# 常微分 (差分) 方程式用有理数対応数値解析パッケージ

高山信毅 (神戸大, 理)

— Numerical linear ODE over $\mathbf{Q}$

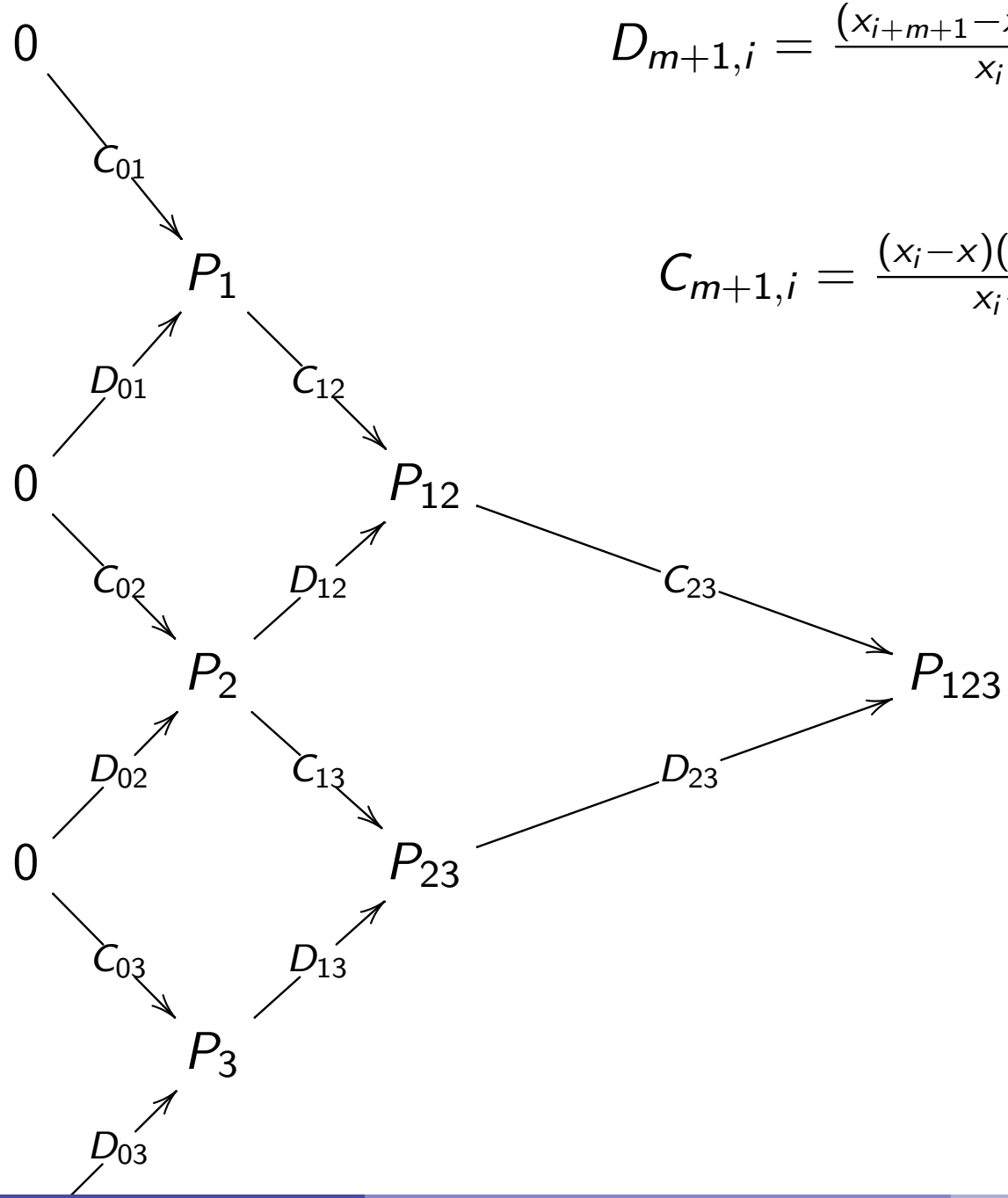
# 補間アルゴリズム, Neville algorithm


1. Press, Tuekolsky, Vetterling, Flannery, Numerical Recipes, 3rd ed, 2007.
2. J.Stoer, R.Bulirsch, Introduction to Numerical Analysis, 1980.


$x_i$ で値 $P_i$ をとる多項式の $x$ での値を評価したい.

$$P_{i(i+1)\cdots(i+m)} = \frac{(x - x_{i+m})P_{i(i+1)\cdots(i+m-1)} + (x_i - x)P_{(i+1)(i+2)\cdots(i+m)}}{x_i - x_{i+m}}$$

$$\begin{aligned} C_{m,i} &= P_{i\cdots(i+m)} - P_{i\cdots(i+m-1)} \\ D_{m,i} &= P_{i\cdots(i+m)} - P_{(i+1)\cdots(i+m)} \end{aligned}$$




$$D_{m+1,i} = \frac{(x_{i+m+1}-x)(C_{m,i+1}-D_{m,i})}{x_i - x_{i+m+1}}$$

$$C_{m+1,i} = \frac{(x_i-x)(C_{m,i+1}-D_{m,i})}{x_i - x_{i+m+1}}$$

# Package tk_exterpolate.rr

```
[3758] tk_exterpolate.initCD([x1,x2,x3],[p1,p2,p3]);
[3759] tk_exterpolate.showCD();
CC=
[ 0 p1 p2 p3 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[3764] tk_exterpolate.nextCD_poly(0,1,x);
// C_{12}, D_{12} を求める.
//  C_{mi}, D_{mi} は nextCD_poly(m+1,i-m,x)
[3765] tk_exterpolate.nextCD_poly(0,2,x);
// C_{13}, D_{13} を求める.
[3766] tk_exterpolate.showCD();
CC=
[ 0 p1 p2 p3 ]
[ 0 ((p1-p2)*x+(-p1+p2)*x1)/(x1-x2) ((p2-p3)*x+(-p2+p3)*x2)/(-x3+x2) 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
```

# 有理式による補間, Stoer-Bulirsch algorithm

$$X_{i,m} = \frac{x - x_i}{x - x_{i+m+1}}$$

$$\begin{aligned} D_{m+1,i} &= \frac{C_{m,i+1}(C_{m,i+1} - D_{m,i})}{X_{i,m} D_{m,i} - C_{m,i+1}} \\ C_{m+1,i} &= \frac{X_{i,m} D_{m,i}(C_{m,i+1} - D_{m,i})}{X_{i,m} D_{m,i} - C_{m,i+1}} \end{aligned}$$

$m = 2\nu$, $m+1$ 個の点をとおる分母分子が $\nu$ 次の有理式.
$m = 2\nu + 1$, $m+1$ 個の点をとおる分母が $\nu + 1$ 次, 分子が $\nu$ 次の有理式.

```
import("tk_exterpolate.rr");
[3665] G=tk_exterpolate.rat_extpl([x1,x2,x3],[p1,p2,p3],x);
(((p2-p3)*p1*x1+(p3*p1-p3*p2)*x3+(-p2*p1+p3*p2)*x2)*x+((-p2*p1+p3*p2)*x3+(p
[3668] red(base_replace(G,[[x,x1]]));
p1
```

# ODE を解く Bulirsch-Stoer method

入力: 微分方程式 $Y' = P(x)Y$ の $P(x)$, $x = t$ での $Y$ の値 $Y(t)$. 数 $H$.

出力: $x = t + H$ での $Y$ の値の近似値 $Y(t + H)$.

For $n \in [2, 4, 6, 8, 12, 16, 24, 32, 48, 64, 96]$,

1. $h = H/n$ を step size として $Y(t)$ より $Y(t + H)$ を通常の数値解法で解く. 値は $n$ に依存するので $Y(t + H, n)$ と書く.
2. $Y(t + H, 2), Y(t + H, 4), \ldots, Y(t + H, n)$ より BS 法有理補間で $Y(t + H)$ の値をきめる. 値が十分妥当なら for を break

例: $Y' = Y, Y(0) = 1$ を Euler 差分法で解くことは次の漸化式で $Y(t)$ を決めていくことに他ならない.

$$Y(t + h) = Y(t) + hY(t) = (1 + h)Y(t).$$

$H = 1$ としよう. この時, $h = H/n = 1/h$, $n = 1/h$ で, $Y(H, n) = (1 + h)^n$ である. BS 法では, $(1 + h)^{1/h} = (1 + 1/n)^n$ の値達から $h = 0$ での値を有理補間推定をする.

# 有理数対応版 BS algorithm

$M(k)$ をパラメータ $k$ に依存する行列とするとき,

$$M(n)M(n-1)\cdots M(2)M(1)$$

を計算する問題を matrix factorial 計算問題とよぶことにする.

1. matrix factorial 計算には chinese remainder Th と分散計算を用いた modular 計算が有効. 応用: 分割表の超幾何分布に対する正規化定数の計算. (橘, 後藤, 高山 2015–, `gtt_ekn.rr`).
2. 通常の linear ODE の数値解法は, 解法に応じた行列 $M(t,h)$ に対して漸化式

   $$Y(t+h) = M(t,h)Y(t)$$

   で $Y$ を決めていくので matrix factorial 計算を使える.

```
import("gtt_ekn.rr");
R=gtt_ekn.g_mat_fac_itor(Y,M,1,N,1,k,Tk_bs_plist,Tk_bs_idl
```

# Package tk_rk-mfac.rr, tk_bs.rr

```
[3769] tk_bs.mid_point_mat(newmat(2,2,[[0,1],[x,0]]),x,t,h);
// 修正中点法の行列 M.
[ 0 0 1 0 ]
[ 0 0 0 1 ]
[ 1 0 0 2*h ]
[ 0 1 2*h*t 0 ]

[3821] tk_rk.rk4_mat(newmat(2,2,[[0,1],[x,0]]),x,t,h);
// 4 次の Runge-Kutta 法の行列 M.
[ 1/24*h^4*t^2+(1/48*h^5+1/2*h^2)*t+1/6*h^3+1 1/6*h^3*t+1/12*h^4+h ]
[ 1/6*h^3*t^2+(1/6*h^4+h)*t+1/24*h^5+1/2*h^2 1/24*h^4*t^2+(1/16*h^5+1/2*h^2
```

# BS algorithm /**Q**

1. 漸化式を解く部分は 分散計算 chinese remainder 版 matrix factorial を用いる (橘, 後藤, 高山 2015–, `gtt_ekn.rr`).

   ```
   gtt_ekn.g_mat_fac_itor(Y,M,1,N,1,k,Tk_bs_plist,Tk_bs_idl);
   ```

2. 微分方程式の近似解法の誤差より精密な巨大有理数で計算しても意味がないので, BS 法の step 毎に巨大有理数は連分数近似で小さい有理数へ. `tk_approx_r.cont_frac(R,Err,Err_rel)`

有理数の利点は, double の演算に伴う誤差 (たとえば分配法則等も成立せず) の解析が不要. 近似誤差の解析と連分数近似の誤差の解析でよいのでより明快.
開発の動機: 小原-高山, ot_hgm_ahg.rr. 不確定特異点の近傍で超高精度で計算しないと解が不安定. 有理数 Runge-Kutta 法で HGM. これを高速化したかった.

# 例

$$y'' - \frac{x}{1-x^2}y' = 0, \quad y(0) = 0, y'(0) = 2$$

答えは $y = 2\arcsin(x)$. $x = 1$ の近くでの近似値を求めたい.

```
[3405] load("tk_bs.rr");
[3439] tk_bs.bs_test3();
Time=9/10, Y=[ 1552/693 3377/736 ]=[ 2.23953823953824 4.5883152173913 ], 2*asin(0.9)=2.23954
Time=99/100, Y=[ 1192/417 16602/1171 ]=[ 2.85851318944844 14.1776259607173 ], 2*asin(0.99)=2.85851
Time=999/1000, Y=[ 1639/537 60881/1361 ]=[ 3.05214152700186 44.7325495958854 ], 2*asin(0.999)=3.05214
Time=9999/10000, Y=[ 2363/759 38609/273 ]=[ 3.1133069828722 141.424908424908 ], 2*asin(0.9999)=3.11331
Time=99999/100000, Y=[ 5172/1651 66635/149 ]=[ 3.13264688067838 447.214765100671 ], 2*asin(0.99999)=3.13265
Time=999999/1000000, Y=[ 2081/663 984293/696 ]=[ 3.13876319758673 1414.21408045977 ], 2*asin(0.999999)=3.13876
H seems to be too large. H<-H/2
Time=19999989/20000000, Y=[ 1733/552 795188/417 ]=[ 3.13949275362319 1906.92565947242 ], 2*asin(0.999999)=3.139
--- snip   38 iterations
Time=19999989999999999999999999999999999999
    /20000000000000000000000000000000000000,
Y=[ 1372/437 29418007/14709 ]
 =[ 3.1395881006865 2000.00047589911 ],
2*asin(0.999999)=3.13959
```

# BS 補間法を HGM に

HGM は微分方程式, 差分方程式を用いて, 定積分の値や和の値を計算する方法.

HGM の欠点: 方程式の特異点の場所では値が計算できない.

例:

```
[3823] import("tk_fd.rr");
[5146] M=tk_fd.umat_abc(a,[-4],3);
[ 1 (4*x_1)/((x_1-1)*a) ]
[ 1 (-a+4*x_1+2)/((x_1-1)*a) ]
```

$$F(a+1) = \begin{pmatrix} 1 & \frac{4x_1}{(x_1-1)a} \\ 1 & \frac{-a+4x_1+2}{(x_1-1)a} \end{pmatrix} F(a)$$

$F(a)$ の第一成分は ${}_2F_1(a,-4,3;x_1)$ で多項式:

$$ {}_2F_1(a,-4,3;x_1) = 1 + \frac{a(-4)}{1\cdot 3}x_1 + \frac{a(a+1)(-4)(-3)}{2!3\cdot 4}x_1^2 + \cdots + \frac{a(a+1)(a+2)(a+3)(-4)(-3)(-2)(-1)}{4!3\cdot 4\cdot 5\cdot 6}x_1^4$$

$x_1 = 1$ での値も存在するが, 上の漸化式は方程式の特異点 $x_1 = 1$ では使えない.

# BS 補間の HGM への応用

1. 特異点の外で HGM を適用して計算.
2. BS 補間で特異点での値を計算.

答えが有理式なら厳密値が求まる.

$A$ を $d \times n$ 行列, 整数成分. $d \leq n$. $A$ の rank は $d$. $A$ の row span は $(1, 1, \ldots, 1)$ を含む. $\beta \in \mathbf{Z}^d_{\geq 0}$.

$$Z(\beta; p) = \sum_{Au=\beta, u \in \mathbf{Z}^n_{\geq 0}} \frac{p^u}{p!}$$

が次の条件付き分布の正規化定数

$$P(U = u \,|\, Au = \beta) = \frac{p^u}{u!} / Z$$

特に $A = (1, 1, \ldots, 1)$ なら, これは多項分布.
例: $2 \times 2$ 分割表で行和列和が一定の条件付き分布を考える時の $A$ は

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

# BS algorithm の 0-cell 問題への応用

1. 後藤, 松本. Pfaffian equations and contiguity relations of the hypergeometric function of type (k+1,k+n+2) and their applications. arxiv:1602.01637 $p$ が特異点の外なら HGM で計算できる.
2. 橘, 後藤, 高山 (2016), 2 元分割表に対する差分ホロノミック勾配法の実装 (RIMS 講究録に掲載予定). 分散計算と chinese remainder で上記計算を効率化.

しかし特異点の上で計算できないので, たとえば 0-cell ($p_i = 0$) がある時の計算ができない. (分割表 0-cell 問題).

**命題**

$A$ の第一行が $(1, 1, \ldots, 1)$ とする. $p \in \mathbf{R}^n_{>0}$ 空間での直線上 $L$ にある異なる $2\beta_1$ 個の点での $E[U_i]$ の値を計算すれば $E[U_i]$ の $L$ 上の任意の点での値が求まる.

$E[U_i]$ を $L$ に制限すると分母分子が高々 $\beta_1$ 次の有理式になる.

# 例

$$\begin{array}{ccc|c} * & * & * & 3 \\ * & * & * & 4 \\ * & * & * & 3 \\ \hline 3 & 4 & 3 & \end{array}, p = \begin{pmatrix} 1 & 1/2 & 0 \\ 1 & 1/3 & 1/4 \\ 1 & 1 & 1 \end{pmatrix}$$

```
[5150] import("gtt_ekn.rr");
0
[5151] E=gtt_ekn.cBasistoE_0(0,[[3,4,3],[3,4,3]],[[1,1/2,0],[1,1/3,1/4],[1,
[ 71076/56575 98649/56575 0 ]
[ 157581/113150 28069/22630 77337/56575 ]
[ 39717/113150 114957/113150 92388/56575 ]
[5153] number_eval(E);  // 頻度の期待値.
[ 1.25631462660186 1.74368537339814 0 ]
[ 1.39267344233319 1.2403446752099 1.36698188245692 ]
[ 0.351011931064958 1.01596995139196 1.63301811754308 ]
```

# HGM への BS algorithm の応用まとめ

— Numerical linear ODE over $\mathbf{Q}$, ODE=ordinary differential or difference equation

| 問題 | HGM | |
|---|---|---|
| 分割表 0-cell 問題 | 後藤による特異点上での漸化式 | `ekn_pfaffian_2.rr` |
| Wishart, 固有値退化問題 | 野呂による特異点上での方程式 | `n_wishartd.rr` |

BS 有理補間, 多項式補間は両方の問題に適用可能. 前者は厳密値, 後者は近似値.

1. 利点: 汎用かつ実装が容易.
2. 欠点: 多くの点での評価をするので遅い. ただし分散計算は可能.

課題

1. 実装の高速化.
2. numerical ODE over $\mathbf{Q}_p$

# 関連して開発したツール等

## oxlistusage

```
def bs_next(P,Y0,X,X0,H,Eps)
"For dY/dX=PY, Y(X0)=Y0, it evaluates Y(X0+H) by the Burlirsch-Store method\
(rational exterpolation). When the step error < Eps, it returns.\
Options: mfac[0], same options with gtt_ekn.setup when mfac=1.\n\
Example: T=tk_bs.bs_next([[0,1],[0,x/(1-x^2)]],[0,2],x,0,9/10,10^(-5));\n\
 number_eval(T); eval(2*asin(0.9));\n\
ref: Numerical_Recipes"
{
   関数の body.
}
```

oxlistusage は oxgentexi の入力を生成. texinfo 形式のマニュアルを自動生成.

```
oxlistusage --oxgentexi  <yang.rr |
   oxgentexi --en --upnode "yang functions" >yang-auto.en
tex myyang.texi  // yang-auto.en を include してる.
```