# 行列超幾何関連の hgm プログラム

Nobuki Takayama

2014.03.20

http://fe.math.kobe-u.ac.jp/Movies/oxvh/2014-03-11-jack-n-c-automatic
(movie)

# Wishart 分布に従う行列達の第一固有値 $\ell_1$ に関する累積分布関数

行列引数の超幾何積分 ${}_1F_1$. $X$ : $m \times m$ 実行列.

$$\int_{0<X<I_m} \exp(\mathrm{Tr}\, XY)|X|^{a-(m+1)/2}|I_m - X|^{c-a-(m+1)/2} dX,$$

$0 < X < I_m$ は $X$ と $I_m - X$ が正定値対称行列を意味する.
$dX = \prod_{i \le j} dx_{ij}$.

$$ {}_pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; Y) = \sum_{k=0}^{\infty} \sum_{\kappa \vdash k} \frac{(a_1)_\kappa \ldots (a_p)_\kappa}{(b_1)_\kappa \ldots (b_q)_\kappa} \frac{C_\kappa(Y)}{k!}.$$

$$\Pr[\ell_1 < x] = C \exp\left(-\frac{x}{2}\mathrm{Tr}\,\Sigma^{-1}\right) x^{\frac{1}{2}nm} {}_1F_1\left(\frac{m+1}{2}; \frac{n+m+1}{2}; \frac{x}{2}\Sigma^{-1}\right),$$

# 実装

1. 正規化定数 OpenXM/src/hgm/mh/src/, Koev-Edelman algorithm (2006, 級数による初期値) + hgm, hgm_jack-n + htm_w-n.
2. Asir package tk_jack.rr

–idata で指定するファイルでパラメータを指定. $\Sigma^{-1}/2$ が $\beta$

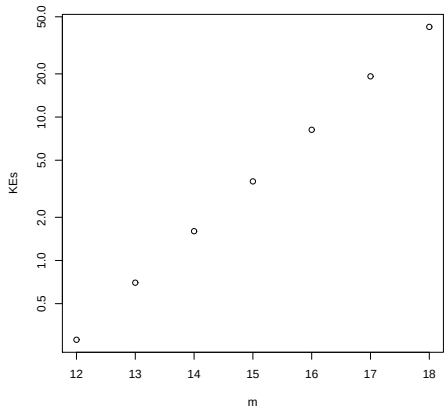
```
taka@orange3m:~/OX4L/OX64/OpenXM/src/hgm/mh/src>
 ./hgm_jack-n --idata Testdata/tmp-idata3.txt >t.txt
%%serror=0.00102773, M_assigned_series_error=1e-05, M_m_estimated_approx_deg=10
%%x0value_min=1e-60, x0g_bound=1e+100
%%F=2.86752,Ef=0.00108105,M_estimated_X0g=0.3, X0g=0.3
serror が 10^{-5} 以下となるように degree を上げる.
%%serror=4.30301e-06, M_assigned_series_error=1e-05, M_m_estimated_approx_deg=9
%%x0value_min=1e-60, x0g_bound=1e+100
%%F=2.86815,Ef=0.00108105,M_estimated_X0g=0.3, X0g=0.3
%Mg=      変数の数
3
%Beta[0]=
1.000000

%Iv[0]=     series で計算した初期値
2.86815
```

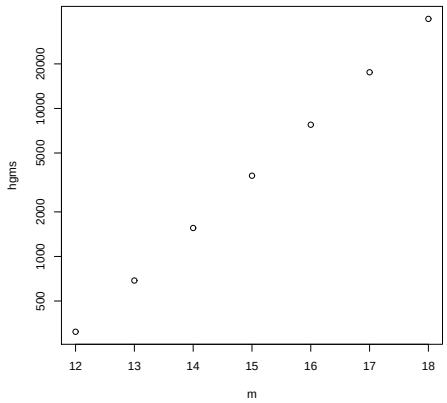
# 実装

```
%Ef=    1F1 以外の部分の値.
0.00108105
%Xng=   x はどこまで計算.
12.000000
%abserror=3.10061e-08      初期値の大きさ/10000 を RK の abserr に.

taka@orange3m:~/OX4L/OX64/OpenXM/src/hgm/mh/src>
   ./hgm_w-n --idata t.txt --verbose
  途中経過も出力するには, --gnuplotf test-g を option へ.
%strategy=1
%abserr=3.10061e-08, %relerr=1e-10
#MH_success=1
#MH_coeff_max=4.62963
#MH_estimated_start_step=0.0231836

x=12.001000, y[0]=0.999958
```

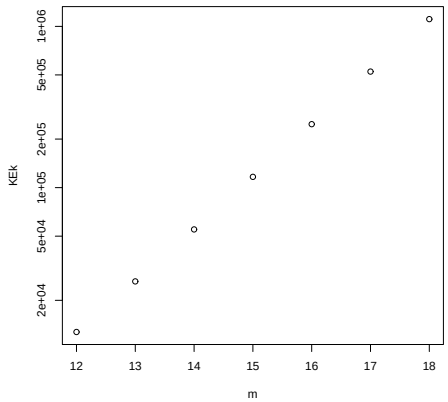
# 現在の実装の Timing Data

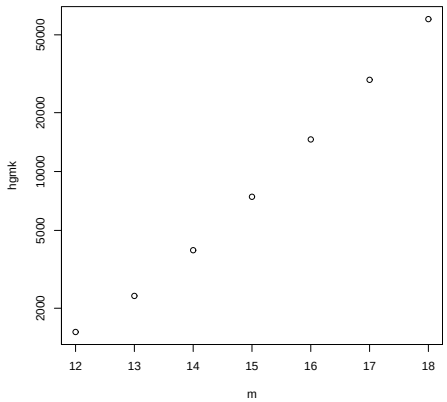
Intel(R) Xeon(R) CPU E5-4650 0 @ 2.70GHz.
$n = 3, \beta = (1.0, 1.2, 1.4, 1.6, \ldots)$, logarithmic scale, degree=6,
$x_0 = 0.1$, –automatic 0, –strategy 0 (h=0.001)




秒. hgm は 11900 点

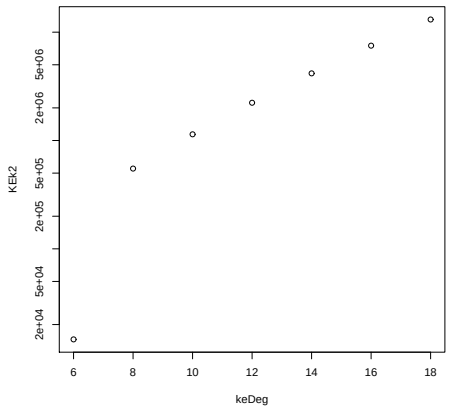


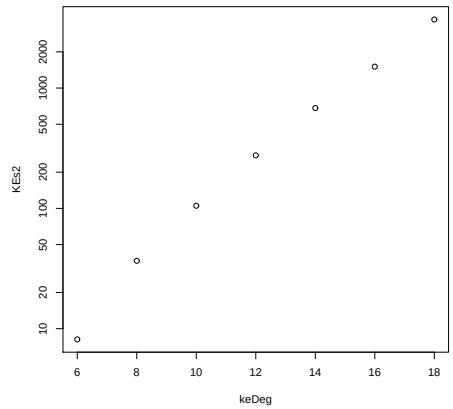

Kbytes.

# 精度のよい初期値が必要. どこまで近似 (degree をどう選ぶか)?

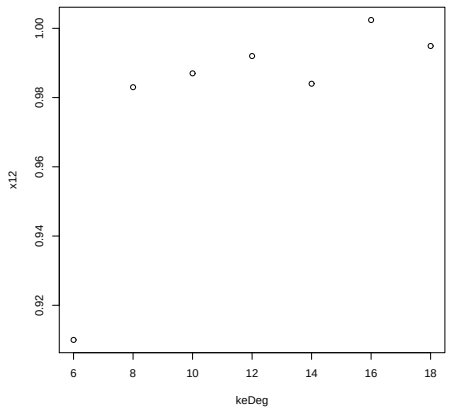
$m = 16$, $x_0 = 0.1$, –automatic 0, –strategy 0 (h=0.001) 最初二つは logarithmic scale.




KB,




秒. (Koev-Edelman)




$x = 12$ での値. 経験則: $m$ を増やすとメモリ、時間は指数関数的. degree を増やすと, メモリ, 時間は指数関数的.

# degree 自動設定の戦略

**少々短絡的戦略** (–automatic 1).

1. log(精度の桁がどれだけ足りないか) で degree を増す.
2. degree $\geq m$. (微分のタイプとランク).
3. 初期値が小さすぎるなら $x_0$ を増やす. 増やす割合は, ちいさすぎの桁のやはり log に比例させる.

`OpenXM/src/hgm/mh/src/jack-n.c` 関数 mh_t の 1373 行前後

**Adaptive Runge-Kutta 法** (rk.c, –strategy 1). $D_i$ = abserr + relerr*$|y_i|$ (ここで $y_i$ は解の $i$ 成分) と置くとき, 検出された誤差が $|D|$ 以上なら step size h を小さくする.

**経験的戦略**

1. 初期値 $y_0$ に
   (級数のエラー上限 [assigned_series_error])*10 をかけたものを abserror の推奨値とする.
2. Todo, relative error の推奨値はまだやってない. 1e-10 固定.

# 改造版の Timing と精度

OpenXM/src/hgm/mh/src/Testdata/m10-paper が論文の例.

$m = 10$, 論文の $x_0$, $n$, $\Sigma$. $x = 30$ での 値 (P) を求める.

| assigned err | 値 | degree | KE-m | KE-s | hgm-s |
|---|---|---|---|---|---|
| 論文 | 0.999545 | 20 | | | 75s |
| $10^{-5}$ | 0.999989 | 19 | 226M | 36m | 4h36m |
| $10^{-7}$ | 0.999998 | 23 | 922M | 2h37m | 4h38m |
| $10^{-8}$ | 0.999998 | 24 | 920M | 1h41m | 4h39m |

1. KE-s は 適切な degree を最初に与えると無駄な再計算がなくなるので早くなる. Todo, プログラムも工夫.
2. “論文” は adaptive Runge-Kutta を使っていない.

# 限界はどのあたり?

数ページ前
と同じパラメータ. ただし $x_0 = 0.2$ (経験的にこのあたりがベスト).

| m | assigned err | 値 | degree | KE-m | KE-s | hgm-s |
|---|---|---|---|---|---|---|
| 15 | $10^{-5}$ | 0.9959464 | 20 | 10G | 1h09m | 1h47m |
| 15 | $10^{-6}$ | 0.9989103 | 22 | 17G | 2h03m | 1h49m |
| 16 | $10^{-5}$ | 0.9945382 | 21 | 28G | 3h36m | 4h09m |
| 16 | $10^{-6}$ | Out of mem | | | | |