n_wishartd

n_wishartd User's Manual
Edition 1.0
Aug 2016

by Masayuki Noro

Copyright © Masayuki Noro 2016. All rights reserved.

1 matrix 1F1 の対角領域上への制限パッケージ n_wishartd.rr

このマニュアルでは, asir-contrib パッケージに収録されている, matrix 1F1 が対角領域上で満たす微分方程式系を計算するパッケージ‘n_wishartd.rr’ について解説する. このパッケージを使うには, まず‘n_wishartd.rr’ をロードする.

```
[...] load("n_wishartd.rr");
```

このパッケージの函数を呼び出すには, 全てn_wishartd. を先頭につける.

1.1 matrix 1F1 の対角領域上への制限

1.1.1 n_wishartd.diagpf

n_wishartd.diagpf(*m*,*blocks*)

*m*変数の 1F1 が満たす方程式を, *blocks* で指定される対角領域上に制限した微分方程式系を計算する.

return *[E1,E2,...]* なるリスト, 各*Ei* は部分分数を係数とする微分作用素で, 制限した 1F1 を零化する.

m 自然数

vars *[[s1,e1],[s2,e2],...]* なるリスト.

options 下の説明参照.

- *m*変数の 1F1 が満たす方程式を, *blocks* で指定される対角領域上に制限した微分方程式系を計算する.
- *blocks* の各成分*[s,e]* は*ys=y(s+1)=...=ye* を意味する. このブロックを代表する変数は*ye* である.
- *blocks* には全ての変数が現れるように指定する. 特に, 一つの変数からなるブロックは*[s,s]* と指定する.
- この関数が常に成功することは証明されていないが, 現在のところ, 各変数ブロックサイズが 36 以下なら成功することは証明されている.
- 出力される微分作用素のフォーマットに関しては⟨undefined⟩ [部分分数を係数とする微分作用素], p. ⟨undefined⟩, を参照.

```
[2649] Z=n_wishartd.diagpf(5,[[1,3],[4,5]]);
[
 [[[[-1,[]]],(1)*<<0,0,0,0,3,0>>],
  [[[-2,[[y1-y4,1]]],[-2,[[y4,1]]]],(1)*<<0,1,0,0,1,0>>],
  [[[9/2,[[y1-y4,1]]],[-3*c+11/2,[[y4,1]]],[3,[]]],(1)*<<0,0,0,0,2,0>>],
  ...
  [[[-6*a,[[y1-y4,1],[y4,1]]],[(4*c-10)*a,[[y4,2]]],[-4*a,[[y4,1]]]],
   (1)*<<0,0,0,0,0,0>>]],
 [[[[-1,[]]],(1)*<<0,4,0,0,0,0>>],

  [[[-6,[[y1-y4,1]]],[-6*c+10,[[y1,1]]],[6,[]]],(1)*<<0,3,0,0,0,0>>],
  [[[5,[[y1-y4,1]]],[-5,[[y1,1]]]],(1)*<<0,2,0,0,1,0>>],
  ...
```

```
      [[[21*a,[[y1-y4,2],[y1,1]]],[(36*c-87)*a,[[y1-y4,1],[y1,2]]],
       [-36*a,[[y1-y4,1],[y1,1]]],[(18*c^2-84*c+96)*a,[[y1,3]]],
       [-9*a^2+(-36*c+87)*a,[[y1,2]]],[18*a,[[y1,1]]]],(1)*<<0,0,0,0,0,0>>]]
    ]
```

1.1.2 n_wishartd.message

`n_wishartd.message(`*onoff*`)`
: 計算中のメッセージ出力を on/off する.

onoff : *onoff=1* のときメッセージを出力し, *onoff=0* のときしない.

- 計算中のメッセージ出力を on/off する. デフォルトは off である.

1.2 制限した関数の計算

1.2.1 n_wishartd.prob_by_hgm

`n_wishartd.prob_by_hgm(`*m*`,`*n*`,`*[p1,p2,...]*`,`*[s1,s2,...]*`,t[|`*options*`])`
: HGM により重複固有値を持つ共分散行列に対する Wishart 行列の最大固有値の分布関数の値を計算する.

return

m : 変数の個数

n : 自由度

[p1,p2,...] : 重複固有値の個数のリスト

[s1,s2,...] : 各重複固有値

- 固有値*si* を*pi* 個もつ対角行列を共分散行列とする Wishart 行列の最大固有値*l1*の分布関数の値*Pr[l1<t]* を計算する.
- ステップ数を指定したルンゲ = クッタ法を, ステップ数を 2 倍しながら一つ前の計算結果との相対誤差が*eps* (デフォルトで*10^(-4)*) になるまで繰り返す.
- *eq* オプション指定がない場合, *[p1,p2,...]* で指定される対角領域に制限した微分方程式系を計算する. 指定がある場合, オプションとして指定されたリストをチェックなしに制限した方程式と見なして計算する.
- *eps*オプションが指定された場合, 指定された値を*eps* として計算する.
- *td* オプションが指定された場合, 初期ベクトル計算のためのべき級数を*td* で指定された全次数まで計算する(デフォルトは 100).
- *rk* オプションが指定された場合, 指定された次数のルンゲ = クッタ法を用いる. 許される値は 4 または 5, でデフォルトは 5 である.
- べき級数解の計算の困難さ, およびパフィアン行列の計算の困難さから, ブロック数が 2 以下の場合にのみ実用性がある.

```
[...] n_wishartd.message(1)$
[...] P=n_wishartd.prob_by_hgm(10,100,[9,1],[1/100,1],100|eps=10^(-6));
...
[x0=,8/25]
Step=10000
```

```
[0]
[8.23700622458446e-17,8.23700622459772e-17]
...
Step=1280000
[0][100000][200000][300000]...[900000][1000000][1100000][1200000]
[0.516246820120598,0.516246820227214]
[log ratio=,4.84611265040128]

Step=2560000
[0][100000][200000][300000]...[2200000][2300000][2400000][2500000]
[0.516246912003845,0.516246912217004]
[log ratio=,4.93705929488356]
[diag,18.6292,pfaffian,1.09207,ps,41.0026,rk,213.929]
0.516246912217004
266.4sec + gc : 8.277sec(276.8sec)
```

1.2.2 n_wishartd.prob_by_ps

`n_wishartd.prrob_by_ps(`*m*`,`*n*`,`*[p1,p2,...]*`,`*[s1,s2,...]*`,t[|`*options*`])`
べき級数により重複固有値を持つ共分散行列に対する Wishart 行列の最大固有値の分布関数の値を計算する.

m 変数の個数

n 自由度

[p1,p2,...] 重複固有値の個数のリスト

[s1,s2,...] 各重複固有値

- 直前の値との相対誤差が*eps* (デフォルト値は*10^(-4)*) 以下になるまで, べき級数を全次数ごとに計算する. その値から分布関数の値を計算して返す.
- *eps*オプションが指定された場合, 指定された値を*eps* として計算する. *eq* オプション指定がない場合, *[p1,p2,...]* で指定される対角領域に制限した微分方程式系を計算する. 指定がある場合, オプションとして指定されたリストをチェックなしに制限した方程式と見なして計算する.
- t の値が小さい場合にのみ実用的に用いることができる.

```
[...] Q=n_wishartd.prob_by_ps(10,100,[9,1],[1/100,1],1/2);
...
[I=,109,act,24.9016,actmul,0,gr,19.7852]
2.69026137621748e-165
61.69sec + gc : 2.06sec(64.23sec)
[...] R=n_wishartd.prob_by_hgm(10,100,[9,1],[1/100,1],1/2|td=50);
[diag,15.957,pfaffian,1.00006,ps,5.92437,rk,1.29208]
2.69026135182769e-165
23.07sec + gc : 1.136sec(24.25sec)
```

1.2.3 n_wishartd.ps

`n_wishartd.ps(`*z*`,`*v*`,`*td*`)`
微分方程式系のべき級数解を指定された全次数まで計算する.

return 多項式リスト

z 部分分数係数の微分作用素のリスト

v 変数リスト

- 結果は*[p,pd]* なるリストで, p はtd 次まで求めたべき級数解, pd はp のtd 次部分である.
- z は, v に指定される変数以外のパラメタを含んではいけない.

```
[...] Z=n_wishartd.diagpf(10,[[1,5],[6,10]])$
[...] Z0=subst(Z,a,(10+1)/2,c,(10+100+1)/2)$
[...] PS=n_wishartd.ps(Z0,[y1,y6],10)$
[...] PS[0];
19723078950274338395363 9/230438384724900975787223158176000*y1^10+
...
+(67388425421319768716722 33/10119537066347794279570342689043 20*y6^9
...+3932525/62890602*y6^2+1025/4181*y6+55/111)*y1
+19723078950274338395363 9/230438384724900975787223158176000*y6^10
+...+1395815/62890602*y6^3+3175/25086*y6^2+55/111*y6+1
```

1.3 部分分数係数の微分作用素

1.3.1 部分分数の表現

matrix 1F1 が満たす微分方程式の係数は*1/yi, 1/(yi-yj)* の定数倍の和として書かれている. さらに, ロピタル則を用いた対角領域への制限アルゴリズムの結果も同様に部分分数の和として書ける.

- 分母に現れる*yi0^n0(yi1-yj1)^n1(yi2-yj2)^n2...(yik-yjk)^nk* は*[[yi0,n0],[yi1-yj1,n1],...,[yik-yjk,nk]]* なる形のリストとして表現される. ここで, 各因子*yi-yj* は*i>j* を満たし, さらに因子はある一定の順序で整列される.
- f を上のようなべき積とし, c を定数とするとき, 単項式にあたる*c/f* は*[c,f]* で表現される. *f=[]* の場合, 分母が 1 であることを意味する.
- 最後に, *c1/f1+...+ck/fk* は*[[c1,f1],...,[ck,fk]]* と表現される. ここでも, 各項はある一定の順序で整列される.
- 部分分数を通分して簡約した結果, 0 になることもあることに注意する.

1.3.2 部分分数係数の微分作用素の表現

前節の部分分数を用いて, それらを係数とする微分作用素が表現できる. *f1,...,fk* を部分分数の表現, *d1,...,dk* を分散表現単項式(現在設定されている項順序で*d1>...>dk*) とするとき, 微分作用素*f1*d1+...+fk*dk* が*[f1,d1],...[fk,dk]]*で表現される.

1.3.3 部分分数係数の微分作用素の演算

1.3.3.1 n_wishartd.wsetup

`n_wishartd.wsetup(`*m*`)`

m 自然数

- *m* 変数の計算環境をセットする. 変数は*y0,y1,...,ym, dy0,...,dym* で*y0, dy0* は中間結果の計算のためのダミー変数である.

1.3.3.2 n_wishartd.addpf

`n_wishartd.addpf(`*p1*`,`*p2*`)`

return 部分分数係数の微分作用素

p1, *p2* 部分分数係数の微分作用素

- 微分作用素*p1*, *p2* の和を求める.

1.3.3.3 n_wishartd.mulcpf

`n_wishartd.mulcpf(`*c*`,`*p*`)`

return 部分分数係数の微分作用素

c 部分分数

p 部分分数係数の微分作用素

- 部分分数c と微分作用素p の積を計算する.

1.3.3.4 n_wishartd.mulpf

`n_wishartd.mulpf(`*p1*`,`*p2*`)`

return 部分分数係数の微分作用素

p1, *p2* 部分分数係数の微分作用素

- 微分作用素*p1*, *p2* の積を計算する.

1.3.3.5 n_wishartd.muldpf

`n_wishartd.muldpf(`*y*`,`*p*`)`

return 部分分数係数の微分作用素

y 変数

p 部分分数係数の微分作用素

- 変数y に対し, 微分作用素dy とp の微分作用素としての積を計算する.

```
[...] n_wishartd.wsetup(4)$
[...] P=n_wishartd.wishartpf(4,1);
[[[[1,[]]],(1)*<<0,2,0,0,0>>],[[[1/2,[[y1-y2,1]]],[1/2,[[y1-y3,1]]],
...,[[[-a,[[y1,1]]]],(1)*<<0,0,0,0,0>>]]
[...] Q=n_wishartd.muldpf(y1,P);
[[[[1,[]]],(1)*<<0,3,0,0,0>>],[[[1/2,[[y1-y2,1]]],[1/2,[[y1-y3,1]]],
...,[[[a,[[y1,2]]]],(1)*<<0,0,0,0,0>>]]
```

1.4 Runge-Kutta 法の試験的実装

`n_wishartd.ps_by_hgm` では, パフィアン行列を計算したあと, 与えられたステップ数で Runge-Kutta 法を実行して近似解の値を計算する組み込み関数`rk_ratmat` を実行している. この関数を, 値が与えられた精度で安定するまでステップ数を 2 倍しながら繰り返して実行する. `rk_ratmat` 自体, ある程度汎用性があるので, ここでその使用法を解説する.

1.4.1 rk_ratmat

`rk_ratmat(`*`rk45,num,den,x0,x1,s,f0`*`)`
有理関数係数のベクトル値一階線形常微分方程式系を Runge-Kutta 法で解く

return 実数のリスト

rk45 4 または 5

num 定数行列の配列

den 多項式

x0, x1 実数

s 自然数

f0 実ベクトル

- 配列*num* のサイズをk とするとき, *P(x)=1/den(num[0]+num[1]x+...+num[k-1]x^(k-1))* に対し$dF/dx = P(x)F$, *F(x0)=f0* を Runge-Kutta 法で解く.
- *rk45* が 4 のとき 4 次 Runge-Kutta, 5 のとき 5 次 Runge-Kutta アルゴリズムを実行する. 実験的実装のため, adaptive アルゴリズムは実装されていない.
- *s* はステップ数で, 刻み幅は*(x1-x0)/s* である.
- *f0* がサイズn のとき, *num* の各成分はn 次正方行列である.
- 結果は, 長さs の実数リスト*[r1,...,rs]* で, *ri* はi ステップ目に計算された解ベクトルの第 0 成分である. 次のステップに進む前に解ベクトルを*ri* で割るので, 最終的に解*F(x1)* の第 0 成分が*rs*r(s-1)*...*r1* となる.
- 方程式が線形なので, Runge-Kutta の各ステップも線形となることを利用し, 第 0 成分を 1 に正規化することで, 途中の解の成分が倍精度浮動小数の範囲に収まることを期待している. 初期ベクトル*f0* の成分が倍精度浮動小数に収まらない場合は, *f0* を正規化してから`rk_ratmat` を実行し, 前項の結果に*f0* の第 0 成分をかければよい.

```
[...] F=ltov([sin(1/x),cos(1/x),sin(1/x^2),cos(1/x^2)]);
[ sin((1)/(x)) cos((1)/(x)) sin((1)/(x^2)) cos((1)/(x^2)) ]
[...] F0=map(eval,map(subst,F,x,1/10));
[ -0.54402111088937 -0.839071529076452 -0.506365641109759 0.862318872287684 ]
[...] N0=matrix(4,4,[[0,0,0,0],[0,0,0,0],[0,0,0,-2],[0,0,2,0]])$
[...] N1=matrix(4,4,[[0,-1,0,0],[1,0,0,0],[0,0,0,0],[0,0,0,0]])$
[...] N=ltov([N0,N1])$
[...] D=x^3$
[...] R=rk_ratmat(5,N,D,1/10,10,10^4,F0)$
[...] for(T=R,A=1;T!=[];T=cdr(T))A *=car(T)[1];
[...] A;
0.0998334
[...] F1=map(eval,map(subst,F,x,10));
[ 0.0998334166468282 0.995004165278026 0.00999983333416666 0.999950000416665 ]
```

Index

（インデックスがありません）

（インデックスがありません）

簡単な目次

1 matrix 1F1 の対角領域上への制限パッケージ n_wishartd.rr 1
Index 7

目次

1 matrix 1F1 の対角領域上への制限パッケージ n_wishartd.rr 1

1.1 matrix 1F1 の対角領域上への制限 1
1.1.1 `n_wishartd.diagpf` 1
1.1.2 `n_wishartd.message` 2
1.2 制限した関数の計算 2
1.2.1 `n_wishartd.prob_by_hgm` 2
1.2.2 `n_wishartd.prob_by_ps` 3
1.2.3 `n_wishartd.ps` 3
1.3 部分分数係数の微分作用素 4
1.3.1 部分分数の表現 4
1.3.2 部分分数係数の微分作用素の表現 4
1.3.3 部分分数係数の微分作用素の演算 4
1.3.3.1 `n_wishartd.wsetup` 4
1.3.3.2 `n_wishartd.addpf` 5
1.3.3.3 `n_wishartd.mulcpf` 5
1.3.3.4 `n_wishartd.mulpf` 5
1.3.3.5 `n_wishartd.muldpf` 5
1.4 Runge-Kutta 法の試験的実装 5
1.4.1 `rk_ratmat` 6

Index 7