Computing the longest polynomial in the world : general discriminant formula of degree 17

Kimura, Kinji
Graduate School of Informatics Kyoto University

出版情報：COE Lecture Note. 49, pp.128-137, 2013-08-09. 九州大学マス・フォア・インダストリ研究所
バージョン：
権利関係：

Computing the longest polynomial in the world -general discriminant formula of degree 17-

Kinji Kimura(Graduate School of Informatics, Kyoto University) *

Abstract

We introduce how to compute the general discriminant formula of degree 17 by using multivariate Newton interpolation. Our approach is not heuristic but deterministic. Therefore, our result is clearly world-record in computing the general discriminant formula.

1 問題の設定

1.1 判別式の定義

Sylvester 表現を利用して, 終結式 (resultant) を以下のように定義する,

$$f(x) = a_m x^m + a_{m-1}x^{m-1} + \cdots + a_0, g(x) = b_n x^n + b_{n-1}x^{n-1} + \cdots + b_0,$$

$$\mathrm{Sylvester}(f(x), g(x)) = \begin{pmatrix} a_m & a_{m-1} & \cdots & a_0 & & & \\ & a_m & a_{m-1} & \cdots & a_0 & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & a_m & a_{m-1} & \cdots & a_0 \\ b_n & b_{n-1} & \cdots & \cdots & b_0 & & \\ & b_n & b_{n-1} & \cdots & \cdots & b_0 & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & b_n & b_{n-1} & \cdots & \cdots & b_0 \end{pmatrix},$$

$$\mathrm{resultant}(f(x), g(x)) = \det(\mathrm{Sylvester}(f(x), g(x))).$$

すると, 判別式 (discriminant) は, 次のように定義できる,

$$\mathrm{discriminant}(f(x)) = (-1)^{\frac{1}{2}m(m-1)}\frac{1}{a_m}\mathrm{resultant}(f(x), f'(x)).$$

ただし, $f(x)$ は, m 次多項式である.

1.2 判別式の特徴

3 次式 ax^3+bx^2+cx+d の判別式は, $27d^2a^3-18dcba^2+4c^3a^2+4adb^3-ac^2b^2$ であり, 5 次の斉次多項式になっている (簡単に証明できる). よって, $a=1$ にしても一般性を失わない. よって, 3 次式 x^3+bx^2+cx+d の判別式を考えると, $27d^2-18dcb+4c^3+4db^3-c^2b^2$ である.

*Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501 JAPAN, kkimur@amp.i.kyoto-u.ac.jp

不変式論の立場では, 平行移動 $x \leftarrow x+\alpha$ することで, 3 次式 $x^3+c'x+d'$ の判別式を計算すれば十分であるということになっている. $\alpha=-b/3$ であるから, 計算結果に対して, 置換

$$c'=-1/3b^2+c, d'=2/27b^3-1/3cb+d$$

を行わなければならないため, 数式処理の立場からみると合理的な計算の方法ではない.

1.3 斉重多項式

3 次式 x^3+bx^2+cx+d の判別式 $27d^2-18dcb+4c^3+4db^3-c^2b^2$ の計算結果は, $\mathrm{weight}([b,c,d])=[1,2,3]$ において weighted homogenious になっている. すなわち, 斉重多項式になっている.

3 次方程式 x^3+bx^2+cx+d の解 x_1,x_2,x_3 とすると, 解と係数の関係より,

$$-b=x_1+x_2+x_3, c=x_1x_2+x_1x_3+x_2x_3, -d=x_1x_2x_3.$$

判別式の定義式は, $\{(x_3-x_1)(x_1-x_2)(x_2-x_3)\}^2$ であるから, x_1,x_2,x_3 について, 6 次の斉次式になる.

判別式が b,c,d の多項式で表現できるならば (容易に証明可能), $b^kc^ld^m$ の次数は, x_1,x_2,x_3 について 6 次の項を生成するため, $k+2l+3m=6$ とならなければならない. すなわち, 3 次式 x^3+x^2+cx+d の判別式が計算できることは, 3 次式 ax^3+bx^2+cx+d の判別式が計算できることに等しい.

以上より, 17 次の判別式は,

$$G'_{17}=x^{17}+x^{16}+a_{15}x^{15}+\cdots, H'_{17}=\mathrm{resultant}_x\left(G'_{17}, \frac{dG'_{17}}{dx}\right)$$

H'_{17} より容易に手に入れることができるため, H'_{17} を計算することを問題とする.

2 Cayley の方法

一般的な多項式の判別式の公式を設計するためにのみ用いることができる方法である. $f_n(x)=x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n$ に対して, $\hat{f}_n(x)=x^n+a_1x^{n-1}+\cdots+a_{n-1}x+0$ とおくと, $\mathrm{disc.}(\hat{f}_n(x))=a_{n-1}^2\mathrm{disc.}(f_{n-1}(x))$ となる. ここから, a_n を変化させて, $\mathrm{disc.}(f_n(x))$ を求める.

2.1 判別式が満たす微分方程式

$f_n(x)=x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n$ の判別式 D は,

$$n\frac{\partial D}{\partial a_1}+(n-1)a_1\frac{\partial D}{\partial a_2}+\cdots+a_{n-1}\frac{\partial D}{\partial a_n}=0$$

を満たすことを証明する.

根を $x_1,\cdots,x_n$ とすると, $D=\prod_{i>j}(x_i-x_j)^2$ となる. $x_i\rightarrow x_i-h$ と平行移動しても不変であるから,

$$\frac{dD}{dh}=\frac{\partial D}{\partial a_1}\frac{da_1}{dh}+\frac{\partial D}{\partial a_2}\frac{da_2}{dh}+\cdots+\frac{\partial D}{\partial a_n}\frac{da_n}{dh}=0.$$

$x_i\rightarrow x_i-h$ と平行移動は,

$$f_n(x+h)=(x+h)^n+a_1(x+h)^{n-1}+\cdots+a_{n-1}(x+h)+a_n.$$

x の $n-1$ 次の係数は, a_1+hn, x の $n-2$ 次の係数は, $a_2+h(n-1)a_1+h^2(\cdots)$, x の $n-3$ 次の係数は, $a_3+h(n-2)a_2+h^2(\cdots)$, $\cdots$ より, 証明終わり.

判別式 D を, a_n でべき級数展開してみる,

$$D_i=\left.\frac{\partial^i D}{\partial a_n^i}\right|_{a_n=0}, D=D_0+\frac{1}{1!}D_1a_n+\frac{1}{2!}D_2a_n^2+\cdots+\frac{1}{(n-1)!}D_{n-1}a_n^{n-1}.$$

べき級数が, $n-1$ 次までであることは Sylvester 表現から自明. さらに,

$$\frac{\partial D}{\partial a_n}=\frac{-1}{a_{n-1}}(n\frac{\partial D}{\partial a_1}+(n-1)a_1\frac{\partial D}{\partial a_2}+\cdots),$$

と変形し, 代入すると,

$$D_1=\frac{-1}{a_{n-1}}(n\frac{\partial D_0}{\partial a_1}+(n-1)a_1\frac{\partial D_0}{\partial a_2}+\cdots), D_2=\frac{-1}{a_{n-1}}(n\frac{\partial D_1}{\partial a_1}+(n-1)a_1\frac{\partial D_1}{\partial a_2}+\cdots),\cdots$$

となる.

2.1.1 アルゴリズム全体像

$\mathrm{disc.}(f_2)=a_1^2-4a_2$ とし, 以下のアルゴリズムを繰り返し適用して, 自分が必要な n まで計算する. $f_n(x)=x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n$ に対して, $\hat{f}_n(x)=x^n+a_1x^{n-1}+\cdots+a_{n-1}x+0$ とおくと, $D_0=\mathrm{disc.}(\hat{f}_n(x))=a_{n-1}^2\mathrm{disc.}(f_{n-1}(x))$ となる. この D_0 を seed solution として, べき級数の係数を求め, $D=D_0+\frac{1}{1!}D_1a_n+\frac{1}{2!}D_2a_n^2+\cdots+\frac{1}{(n-1)!}D_{n-1}a_n^{n-1}$ を計算する.

3 小行列式展開による方法

3.1 終結式 (resultant) の関孝和, Bezout 表現

x について m 次の多項式 f と n 次の多項式 g の終結式を計算する. ただし, $m>n$ とする.

$$\begin{aligned}f &= a_mx^m+a_{m-1}x^{m-1}+\cdots+a_1x+a_0=0\\ x^{m-n}g &= b_nx^m+b_{n-1}x^{m-1}+\cdots+b_1x^{m-n+1}+b_0x^{m-n}=0\end{aligned}$$

この 2 本の式をもとに以下の操作を行い, x についての新しい多項式を n 本用意する.

$$\begin{aligned}&(a_mx^{n-1}+\cdots+a_{m-n+1})x^{m-n}g-(b_nx^{n-1}+\cdots+b_1)f\\ &=(a_mb_0-a_{m-n}b_n)x^{m-1}+(a_{m-1}b_0-a_{m-n}b_{n-1}-a_{m-n-1}b_n)x^{m-2}+\cdots=0\\ &\cdots\\ &a_mx^{m-n}g-b_nf=(a_mb_{n-1}-a_{m-1}b_n)x^{m-1}+(a_mb_{n-2}-a_{m-2}b_n)x^{m-2}+\cdots=0\end{aligned}$$

効率的に計算を行うために,

$$(a_mx+a_{m-1})x^{m-n}g-(b_nx+b_{n-1})f=x(a_mx^{m-n}g-b_nf)+(a_{m-1}x^{m-n}g-b_{n-1}f)$$

などが成立することに注意する. さらに, x についての多項式を $m-n$ 本を用意する,

$$\begin{aligned}x^{m-n-1}g &= b_nx^{m-1}+b_{n-1}x^{m-2}+\cdots+b_0x^{m-n-1}=0,\\ \cdots &\quad ,\\ g &= b_nx^n+b_{n-1}x^{n-1}+\cdots+b_0=0.\end{aligned}$$

これを行列の形式で書くと,

$$A = \begin{pmatrix} a_m b_0 - a_{m-n} b_n & \cdots & \cdots \\ \cdots & & \\ a_m b_{n-1} - a_{m-1} b_n & \cdots & \cdots \\ b_n & b_{n-1} & \cdots \\ & b_n & \cdots \\ & & \cdots \end{pmatrix}, v = \begin{pmatrix} x^{m-1} & \cdots & 1 \end{pmatrix}^\top, Av = 0$$

$\det(A)$ は, 関孝和, Bezout による終結式の行列式表現である.

3.2 関孝和, Bezout による終結式の行列式表現の小行列式展開法

4×4 の行列式を使って説明する.

$$\begin{vmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{vmatrix} = \begin{array}{l} +a_1(b_2(\underline{c_3d_4 - d_3c_4}) - c_2(\underline{b_3d_4 - d_3b_4}) + d_2(\underline{b_3c_4 - c_3b_4})) - b_1(a_2(\underline{c_3d_4 - d_3c_4}) \\ -c_2(\underline{a_3d_4 - d_3a_4}) + d_2(\underline{a_3c_4 - c_3a_4})) + c_1(a_2(\underline{b_3d_4 - d_3b_4}) - b_2(\underline{a_3d_4 - d_3a_4}) \\ +d_2(\underline{a_3b_4 - b_3a_4})) - d_1(a_2(\underline{b_3c_4 - c_3b_4}) - b_2(\underline{a_3c_4 - c_3a_4}) + c_2(\underline{a_3b_4 - b_3a_4})) \end{array}$$

下線部は, 使い回しができる. その事実を有効に使い, メモリに計算結果を蓄え計算を行う方法である.

4 提案手法

提案手法を説明するために, 準備をする.

4.1 linear assignment problem に基づく行列式の次数の上界公式

4.1.1 linear assignment problem:LAP とは

「あなたは, 働き手を持っています. Jim, Steve と Alan です, 一人に, 浴室の掃除をさせます. もう一人に, 床の掃き掃除をさせます. 三人目の人には, 窓の洗浄をさせます. 最大 (最小) のコストになるには, どのように仕事を割り当てればよいでしょうか? それぞれの仕事に対する 3 名のコストは, 以下のテーブルで与えられています.」このような問題を, linear assignment problem という. この問題を解くことと, 行列式の次数の上界公式を設計することは同値である.

	Clean bathroom	*Sweep floors*	*Wash windows*
Jim	\$1	\$2	\$3
Steve	\$3	\$3	\$3
Alan	\$3	\$3	\$2

我々の方法では, Jonker-Volgenant algorithm(LAPJV), 1987 を用いてこの問題を解いている.

4.1.2 多変数多項式を要素とする行列式の次数の上界

多変数多項式を要素とする行列式の次数の上界について考える. 具体的には, 次の A の次数の上界を考える,

$$A = \begin{vmatrix} x+y+z & xy \\ 2 & xyz \end{vmatrix}.$$

変数とパラメータの解釈により, 以下の表が得られる.

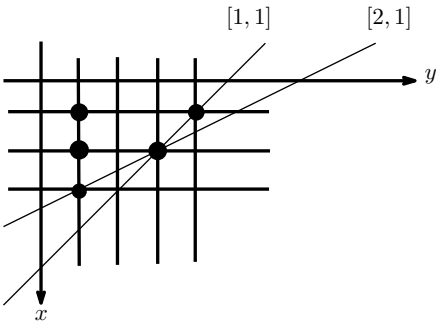
変数	パラメータ	(partial) total degree
x	y, z	2
y	x, z	2
z	x, y	2
x, y	z	3
y, z	x	3
z, x	y	3
x, y, z		4

ここでは, weight$([x, y, z])$=$[1, 1, 1]$ で考えている.

4.2 カット面付き多変数 Newton 補間

4.2.1 weight による計算量の削減

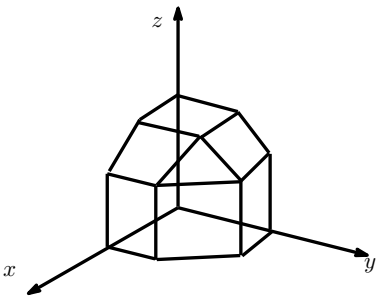
●で解の項の位置を表す. 2 種類の weight を用いて, cut 面を生成することを考える.
weight$([x, y])$=$[1, 1]$,weight$([x, y])$=$[2, 1]$ とする.



上記の図より, 2 つの weight を利用することで, より少ない評価点から解を補間できることがわかる. たくさんの weight を使えば, 原理的にはよりタイトな上界になる. しかし, 凸包の内点格子を生成する部分で, 多くの時間を費やすことになる.

4.2.2 カット面付き多変数 Newton 補間の具体例

下の図は, weight が 1 種類の場合である. 複数の weight でカットを入れる場合もあるが, ここでは, 1 種類の場合のみとする. 具体的には, weight$([x, y, z])$=$[1, 1, 1]$ を用いている. 下の凸包の内点格子の数分の評価点で, 終結式や行列式の値をサンプリングし, 補間する. 詳細は, [4] を参照されたい.

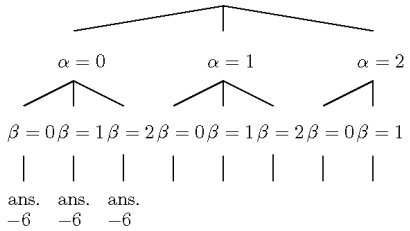


具体例として, 次の B を用いて説明する.

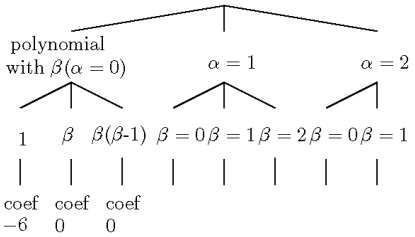
$$B = \begin{vmatrix} \alpha+\beta & 2 \\ 3 & \alpha\beta \end{vmatrix}$$

B の展開後の結果は, $(c_0+c_1\beta+c_2\beta(\beta-1))+\alpha(c_3+c_4\beta+c_5\beta(\beta-1))+\alpha(\alpha-1)(c_6+c_7\beta+c_8\beta(\beta-1))$, $c_8=0$, と仮定できる. $c_0,\cdots,c_7$ は, 未知数である.

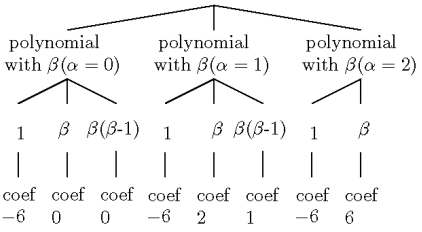
ここでは説明のため $\mathbb{Q}$ 上で表記するが, 実際のプログラムでは, $\mathbb{Z}/p\mathbb{Z}$ を利用して計算する. はじめに, $\alpha=0$, $\beta=0,1,2$ における値を計算したとする.



次に, $\alpha=0$ において Newton 補間をする.

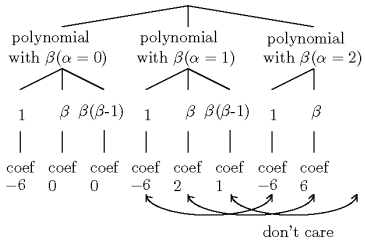


$\alpha=1,\alpha=2$ においても同様の計算をおこなう.

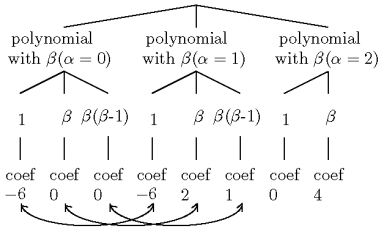


$\beta = 2$ の多項式は, あえて次数 1 で打ち切っている. Newton 補間は, 逐次補間であるため, このような途中で打ち切ったものを入力としても正しい計算ができる.

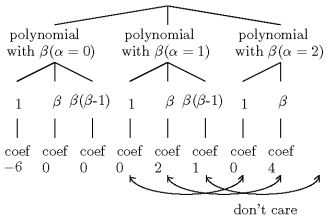
ここからは, 多項式を入力として Newton 補間をする. まず, $\alpha = 2$ の多項式から $\alpha = 1$ の多項式の減算をおこない, その結果を再度 $\alpha = 2$ に格納する.



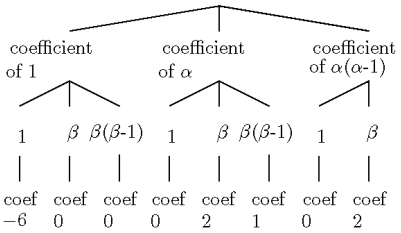
つぎに, $\alpha = 1$ の多項式から $\alpha = 0$ の多項式の減算をおこない, その結果を $\alpha = 1$ に格納する.



最後に $\alpha = 1$ の多項式から $\alpha = 0$ の多項式の減算をおこなうと, 多項式を入力とする Newton 補間が完了する.



以上より, つぎの結果を得る.



$B' = (-6 + 0\beta + 0\beta(\beta - 1)) + \alpha(0 + 2\beta + 1\beta(\beta - 1)) + \alpha(\alpha - 1)(0 + 2\beta + 0\beta(\beta - 1))$. 最後に, B' を展開し, B を得る.

5 アルゴリズムの比較

5.1 16 次までの次数の問題を使った Cayley の方法, 小行列式展開とカット面付き多変数 Newton 補間法の計算時間の比較

k	Cayley method Singular-3-1-6	TRIP minor Serial	TRIP minor Parallel	Newton Serial	Newton Parallel
12	19.337s	3m6.419s	3m5,686s	3m11.984s	13.596s
13	2m40.525s	27m38.563s	21m26.231s	22m29.047s	1m16.732s
14	12m40.276s	244m17.104s	146m44.051s	210m58.967s	10m17.354s
15	105m3.749s	-	-	1495m47.376s	67m23.748s
16	725m14.435s	-	-	???	462m46.799s

Cayley method Singular-3-1-6 は, 数式処理ソフト Singular[1] を用いて Cayley の方法を用いて計算した場合のタイミングデータを示している. TRIP minor は, 数式処理ソフト TRIP[2] を用いて小行列式展開を行った場合のタイミングデータを示している. Newton は, カット面付き多変数 Newton 補間法のタイミングデータを示している. -は, out of memory を意味する. TRIP は, 14 次の判別式を計算するために, メモリとして 483056 MB を必要とするため, それよりも次数の高い判別式を計算することは, 明らかにできない. ???は, あまりにも計算時間を必要とするため測定していない. なお, この表には, 凸包を生成する時間が含まれていない. なぜならば, 凸包を生成する部分は現在, 数式処理ソフト Risa/Asir[3] にて動作しているためである. 当日は, 全てのプログラムを C 言語で書き, その上で, すべてのタイミングデータを示す予定である. 実験環境は, CPU:E5-4650L, 4 CPU, 32 コア, mem:1440Gbyte, Intel C++ Compiler:13.1.2 である.

5.1.1 利用している weight について

16 方程式の判別式の計算を例に, 利用している weight を紹介する,

$$f(x) = x^{16} + x^{15} + a_{14}x^{14} + a_{13}x^{13} + a_{12}x^{12} + a_{11}x^{11} + a_{10}x^{10} + a_9x^9 + a_8x^8 + a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0.$$

下記の 2 種類を使う.

	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
weight_1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
weight_2	14	13	12	11	10	9	8	7	6	5	4	3	2	1	16

6 17 次方程式の判別式の計算について

数式処理ソフト Singular-3-1-6 は, ディスクを使って計算を行うように改良することが容易ではないため, Cayley の方法を用いて 17 次方程式の判別式を計算することはしない. もちろん, 並列計算が容易に可能な算法でないことも, 選択肢から外した理由である. 小行列式展開は, 莫大なメモリ使用量を必要とするため,

選択肢から外した. よって, ここでは, カット面付き多変数 Newton 補間法のみを用いて, 17 次方程式の判別式を計算することを試みる.

$$f(x) = x^{17} + x^{16} + a_{15}x^{15} + a_{14}x^{14} + a_{13}x^{13} + a_{12}x^{12} + a_{11}x^{11} + a_{10}x^{10} + a_9x^9 + a_8x^8 + a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0.$$

の判別式を計算することを考える.

6.1 計算結果

17 次方程式の判別式の項数は, 21976689397 であり, ディスク容量では, 427,470,114,659byte であった.

6.2 ディスクを使った使用メモリ量の削減方法について

計算結果について考察すると, a_{15} について, 17 次式であることから, a_{15} に, -9 から 8 まで数字を代入し, その結果を, 1 度ディスクに蓄え, さらに, ディスク上にて 1 変数の Newton 補間を行うことで使用メモリを削減できる. 次に示す結果は, a_{15} に, -9 から 8 まで数字を代入することによって得られた子問題を計算する際に要した時間を表している.

問題番号	計算時間
子問題 0	919m48.880s
子問題 1	928m1.009s
子問題 2	930m53.303s
子問題 3	933m17.794s
子問題 4	922m18.824s
子問題 5	929m57.208s
子問題 6	930m48.341s
子問題 7	922m16.684s
子問題 8	924m56.980s
子問題 9	864m15.264s
子問題 10	910m48.913s
子問題 11	924m9.816s
子問題 12	927m40.655s
子問題 13	922m10.048s
子問題 14	925m17.299s
子問題 15	934m1.590s
子問題 16	942m42.936s
子問題 17	937m48.058s

ディスク上にて 1 変数の Newton 補間を行った際の計算時間は, 612m15.268s であった. 項数のカウントに, 20m34.528s を必要とした. さらに, a_{15} から a_0 に具体的な数値を代入して, 判別式であることを確認するために, 109m47.591s を必要とした. 検算のため, そのような数値の代入を, 10 回行った.

7　まとめ

17 次方程式の判別式を計算する方法を紹介した.

参　考　文　献

[1] Singular, http://www.singular.uni-kl.de.

[2] TRIP, http://www.imcce.fr/Equipes/ASD/trip/trip.php.

[3] Risa/Asir, http://www.math.kobe-u.ac.jp/Asir/asir-ja.html.

[4] H. Werner, Mtinster, Remarks on Newton Type Multivariate Interpolation for Subsets of Grids, Computing 25(1980), 181-191.