解　説　数式処理の最近の研究動向

5．数式処理と並列処理

Parallel Processing in Formula Manipulation by Hirokazu MURAO (Computer Center, The University of Tokyo) and Tetsuro FUJISE (Mitsubishi Research Institute).

村尾　裕一[1]　藤瀬　哲朗[2]

1 東京大学大型計算機センター
2 三菱総合研究所

1. はじめに

スーパーコンピュータは，一般には科学技術計算における花形的存在と認知され，事実，これまでに多大な貢献をしてきている．しかしながら，多くの場合，それは数値計算におけることである．実際，スーパーコンピュータは，数値処理を主なターゲットとして開発されてきており，その最たるものがベクトル・プロセッサである．本稿では，数式処理の分野における並列処理の研究動向について解説し，記号処理非（数値処理）の1つである数式計算に対しても，アルゴリズムを適切に選び処理法を適宜構築することにより，ベクトル・プロセッサまで含めた高性能計算機を効率よく適用できる例のいくつかを紹介する．紹介する具体例は，ページ数の制限により，筆者らによる研究を中心とし，それ以外については参考文献に挙げるに留めさせていただく．また，並列処理に関する一般的な事柄や注意については，たとえば富田の教科書[1]などを参照されたい．具体的な研究例を挙げる前に，一般的な注意点を述べておこう．

ある計算処理に対し並列処理を施すには，まず，その処理を構成する部分計算群の中に並行性（concurrency）を見い出す必要がある．数式処理においては，具体的な処理内容の多くは数学的に定式化されており，たとえば分配則が成り立つ演算の場合のように，その定式化の範囲内で，十分な粒度を持った複数の処理中に自明な並行性を見い出せる場合が数多くある．極端には，多項式の足し算における，各項の係数の計算が一例である．並行性が自明でない場合でも，処理を数理的に変形することによって，並行性を導き出しうる場合がある．代数的独立性に基づいた並行性を導出することは，並列処理アルゴリズム開発の第一歩であり，それ自身で数理的に興味深い題材であることも多いが，それだけでは並列処理アルゴリズムとしては十分ではない．効率のよい並列アルゴリズムとして設計するには，処理全体をある程度の粒度を保った細かな処理へと適宜分割し，その部分計算群を，並行性と逐次性に基づき，並列計算機を構成する複数の演算要素の全体ができるだけ効率よく稼働するようにマッピングすることが必要となる．この時，部分計算群の間で共用すべきデータの配置と流れや，その結果として部分計算をどの程度の粒度とすべきかなどについても十分に考慮する必要がある．特に，数式処理の場合には，扱うデータに構造があり，かつ，その演算は複雑であるため，計算機ハードウェアで直接扱われる通常の数値計算の場合のように均質で等価であるとは，一般には期待できないことに注意する必要がある．たとえば数値データ1つとっても，数式処理では任意多倍長の整数として扱われるのが普通であるため，演算にかかるコストは決して一様であるとは期待できない．つまり，並行性が明らかであるからといって，そのまま並列処理を実際に適用したとしても，まったく功を奏さないということにもなりかねないのである．

2. 並列処理研究の歴史

数式処理における並列処理の研究は，意外と古く，1980年前後に出版された論文のタイトルに“parallel”の単語を見い出すことができる．もちろん，並列処理の研究が実機上での実験と実証を伴って盛んに行われるようになったのは，高々ここ10年程のことで，それ以前の研究は，理論的に並列性（や並列処理の方法）を示すことと，計算量の解析が主である．実践的には，いまだ，並列Lisp処理系の開発が主な話題であった．この頃までの並列処理に関する研究については，ACM SIGSAM Bulletinに掲載されたPonderによるreview[2]（並列Lisp処理系に関するreviewもある）が参考になる．1980年代も後半になると，このPonder（U. C. Berkeley）やWatt（Mapleの開発元であるカナダWaterloo大）が並列数式処理を学位論文の題材としても取り上げ，より現実的な並列処理の研究が行わ

表-1　主な国際会議

CAP	並列数式処理に関するワークショップComputer Algebra and Parallelism. 1988年[3]と1990年[4]の2回開催された.
PASCO	並列記号処理に関する国際会議International Symposium on PArallel Symbolic COmputation. 1994年[5]と1997年[6]に開催.
ISSAC	ACM SIGSAMが毎年開催する数式処理全般に関する国際会議International Symposium on Symbolic and Algebraic Computation.

れるようになった．数式処理研究者にとって並列計算機が比較的身近な存在となってきたのもこの頃のことであり，実践的に得られた研究成果が各種のワークショップや会議（**表-1**）で報告されるようになった．

研究の内容は，並列計算機が一般的にはなっていない初期の頃には，理想的な計算機モデルを用いた並列アルゴリズムや理論的な並列性（代数的な独立性）の指摘であり，多項式の算術演算や行列の変換や演算を，一見均等な複数の処理に分割するという類のものが主であった．そうした理論的な並列性に基づいたアルゴリズムや計算量は，すでに，BiniとPanによる教科書[7]にもまとめられている．

その後，並列計算機が普及してくると，研究内容は，実践的側面が重視されるようになり．その一方，理論的な方面では，より高度な問題へと対象を広げ，また，より複雑な技法を用いるようになってきた．実践的な研究は，当初は，既存の数式処理システムの移植や単機能なソフトウェア部品の開発が，主として共有メモリ型の並列計算機を対象として行われた．当時の共有メモリ型の場合，データの競合が起きないように処理を分割してやれば，多項式の項ごとの並列処理というところまで粒度を細かくすることも現実的であった．そのため，多倍長整数や多項式の算術演算といった最も基本的な処理からの計算法の見直しが幅広く行われた．並列計算の題材としては，こうした基本的な演算に始まり，

- 多項式の最大公約式（GCD）と因数分解
- 行列への演算
- グレブナ基底の計算（代数方程式系の解法）

などが主として扱われ，これらに関しては，現在でも幅広く研究が続けられており，主要な研究題材となっている．多項式GCDや因数分解では，後述の中国剰余算法（以下CRAと略す）やモジュラー算法を用いるのが一般的だが，そうした算法で必要な計算では式の膨張がなく，ある程度の計算量の予測が可能となる．その結果，負荷のバランスを考慮した並列処理の方法を構築することができ，並列処理の研究には好適の題材なのである．上記のそれ以外の題材は，計算量が計算の規模に従って爆発的に増大し，現実に計算するのが難しくなる可能性のある問題である．

さらに，並列計算機の形態として分散メモリ型が増えてきた昨今では，データの配置や交換に関連した事柄も研究対象となってきている．ソフトウェア関連では，分散メモリ型並列計算機を分散処理環境の一形態と捉え，分散処理環境下において，複数の計算機上で稼働する数式処理システムを結合したシステムの開発やそれらをつなぐ方法に関する研究が主流である．また，アルゴリズムの研究では，データの配置や通信コストを考慮した並列アルゴリズムが発表されるようになってきている．

表-2に，現在も並列数式処理の研究開発を実践的に進めている代表的な研究グループとその成果内容をまとめた．次章でも示すが，並列数式処理ソフトウェアの開発や移植は，表のグループ以外にも，RISC-LinzのH. Hong（現在は，Kaltofenのグループに移動し合流）や，Mapleの開発者の1人であるB. Char（米国・Drexel大）らによっても過去に行われている．また，アルゴリズムの理論的な面での研究は，表中のグループや前掲の教科書の筆者であるBini（イタリア・Pisa大）とPan（ニューヨーク市立大学）も含め，世界中で幅広く行われている．

3. 並列処理・分散処理向けの数式処理システム

前章の表中にも示したとおり，並列処理や分散処理を本格的に目指した汎用的な数式処理のソフトウェアがすでに何種類も開発され，さまざまな並列計算の実験に用いられている．その多くは，既存の数式処理の実行時ライブラリやシステムを並列処理用に改訂・移植したものである．その並列処理化の方法には，実行時ライブラリをソースレベルで改訂して細かくスレッド化するという方法と，既存のシステムにデータ通信用の拡張を行った上で複数のシステムを同時に稼働する機構を設ける，という2つの方法がある．粒度を細かくすることができる前者の方法は，主に，共有メモリ型の並列計算機を対象として用いられ，PARSAC-2や，SAC-2を元にしたライブラリSACLIBをHongらが共有メモリ型の並列計算機上に移植したPACLIBがその例である．既存の数式処理システムを，分散メモリ型の並列計算機や分散環境に実現するのであれば，後者の方法が簡単である．汎用の数式処理システムとして広く流布しているREDUCEやMapleに関しては，この方法による並列化の試みが報告されている．特に，Mapleに関しては，CharよるMaple/Linda[14]およびその発展形の*Sugarbush*[15]（複数のMapleをLindaによ

表-2　主な研究開発グループと成果

米国Kent州立大のP. S. Wangのグループ	Macsymaの開発で，多項式GCDと因数分解に関し多大な貢献をしたWangは，これらの演算に関する共有メモリ型の並列計算機向けの並列アルゴリズムを提案し，個別プログラムとしての実装を行っている．最近では，分散環境下で，複数の並列処理システムを結合しデータの交換を行うためのMP[8]を発表・実装している． http://icm.mcs.kent.edu/~pwang．また，SymbolicNet（http://SymbolicNet.mcs.kent.edu/）を開設．
米国North Carolina州立大のE. Kaltofenのグループ	数式処理の高性能計算に関しては，多項式因数分解に関連のアルゴリズムの開発と実装を行っている．また，分散数式処理のためのインタフェース／システムDSC[9]を開発．C, C++, Common Lisp, Mapleで書かれたソースプログラムを，ネットワーク環境下の登録された計算機群に自動配布し，分散処理を行う．実際に，大規模な多項式因数分解の問題などに適用している． http://www.math.ncsu.edu/~kaltofen/public.html
富士通研究所の数式処理研究のグループ	同グループが開発した数式処理システムRisa/Asirに分散処理用の機能を追加し，ワークステーション・クラスタや分散メモリ型並列計算機上で実装．グレブナ基底の並列・分散処理による効率のよい計算法．
独Tübingen大のW. Küchlinのグループ	共有メモリの並列計算機向けに，リスト処理のためのメモリ管理機構の機能を有するスレッド機構S-threadを準備し，それを用いて，既存の数式処理システムSAC-2を並列計算機上に移植したPARSAC-2[10]を実現．分散環境へも拡張済み．CRAの適用による多項式GCDの並列計算や，グレブナ基底の計算の並列処理などの試み．http://www-sr.informatik.uni-tuebingen.de/
仏IMAG研究所のLMC, G. Villard他	分散メモリ型の並列計算機（Transputer）向けに並列数式処理用のライブラリPAC[11]を開発．さまざまな計算へのPACの応用がCAP '88で報告されている．その後，動的な負荷分散を可能とするよう設計し直されたPAC++[12]を経て，現在は，自動負荷分散の機能を備えたGivaroへと発展． http://www-lmc.imag.fr/
英国A. C. Norman（Cambridge大）とJ. Fitch（Bath大）	分散メモリの並列計算機上で，分散したメモリのすべてを使って多項式を表現し，巨大な規模の数式の計算を行えるようにすることを主な目的とした，新たな数式処理システムCABAL[13]を開発中．
スイスETH	分散メモリ型の超並列計算機上での，数式処理ライブラリの開発およびシステムの移植．

って結合），RISC-Linzにおける‖MAPLE‖[16]（複数のMapleタスクを並列論理型プログラミング言語Strandを介して結合）の開発，および，ETHにおける超並列計算機への移植など，複数の例がある．

既存の数式処理システムを用いる方法としては，分散処理環境において，複数の数式処理システムを結合して並列・分散処理システムを構築するという方法がある．そのためのソフトウェアとしては，KaltofenグループのDSCや，会話型ソフトウェアとMPIとを結合してマスタ／スレーブによるSPMDモデルを実現するためのSTAR/MPI[17]がある．こうした分散処理向けの通信に関しては，数式処理の通信プロトコルとしてMP[8]やMPP[18]が提案されている．こうした通信を規定するソフトウェアは，単に分散処理を行うためだけではなく，ヨーロッパを中心に開発された数多くの数式処理のソフトウェアを，必要な個別の機能ごとに呼び出して接続して利用するという形態を想定してのことである．特に，個別の機能が高性能計算機上で実現されていれば，共用の高性能計算機を分散環境下で計算サーバとして利用するという形態も現実的である．実際，近年新たに登場した汎用の数式処理システムとして脚光を浴びつつあるMuPAD[19]では，すでにMPも採用して，このような他のソフトウェアとの結合を積極的に行い，異なる研究グループが開発したソフトウェアによるさまざまな機能が簡単に提供されうるようになっている．

並列数式処理システムに関しては，ISSAC97とPASCO97のチュートリアルの資料[20]にも，より詳しいレビューがあるので，そちらも参照されたい．

4. ベクトル処理の研究

複数の独立なデータに対して同一の演算を同時並列に実行するデータ並列処理では，プロセッサ全体を効率よく利用するために，各プロセッサ上での演算量を等しくする努力が必要だが，そのためにはデータの均一性が要求される．特に，その究極の形であるベクトル処理を施すには，均質なデータは機械表現された数値データである必要があり，しかも，数学的な意味でのベクトルとして処理されるデータ列の間には依存関係が生じないような処理としてアルゴリズムを構築しなければならない．数式処理においては，そのような処理は，非常に限られた場合にしか現れない．数式処理では，通常扱う数は任意多倍長の整数であって構造を持つため，数の演算においてさえ均質性が存在しない．むしろ，任意多倍長の整数そのものが，均質なデータの列として捉えられるべきもので，ベクトル処理の対象ともなる．これに関する研究報告もCAP '88に見られるが，ベクトル計算機本来の性能を出し切っているとは言えない．

筆者らは，これに対し，ベクトル処理に適した計算を既存の数式処理の計算や算法の中から探し出し，できる限り効率よくベクトル処理を施すという方針で研

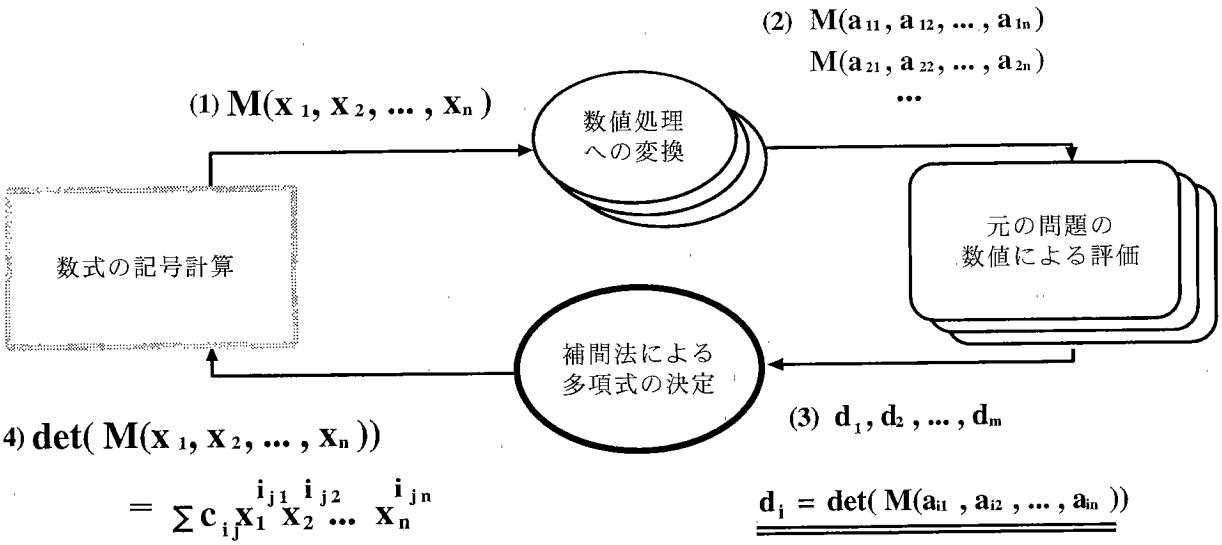

図-1　多項式補間法の応用：行列式の展開計算

究を続けてきている．基本は，数については多倍長整数を扱わずに剰余数の演算に落として扱うこと☆と，対象を，そうした数値の単純な集まりである行列や一変数多項式とすることである．最も成功した例は，最終結果として多項式を求める計算において，多項式の補間（interpolation）による決定と数値処理に中国剰余定理（CRA）を適用するという方法である．この2つの方法については，次章で述べる．多項式補間を用いることにより，途中の計算を数値による計算で置き換えることが可能となり，さらに，その数値処理にはCRAを適用し，法計算を行うことにより，すべての数値処理が固定精度となる．その結果，ベクトル演算器をきわめて効率よく利用することが可能となる．求めるべき多項式は，既存の数式処理システム上で構成する．この方法は，高橋と石橋により1960年に，当時の計算機を用いて線形方程式をexactに解くための方法として発表されており[21]，筆者らは，ちょうど30年後に，行列式の計算にベクトル処理を活用するために利用した[22]．**図-1**は，整数係数の$x_1, \cdots, x_n$の多項式を要素とする行列Mが与えられて，その行列式det Mを求める場合の模式図である．行列式は多項式になるので，その多項式の数値列から，補間法により決定することができる．その各数値は，元の行列の多項式の変数に数値を代入して得られる数値行列の行列式の値d_iである．その行列式の値は，多倍長整数となるが，理論的に定められた上限値の元でCRAを適用すれば，十分に多くの法による剰余の集合として求められる．つまり，「数値による評価」において実際に扱う数は，精度の定まった剰余の値だけであり，ベクトル処理が可能となる．この計算法は，データ並列化するための有効な方法であり，行列式以外にもさまざまな応用が考えられる．

Knuthの教科書にも紹介されているように，多項式因数分解の基本は有限体上の多項式の因数分解である．そのための算法としてはBerlekampアルゴリズムがよく知られているが，そのアルゴリズムにおける行列の消去演算そのものは，CAP '88でNormanらも指摘しているとおり，ベクトル処理に適合する．筆者らは，有限体上の多項式の因数分解のためのいくつかのアルゴリズムをベクトル処理向けに実現し，その因子の分離に用いる多項式の導出における有効性（計算量の劇的な緩和）を確認している[23]．同時に，有限体上の多項式に対する演算全般に，ベクトル処理が有効であることも確認している．

5. 並列・分散処理とそのアルゴリズムの研究

多項式演算のモジュラー算法

多項式関連の計算では，途中の計算をある法の元で剰余として進め，最終結果の式を何らかの方法で構築するという算法が，一般的な方法として広く使われている．そのような算法を，一般に，モジュラー算法と呼ぶ．モジュラー算法では，剰余の式として表される中間式は過度な膨張を起こさないという利点があるのだが，以下に述べるように，並行性を導き出す手法としても有用である．モジュラー算法の基礎となる数学的事実の1つとして中国剰余定理がある．

（中国剰余定理）「$M_1, M_2, \cdots, M_n$を互いに素な法として，法の各々に対する剰余 $a_1, a_2, \cdots, a_n$ が与えられた時，各iに対して $c \equiv a_i \bmod M_i$ を満たす（$c \bmod \Pi_{i=1}^{n} M_i$）が唯一つ存在する．」

上の定理は，式の存在性だけでなく，その式の具体

☆ Kunthの著名な教科書でも，有用な方法として紹介されている．

的な構成法を与えて証明されている．M_iを素数とすれば，数に対するごく普通のCRAとなるし，$M_i=x-\alpha_i$とxの多項式とすれば，$x=\alpha_i$における値からxの多項式$c(x)$を定めるという補間法となる．各々の法M_iについての途中の計算は，代数的に独立なため，容易に並行性を導き出すことができ，しかも，同等の法を選べば，計算の負荷はほぼ均等となり並列アルゴリズムの設計がしやすい．

多変数多項式の補間は，上の方法を帰納的に用いれば可能だが，多くの場合多変数多項式は疎であるのに対し，あらゆる単項式の存在の仮定に基づいたそのような方法は，きわめて多くの数値が必要となるため，実用上はほとんど使い物にならない．しかしながら，疎な多変数多項式を補間する画期的な方法が，1988年にBenOrとTiwariにより発表され[24]，さらに，その後，改良が重ねられ[25]，実用性の高い算法ができ上がっている．

また，整数係数の多項式の因数分解やGCDの計算では，適当な法の元で多項式の因子の個数や次数などを求め，元の多項式に関するそれらの値を推定するという場合があるが，さまざまな法の元での推定を行い確度をあげるのが一般的である．そのような場面でも，単純な並列処理が可能である．

グレブナ基底の計算に関して

グレブナ基底の計算は，規模（変数の個数や次数）に応じて，計算量は爆発的に増大する一方で，自明な並行性（集合の2要素間での簡約化）があり，格好の題材として，メモリの共有／分散を問わずさまざまな並列計算機で並列処理が試みられてきた．しかし，直接並列化しても，多倍長整数の係数の膨張が激しく，その計算がネックとなって計算負荷の極端な不均衡を招くこともある[26]．最近では，モジュラー計算の適用も含め，より効率のよい並列処理が可能となっている．グレブナ基底の説明とより詳しい計算法については，本特集「2. 方程式を解く」を参照されたい．

多項式因数分解に関して

有限体Z_p（pは素数）上の一変数多項式の因数分解は，多項式の因数分解の基本である．Z_p上の一変数多項式$F\in Z_p[z]$の因数分解は，基本的には，代数的な探索問題へと帰着される．探索は，$V^p\equiv V \bmod F$を満たすFの分離多項式$V\in Z_p[z]$を用いて，

$$F=\prod_{\alpha\in Z_p}\gcd(F, V-\alpha)$$

という等式に基づいたGCD計算の繰り返しで行う．Berlekampのアルゴリズムでは，分離多項式を行列の消去により求め，消去や付随する多項式演算にベクトル処理が有効なことは前述のとおり．特に，行列消去に関しては，係数を有限体に持つ線形方程式を反復法により解くWiedemannの算法をブロック化した算法も知られており，並列処理も可能である[27),28)]．探索問題は，一般に，並列処理により高速化が期待されるが，上の探索も，異なるVやαを用いたGCD計算の繰り返しを単純に並列化してしまい，投棄的計算を行えばよい．得られた因子の計算中の除去は，GCD計算の高速化以外に意味はない．異なるαで並列に計算する場合，因子を非同期に各計算間で送受信して除去し合うこと（代数的並列枝刈りと呼ぶ）は，通信遅延がある場合でも全体の高速化に有効である．次数別因数分解（Fを，等しい次数の因子の積f_kの列へと分解すること）も，代数的な探索$(\gcd(F, z^{p^k}-z)=\Pi_{1\le\forall i\le k, i\mid k} f_i)$によって得るが，探索の幅を徐々に狭めていくという，探索の一般的手法を用いたアルゴリズムが発表されており，ごく自然に並列化することが可能である[29)]．

6. おわりに

数式処理システムは，すでに科学技術計算に必須の道具として定着し，数学的機能の更なる強化を目指して研究が進められている．機能や扱いうる問題の難しさが増すに従い，必要な計算量が増大していくことは必須である．その緩和や対処のための研究は，数理的側面を主体に続けられているが，高性能計算機の利用という力技を駆使するための研究も大切である．

本稿では，紙面の都合で，アルゴリズムなどにおける詳細な事柄は割愛せざるを得なかったので，それらについては，筆者らのより詳しい解説[30)]も参照されたい．

参考文献

1) 富田: 並列コンピュータ工学, 昭晃堂 (1996).
2) Ponder, C. G.: Parallelism and Algorithms for Algebraic Manipulation: Current Work, SIGSAM Bulletin, Vol.22, No.3, pp.7-14 (1988).
3) Della Dora, J. and Fitch, J. (eds.): Computer Algebra and Parallelism, Computational Mathematics and Applications, Academic Press (1989).
4) Zippel, R. E. (ed.): Computer Algebra and Parallelism, Second International Workshop Proceedings (CAP '90), LNCS, No.584, Springer-Verlag (1990).
5) Hong, H. (ed.): Proc. PASCO '94, Lecture Note Series in Computing, Vol.5, World Scientific (1994).
6) Hitz, M. and Kaltofen, E. (eds.): Proc. PASCO '97, ACM Press (1997).
7) Bini, D. and Pan, V.: Polynomial and Matrix Computations, Fundamental Algorithms, Vol.1, Birkhauser (1994).
8) Gray, S., Kajler, N. and Wang, P.: MP: A Protocol for Efficient Exchange of Mathematical Expressions, In von zur Gathen and Giesbrecht[31)], pp.330-335.
9) Diaz, A., Kaltofen, E., Schmitz, K. and Valente, T.: DSC A System for Distributed Symbolic Computation, Proc. ISSAC '91 (Watt, S. M. (ed.)), pp.323-332 (1991).

10) Küchlin, W.: PARSAC-2: A Parallel SAC-2 Based on Threads, Proc. AAECC-8 (Sakata, S. (ed.)), LNCS, No.508, Springer-Verlag, pp.341-353 (1990).
11) Roch, J.-L.: PAC: Towards a Parallel Computer Algebra Co-processor, In Della Dora and Fitch[3], pp.33-50.
12) Gautier, T. and Roch, J. L.: PAC++ System and Parallel Algebraic Numbers Computation, In Hong[5], pp.145-153.
13) Norman, A. and Fitch, J.: CABAL: Polynomial and Power Series Algebra on a Parallel Computer, In Hitz and Kaltofen[6], pp.196-203.
14) Char, B. W.: Progress Report on a System for General-purpose Parallel Symbolic Algebraic Computation, Proc. ISSAC '90 (Watanabe, S. and Nagata, M. (eds.)), pp.96-103 (1990).
15) Char, B. W.: A User's Guide to Sugarbush-Parallel Maple through Linda, Technical report, Drexel Univ., Dept. Math. and Comp. Sci. (1994).
16) Siegl, K.: ‖ MAPLE ‖: A System for Parallel Symbolic Computation, Technical Report 93-07, RISC-Linz (1993).
17) Cooperman, G.: STAR/MPI: Binding a Parallel Library to Interactive Symbolic Algebra System, Proc. ISSAC '95 (Levelt, A. H. M. (ed.)), pp.126-132 (1995).
18) Bachmann, O., Schönemann, H. and Gray, S.: MPP: A Framework for Distributed Polynomial Computations, In Lakshman[32], pp. 103-112.
19) MuPAD, Multi Processing Algebra Data tool, MuPAD Group, Automath, Univ. Paderborn. http://www.mupad.de/MuPAD, ftp: ftp://ftp.mupad.de/MuPAD.
20) Roch, J.-L. and Villard, G.: Parallel Computer Algebra (ISSAC97/PASCO97 tutorial), http://www-lmc.imag.fr/~gvillard/CFPAR/.
21) 高橋, 石橋: 電子計算機によるexactな計算の新方法, 情報処理, Vol.1, No.2, pp.78-86 (Feb. 1960).
22) Murao, H.: Vectorization of Symbolic Determinant Calculation, SUPERCOMPUTER, Vol.VIII, No.3, pp.36-48 (1991).
23) 村尾: Z_p上の多項式の因数分解－高速化技法・ベクトル処理・並列処理－, 講究録920「数式処理における理論とその応用の研究」, 京都大学数理解析研究所 (1995).
24) Ben-Or, M. and Tiwari, P.: A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation, Proc. 20th Symp. Theory of Comput., pp.301-309 (1988).
25) Murao, H. and Fujise, T.: Modular Algorithm for Sparse Multivariate Polynomial Interpolation and its Parallel Implementation, J. Symb. Comp., Vol.21, No.46, pp.377-396 (1996).
26) Sawada, H., Terasaki, S. and Aiba, A.: Parallel Computation of Groebner Bases on Distributed Memory Machines, J. Symb. Comp., Vol.18, No.3, pp.207-222 (1994).
27) Kaltofen, E. and Lobo, A.: Factoring Hig-Degree Polynomials by the Black Box Berlekamp Algorithm, In von zur Gathen and Giesbrecht[31], pp.90-98.
28) Kaltofen, E.: Analysis of Coppersmith's Block Wiedemann Algorithm for the Parallel Solution of Sparse Linear Systems, Math. Comp., Vol.64, No.210, pp.777-806 (1995).
29) Fujise, T. and Murao, H.: Parallel Distinct Degree Factorization Algorithm, In Lakshman[32], pp.18-25.
30) 村尾, 藤瀬: 並列処理の多項式計算への応用, 数式処理, Vol.5, No.1, pp. 2-17 (1996).
31) von zur Gathen, J. and Giesbrecht, M. (eds.): Proc. ISSAC '94 (1994).
32) Lakshman, Y. N. (ed.): Proc. ISSAC '96 (1996).

(平成9年12月9日受付)

村尾　裕一（正会員）

昭和30年生．昭和56年東京大学大学院理学系研究科情報科学専攻修士課程修了．昭和59年同博士課程単位取得のうえ退学．同年より東京大学大型計算機センター助手．主な興味は，記号処理，数式処理のシステムとアルゴリズムに関する実践的な研究，および，スーパーコンピュータの応用（特に，数式処理への応用）．理学博士．

e-mail:murao@cc.u-tokyo.ac.jp

藤瀬　哲朗（正会員）

1959年生．1984年電気通信大学大学院電気通信学研究科情報数理工学専攻修士課程修了．同年三菱総合研究所入社．1992年から1995年にかけて(財)新世代コンピュータ技術開発機構出向，並列論理型プログラミング言語処理系の研究開発に従事．現在三菱総合研究所メディア&ネットワークシステム部研究室長．並列記号処理系および記号処理算法，特に数式処理算法に興味をもつ．また映像処理についても興味をもつ．

e-mail:fujise@mri.co.jp