan 10 月 6 日の宿題の答え」のような標題のメールにしてください。本文末には学籍番号と氏名も必要です。

最初のメールに対して返信としてアンケートを送りますので、それに答えてください。各個人のメールアドレスはこの講義についての本人とのやりとりにしか使いません。

## 数学ソフト SAGE のインストール

インターネットにつながったパソコンを持っていれば下記の SAGE のホームページに行ってインストール出来ます。ただし 300 MB 以上のファイルを読み込む必要があります。

<http://www.sagemath.org/>

Macintosh の場合は Mac OS X 10.4 または 10.5 で Xcode がインストールされている必要があります。Xcode はパソコン付属のディスクに入っています。インストールや起動にはアプリケーション/ユーティリティにあるターミナルを使うことになります。

Windows の場合は VMware という会社の VMware Player という無料ソフトをインストールして、その上に SAGE から読み込んだファイルをインストールする必要があります。 <http://www.vmware.com/products/player/>

Linux にもインストールできます。

インストールに成功すると Web ブラウザを通じて SAGE が使えるようになります。

ブラウザとしては Firefox が推奨されています。

Windows の場合の起動は `sage_vmx` を起動して VMware の黒いウインドウが開き、

1 分程度待つとメニューが表示されるので `notebook` と入力してから、ウインドウの外に出て、ブラウザを起動してアドレス `http://192.168.145.128` などを指定すると使えるようになります。アドレスの 145 の部分はインストールするパソコンによって異なります。

ブラウザに SAGE Notebook のページが表示されたら `New Worksheet` をクリックして、出てきたページの 1 行目に `2+2` を入れてシフトを押しながらリターンして `4` と表示されたら成功です。自分のパソコンに SAGE がインストールできたら OS の種類も書いて知らせてください。

<http://www.sagemath.org/library/books.html> にある

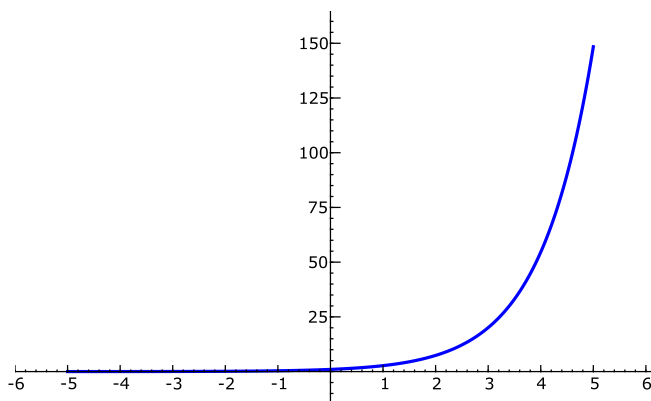
Sage for Newbies - by T. Kosan/sage\_for\_newbies\_v1.23.pdf  
を SAGE 自習のテキストとして推薦します.

10月6日の講義で使用したプログラム

一部は SAGE のページ <http://wiki.sagemath.org/interact> からのコピー  
(削除済み)

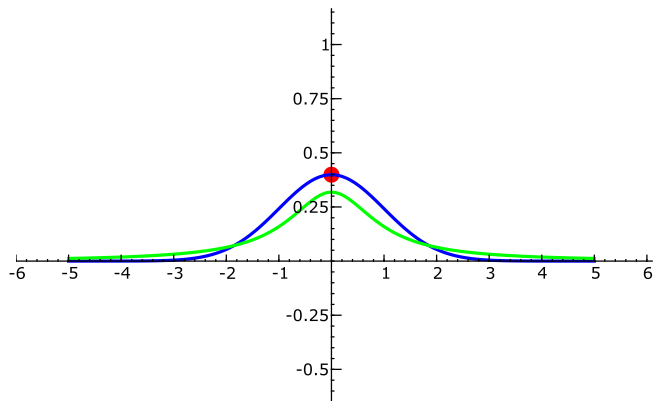
関数の表示

```
var('x')
f = e^x
p = plot(f, -5, 5, color='blue', thickness=2)
show(p)
```



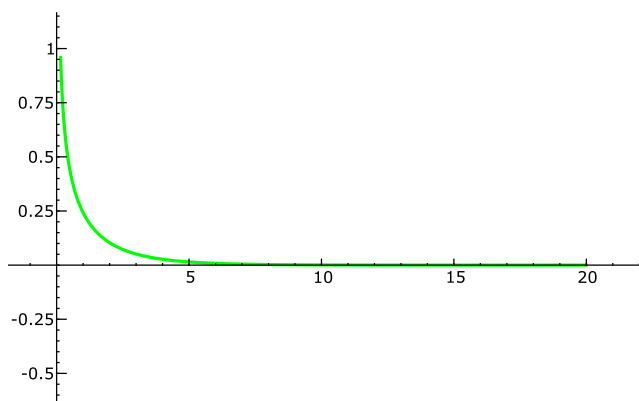
t 分布, t\_1 から t\_12 まで

```
var('x')
x0 = 0
f = e^(-x^2/2)/sqrt(2*pi)
p = plot(f, -5, 5, thickness=2)
dot = point((x0, f(x0)), pointsize=80, rgbcolor=(1, 0, 0))
@interact
def _(n=(1..12)):
    ft = gamma((n+1)/2)/(sqrt(n*pi)*gamma(n/2)*(1+x^2/n)^((n+1)/2))
    pt = plot(ft, -5, 5, color='green', thickness=2)
    html('$f(x)\;=\; %s$'%latex(f))
    html('t_%s(x)'%latex(n))
    show(dot + p + pt, ymin = -.5, ymax = 1)
```



カイ 2 乗分布, chi\_1 から chi\_12 まで

```
var('x')
x0 = 0
@interact
def _(n=(1..12)):
    ft = (x/2)^(n/2-1)/(2*gamma(n/2)*e^(x/2))
    pt = plot(ft, 0, 20, color='green', thickness=2)
    html('$f(x)\;=\;{}s$'.format(latex(f)))
    html('$\chi^2_{}s(x)$'.format(n))
    show(pt, ymin = -.5, ymax = 1)
```



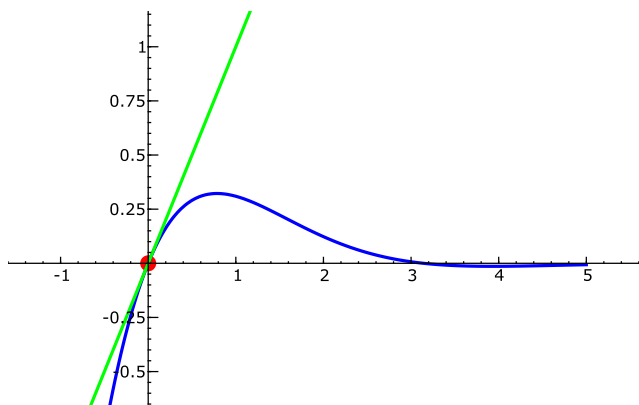
$e^{-x} \sin x$  のテーラー展開

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

```

x0 = 0
f = sin(x)*e^(-x)
p = plot(f,-1,5, thickness=2)
dot = point((x0,f(x0)),pointsize=80,rgbcolor=(1,0,0))
@interact
def _(order=(1..12)):
    ft = f.taylor(x,x0,order)
    pt = plot(ft,-1, 5, color='green', thickness=2)
    html('$f(x)\;=\; %s$'%latex(f))
    html('$\hat{f}(x;%s)\;=\; %s+\mathcal{O}(x^{\%s})$'%(x0,latex(ft),order+1))
    show(dot + p + pt, ymin = -.5, ymax = 1)

```



### 三角関数表

```

RR20 = RealField(20)
rads = [n*float(pi)/180 for n in range(1,90)]
trigs = ["sin", "cos", "tan"]
tbl = [[RR20(y)]+[RR20(eval(x+"(%s)"%y))] for x in trigs] for y in rads]
print 'degree radian      sin      cos      tan'
for i in range(1,90):
    print '%2s %40s'%(i,Matrix(tbl)[i-1])

```

このプログラムを実行すると三角関数表が表示される。

### 素数

```
n=1000; c=0
for i in range(n):
    if is_prime(n-i) and is_prime(i):
        print i, '+', n-i
        c=c+1
print c
```

このプログラムを実行すると 1000 が二つの素数の和で表される.

10 月 20 日の講義で使用したプログラムと宿題

短半径 1, 長半径 2 の楕円周の長さ

(1) パラメータ表示 ( $\cos(t)$ ,  $\sin(t)$ ) を用いて計算する方法

```
var('a,t')
p = 2
f = sqrt(sin(t)^2 + a^2*cos(t)^2)
A = f(a=p).nintegral(t,0,pi/2)
A; 4*A[0]
```

結果:

```
(2.4221120551369189, 8.6681765533880445e-11, 21, 0)
9.6884482205476754
```

(2) x 座標について積分する方法

```
var('a,x')
p = 2
f = sqrt(1+diff(sqrt(1-(x/a)^2),x)^2)
A = f(a=p).nintegral(x,0,p)
A; 4*A[0]
```

結果:

```
(2.4221120551368052, 2.8467179724600555e-09, 315, 0)
9.6884482205472207
```

結果の 1 行目は

(1/4 の積分値, 誤差の範囲, 積分計算に使った値の数, エラーコード)  
である。(1) の方法の方が少ない計算で誤差が少ないことがわかる。

$p = 2$  の左辺を任意の長半径に変えることができる.

10 月 20 日の宿題:

(1) 解析学 A で利用した教科書から 10 個の関数を選んで微分または積分を SAGE で実行してみる. そのうち微分 1 個と積分 1 個について結果を回答する.

```
diff(5*x^4+3*x-2)
20*x^3 + 3
```

```
integral(1/sin(x))
log(cos(x) - 1)/2 - log(cos(x) + 1)/2
```

(2) 短半径 1, 長半径  $a$  の楕円周の長さが 10 となる  $a$  の値を数値として求める. 回答ではどのように求めたかも説明する.

提出先: [ishida@math.tohoku.ac.jp](mailto:ishida@math.tohoku.ac.jp)

件名: keisan 10 月 20 日の宿題

提出期限: 2008 年 11 月 4 日

10 月 20 日

石田

## 計算機数学 A (2008/10/27)

(1) 簡単な既知関数で定義された関数なら数値解を求めるのに `find_root` 関数が見える。 `solve` 関数は役に立たない。

```
f = sin(x) + log(1+x)
print find_root(f==1,0,1)
```

しかし、 `numerical_integral` や `nintegral` で定義した関数には使えないようである。

(2) `for` を使ったループの例

```
f = sin(x) + log(1+x)
b = 0
for i in range(0,11,1):
    print i, n(f(b+i/10))
```

(3) 2 項係数を再帰呼び出しで求める。全然速くないがこのような関数の使い方は応用上便利である。

```
var('m,n')
def C(m,n):
    if m < 0 or n < m:
        return 0
    if m == 0 or m == n:
        return 1
    return C(m-1,n-1) + C(m,n-1)
C(10,20)
```

(3) 2 項係数を階乗を使って定義する。

```
var('m,n')
def C(m,n):
    if m < 0 or n < m:
        return 0
```

```
        return (factor(n)/factor(m))/factor(n-m)
C(10,20); C(1000,2000)
```

(4) if, else, elif を使ったプログラム

```
def test(x):
    if x == 1:
        print 'A'
    elif x == 2:
        print 'B'
    elif x == 3:
        print 'C'
    else:
        print 'D'
test(1); test(2); test(3); test(4); test(5)
```

(5) 集合演算の例

```
A = Set(range(10))
B = Set(range(0,20,2))
print A, B
A.intersection(B); A.union(B); A.difference(B); B.difference(A)
```

10 月 20 日の宿題について (2008/11/10)

短半径を 1 として楕円周が 10 となるように長半径を定める問題であるが, 長半径を  $a$  とすれば楕円周は  $a$  についての単調増加関数であることは想像される. 楕円周を  $g(a)$  とすれば  $a = p$  での値  $g(p)$  は

$$4 * f(a=p) . \text{nintegral}(x, 0, \pi/2) [0]$$

で計算される.  $g(2) < 10$  で  $g(3) > 10$  であるから  $g(a) = 10$  の解は开区間  $(2, 3)$  にあると考えられる.

$(p, g(p))$  と  $(q, g(q))$  を直線で結べば直線  $y = k$  と交わるのは

$$x_0 = (g(p)q - g(q)p - k(q - p)) / (g(p) - g(q))$$

となる.  $k = 10$  と置いて  $a = x_0$  とすれば  $g(a)$  が 10 に近い値をとると期待できる. 次は  $g(x_0) > 10$  なら开区間  $(2, x_0)$  で考え,  $g(x_0) < 10$  なら开区間  $(x_0, 3)$  を考える. 以下これを繰り返せば  $g(a) = 10$  となる  $a$  に近づくと考えられる.

下記のプログラムはこの方針で作られている. ただし, 繰り返しはループはせずに, 同じ操作をコピーして行った.

```
var('a,x')
k = 10; p = 3; q = 2
f = sqrt(sin(x)^2 + a^2*cos(x)^2)
u = 4*f(a=p).nintegral(x,0,pi/2)[0]
v = 4*f(a=q).nintegral(x,0,pi/2)[0]
r = ((u-k)*q+(k-v)*p)/(u-v); u = v; p = q; q = r
v = 4*f(a=q).nintegral(x,0,pi/2)[0]
r = ((u-k)*q+(k-v)*p)/(u-v); u = v; p = q; q = r
v = 4*f(a=q).nintegral(x,0,pi/2)[0]
r = ((u-k)*q+(k-v)*p)/(u-v); u = v; p = q; q = r
v = 4*f(a=q).nintegral(x,0,pi/2)[0]
r = ((u-k)*q+(k-v)*p)/(u-v); u = v; p = q; q = r
```

```

v = 4*f(a=q).nintegral(x,0,pi/2)[0]
r = ((u-k)*q+(k-v)*p)/(u-v); u = v; p = q; q = r
print q, 4*f(a=q).nintegral(x,0,pi/2)[0]

```

このプログラムの  $k$  の値を変えれば楕円周が  $k$  となる長半径  $a$  (場合によっては  $1$  が長半径で  $a$  が短半径となる) が得られる.

この問題は  $u = g(p)$  の逆関数の  $u = 10$  での値を求める問題であることがわかる. 下記のように書けば, 後半が誤差  $d$  で逆関数を計算していることになる.

```

a0 = 2; b0 = 3; maxroop = 20
var('t,x')
g = sqrt(sin(x)^2 + t^2*cos(x)^2)
def f(t):
    return 4*g(t).nintegral(x,0,pi/2)[0]
# f(t) monotone increasing at [a0,b0]
d = 1/10^14
def finv(y):
    a = a0; b = b0; c = (a0+b0)/2; m = 0
    u = f(a); v = f(b); w = f(c)
    if u < y and y < v:
        while abs(w-y) > d and m < maxroop:
            print m, c; m = m + 1
            if w-y < 0:
                a = c; u = w
            else:
                b = c; v = w
            c = (a*v-b*u-(a-b)*y)/(v-u)
            w = f(c)
        return c
    else:
        print "out of range!"
finv(10)

```

これを別の関数  $y = \sin(x) + x$  に適用して逆関数を作る. この場合は

ループ回数の上限 maxroop を増やす必要がある.

```
a0 = 0; b0 = n(pi); maxroop = 40
def f(x):
    return n(sin(x) + x)
# f(t) monotone increasing at [a0,b0]
d = 1/10^14
def finv(y):
    a = a0; b = b0; c = (a0+b0)/2; m = 0
    u = f(a); v = f(b); w = f(c)
    if u < y and y < v:
        while abs(w-y) > d and m < maxroop:
            print m, c; m = m + 1
            if w-y < 0:
                a = c; u = w
            else:
                b = c; v = w
            c = (a*v-b*u-(a-b)*y)/(v-u)
            w = f(c)
        return c
    else:
        print "out of range!"
f(finv(1)), f(finv(2)), f(finv(3))
```

結果  $c = \text{finv}(3)$  は  $m = 34$  で  $c = 2.17975706648007$  となり,  
 $f(\text{finv}(3)) = 3.000000000000001$  と表示される.

計算機数学 A (2008 年 11 月 10 日)  
線形代数

SAGE での行列は

```
A = matrix(2,3,[[1,3,5],[5,7,9]])
```

のように書く. `matrix` のすぐ右の 2, 3 は行の数と列の数である. これを簡単に

```
A = matrix([[1,3,5],[5,7,9]])
```

や

```
A = matrix(2,3,[1,3,5,5,7,9])
```

と書くこともできる.

```
A = matrix(2,3,[[1,3,5],[5,7,9]])  
A; transpose(A)
```

とすると

```
[1 3 5]  
[5 7 9]  
[1 5]  
[3 7]  
[5 9]
```

と表示される. 上が  $A$  で, 下が  $A$  の転置行列 `transpose(A)` である. 実数や複素数の混じった行列

```
A = matrix(2,2,[1/2,3+I,5.0,pi])
```

や, 変数の入った行列

```
var('x1,x2,x3,y1,y2,y3')
A = matrix(2,3,[x1,x2,x3,y1,y2,y3])
```

も定義できる。型が適当であれば、和は + 積は \* ベキは ^ でできる。

```
A = matrix([[2,3],[1,2]])
A^50
```

正方行列ならそれらで使える関数がある、例えば

```
A = matrix(3,3,[[1,3,5],[5,7,9],[9,3,1]])
~A; det(A)
```

で逆行列  $\sim A$  や行列式  $\det(A)$  が

```
[ 5/8 -3/8 1/4]
[-19/8 11/8 -1/2]
[ 3/2 -3/4 1/4]
-32
```

と表示される。

6 次の Vandermonde 行列式の展開した式と因数分解した式が、下記を実行すれば得られる。

```
var('x1,x2,x3,x4,x5,x6')
A = matrix(6,6,[1,1,1,1,1,1,
                x1,x2,x3,x4,x5,x6,
                x1^2,x2^2,x3^2,x4^2,x5^2,x6^2,
                x1^3,x2^3,x3^3,x4^3,x5^3,x6^3,
                x1^4,x2^4,x3^4,x4^4,x5^4,x6^4,
                x1^5,x2^5,x3^5,x4^5,x5^5,x6^5])
F = det(A); expand(F); factor(F)
```

行列の基本変形による簡約化：

行列は行ベクトルの基本変形により階段状の簡約化された行列に変形

できる。英語では row-echelon form といい、段の角の位置の成分がすべて 1 となっているものを reduced row-echelon form という。角の 1 を pivot といい、その 1 を含む列を pivot column という。SAGE のページの Library にリンクのある線形代数の電子本 "A First Course in Linear Algebra" の p.38 に変形の例がある。この例について

```
A = matrix(3,4,[-7,-8,-12,-33,5,5,7,24,1,0,4,5])
A.echelon_form()
```

とすれば A の row-echelon form

```
[ 1  0  4  5]
[ 0  1  7 -1]
[ 0  0 24 -2]
```

が表示され、

```
A = matrix(QQ,3,4,[-7,-8,-12,-33,5,5,7,24,1,0,4,5])
A.echelon_form()
```

なら A の reduced row-echelon form

```
[ 1  0  0  16/3]
[ 0  1  0 -5/12]
[ 0  0  1 -1/12]
```

が表示される。これは整数の行列でも reduced row-echelon form にしようとするとな分数が必要になることが多いことによる。'QQ' は有理数で考えていることを表す。整数は 'ZZ' である。

連立一次方程式は solve 関数を使って

```
var('x,y,z')
solve([3*x+y+z==1,x-y+2*z==4,x-5*y-8*z==7],x,y,z)
```

で解が得られる.

固有値とそれらに対応する固有空間を表示する関数もある.

```
B = diagonal_matrix([1,2,3,4,5])  
B[0,4] = 1; B; B.eigenspaces()
```

## 有限体上の行列

有理数体  $\mathbf{Q}$ , 実数体  $\mathbf{R}$ , 複素数体  $\mathbf{C}$  など四則演算のできる集合を体という。  $\mathbf{Q}$  は  $\mathbf{R}$  に含まれるが  $\mathbf{Q}$  と  $\mathbf{R}$  の間にも  $\mathbf{Q}(\sqrt{2})$  など多くの体が存在する。

そのほか元の数が有限の体も存在する。  $p$  を素数とすると  $\mathbf{Z}$  剰余加群  $\mathbf{Z}/p\mathbf{Z}$  には積も定義できて体となる。 数学の記号としてはこれを  $\mathbf{F}_p$  と書くが, SAGE ではこの体を  $\text{GF}(p)$  と書く。 例として  $p = 5$  なら  $\mathbf{F}_p = \{0, 1, 2, 3, 4\}$  で,

```
K = GF(5)
for a in K:
    for b in K:
        print a, '+', b, '=' , a+b
```

を実行すると, 全部の組み合わせで 2 元の和が表示される。 積は

```
K = GF(5)
for a in K:
    for b in K:
        print a, '*', b, '=' , a*b
```

でわかる。 普通の整数の積では  $3 * 3 = 9$  だが  $\mathbf{F}_5$  では  $3 * 3 = 4$  である。 すなわち, 右辺は 9 を 5 で割った余りの 4 となる。

もっと一般に, 素数の冪乗  $q = p^l$  についても, 元の数  $q$  の有限体  $\mathbf{F}_q$  が存在する。 定義は難しいので説明は省略するが, SAGE では

```
K.<t> = GF(3^2,'t')
for a in K:
    print a
```

のように書いて実行すると, 体  $K$  に含まれる 9 個の元が表示される。  $3^2$  のところをいろいろ変えて試して見るとよい。  $K$  の元を見慣れたら和や積も  $\mathbf{F}_5$  と同様に表示して見るとよいだろう。 ただし  $a, b$  にかっこをつけないと見にくくなる。

$K$  を有限個の元からなる体, すなわち有限体とする。  $K$  の元について和と積が定義されているので,  $K$  の元を要素とする行列を考えることができる。 次の例は  $\mathbf{F}_7$  で  $3 \times 3$  行列

$$A = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

を考え、 $A$  を何乗すると単位行列となるかを調べている。この数を  $A$  の位数という。

```
M = MatrixSpace(GF(7),3)
I = M([1,0,0,0,1,0,0,0,1])
B = I
A = M([0,2,0,0,0,1,1,0,1])
if det(A) == 0:
    print "not regular"
else:
    for i in range(1,1000):
        B = B*A
        if B == I:
            print i
            break
```

上記のプログラム中で  $I$  は 3 次の単位行列, “not regular” は行列式が 0, すなわち正則行列でないことを知らせる。

### 11 月 17 日の宿題 (期限 11 月 28 日)

(1)  $F_7$  の元の 3 次正則行列の位数となる自然数をすべて求めよ。

(2)  $F_8$  の元の 3 次正則行列の位数となる自然数をすべて求めよ。

位数全部が無理であればできるだけ多く見つけて欲しい。行列は書かなくて良い。回答ではどのように求めたかも説明する。

提出先: [ishida@math.tohoku.ac.jp](mailto:ishida@math.tohoku.ac.jp)

件名: keisan 11 月 17 日の宿題

提出期限: 2008 年 11 月 28 日

2 次行列であれば下記のプログラムのように全部の行列について調べることもできる。このプログラムでは位数を求める関数 `ord()` を定義して、最初は空集合の  $L$  に見つけた位数を追加していく。正則でない行列の位数はとりあえず  $-1$  としておき、最後に削除する。

```
K = GF(7)
N = MatrixSpace(K,2)
I = N([1,0,0,1])
def ord(A):
    B = I
    if det(A) == 0:
```

```

        return -1
    for i in range(1,1000):
        B = B*A
        if B == I:
            return i
    print A,"ord > 999"
L = Set([])
for a in range(7):
    for b in range(7):
        for c in range(7):
            for d in range(7):
                i = ord(N([a,b,c,d]))
                if i not in L:
                    L = L.union(Set([i]))
print L.difference(Set([-1]))

```

ここで  $L = \text{Set}([])$  の右辺は空集合,  $\text{Set}([i])$  は  $\{i\}$ ,  $A.\text{union}(B)$  は  $A \cup B$ ,  $A.\text{difference}(B)$  は  $A \setminus B$  である.

有限体の中から乱数として元を選んで行列の要素を取り替える関数もある. 今回の宿題にも使えそうである.

```

A = matrix(GF(5), 5, 5)
for i in range(10):
    A.randomize(); print i; print A

```

## 11 月 17 日の宿題について

$K$  を体とし  $n$  を正の整数とすると,  $K$  の元を成分とする  $n$  次正方行列全体は加法と乗法が定義される集合となる. 代数の言葉ではこのような集合を環という. ただし, 乗法について  $AB = BA$  が成り立つとは限らないので非可換環に属する. 通常この環は  $M(n, K)$  と書かれるが, Sage ではこの非可換環を `MatrixSpace(K, n)` と書いている.

各行列  $A \in M(n, K)$  はその行列式  $\det(A)$  が実数行列の場合と同じ式で定義され, その値は  $K$  の元となる.  $\det(A) \neq 0$  となる行列  $A$  全体は  $GL(n, K)$  と書かれ群となる. すなわち, 単位行列がこの群の単位元であり, 行列式が 0 でないことにより, すべての元が逆元をもつ. 行列の成分は  $n^2$  個なので,  $K$  が有限体で元の数  $q$  個であれば,  $M(n, K)$  の元の数  $q^{n^2}$  個である. これに含まれる群  $GL(n, K)$  の元の個数, つまり位数についての説明は省略するが,  $GL(2, K)$  の位数は  $(q^2 - 1)(q^2 - q)$  で,  $GL(3, K)$  は  $(q^3 - 1)(q^3 - q)(q^3 - q^2)$  である.  $GL(2, \mathbf{F}_7)$  なら位数は  $(7^2 - 1)(7^2 - 7) = 2016 = 2^5 \cdot 3^2 \cdot 7$  となる. 前回のプログラムでわかるように, この群の元の位数として現れる自然数は

$$\{1, 2, 3, 4, 6, 7, 8, 12, 14, 16, 21, 24, 42, 48\}$$

である. 当然ながら, すべて群の位数の約数となっている.

さて, 宿題 (1) の  $GL(3, \mathbf{F}_7)$  であるが,  $M(3, \mathbf{F}_7)$  の元の個数は 40353607 もあるので, 全部ループで回すには多すぎる. ここで群の共役な 2 元の位数は互いに等しいことを使うと使うと, 調べる個数を減らすことができる. 標準基底で  $A$  で表される線形写像は, 縦ベクトル  $v_1, v_2, v_3$  を基底に取り直すと, 行列  $P = (v_1, v_2, v_3)$  により表現行列が  $A$  と共役な  $P^{-1}AP$  となる. このことを使って  $A$  を共役元と取り替える.

1. あるベクトル  $v_1$  について  $v_2 = Av_1$  が  $v_1$  と一次独立で, さらに  $v_3 = Av_2$  が  $v_1, v_2$  と一次独立になる場合: この場合は  $v_1, v_2, v_3$  が基底となり,  $Av_3 = av_1 + bv_2 + cv_3$  とすると表現行列は

$$\begin{bmatrix} 0 & 0 & a \\ 1 & 0 & b \\ 0 & 1 & c \end{bmatrix}$$

となる.

2. あるベクトル  $v_1$  について  $v_2 = Av_1$  が  $v_1$  と一次独立となるが,  $Av_2$  は  $v_1, v_2$  で生成される 2 次元部分空間に含まれる場合: この場合, まず  $Av_1 = av_1 + bv_2$  と

する. つぎに  $v_1, v_2$  と一次独立なベクトル  $v$  をとる.  $Av = a_1v_1 + a_2v_2 + a_3v$  としたとき,  $v_3 = v - a_2v_1$  と置けば  $Av_3 = Av - a_2v_2 = a_1v_1 + a_3v = (a_1 + a_2a_3)v_1 + a_3v_3$  となる. このとき,  $v_1, v_2, v_3$  は基底で  $c = a_1 + a_2a_3, d = a_3$  と置けば線形写像の表現行列は

$$\begin{bmatrix} 0 & a & c \\ 1 & b & 0 \\ 0 & 0 & d \end{bmatrix}$$

となる.

3. 上記のどちらでもなければ, 任意のベクトル  $v$  について  $v$  と  $Av$  が一次従属となるが, この場合  $A$  は単位行列の定数倍となる. 実際,  $A = [a_{ij}]$  が対角行列でなければ, ある  $i \neq j$  について  $a_{ij} \neq 0$  だが, 基本ベクトル  $e_j$  について  $e_j$  と  $Ae_j = a_{ij}e_i$  が一次独立となる. また  $a_{ii} \neq a_{jj}$  であれば  $e_i + e_j$  と  $A(e_i + e_j) = a_{ii}e_i + a_{jj}e_j$  は一次独立となる. したがって,  $a = a_{11}$  とすれば  $A$  は単位行列の  $a$  倍に等しい.

これら 3 つの場合について, 基底を取り替えて簡易化した行列だけについて位数を調べればよい.

```
K = GF(7)
M = MatrixSpace(K,3)
I = M([1,0,0,0,1,0,0,0,1])
def ord(A):
    B = I
    if det(A) == 0:
        return -1
    for i in range(1,1000):
        B = B*A
        if B == I:
            return i
    print A,"ord > 999"
L = Set([])
for a in range(7):
    for b in range(7):
        for c in range(7):
            i = ord(M([0,0,a,1,0,b,0,1,c]))
            if i not in L:
                L = L.union(Set([i]))
for a in range(7):
```

```

for b in range(7):
    for c in range(7):
        for d in range(7):
            i = ord(M([0,a,c,1,b,0,0,0,d]))
            if i not in L:
                L = L.union(Set([i]))
for a in range(7):
    i = ord(M([a,0,0,0,a,0,0,0,a]))
    if i not in L:
        L = L.union(Set([i]))
A = list(L.difference(Set([-1])))
A.sort(); A

```

このプログラムを実行すると

[1, 2, 3, 4, 6, 7, 8, 9, 12, 14, 16, 18, 19, 21, 24, 38, 42, 48, 57, 114, 171, 342]

と表示される。プログラムの最後の 2 行は集合をリストに変換し、ソート関数を適用して番号順に並べ直している。GL(3,  $\mathbf{F}_7$ ) の位数は  $(7^3 - 1)(7^3 - 7)(7^3 - 7^2) = 33784128 = 2^6 \cdot 3^4 \cdot 7^3 \cdot 19$  である。

(2) の GL(3,  $\mathbf{F}_8$ ) については、このプログラムの 1 行目を  $K.\langle t \rangle = \text{GF}(2^3, 't')$  に代えて、range(7) をすべて K に代えればよい。結果は

[1, 2, 3, 4, 7, 9, 14, 21, 28, 63, 73, 511]

が得られる。なお、この場合の “for i in K:” は K のリスト化

$$\text{list}(K) = [0, t, t^2, t + 1, t^2 + t, t^2 + t + 1, t^2 + 1, 1]$$

の要素を順に変数  $i$  に代入するループである。GL(3,  $\mathbf{F}_8$ ) の位数は  $(8^3 - 1)(8^3 - 8)(8^3 - 8^2) = 115379712 = 2^9 \cdot 3^2 \cdot 7^3 \cdot 73$  である。元の最大位数の 511 は  $7 \cdot 73$  である。

計算機数学 A (2008/12/1, 石田)

RSA 社の懸賞問題

次の 174 桁の自然数の積への分解は 1 万ドルの賞金がかかっていた。解は 2003 年に得られた。

```
1881988129206079638386972394616504398071635633794173827007\  
6335642298885971523466548531906060650474304531738801130339\  
6716199692321205734031879550656996221305168759307650257059\  
==398075086424064937397125500550386491199064362342526708406\  
385189575946388957261768583317\  
*472772146107435302536223071973048224632914695302097116459\  
852171130520711256363590397527
```

最新では次の 212 桁の数の分解が問題として出されている。Is\_prime 関数を使えば素数でないことはわかる。

```
n = 7403756347956171282804679609742957314259318888\  
9231289084936232638972765034028266276891996419625117\  
8439958943305021275853701189680982867331732731089309\  
0055250511687706329907239638078671008609696253793465\  
0563796359  
len(n.str(10)); n.is_prime()
```

44 番目のメルセンヌ素数

$p = 2^q - 1$  が素数であれば  $2^{q-1}$  が完全数となる。このような素数  $p$  をメルセンヌ素数という。メルセンヌ素数は大きな素数を作る競争のわかりやすい目標となっている。下記は 44 番目の数である。これは約 980 万桁である。 $s$  はこれを 10 進数の文字列にしたもので、任意の部分を表示させることができる。

```
p = 232582657 - 1  
p.ndigits()
```

```
s = p.str(10)
len(s); s[:100]
s[1000:2000]
```

最大桁を 0 番目として m 番目から n-1 番目の桁を表示したい場合は s[m:n] を実行する. 最初の n 桁は s[:n], 最後の n 桁は s[-n:] とする.

#### 素数の密度

n までの素数の数を返す関数 prime\_pi(n) を使って, 1 億までの自然数のうちどれだけの割合で素数が現れるかが下記を実行するとわかる. 約 5 % である.

```
n = 100000000
float(prime_pi(n)/n)
```

#### 12 月 1 日の宿題:

計算機で何かを計算して得られた結果を報告すること.

例: (1) 何かの曲線の長さを計算する. (2) 曲線の長さが与えられた値になるようにパラメーターを

調節する. (3) 特定の群の元の位数を調べる. (4) 特定の群の共役類の数を求める. (5) 対称群や

交代群について何か計算する.

提出先: ishida@math.tohoku.ac.jp

件名: keisan 12 月 1 日の宿題

提出期限: 2008 年 12 月 12 日

#### 参考文献

William Stein, Elementary Number Theory: Primes, Congruences, and Secrets, Springer-Verlag, 2008.

計算機数学 A (2008/12/15)

自然数  $a, b$  が互いに素であれば, ある整数  $s, t$  について  $sa + tb = 1$  となる.

**フェルマの小定理**  $p$  を素数とする.  $0 < a < p$  を満たす任意の整数  $a$  について  $a^{p-1} \equiv 1 \pmod{p}$  が成り立つ. 逆に, これがすべての  $a$  について成り立てば  $p$  は素数である.

この定理により, 特定の  $a$  だけについてでも  $a^{p-1} \equiv 1 \pmod{p}$  が成り立たなければ  $p$  が素数でないことがわかる. 次のプログラムにより,  $a = 2$  が素数の判定に役立つことがわかる.

```
for p in range (3, 1000):
    A = (Mod(2, p)^p-1 == 1)
    B = is_prime(p)
    print '%6s %6s %6s %6s'%(p, A, B, A == B)
```

$\mathbf{Z}/n\mathbf{Z}$  での計算を SAGE で行うには  $\text{Mod}(a, n)$  や  $\text{Integers}(n)$  を使う. 暗号理論への応用上, 非常に高い指数の冪乗を行う必要がある.

冪乗の計算方法:  $a^4 = (a^2)^2$  であるから,  $a^4$  はかけ算 2 回で計算できる.  $a^8 = (a^4)^2$  より,  $a^8$  は 3 回でできる. 一般に  $a^{2^n}$  は  $n$  回のかけ算でできることがわかる.  $a^{13}$  なら  $a^{13} = a^8 a^4 a$  であるから,  $a^8$  は 3 回,  $a^4$  は 2 回で計算できて,  $a^8, a^4, a$  をかけ合わせるのにさらに 2 回必要で, 結局 7 回のかけ算で計算できる.

$a$  の冪乗を数多く計算する場合は,

$$a, a^2, a^4, a^8, \dots, a^{2^l}$$

の数値を用意しておいて,  $a^n$  であれば  $n$  を 2 冪数の和に表して, 各項に対応する  $a^{2^i}$  をかけ合わせれば計算できる. 結局, 必要なかけ算の回数は  $n$  の桁数に比例する程度しか増えて行かない.

```
R = Integers(10)
a = R(3) # create an element of Z/10Z
a.multiplicative_order()
```

これは  $R$  を剰余環  $\mathbf{Z}/10\mathbf{Z}$  と定義して, その元 3 を何乗すると 1 となるかを求めている. 実行すると 4 が表示される.  $3^4 = 81 \equiv 1 \pmod{10}$  である.

そのほか,  $n$  と互いに素な  $n$  以下の自然数の数を求めるオイラー関数  $\text{euler\_phi}(n)$ , 中国の剰余定理に従い,  $m$  で割って  $a$ ,  $n$  で割って  $b$  となる自然数を求める  $\text{CRT}(a, b, m, n)$ ,

$a, b$  の最大公約数  $d$  を  $d = sa + tb$  と表す  $s, t$  を求める `xgcd(a, b)` などの関数がある。また、`100.str(2)` により 100 を 2 進数で表すことができる。数字の次にアルファベットを使うので 36 進数まで可能である。10 の 1000 乗を 36 進数で表すと下記のようになる。

```
(10^1000).str(36)
```

```
'753hplxbwkpvgnvq5g47vsoyvusdy70m4riv8tu0acgdsky4tggh7qcoxaie5nrsg59gpq4\  
mv7epycy9q0fsij4iun8a7qqhvsksnf7gli8naj4o5zv9v9ofd9uwwops9zyefc01g8fbbhs\  
owe8prof51alijhnxtl7sks80sgzzvq9urr5kdv5rbv45isaki6zastlts5ffb4gr8unrs5b\  
grdsv2f8jxx4t7nlmogdigkifeczacrlqrwjziumajujqlootlzojlrincuuxl69ljjity3a\  
ltx0o6cnynygwazpw2l3cgcosc0meo6cjsvms63q49h1b3ixyfuen09oxlu2c5y2qu9hmhln\  
idu9isb37nl0yktp2dfm9dizhg3rni jwtrkpe4yvvd06bs525yhn098pzjipg0duhvfia73p6\  
ins7pr91tiea6v1tcxzlw0aftzwvewj7aporw1fa1t0irwrhy6jit3u1rhpa0ieezjzgtbn\  
h1axyetf1wzzw3rw9qwl98fw6t7l28dwh8z4wnz55jynjpf1s8izpkrmqjba1r7i2sv5u\  
7fh0flarc67hiks9hdpwd8h9a2f9kaxgi3u3zcqnw56mxhlohr8jkonnayywmk724xds'
```

通常のアルファベットの集合を  $\mathcal{A}$  とし, 暗号化のため置き換える文字の集合を  $\mathcal{A}'$  とする. 実際の英文を作るには大文字と小文字の他, 空白, カンマ, ピリオドなどが必要だが, ここでは話を簡単にするため  $\mathcal{A}$  は  $A$  から  $Z$  までの 26 個の大文字と空白からなる 27 個の元からなる集合とする.

単純な代入による暗号化は全単射  $E_K : \mathcal{A} \rightarrow \mathcal{A}$  から作られる.  $s$  を  $\mathcal{A}$  の元からなる長さ  $l$  の文字列としたとき, 各  $0 \leq i \leq l$  について  $t[i] = E_K(s[i])$  と置くことにより長さ  $l$  の文字列  $t$  をつくる.  $s$  を平文とすると  $t$  が暗号文となる. 暗号文の平文化は  $E_K$  の逆写像  $D_K$  を用いて, 各  $0 \leq i \leq l$  について  $s[i] = D_K(t[i])$  と置けば, 暗号文  $t$  から元の文  $s$  が得られる.  $E_K$  としてアルファベットの順のまま巡回置換をとったものをシーザー暗号という.

このような暗号は, 暗号文に現れる文字の頻度を調べることにより解読されやすい.

暗号の解読: 英文で使われるアルファベットの出現頻度 (William Stein, Elementary Number Theory: Primes, Congruences, and Secrets, Springer-Verlag, 2008.)

A	B	C	D	E	F	G	H	I	J	K	L	M
7.3	0.9	3.0	4.4	13	2.8	1.6	3.5	7.4	0.2	0.3	3.5	2.5
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
7.8	7.4	2.7	0.3	7.7	6.3	9.3	2.7	1.3	1.6	0.5	1.9	0.1

現在実際に使われている暗号はもっと複雑な仕組みによるもので, 対称鍵暗号と公開鍵暗号に分類される. 対称鍵暗号は通信を行う二者が同じ平文化のための鍵を持つものである. 公開鍵暗号では, 一方は暗号化の鍵だけを持ち, 他方が平文化のための鍵も持つ. 一般に公開鍵暗号は暗号化や平文化のための計算に時間がかかる.

公開鍵暗号 会社 A から個人 B がインターネットで買い物をするためクレジットカード番号などの個人情報を暗号化して送るという設定で考えるとよい. 通信内容は第三者に知られることを予期しておく.

(1) RSA 暗号: A から B へ, 素数の積  $n = pq$  と  $\gcd(e, p-1) = \gcd(e, q-1) = 1$  を満たす自然数  $e$  を公開鍵として送る. B から A に送りたい内容の平文を  $0 < m < n$  に数値化する.

$$D : \mathbf{Z}/n\mathbf{Z} \longrightarrow \mathbf{Z}/n\mathbf{Z}$$

$D(m) = m^e$  が暗号化となる. B から数値  $m^e$  を暗号文として受け取った A は,  $de \equiv 1 \pmod{(p-1)(q-1)}$  なる  $d$  を使って  $m = (m^e)^d$  で平文化する.  $n$  から素因数の  $p, q$  がわかれば  $(p-1)(q-1)$  や  $d$  がわかり, 暗号が解読できることになるが, 大きな素数の積の素因数分解が計算困難であることにこの暗号化の意味がある.



$p, q, e$  の作り方と  $d$  の見つけ方は次のようにすればよい。ここで使っている lift は  $\mathbb{Z}/l\mathbb{Z}$  などの元を整数に戻す関数である。

```
p = next_prime(ZZ.random_element(10^30))
q = next_prime(ZZ.random_element(10^30))
n = p*q
phi_n = (p-1)*(q-1)
print p, q, n, phi_n
while True:
    e = ZZ.random_element(n)
    if gcd(e,phi_n) == 1: break
d = lift(Mod(e,phi_n)^(-1))
print e, d
```

### 12 月 22 日の宿題 (期限 1 月 9 日)

英文を RSA 暗号化するプログラムと、暗号を平文化するプログラムを作れ。次に書く手順を組み合わせるとできるはずである。

(1) 英文テキストを  $N$  字ずつに分解する。

```
s = 'The Integer class represents arbitrary precision integers. It derives\
from the Element class, so integers can be used as ring elements\
anywhere in SAGE.'
N = 20
l = len(s)
for i in range(integer_ceil(l/N)):
    t = s[N*i:N*(i+1)]
    print t
```

ここで  $l$  は文全体の文字数で、 $\text{integer\_ceil}(l/N)$  は  $l/N$  以上の最小の整数を表す。  $s[a:b]$  は文字列  $s$  の  $a$  文字目から  $b-1$  文字目までを取り出した文字列を表す。  $s$  の左端が 0 文字目で右端が  $l-1$  文字目であることに注意が必要である。  $s$  を他の英文で置き換えて試すとよい。

(2) 文字列をアスキーコードの列に変換する。

```
Hex = HexadecimalStrings()
t = 'The Integer class'
a = Hex.encoding(t)
a
```

HexadecimalStrings() は文字列を 16 進数で表したり戻したりするための目印で、ここではそれを短く Hex に置き換えている。

(3) アスキーコードの列を整数値に変換する。

```
a = '54686520496e746567657220636c61'  
m = Integer('0x'+str(a))  
m
```

16 進数は左端に  $0x$  をつけて記述する。  $0x8 = 8$ ,  $0x9 = 9$ ,  $0xa = 10$ ,  $0xb = 11$  である。文字列 '0x9bc5' は Integer('0x9bc5') により整数

$$0x9bc5 = 9 * 16^3 + 11 * 16^2 + 12 * 16 + 5 = 39877$$

に変換される。

(4)  $n$  を法とした冪乗の計算をして、結果を整数にする。

```
n = 74349214164438966063956566993552725620206028600147093192213  
e = 35766809327297305724406723217696438904514350075886769591457  
m = 10000  
mm = lift(Mod(m,n)^e)  
mm
```

ここで  $\text{Mod}(m,n)^e$  だけでは  $\mathbf{Z}/n\mathbf{Z}$  の元となるので、lift 関数を使って整数に戻す。ここで RSA 暗号作成のため用意した  $d$  を使い、 $mm$  を  $d$  乗して  $m$  と等しいはずの整数  $mmm$  を得る。

(5) 整数  $mmm$  を 16 進数にして、アスキーコードの列と考えて、英文に戻す。

```
d = 71175607633178390991276629813850593839840420591419976447193  
mmm = lift(Mod(mm,n)^d)  
tt = Hex(hex(mmm)).decoding()  
tt
```

ここで hex(mmm) が数値  $mmm$  を 16 進数にしたもの、Hex(hex(mmm)) がそれをアスキーコードの列と考えたもので、最後に .decoding() をつけて英文字の列に戻している。

## TeX の使い方 (2008/12/24) 石田

数式の記述には TeX を使いやすくした L<sup>A</sup>T<sub>E</sub>X を使うのが普通である。

この上の一行を表示する文書を TeX (テック, テフ) で作るには, 下記の 4 行からなるテキストファイルを作り, ファイル名を aaa.tex のように適当に付けて, LaTeX に読み込ませればよい. DVI ファイルまたは PDF ファイルが生成される. このことを TeX ファイルを「TeX にかける」, 「TeX でコンパイルする」, 「タイプセットする」などという.

```
\documentclass{jarticle}
```

```
\begin{document}
```

数式の記述には \TeX を使いやすくした \LaTeX を使うのが普通である.

```
\end{document}
```

\ の記号は普通の文書の中で見かけることは少ないが, 計算機のプログラムなどでは制御用の文字としてよく用いられる. TeX では \ で始まる単語は何かのコマンドである.

記述したい文書は \begin{document} と \end{document} の間に記述する. TeX にかけての結果がどうなるかは, 実際にいろいろやってみて観察するのがよい.

1. 数式の無い文書はそのまま書けばよい. ただし, 一つの改行は無視される. その位置で改行したい場合は改行のキーを二度打って一行あける. 文節の始めの空白は入れない. TeX にかけると自動的に空白が入った文書ができる.
2. 文中に数式を入れるときは  $x^2 - y^2 = (x + y)(x - y)$  のようにドル記号 \$ ではさんで書く. TeX にかけると  $x^2 - y^2 = (x + y)(x - y)$  と数式になる. 数式の文字が少し右に傾いたイタリック体の文字になっていることに注意する.
3. 独立した行で数式を書きたい場合は

任意の実数  $x, y$  と正の整数  $n$  に対して

\$\$

$$(x + y)^n = \sum_{k=0}^n \binom{n}{k} x^{n-k} y^k$$

\$\$

が成り立つ.

のように書く. このように書いた TeX ファイルをコンパイルすると

任意の実数  $x, y$  と正の整数  $n$  に対して

$$(x + y)^n = \sum_{k=0}^n {}_n C_k x^{n-k} y^k$$

が成り立つ.

と書かれた文書が得られる. ここで TeX ファイル中の `\rm C` は, C をイタリックでなくまっすぐ立ったローマン体にしたいのでこう書いている.

- TeX ファイル (正確には LaTeX ファイルの) の `\documentclass{jarticle}` と `\begin{document}` の間をプリアンブルという. ここには文書全体に関する書式設定や独自のコマンドの定義を行うことができる.

例えば

```
\setlength{\oddsidemargin}{10pt}
\setlength{\topmargin}{0pt}
\setlength{\textwidth}{427pt}
\setlength{\textheight}{592pt}
```

のように書いて, 印刷したときの上下左右の余白を調節することができる. また  $\gcd(a, b)$  の記号はもともと `\gcd` があるので書けるが, 新しく `\xgcd(a, b)` という記号を使いたいときは, プリアンブルに

```
\newcommand{\xgcd}{\mathop{\mathrm{xgcd}}\nolimits}
```

と書いて定義しておけば, 必要なときに `\xgcd(a, b)` のように書いて使うことができる.

- TeX は元々活版印刷の活字を組む作業を自動化するためのソフトなので, 文書の体裁を整えるためのコマンドがたくさんある. `\vspace{3mm}` は縦に 3 mm 分スペースをとってその下から始める, `\hspace{10mm}` は, その行で 10 mm 分スペースをとって右から始める, `\noindent` は, 改行した位置につけて, 文頭に空白を入れないことを表す.
- `{ }` は TeX ファイルの中で制御用に使われるのでコンパイルすると表示されない. 表示させたい場合は `\{ }` と書く. また, この文書の TeX ファイルを見て `\begin{verbatim}`, `\end{verbatim}` や `\verb` の意味を読み取って欲しい.

## 計算機数学 A (2009/1/5)

Sage は Python の上で動作しているので、ファイルや文字列の操作を行うには Python の知識が必要となる。

ファイルの操作は C の場合の知識があれば、Python の場合も同じようなところが多く、それより扱い易い部分もある。テキストデータを aaa という名前のファイルとして保存したい場合は、まず

```
f = open("/Users/ishida/keisanki/aaa","w")
```

をプログラムに入れておく。左辺の  $f$  は C のファイルポインタのようなものである。

```
f = open("/Users/ishida/keisanki/aaa","w")
f.write('0123456789\n')
f.write('abcdefghijklmnopqrstuvwxyz\n')
f.close()
```

として実行すると

```
0123456789
```

```
abcdefghijklmnopqrstuvwxyz
```

の 2 行からなる aaa というファイルが得られる。  $\backslash n$  は改行コードである。ファイル名 aaa の前の /Users/ishida/keisanki/ の部分は aaa を入れるフォルダ（ディレクトリ）の絶対アドレスである。Windows の場合はこの部分を C:\Users\ のように使うディレクトリを指定する。（C:/Users/ が良いのかも知れない。）

### Windows の場合についての訂正と補足：

講義のときに上記のように言ったが、Window 版の Sage は VMware Player という仮想 OS の上で動いているので、ハードディスクの C: ドライブなどは見えておらず、ディスクへのファイル操作は Sage を普通にインストールしたままではできない。

パソコンの操作に慣れた人なら、次のようにして VMware 側からも見えるフォルダ（共有フォルダ）を作ることができる。まず Windows の方で C: ドライブに SharedFolder というフォルダを作る。フォルダの名前と場所は変えることもできる。次に Sage の起動に使っている sage\_vmx をワードパッドなどのエディターで開いて最後に下記を書き加える。

なお、sage\_vmx の書き換えに失敗すると Sage が使えなくなるので、書き換える前にコピーを作って保存しておいた方がよい。

```
##### Shared Folders #####
```

```

sharedFolder0.enabled = "TRUE"
sharedFolder0.present = "TRUE"
sharedFolder0.writeAccess = "TRUE"
sharedFolder0.readAccess = "TRUE"
sharedFolder0.hostPath = "C:\SharedFolder"
sharedFolder0.guestName ="share"
sharedFolder0.eXpiration = "never"
sharedFolder.maxNum = "1"

```

このあと修正した sage\_vmx をダブルクリックして VMware Player を起動し、現れたウィンドウの上部の“VMware Player”の部分をクリックするとメニューが現れるので「共有フォルダ (F)」を選ぶ。出てきたウィンドウの「フォルダ」の欄に

```
share C:\SharedFolder
```

と出ていることを確認して、その上の「フォルダの共有」で「常に有効 (E)」のボタンを押して“OK”で閉じる。ここでエラーメッセージが出るが、気にせずクローズボタンで VMware Player をいったん終了する。この後 sage\_vmx を再びダブルクリックして Sage を起動して任意の worksheet に入り、

```
%sh ls /mnt/hgfs
```

を実行して 1 行目に share と表示されれば成功である。Unix や Linux などを使ったことがある人なら ls の意味は知っているはずだが、その前の %sh は Sage で Unix などのシェルコマンドを実行するために用いる。つまり share という名のフォルダがあることが、これで確認される。

Sage 側では /mnt/hgfs/share が共有フォルダとなる。つまり Windows 側で C:\SharedFolder にファイルを入れて置けば、Sage からは /mnt/hgfs/share の中からファイルが取り出せ、書き換えて保存すれば、Windows に戻って C:\SharedFolder にできたファイルがあるので読んだり取り出したりできる。C:\SharedFolder にサブフォルダを作れば、これも共有フォルダとなる。この文書の他のところで使っているファイルのパスなら /Users/ishida/keisanki の部分を /mnt/hgfs/share に置き換えて実行すればよい。(補足はここまで)

Python ではリストを作って

```

A = ['0123456789\n', 'abcdefghijklmnopqrstuvwxy\n']
f = open("/Users/ishida/keisanki/aaa", "w")
f.writelines(A)
f.close()

```

と一気に書き込むこともできる。ファイルの読み書きが終わったら `f.close()` として閉じる。

ファイルを読み込むときは

```
g = open("/Users/ishida/keisanki/aaa","r")
B = g.readlines()
g.close()
```

とすると、ファイル全体を一行ずつのリストとした `B` ができる。最初から一行ずつ読み込みたいときは `readline()` を使う。読み込んだ各行は数字であっても文字列なので、数値として扱うためには `int(B[0])` や `int(B[0])` とするか、文字列関数を使って `string.atoi(B[0])` や `string.atof(B[0])` とする。ただし文字列関数を使うには、先にプログラムに

```
import string
```

と書いておいて、`string` モジュールを読み込んでおく必要がある。

Python の文字列はシングルクォーテーションではさんで `'abc'` でも、ダブルクォーテーションではさんで `"abc"` でもよい。英文 He said "I don't know". を文字列にする場合、同じ記号が出てきた場合、`\` を入れて `'He said "I don\'t know".'` かまたは `"He said \"I don't know\"."` とする必要があるが、シングルクォーテーションかダブルクォーテーションを 3 つ使った `'''He said "I don't know".'''` も使える。この書き方は強力で、途中で改行があっても文字列の要素とみなす。

テキストファイルにおける改行のアスキーコードは Linux などの UNIX 系のファイルでは LF (0x0A) が使われるが、Windows では CR+LF (0x0D0A) が、MacOS では CR (0x0D) が標準的に使われるので注意が必要である。ファイル読み込みのモード `"r"` を `"Ur"` とすると改行コードを LF に変換して読み込む。

12 月 24 日にパソコンの歴史について話したときに、年について間違えていたので訂正しておきます。

日本で最初にパソコンとして一般的になった NEC PC8001 は 1979 年に発売されています。富士通の Micro 8 (FM-8) は 1981 年に発売されました。これらや NEC PC8801、富士通 FM-7 までは 8 ビット CPU で電源を入れると BASIC が起動するパソコンでした。16 ビット CPU で、主に MS-DOS 上で使われることになる PC-9801 は 1982 年に発売されています。

**ラプラス変換**：関数  $x(t)$  のラプラス変換は

$$\mathcal{L}(x(t), t, s) = \int_0^{\infty} e^{-st} f(t) dt$$

で定義され、微分方程式の初期値問題の解を求めるのに有効である。Sage ではラプラス変換の計算は Maxima という別の計算ソフトの関数を使う。関数  $x(t)$  の 2 回微分のラプラス変換は

```
view(maxima("laplace(diff(x(t),t,2),t,s)"))
```

でできる。3 回微分なら `view(maxima("laplace(diff(x(t),t,3),t,s)"))` とすればよい。  $1/(s+2) + 1/(s+5)$  の逆ラプラス変換は

```
view(inverse_laplace(1/(s+2) + 1/(s+5),s,t))
```

で求めることができる。微分方程式

$$y'' + 3y' + 2y = 0$$

の初期値問題  $y(0) = 4, y'(0) = 5$  を解きたいときは

```
x = function('x', t)
de = lambda y: diff(y,t,2) + 3*diff(y,t) + 2*y
soln = desolve_laplace(de(x(t)),["t","x"],[0,4,5])
print soln
view(eval(soln.replace("^","**").replace("%","")))
```

とする。下から 2 行目の `print soln` は必要ないが、これで Maxima が返した式の形がわかる。Sage で使える式にするため、文字列を Python の式の形に修正して、`eval` で式としている。`replace(s1,s2)` は “replace string s1 with string s2” (文字列 s1 を文字列 s2 で置き換える) という意味である。なお  $e^t$  を `e**t` に直しているのは Python で指数をこのように書くからである。

## 12月22日の宿題の解答例

```
#
# RSA 暗号鍵作成のプログラム例 (石田)
#
p = next_prime(ZZ.random_element(10^30)) # 30桁の素数 p
q = next_prime(ZZ.random_element(10^30)) # 30桁の素数 q
n = p*q
phi_n = (p-1)*(q-1)
print 'p = ', p, '\nq = ', q, '\nn = ', n, '\nphi_n = ', phi_n
while True: # このループで  $0 < e < n$  で  $(\text{phi}_n, e) = 1$  となる整数  $e$  をと
る.
    e = ZZ.random_element(n)
    if gcd(e, phi_n) == 1: break
d = lift(Mod(e, phi_n)^(-1)) # d は  $de$  の  $\text{phi}_n$  による剰余が 1 となる  $[0, n]$  の
整数.
print 'e = ', e, '\nd = ', d
mykey = [n.str(10)+'\n', d.str(10)+'\n'] # n と秘密鍵 d の保存
f0 = open("/Users/ishida/A/secretkey090119", "w")
f0.writelines(mykey)
f0.close()
openkey = [n.str(10)+'\n', e.str(10)+'\n'] # 公開鍵 n, e の保存 (B に送
る)
f1 = open("/Users/ishida/B/openkey090119", "w")
f1.writelines(openkey)
f1.close()

#
# RSA 暗号文作成のプログラム例 (石田)
#
f = open('/Users/ishida/B/Binfo.txt', 'r') # 読み込み用のファイルを開く
Binfo = f.readlines() # ファイルを読み込んで各文節を要素とするリスト Binfo を
作る
ParNum = len(Binfo) # リスト Binfo の長さ, すなわち文節の数を ParNum と
する.
```

```

print ParNum
for i in range(ParNum):
    print Binfo[i] # i 番目の文節を表示
f.close()
import string
f = open('/Users/ishida/B/openkey090119', 'r') # 公開鍵 n, e の読み込み
givenkey = f.readlines()
n = string.atoi(givenkey[0])
e = string.atoi(givenkey[1])
Hex = HexadecimalStrings() # 16 進数と文字の変換をするときの記号を Hex に
する.
N = 20
Bcode = []
for i in range(ParNum): # 文節ごとに暗号化の作業をする.
    l = len(Binfo[i])
    for j in range(integer_ceil(l/N)): # 第 i 文節を # 20 字ずつに分割
        して暗号化する.
        t = Binfo[i][N*j:N*(j+1)]
        a = Hex.encoding(t) # アスキーコードの列に変換する.
        m = Integer('0x'+str(a)) # 16 進数とみなして数値化する.
        Bcode.append(lift(Mod(m,n)^e).str(10)+'\n') # 得られた数値を配
列 Bcode の最後に追加する.
print len(Bcode) # 配列 Bcode の長さ
for i in range(len(Bcode)):
    print Bcode[i] # 数値 Bcode[i] を表示する.
g = open('/Users/ishida/A/Bcode.txt', 'w') # 作成した暗号の保存 (A に送
る)
g.writelines(Bcode)
g.close()

#
# RSA 暗号分平文化のプログラム例 (石田)
#
import string
Bdecode = '' # 平文化のために空の文字列を用意する.

```

```

g = open('/Users/ishida/A/Bcode.txt','r')
C = g.readlines()
g.close()
g = open('/Users/ishida/A/secretkey090119','r') # n と秘密鍵 d の読み込み
mykey = g.readlines()
n = string.atoi(mykey[0])
d = string.atoi(mykey[1])
g.close()
for i in range(len(C)):
    mmm = lift(Mod(string.atoi(C[i]),n)^d) # 鍵を使って数値を暗号化前に戻す。
    Asc = Hex(hex(mmm)) # 数値を 16 進数に変換し、アスキーコードの列とみなす。
    Bdecode += Asc.decoding() # アスキーコードを文字に変換し文字列の最後に追加する。
print Bdecode # 平文化した文字列を表示する。
g = open('/Users/ishida/A/Bdecode.txt','w') # 平文化した文字列を保存する。
g.write(Bdecode)
g.close()

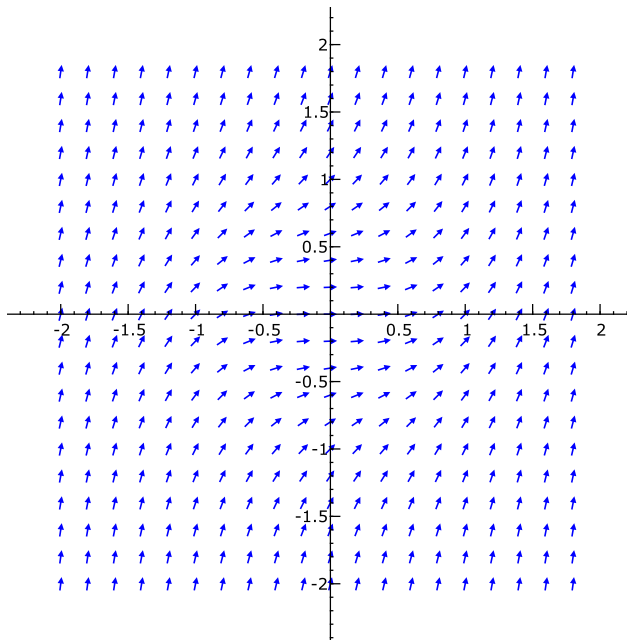
```

1 階の微分方程式

$$y' = f(x, y), \quad y(a) = c$$

は座標平面の  $(x, y)$  に傾き  $f(x, y)$  のベクトルを置くことにより解の関数のグラフが予測できる. 下記のプログラムにより,  $f(x, y) = x^2 + y^2$  の場合に間隔を  $1/5$  としてベクトルを配置する.

```
pts = [(-2+i/5,-2+j/5) for i in range(20) for j in range(20)]
f = lambda p:p[0]^2+p[1]^2
arrows = []
for p in pts:
    x = 0.1/sqrt(1+f(p)^2)
    y = 0.1*f(p)/sqrt(1+f(p)^2)
    a= arrow((p[0]-x/2,p[1]-y/2), (p[0]+x/2,p[1]+y/2), width=1, arrowsize=4)
    arrows.append(a)
show(sum(arrows), aspect_ratio=1)
```



なお, プログラムの実行とともに上記の図を pdf ファイルとして保存するためには, プログラムの最後に下記の 1 行を付け加える. /Users/ishida/aaa.pdf は保存するディレクトリのパスとファイル名である.

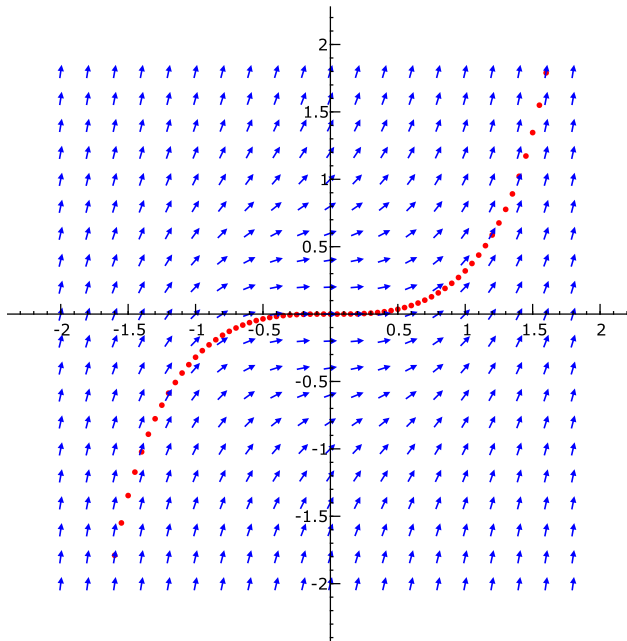
```
save(sum(arrows), '/Users/ishida/aaa.pdf', aspect_ratio=1)
```

初期値を  $y(0) = 0$  として  $x$  の値を  $1/20$  間隔で変化させて近似解を求める.  
 $x_i = i/20$  とすると,  $(x_0, y_0) = (0, 0)$  で  $i \geq 0$  では

$$\frac{y_{i+1} - y_i}{x_{i+1} - x_i} = f(x_i, y_i)$$

となるように  $(x_i, y_i)$  を帰納的に定める.  $i \leq 0$  の場合も同様である.

```
pts = [(-2+i/5,-2+j/5) for i in range(20) for j in range(20)]
f = lambda p:p[0]^2+p[1]^2
arrows = []
for p in pts:
    x = 0.1/sqrt(1+f(p)^2)
    y = 0.1*f(p)/sqrt(1+f(p)^2)
    a = arrow((p[0]-x/2,p[1]-y/2), (p[0]+x/2,p[1]+y/2), width=1, arrowsize=4)
    arrows.append(a)
pp = [point((0, 0), rgbcolor=(1,0,0))]
step = 0.05
x0 = 0; y0 = 0
for i in range(32):
    a = f((x0, y0))
    x0 += step
    y0 += step*a
    pp.append(point((x0, y0), rgbcolor=(1,0,0)))
x0 = 0; y0 = 0
for i in range(32):
    a = f((x0, y0))
    x0 -= step
    y0 -= step*a
    pp.append(point((x0, y0), rgbcolor=(1,0,0)))
show(sum(arrows)+sum(pp), aspect_ratio=1)
```



2 行目の  $f$  の定義を変えれば別の微分方程式の様子がわかる。

### 1 月 19 日の宿題 (期限 1 月 23 日)

$f$  を別の 2 変数関数に代えて試し、結果を報告する。初期値を変えてもよい。必要であれば平面の範囲や増分を変更してみる。微分方程式の知識で正しい解が求まる場合は、そのグラフを表示して見比べる。

グラフの書き方：

```
f = sin(x)
p = plot(f, -5, 5, color='green', thickness=2)
show(p, ymin = -2, ymax = 2, aspect_ratio = 1)
```

数学の多くの知識はインターネットで知ることができる。数学の勉強で知らない言葉が出てきたら、インターネットで検索して解説を探すことは多くの場合有効である。検索で見つかった記述がすべて正しいとは限らないが、Wikipedia の記述はかなり充実してきている。

無料で公開されている書籍や論文を見つけることもできる。

線形代数の英文教科書

<http://joshua.smcvt.edu/linearalgebra/>

<http://linear.ups.edu/>

解析学の英文教科書

<http://www.its.caltech.edu/~sean/book/unabridged.html>

正式の出版より前の数学の論文なら

<http://front.math.ucdavis.edu/math>

に沢山ある.

計算機数学 A (2009/1/26)

1 月 19 日の宿題：

例として変数分離型の方程式

$$y' = x(1 + y^2)$$

の場合、一般解は  $y = \tan(x^2/2 + C)$  となる。

```
pts = [(-2+i/5,-2+j/5) for i in range(21) for j in range(21)]
f = lambda p:float(p[0]*(1+p[1]^2))
arrows = []
for p in pts:
    u = 0.1/sqrt(1+f(p)^2)
    v = 0.1*f(p)/sqrt(1+f(p)^2)
    a = arrow((p[0]-u/2,p[1]-v/2), (p[0]+u/2,p[1]+v/2), width=1, arrowsize=4)
    arrows.append(a)
@interact
def _(n=(-10..10)):
    html('$y\'' = x(1+y^2)$')
    html('$y = \tan(x^2/2+C)$')
    pp = [point((0, n/10), rgbcolor=(1,0,0))]
    step = 0.05
    x0 = 0; y0 = n/10
    for i in range(32):
        a = f((x0, y0))
        x0 += step
        y0 += step*a
        pp.append(point((x0, y0), rgbcolor=(1,0,0)))
    x0 = 0; y0 = n/10
    for i in range(32):
        a = f((x0, y0))
        x0 -= step
        y0 -= step*a
        pp.append(point((x0, y0), rgbcolor=(1,0,0)))
    ppp = plot(tan(x^2/2+atan(n/10)), -2, 2, color='green', thickness=2)
    show(sum(arrows)+sum(pp)+ppp, ymin = -2, ymax = 2, aspect_ratio=1)
```

なお、上記プログラム中の

```
@interact
def _(n=(-10..10)):
```

の部分は Python のインタラクティブ機能で

```
fns = [sin,cos,tan,x^2,sqrt]
@interact
def _(n=(0..10),f=selector(fns,default=sin,label="function"),\
a=input_box('2',label="Value")):
    print float(f(n*int(a)))
```

のように使うことができる。  $n=(0..10)$  はスライダーを表示し、 $n$  に 0 から 10 の整数をスライダーを移動させて代入することができる。

$f=selector(fns,default=sin,label="function")$  はリスト  $fns$  のメンバーがメニュー表示され、選んだものが  $f$  に代入される。

$a=input\_box('2',label="Value")$  は入力枠が表示され、任意の文字列を  $a$  に代入することができる。ループや if 文と同じようにセミコロンを行末に付け、次の行からインデントして記述する。

提出されたレポートにあった方程式も 2 つ紹介しよう。方程式  $y' = y + xy^2$  の場合は、一般解は  $y = 1/(Ce^{-x} - x + 1)$  となる。

```
pts = [(-2+i/5,-2+j/5) for i in range(21) for j in range(21)]
f = lambda p:float(p[1] + p[0]*p[1]^2)
arrows = []
for p in pts:
    u = 0.1/sqrt(1+f(p)^2)
    v = 0.1*f(p)/sqrt(1+f(p)^2)
    a = arrow((p[0]-u/2,p[1]-v/2),(p[0]+u/2,p[1]+v/2),width=1,arrowsize=4)
    arrows.append(a)
@interact
def _(n=(-10..10)):
    html('$y\ ' = y+xy^2$')
    html('$y = \\frac{1}{Ce^{-x}-x+1}$')
    pp = [point((0, n/10), rgbcolor=(1,0,0))]
    step = 0.05
    x0 = 0; y0 = n/10
```

```

for i in range(32):
    a = f((x0, y0))
    x0 += step
    y0 += step*a
    pp.append(point((x0, y0), rgbcolor=(1,0,0)))
x0 = 0; y0 = n/10
for i in range(32):
    a = f((x0, y0))
    x0 -= step
    y0 -= step*a
    pp.append(point((x0, y0), rgbcolor=(1,0,0)))
if n == 0:
    ppp = plot(0, -2, 2, color='green', thickness=2)
else: ppp = plot(1/((10/n-1)*e^-x-x+1),-2,2,color='green',thickness=2)
show(sum(arrows)+sum(pp)+ppp,ymin=-2,ymax=2,aspect_ratio=1)

```

2行目で関数に float を付けているのは、シンボリックな計算を行って実際の値を代入するのが遅くなり、計算に時間が多くかかるのを防ぐためである。sin 関数などが入る場合はこれをつけた方がよい。

方程式  $y' = (y-1)(xy-x-y)$  の場合は、一般解は  $y = (Ce^x - x - 1)/(Ce^x - x)$  となる。

```

pts = [(-2+i/5,-2+j/5) for i in range(21) for j in range(21)]
f = lambda p:float((p[1]-1)*(p[0]*p[1] - p[0] - p[1]))
arrows = []
for p in pts:
    u = 0.1/sqrt(1+f(p)^2)
    v = 0.1*f(p)/sqrt(1+f(p)^2)
    a = arrow((p[0]-u/2,p[1]-v/2),(p[0]+u/2,p[1]+v/2),width=1,arrowsize=4)
    arrows.append(a)
@interact
def _(n=(-10..10)):
    html('$y\prime = (y-1)(xy-x-y)$')
    html('$y = \frac{Ce^x-x-1}{Ce^x-x}$')
    pp = [point((0, n/10), rgbcolor=(1,0,0))]
    step = 0.05
    x0 = 0; y0 = n/10;

```

```

if n == 10:
    g(x) = 1
else:
    C = 10/(10 - n)
    g(x) = (x+1-C*e^x)/(x-C*e^x)
for i in range(32):
    a = f((x0, y0))
    x0 += step
    y0 += step*a
    pp.append(point((x0, y0), rgbcolor=(1,0,0)))
x0 = 0; y0 = n/10
for i in range(32):
    a = f((x0, y0))
    x0 -= step
    y0 -= step*a
    pp.append(point((x0, y0), rgbcolor=(1,0,0)))
ppp = plot(g(x), -2, 2, color='green', thickness=2)
show(sum(arrows)+sum(pp)+ppp, ymin=-2, ymax=2, aspect_ratio=1)

```

ラプラス変換による定数係数常微分方程式の解法：

微分方程式の教科書に記述があると思うが、先週紹介した解析学の英文教科書 <http://www.its.caltech.edu/~sean/book/unabridged.html> にも書かれている。

Sage を使って方程式  $y'' + 3y' + 2y = 0$  を初期値  $y(0) = 4, y'(0) = 5$  で解く場合は下記のようにする。

```

var('t')
x = function('x', t)
de = lambda y: diff(y,t,2) + 3*diff(y,t) + 2*y
soln = desolve_laplace(de(x(t)), ["t", "x"], [0, 4, 5])
view(eval(soln.replace("^", "**").replace("%", "")))

```

インタラクティブに係数や初期値を変えるプログラムもできる。

```

fns = [e^t, 0*t, 0, sin(t), t^2, 3*e^t]
var('t')
u = function('u', t)
@interact

```

```

def _(m=(0..10),n=(0..10),f = selector(fns,default=e^t,label="function"),\
a = input_box('0', label = "u(0)", b = input_box('0', label = "u'(0)")):
    html('$y\'' + %dy\'' + %dy + %s = 0$'%(m,n,latex(f(t))))
    de = lambda y: diff(y,t,2) + m*diff(y,t) + n*y + f(t)
    soln = desolve_laplace(de(u(t)),["t","u"],[0,float(a),float(b)])
    print '\nsolution = ',
    view(eval(soln.replace("^","**").replace("%","")))

```

この講義で示したように、最近のパソコンは非常に性能が良くなっているので、いろいろな計算を行って数学の勉強に役立ててください。