JAERI-Data/Code
97-032

運動論的バルーニングシューティングコード KBSHOOTの並列化

1997年7月

山極　満・根本俊行*・広瀬　章**・Mike ELIA**

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所研究情報部研究情報課（〒319-11 茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　㈱原子力資料サービス

JAERI－Data/Code 97－032

運動論的バルーニングシューティングコードKBSHOOTの並列化

日本原子力研究所那珂研究所炉心プラズマ研究部
山極　満・根本　俊行*・広瀬　章**・Mike ELIA**

（1997年7月1日受理）

カナダ・サスカチュワン大学・プラズマ物理研究所において開発された運動論的バルーニングモード解析用シューティングコードKBSHOOTについて，VPP500向きに行った並列化について報告する。このコードのベクトル化率は十分高く，並列化により，さらに高速化を図ることができた。オリジナルコードの1PE (Processing Element)による結果に比して，4PEで約3倍，16PEで約8倍の高速化が達成された。

那珂研究所：〒311-01 茨城県那珂郡那珂町向山801-01
*　富士通（株）
**　カナダサスカチュワン大学プラズマ物理研究所

JAERI－Data/Code 97－032

Parallelization of Kinetic Ballooning Shooting Code KBSHOOT

Mitsuru YAMAGIWA, Toshiyuki NEMOTO*, Akira HIROSE**
and Mike ELIA**

Department of Fusion Plasma Research
Naka Fusion Research Establishment
Japan Atomic Energy Research Institute
Naka-machi, Naka-gun, Ibaraki-ken

(Received July 1, 1997)

This report describes parallelization of the kinetic ballooning shooting code, KBSHOOT, developed at the Plasma Physics Laboratory, University of Saskatchewan, Canada, for running on the VPP500 machine. The speed of the code, which was highly vectorized, has been further enhanced by parallelization. The CPU time has been reduced by three times for the 4PE (Processing Element) mode and eight times for the 16PE mode, as compared to that of the original code with the 1PE mode.

Keywords: Kinetic Ballooning Mode, Tokamak, Plasma, Shooting Code, Parallelization, VPP500, NEXT

* On leave from Fujitsu Ltd.
**Plasma Physics Laboratory, University of Saskatchewan, Canada

目　　次

Contents

1．はじめに

本報告書では、トカマクプラズマにおける有限ラーマー半径等運動論的効果によって修正を受けた圧力勾配駆動型不安定性の一種である「運動論的バルーニングモード（KBM）」を解析するために開発された運動論的バルーニングシューティングコードKBSHOOTの並列化について述べる。

バルーニングモードは、トカマクのような環状プラズマ磁場閉じ込めシステムにおいて、磁場曲率の悪いトーラス外側領域での圧力勾配によって駆動される電磁不安定性で、閉じ込め可能なプラズマの圧力限界をもたらす要因となり得る。バルーニング安定性の理論的基礎付けは理想電磁流体力学（MHD）近似に基づく解析により行われ［1］、低圧力勾配側と高圧力勾配側の両方に安定領域が現われることが発見された［2］。これらの領域は、それぞれ第1および第2安定領域と呼ばれている。さらに、バルーニングモードの運動論的取り扱いにより、理想MHD近似の妥当性が確認されたが［3］、近年、イオン温度勾配が有限な場合、かつ、不安定モードの固有関数が従来より拡がることを考慮に入れることにより、KBMが理想MHD安定領域においても存在し得ることが見いだされた［4、5］。また、米国の大型トカマク装置（TFTR）での重水素（D）－三重水素（T）実験において、DT核融合反応の結果生じた高速アルファ粒子（^{4}He）の損失をもたらすアクティビティとしてKBMが観測され［6］、その解析も行われた［7］。

これらの数値計算において用いられた運動論的バルーニングシューティングコードKBSHOOTはカナダ・サスカチュワン大学プラズマ物理研究所長、広瀬章教授を中心として開発されたものである。本コードは外国人研究者招聘制度に基づき、氏を平成7年11月19日～平成7年12月16日の期間受け入れた際に、原研に導入された。この間、主として、運動論的バルーニングモード方程式におけるイオン速度積分を完全数値的に行うバージョンのVPP500用ベクトル化を行い1PE（Processing Element）による高速化を達成した。その後、並列化を行うことにより、CPU時間をさらに短縮することができた。

並列化技法についての知見を得ることは、炉心プラズマ研究部が計算科学技術推進センターの協力を得て進めている、数値トカマク（NEXT）研究における重要テーマの一つでもある。本報告書ではKBSHOOTの並列化のために行った作業について述べるが、プログラムの追加・変更点に焦点を置き、数値計算法等についての詳細には触れない。

第2章において、KBM解析の基礎となる運動論的バルーニングモード方程式を示す。第3章においては、KBSHOOTの並列化およびその結果について述べる。第4章ではまとめを行う。

2．運動論的バルーニングモード方程式

KBSHOOTでは次の運動論的バルーニングモード方程式を解くことによって、不安定モードの固有関数、成長率、および周波数が評価される［4、5、7］：

$$\frac{d}{d\theta}\left[\{1+(s\,\theta-\alpha\sin\theta)^2\}\frac{d\phi}{d\theta}\right]+\frac{\tau}{2(1+\tau)}\frac{\alpha}{\varepsilon_n(1+\eta)}$$

$$\times\left[(1-0.6\sqrt{\varepsilon})\{(\Omega-1)(\Omega-f(\theta))+\eta_e f(\theta)\}-I_{i2}\delta^2/\tau\right.$$

$$-\{(1-\sqrt{\varepsilon})(\Omega-1)-I_{i1}(\theta)\,\delta\}\times\{(1-0.6\sqrt{\varepsilon})(\Omega-1)-I_{i1}(\theta)\,\delta\}$$

$$\left./(1+\tau-\tau I_{i0}-I_{eT})\right]\phi=0 \qquad (1)$$

ここで、ϕはスカラーポテンシャル（固有関数）、θはバルーニング座標、αは圧力勾配に関するバルーニングパラメータ、sは磁気シアパラメータ、εは逆アスペクト比、$\Omega=\omega/\omega_{*e}$は電子の反磁性ドリフト周波数で規格化した複素周波数、$\tau=T_e/T_i$はイオン温度に対する電子温度の比、$\varepsilon_n=L_n/R$はプラズマ主半径で規格化した密度勾配スケール長である。また、$\eta=(\tau\,\eta_e+\eta_i)/(\tau+1)$、$\eta_j=d\ln T_j/d\ln n$ $(j=e,i)$、nはプラズマ密度、$f(\theta)=2\,\varepsilon_n[\cos\theta+(s\theta-\alpha\sin\theta)\sin\theta]$である。

イオンのトランジット周波数の効果は次式で定義されるδによって、摂動的に取り入れられるようになっている：

$$\delta=\frac{\sqrt{\langle k_{\parallel}^2\rangle}\,v_{Ti}}{\omega_{*i}}=R\sqrt{\langle k_{\parallel}^2\rangle}\frac{\sqrt{2}\,\varepsilon_n}{k_\theta\,\rho_i} \qquad (2)$$

$$\langle k_{\parallel}^2\rangle=\frac{1}{(qR)^2}\int d\theta\left|\frac{d\phi}{d\theta}\right|^2\Big/\int d\theta\,|\phi|^2 \qquad (3)$$

ここで、$k_\theta\rho_i$は有限イオンラーマー半径パラメータ、qは安全係数である。

イオン速度積分および捕捉電子積分はそれぞれ次式で与えられる：

$$I_{in}(\theta)=\frac{2}{\sqrt{\pi}}\int_0^\infty x\,dx\int_{-\infty}^\infty y^n\,dy\,\frac{\tau\,\Omega+1+\eta_i(x^2+y^2-\frac{3}{2})}{\tau\,\Omega+f(\theta)(\frac{1}{2}x^2+y^2)-\delta\,y}\times J_0^2[\sqrt{2}\,\xi(\theta)x]\,e^{-x^2-y^2} \qquad (4)$$

$$\xi(\theta)=k_\theta\rho_i\sqrt{1+(s\theta-\alpha\sin\theta)^2} \qquad (5)$$

$$I_{eT}(\theta)=\frac{4}{\sqrt{\pi}}\int_0^\infty x\,dx\int_0^\infty dy\,\frac{\Omega-1-\eta_e(x^2+y^2-\frac{3}{2})}{\Omega-f(\theta)(\frac{1}{2}x^2+y^2)}\,e^{-x^2-y^2}\quad(for\ \varepsilon x^2>y^2) \qquad (6)$$

3．KBSHOOTの並列化

3．1　コード解析

KBSHOOTでは式（1）を境界条件、$\frac{d\phi}{d\theta}=0 \ at \ \theta=0$　および　$\phi(\theta=\infty)=0$、のもとで、シューティング法により解く。主たるサブルーチン、ファンクションの機能は以下の通りである。

FUNCTION Y	：	固有関数、$\phi(\theta)$、を求める。
FUNCTION Y2	：	式（1）の $\frac{d^2\phi}{d\theta^2}$ 項を除く部分を評価する。
FUNCTION BESJ0	：	Bessel関数、J_0、を定義する。
SUBROUTINE ZROOT	：	固有値、ω、を求める（Root Finder）。

コードにおけるCPU時間分布を富士通㈱製の動的挙動解析ツールANALYZER［8］を用いて解析した。そのサブルーチンごとの解析結果を表1に示す。この表によると、ファンクションY2がスカラ実行で全体の99.5%、ベクトル実行で全体の86.1%のCPU時間を消費している。さらに、Y2の内部には2組の2重ループがあり、この一方がY2で消費するCPU時間の大部分を占めている。また、呼び出し回数で注目してみると、ファンクションBESJ0が5400万回と莫大な回数になっている。副プログラムの呼び出しにはアドレス計算等のオーバーヘッドがかかるので、これらを中心に高速化を行うことにする。参考までに、KBSHOOTのツリー構造を図1に示す。

3．2　並列化のための指針

ファンクションY2は、他の副プログラムで共通に使用される配列はなく、単純にY2自身の値を定義するだけである。VPP500は分散メモリ型計算機なので、配列の分割等を行う場合には、他の副プログラムでも配列の分割を考慮した変更を施すことが多い。しかし、Y2にはこれが無いので、このルーチンのみの並列化を行う。

Y2は十分にベクトル化が施されている。そこで、特にベクトル化効率を損なうことなく並列化を行うために、すでにベクトル化されているDOループを並列処理のための（分割の）対象にせず、これ以外のDOループを分割する。これは、もちろん、ベクトル化しているDOループのベクトル長（ループ長）が長いほどベクトル化効果が発揮され、分割してしまうと、ベクトル長が短くなってしまい、分割数によってはスカラ処理より遅くなる場合があるからである。

図2にANALYZERを実行したY2のオリジナルプログラムを示す。これによるとDO 10およびDO 20の2重ループのコストが高いので、これを並列化の対象とする。さらにY2では、DO 10ループのところで変数、XI2、J0、を定義し（図2の①の部分）、内側ループでこれら変数を使用して、計算を行う。このループ内で、3つの総和計

算を行い、変数、AI0、AI1、AI2、にその結果を代入している（図２の②の部分）。並列処理を行うには、DO 10ループを分割し、各PEでの計算終了後、AI0、AI1、AI2の値の総和を求めればよい。例えば、DO 10ループにおいてDO変数Ｉの範囲は1~200 までである。これを4PEで並列実行した場合、それぞれのPEではI=1, 50、I=51, 100、I=101, 150、I=151, 200の範囲で実行される。そして、実行終了後、４つのPEが持っているデータを総和する。このイメージを図３に示す。

３．３　並列化のための変更

並列処理をさせるには、!XOCL で始まるVPP用拡張最適化制御行［９］をコード内に挿入する。

（１）メインルーチンMAIN

VPP用拡張最適化制御行が挿入されたメインルーチンMAINを図４に示す。①はコードが使用するPEの数の宣言を行うPROCESSOR文である。PROCESSOR文で指定されているPP(NN)はPPがプロセッサグループ名で、NNは使用するPEの数で、パラメータ文によって定義される。パラメータ文はインクルードファイルPARMで定義される。これはVPP用拡張最適化制御行を挿入して並列処理を行うファンクションY2でもPROCESSOR文が必要であるためで、PEの数を変えて実行する場合、変更しやすいようにした処置である。②のPARALLEL REGIONは、③のEND PARALLELと対で使用するもので、この間にあるものをパラレルリージョンと呼び、並列処理の対象となる。これ以外に並列化のための命令が含まれない場合は、すべてのPEで同じ処理（冗長実行という）が行われる。

（２）ファンクションY2

VPP用拡張最適化制御行が挿入されたファンクションY2を図５に示す。①のPROCESSOR文はMAINと同じものである。②のSUBPROCESSOR文は副プログラムで宣言されるもので、後に続くQQ(NN)のQQは仮プロセッサグループ名と呼び、NNによってこの副プログラムで使用するPEの数が指定される。コード全体で使用するPE数とファンクションY2で使用するPE数は同じなので、①、②には同じNNを指定する。③のSPREAD DOは、直後にあるDOループを分割して並列処理をさせる命令である。これは④のEND SPREADと対で使用し、この間で並列処理を行う。SPREAD DOの後の /(QQ) は、どのようにDOループを分割するかの指示である。QQは②で宣言した仮プロセッサグループ名であり、使用PE数がNNである。これは、NN個のPEに均等にDOループを分割して割り当てることである。仮にNNが4である場合は、前述した通り、ループ長が200なので、50ずつに分割して割り当てることになる。なお、NNが16の場合は15のPEに13ずつ割り当てられ、残りの1PEに5となる。

④のEND SPREADは並列処理の終端を示すものであるが、その後の、SUM(AI0),SUM(AI1),SUM(AI2) は総和をとるためのものである。DOループ内では、

ここで指定している変数を使って総和計算を行っている。並列処理をした場合、各PEで計算した総和のみがこれらの変数に格納されているので、並列処理終了後、各PEの持つ値のさらなる総和を計算する必要がある。この役目をするのがSUMである。DO 11およびDO 21の２重ループについても同様な処置を施した（⑤、⑥参照）。

（３）並列化以外の高速化

ファンクションBESJ0は、ファンクションY2から呼ばれているが、呼び出し回数が非常に多いことが動的挙動解析で判明した。副プログラムの呼び出しには、アドレス計算等のオーバーヘッドが発生する。呼び出し回数の多い場合は、このオーバーヘッドが無視できなくなるため、インライン展開（親ルーチンへの展開）を行うことによって、オーバーヘッドを無くすことにする。このファンクションは、コンパイラによるインライン展開ができる構造になっている。コンパイルオプション -Of（最適化レベルF）を指定して、ロードモジュールを作成するようにする。

３．４　並列化の結果

（１）並列化版の倍率

実行シェルで時間測定ツールtimexを使用し、オリジナル実行、並列化版の4PE、および16PEでの計算時間を計測した。その結果を表２に示す。倍率はオリジナル実行のCPU時間と並列化版の経過時間（ELAPS TIME）で求めた。並列化版の本来の倍率を求めるには、経過時間によって比較するが、原研の場合、シングル（非並列）ジョブは、実行クラスによっては２多重でジョブが実行され、かつ、入出力を行うCP（Control Processor）は他のジョブの処理も行うので、これらのジョブの影響によって経過時間が左右されるため、このような求め方を行った。

この表によると、PE台数分の効果は出ていない。これは、並列処理の命令でデータ転送、および同期をとるものがあり、オーバーヘッドを伴うためである。データ転送は総和計算部分で行われ、転送量が少ないため１回当たりの転送時間は短いが、ここで同期とる処置がとられる。KBSHOOT の場合、この処理の実行回数が多いためにオーバーヘッドが大きくなってしまう。また、粒度と呼ばれる、並列処理単位の演算数（１回の並列処理における1PEでの演算数) も少ない。結果として、並列処理のためのオーバーヘッドが目立ち、顕著な台数効果が得られないのである。

（２）並列化版の計算結果の評価

総和計算部分を並列処理した場合と、シングルで実行した場合の計算結果が異なることがある。これは、計算順序が異なるためである。KBSHOOT の場合、並列処理に総和計算が含まれているが、論理機番６番の出力結果で比較したところ、シングル、4PE実行、16PE実行のいずれの場合も全桁一致した。従って、並列手法に誤りはないと判断される。

4．まとめ

運動論的バルーニングシューティングコードKBSHOOTのVPP500用並列化を行った。結果的には、メモリ分散型計算機でよく見受けられる配列等の各PEへの分割・割当作業がなく、CPU時間が一つの副プログラムに集中していたため、単独ルーチンの並列化のみを施した。オリジナルコードの1PEによる結果に比して、4PEで約3倍、16PEで約8倍の高速化が達成された。台数効果が顕著でないのは並列処理命令実行時の同期調整によるオーバーヘッドのためである。特にメモリ分散型計算機においては、PE間でデータ通信が行われる。データ通信はデータ転送時間・同期を伴うものであり、これによりオーバーヘッドが生じる。これをいかに克服するかが効率の良い並列化プログラム作成のポイントとなる。

謝　辞

本研究遂行にあたり終始激励いただきました平山俊雄プラズマ理論研究室長ならびに藤井実情報システム管理課長に感謝いたします。

参考文献

[1] J.W. Connor, R.J. Hastie, and J.B. Taylor, Phys. Rev. Lett. 40, 396 (1978).

[2] L.E. Zakharov, in *Plasma Physics and Controlled Nuclear Fusion Research 1978* (IAEA, Vienna, 1979), Vol. 1, p. 689.

[3] C.Z. Cheng, Phys. Fluids 25, 1020 (1982); Nucl. Fusion 22, 773 (1982).

[4] A. Hirose, L. Zhang, and M. Elia, Phys. Rev. Lett. 72, 3993 (1994); Phys. Plasmas 2, 859 (1995).

[5] A. Hirose and M. Elia, Phys. Rev. Lett. 76, 628 (1996).

[6] Z. Chang *et al.*, Phys. Rev. Lett. 76, 1071 (1996).

[7] M. Yamagiwa, A. Hirose, and M. Elia, Plasma Phys. Control. Fusion 39, 531 (1997).

[8] UXP/M アナライザ使用手引書, 富士通株式会社, 1992年2月.

[9] UXP/M VPP FORTRAN77EX/VPP 使用手引書V12用, 1994年1月.

4．まとめ

運動論的バルーニングシューティングコードKBSHOOTのVPP500用並列化を行った。結果的には、メモリ分散型計算機でよく見受けられる配列等の各PEへの分割・割当作業がなく、CPU時間が一つの副プログラムに集中していたため、単独ルーチンの並列化のみを施した。オリジナルコードの1PEによる結果に比して、4PEで約3倍、16PEで約8倍の高速化が達成された。台数効果が顕著でないのは並列処理命令実行時の同期調整によるオーバーヘッドのためである。特にメモリ分散型計算機においては、PE間でデータ通信が行われる。データ通信はデータ転送時間・同期を伴うものであり、これによりオーバーヘッドが生じる。これをいかに克服するかが効率の良い並列化プログラム作成のポイントとなる。

謝　辞

本研究遂行にあたり終始激励いただきました平山俊雄プラズマ理論研究室長ならびに藤井実情報システム管理課長に感謝いたします。

参考文献

[1] J.W. Connor, R.J. Hastie, and J.B. Taylor, Phys. Rev. Lett. 40, 396 (1978).

[2] L.E. Zakharov, in *Plasma Physics and Controlled Nuclear Fusion Research 1978* (IAEA, Vienna, 1979), Vol. 1, p. 689.

[3] C.Z. Cheng, Phys. Fluids 25, 1020 (1982); Nucl. Fusion 22, 773 (1982).

[4] A. Hirose, L. Zhang, and M. Elia, Phys. Rev. Lett. 72, 3993 (1994); Phys. Plasmas 2, 859 (1995).

[5] A. Hirose and M. Elia, Phys. Rev. Lett. 76, 628 (1996).

[6] Z. Chang *et al.*, Phys. Rev. Lett. 76, 1071 (1996).

[7] M. Yamagiwa, A. Hirose, and M. Elia, Plasma Phys. Control. Fusion 39, 531 (1997).

[8] UXP/M アナライザ使用手引書, 富士通株式会社, 1992年2月.

[9] UXP/M VPP FORTRAN77EX/VPP 使用手引書V12用, 1994年1月.

4．まとめ

運動論的バルーニングシューティングコードKBSHOOTのVPP500用並列化を行った。結果的には、メモリ分散型計算機でよく見受けられる配列等の各PEへの分割・割当作業がなく、CPU時間が一つの副プログラムに集中していたため、単独ルーチンの並列化のみを施した。オリジナルコードの1PEによる結果に比して、4PEで約3倍、16PEで約8倍の高速化が達成された。台数効果が顕著でないのは並列処理命令実行時の同期調整によるオーバーヘッドのためである。特にメモリ分散型計算機においては、PE間でデータ通信が行われる。データ通信はデータ転送時間・同期を伴うものであり、これによりオーバーヘッドが生じる。これをいかに克服するかが効率の良い並列化プログラム作成のポイントとなる。

謝　辞

本研究遂行にあたり終始激励いただきました平山俊雄プラズマ理論研究室長ならびに藤井実情報システム管理課長に感謝いたします。

参考文献

[1] J.W. Connor, R.J. Hastie, and J.B. Taylor, Phys. Rev. Lett. 40, 396 (1978).

[2] L.E. Zakharov, in *Plasma Physics and Controlled Nuclear Fusion Research 1978* (IAEA, Vienna, 1979), Vol. 1, p. 689.

[3] C.Z. Cheng, Phys. Fluids 25, 1020 (1982); Nucl. Fusion 22, 773 (1982).

[4] A. Hirose, L. Zhang, and M. Elia, Phys. Rev. Lett. 72, 3993 (1994); Phys. Plasmas 2, 859 (1995).

[5] A. Hirose and M. Elia, Phys. Rev. Lett. 76, 628 (1996).

[6] Z. Chang *et al.*, Phys. Rev. Lett. 76, 1071 (1996).

[7] M. Yamagiwa, A. Hirose, and M. Elia, Plasma Phys. Control. Fusion 39, 531 (1997).

[8] UXP/M アナライザ使用手引書, 富士通株式会社, 1992年2月.

[9] UXP/M VPP FORTRAN77EX/VPP 使用手引書V12用, 1994年1月.

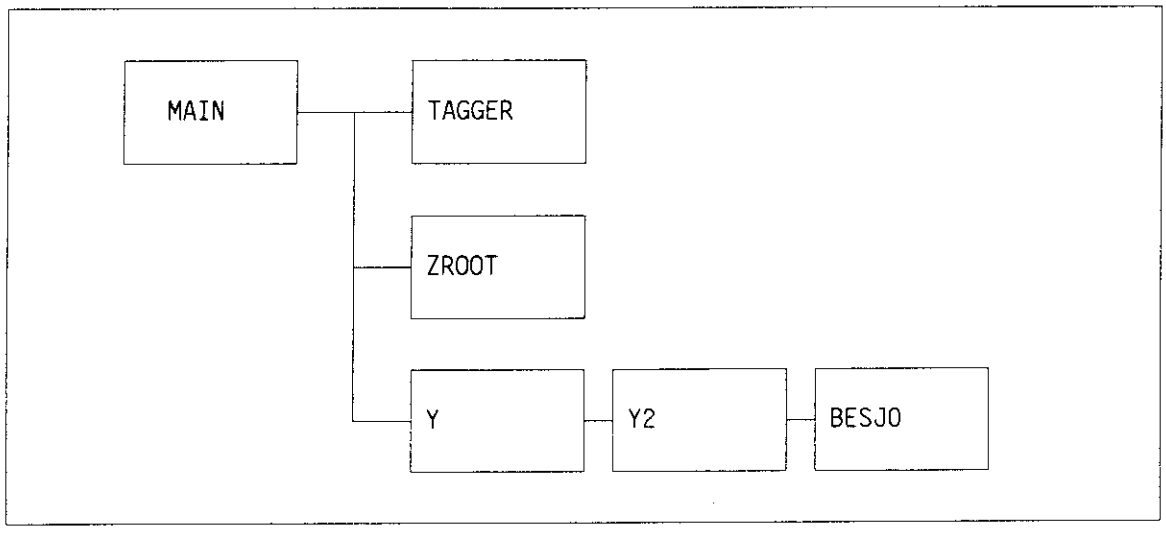

図1 The Tree Structure of KBSHOOT code

ex-count	true	v-cost	s-cost	
270000		1350000	1350000	
270000		13770000	13770000	
270000		13770000	13770000	
270000		13770000	13770000	
270000		810000	810000	
270000		1080000	1080000	
270000		14580000	14580000	
270000		19440000	19440000	
270000		270000	270000	
270000		270000	270000	
270000		810000	810000	
270000		810000	810000	
270000		810000	810000	
270000		810000	810000	
270000		810000	810000	
270000		810000	810000	
270000		540000	540000	S
54000000		.2700E09	.2700E09	M
54000000		54000000	54000000	S
54000000		.2970E10	.2970E10	S
54000000		.1080E09	.1080E09	V
.2160E11		.2374E10	.8640E11	V
.2160E11		.5934E09	.2160E11	V
.2160E11	21.5	.1780E10	.6480E11	V
.4634E10		.9267E10	.9267E10	

```
      COMPLEX FUNCTION Y2(T,Y,P,W)
      COMPLEX Y,P,W,AINTEG,AIO,AI1,AI2,UeT,UINTEG
      REAL PI,  T,JO,  DXI,DXJ
      COMMON/COMMO1/ HH,MAX,  EP,ETA,BO,  N,  S,A,AKPARA2,Q,
     &               E, TAU, ETAe, ETAi, IFLUIDe
C
      PI = 4.*ATAN(1.)
      SI = SIN(T)
      CO = COS(T)
      ZETA = S*T-A*SI
      F = 2.*EP*(CO+ZETA*SI)
      ARG = SQRT(BO*(1.+ZETA**2))
      DELTA=SQRT(AKPARA2*2./BO)*EP/Q
C
      DXI = 0.025
      DXJ = 0.025
      AINTEG = CMPLX(0.)
      AIO=CMPLX(0.)
      AI1=CMPLX(0.)
      AI2=CMPLX(0.)
      UINTEG = CMPLX(0.)
      UeT=CMPLX(0.)
C
Cvocl loop,noeval,nopreex
      DO 10 I=1,200
         XI  = FLOAT(I-1)*DXI +.5*DXI
         XI2 = XI*XI                                  ⇦ ①
         JO  = BESJO( SQRT(2.)*ARG*XI )
Cvocl loop,noeval,nopreex
      DO 20 J=-200,199
         XJ  = FLOAT(J)*DXJ +.5*DXJ
         XJ2 = XJ*XJ
C        ----------------------------------
         IF( (XI2+XJ2) .GT. 25.0 ) GO TO 20
```

図2 The Original Subroutine Y2

```
                                                C          ---------------------------------
.1697E11          .7877E11  .2867E13 V                     AINTEG = ( W*TAU+1.+ETAi*( XI2+XJ2-1.5 ) )
                                                     &           / ( W*TAU+F*( XI2/2.+ XJ2 )-DELTA*XJ ) * JO**2
                                                     &           * EXP(- XI2 - XJ2) * XI
.1697E11          .7458E10  .2715E12 V                     AIO=AIO+AINTEG*DXI*DXJ
.1697E11          .1072E11  .3902E12 V                     AI1=AI1+AINTEG*DXI*DXJ*XJ          ⇦ ②
.1697E11          .1072E11  .3902E12 V                     AI2=AI2+AINTEG*DXI*DXJ*XJ2
.2160E11          .1780E10  .6480E11 V                20   CONTINUE
54000000          .1620E09  .1620E09 V                10   CONTINUE
  270000          29970000  29970000                       AIO=2.*AIO/SQRT(PI)
  270000          29970000  29970000                       AI1=2.*AI1/SQRT(PI)
  270000          29970000  29970000                       AI2=2.*AI2/SQRT(PI)
                                                C          ================================================
  270000    0.0     540000    540000                       IF( E .GT. 0.0 ) THEN
                                                C
                                                C          ---------------------
       0      0          0         0                       IF(IFLUIDe.EQ.1) THEN
                                                C          ---------------------
       0                 0         0                       UeT=((W-7.0/3.0*F/2.)*(W-1.0)-ETAe*F/2.)
                                                     &        /((W-5.0/3.0*F/2.)**2-10.0/9.0*(F/2.)**2)
       0                 0         0                       UeT=SQRT(E)*UeT
                                                C          ----
       0                 0         0                       ELSE
                                                C          ----
       0                 0         0 V                     DO 11 I=1,200
       0                 0         0 V                        XI  = FLOAT(I-1)*DXI + 0.5*DXI
       0                 0         0 V                        XI2 = XI*XI
       0                 0         0 V                     DO 21 J=1,200
       0                 0         0 V                        XJ  = FLOAT(J-1)*DXJ + 0.5*DXJ
       0                 0         0 V                        XJ2 = XJ*XJ
                                                C
       0      0          0         0 V                        IF( ( (XI2+XJ2) .GT. 25.0 ) .OR.
                                                     &           ( XI2 .LT. (XJ2*(1.0-E)/(2.0*E)) ) ) GO TO 21


       0                 0         0


                                                C
       0                 0         0 V                        UINTEG = ( W-1.0-ETAe*( XI2+XJ2-1.5 ) )
```

図2 The Original Subroutine Y2 (continued)

```
                                        &            / ( W-F*( XI2/2. + XJ2 ) )
                                        &            * EXP(- XI2 - XJ2) * XI
                                    C
         0             0          0 V              UeT = UeT+UINTEG*DXI*DXJ
                                    C
         0             0          0 V        21    CONTINUE
         0             0          0 V        11    CONTINUE
         0             0          0                UeT = 4.*UeT/SQRT(PI)
                                    C           -----
         0             0          0                ENDIF
                                    C           -----
                                    C
                                    CMY         WRITE(6,*) UeT
                                    CMY         WRITE(6,*) HH,MAX,EP,ETA,BO,N,S,A,AKPARA2,Q
                                    CMY         WRITE(6,*) E,TAU,ETAe,ETAi,IFLUIDe
                                    CMY         STOP
                                    C
    270000             0          0            ENDIF
                                    C           ================================================
                                    C
    270000       .1094E09   .1094E09            Y2 = -2.*ZETA*(S-A*CO)/(1.+ZETA**2) * P
                                        &          + A/(4.*(ETA+1.)*EP*(1.+ZETA**2))
                                        &          * ( ( (1.0-sqrt(E))*(W-1.0) - AI1*DELTA )
                                        &            * ( (1.0-0.6*sqrt(E))*(W-1.0) - AI1*DELTA )
                                        &            / ( 1.0 + TAU - TAU*AIO - UeT )
                                        &            - (1.0-0.6*sqrt(E))*( (W-F)*(W-1.0) + ETAe*F )
                                        &            + AI2*DELTA**2/TAU ) * Y
                                    C
    270000             0          0            RETURN
    270000        540000     540000            END
```

図2 The Original Subroutine Y2 (continued)

```
      DO 10 I=1,200
           XI  = FLOAT(I-1)*DXI +.5*DXI
           XI2 = XI*XI
           J0  = BESJ0( SQRT(2.)*ARG*XI )
      DO 20 J=-200,199
            .
            .
         AI0=AI0+AINTEG*DXI*DXJ
         AI1=AI1+AINTEG*DXI*DXJ*XJ
         AI2=AI2+AINTEG*DXI*DXJ*XJ2
20    CONTINUE
10    CONTINUE
```

この２重ループの
並列処理を行う

1PE

```
      DO 10 I=1,50  ⇦
           XI  = FLOAT. . .
           XI2 = XI*XI
           J0  = BESJ0. . .
      DO 20 J=-200,199
            .
            .
         AI0=AI0+AINTEG*. . .
         AI1=AI1+AINTEG*. . .
         AI2=AI2+AINTEG*. . .
20    CONTINUE
10    CONTINUE
```

2PE

```
      DO 10 I=51,100  ⇦
           XI  = FLOAT. . .
           XI2 = XI*XI
           J0  = BESJ0. . .
      DO 20 J=-200,199
            .
            .
         AI0=AI0+AINTEG*. . .
         AI1=AI1+AINTEG*. . .
         AI2=AI2+AINTEG*. . .
20    CONTINUE
10    CONTINUE
```

3PE

```
      DO 10 I=101,150  ⇦
           XI  = FLOAT. . .
           XI2 = XI*XI
           J0  = BESJ0. . .
      DO 20 J=-200,199
            .
            .
         AI0=AI0+AINTEG*. . .
         AI1=AI1+AINTEG*. . .
         AI2=AI2+AINTEG*. . .
20    CONTINUE
10    CONTINUE
```

4PE

```
      DO 10 I=151,200  ⇦
           XI  = FLOAT. . .
           XI2 = XI*XI
           J0  = BESJ0. . .
      DO 20 J=-200,199
            .
            .
         AI0=AI0+AINTEG*. . .
         AI1=AI1+AINTEG*. . .
         AI2=AI2+AINTEG*. . .
20    CONTINUE
10    CONTINUE
```

計算終了後、各PEで計算されたAI0, AI1, AI2 の総和計算を行う。

図3 The Image of Parallel Processing

```
*INCLUDE PARM
!XOCL PROCESSOR PP(NN)                    ⇦ ①
        COMPLEX W,O
        COMPLEX FAI
        COMPLEX Y
             .
             .
             .
C
        COUNT = 0
!XOCL PARALLEL REGION                     ⇦ ②
        IF (VAR.GE.0.0) THEN
   10       CONTINUE
            BO = VAR
            A2 = A * A
            S2 = S * S
            ETA  = ( TAU*ETAe + ETAi ) / ( TAU + 1.0 )
            AKPARA2 = 1.0 / 3.0
     &              * ( 1.0 + S2*(PI2/3.0-0.5)
     &                - 8.0*A*S/3.0 + 3.0*A2/4.0 )
     &              / ( 1.0 + S2/3.0*(PI2-7.5)
     &                - 10.0*A*S/9.0 + 5.0*A2/12.0 )
          AKPARA2  = AKPARA2 * PARAKF
C
            CALL ZROOT(Y,EPS,NSIG,PRTB,W,ITMAX,INFER,IER)
C08/03/96   O = W*SQRT(BO*A/(4.*EP*(1.+ETA)))
            O = W*SQRT(BO*A/(2.*(1.+TAU)/TAU**2*EP*(1.+ETA)))
C
            COUNT = COUNT + 1
            WRITE(6,99) VAR,W,O,INFER,INSERT(COUNT)
C           WRITE(2,92) VAR,W,O,INFER,INSERT(COUNT)
C
            IF ((AIMAG(W).LT.0.0).OR.(VAR.LT.VARMIN)) GOTO 100
C
            VAR = VAR + INCLUDE
C
            GOTO 10
        ENDIF
C
 100    CONTINUE
!XOCL END PARALLEL                        ⇦ ③
C
                                    WRITE(6,93)
           IT = 0
        DO T = 0., MAX, HH
           IT = IT + 1
           IF( MOD(IT,25) .EQ. 1 ) WRITE(6,94) THETA(IT),FAI(IT)
        END DO
C
        STOP
  91       FORMAT(1X,F7.4,2X,2(1X,F9.4),2X,2(1X,F9.4),3X,I3,3X,A)
  92       FORMAT(   F7.4,2X,2(1X,F9.4),2X,2(1X,F9.4),3X,I3,3X,A)
  93       FORMAT(//,'  THETA','         ', ' REAL FAI','         ',
     &                                      ' IMAG FAI')
  94       FORMAT(F7.4,'         ', F9.4,'         ', F9.4)
  99       FORMAT(F7.4,'         ', 4(F9.4,'      '), I3,'         ', A)
        END
```

図4 The Parallel Version of MAIN routine

```
      COMPLEX FUNCTION Y2(T,Y,P,W)
*INCLUDE PARM
!XOCL PROCESSOR PP(NN)            ⇦ ①
!XOCL SUBPROCESSOR QQ(NN)         ⇦ ②
      COMPLEX Y,P,W,AINTEG,AIO,AI1,AI2,UeT,UINTEG
      REAL PI,  T,JO,  DXI,DXJ
      COMMON/COMMO1/ HH,MAX,  EP,ETA,BO,  N,  S,A,AKPARA2,Q,
     &               E, TAU, ETAe, ETAi, IFLUIDe
C
      PI = 4.*ATAN(1.)
      SI = SIN(T)
      CO = COS(T)
      ZETA = S*T-A*SI
      F = 2.*EP*(CO+ZETA*SI)
      ARG = SQRT(BO*(1.+ZETA**2))
      DELTA=SQRT(AKPARA2*2./BO)*EP/Q
C
      DXI = 0.025
      DXJ = 0.025
      AINTEG = CMPLX(0.)
      AIO=CMPLX(0.)
      AI1=CMPLX(0.)
      AI2=CMPLX(0.)
      UINTEG = CMPLX(0.)
      UeT=CMPLX(0.)
C
Cvocl loop,noeval,nopreex
!XOCL SPREAD DO/(QQ)               ⇦ ③
      DO 10 I=1,200
         XI  = FLOAT(I-1)*DXI +.5*DXI
         XI2 = XI*XI
         JO  = BESJO( SQRT(2.)*ARG*XI )
Cvocl loop,noeval,nopreex
      DO 20 J=-200,199
         XJ  = FLOAT(J)*DXJ +.5*DXJ
         XJ2 = XJ*XJ
C        ---------------------------------
         IF( (XI2+XJ2) .GT. 25.0 ) GO TO 20
C        ---------------------------------
         AINTEG = ( W*TAU+1.+ETAi*( XI2+XJ2-1.5 ) )
     &          / ( W*TAU+F*( XI2/2.+ XJ2 )-DELTA*XJ ) * JO**2
     &          * EXP(- XI2 - XJ2) * XI
         AIO=AIO+AINTEG*DXI*DXJ
         AI1=AI1+AINTEG*DXI*DXJ*XJ
         AI2=AI2+AINTEG*DXI*DXJ*XJ2
   20    CONTINUE
   10    CONTINUE
!XOCL END SPREAD SUM(AIO),SUM(AI1),SUM(AI2)      ⇦ ④
      AIO=2.*AIO/SQRT(PI)
      AI1=2.*AI1/SQRT(PI)
      AI2=2.*AI2/SQRT(PI)
C     ========================================================
      IF( E .GT. 0.0 ) THEN
C
C     ---------------------
      IF(IFLUIDe.EQ.1) THEN
C     ---------------------
      UeT=((W-7.0/3.0*F/2.)*(W-1.0)-ETAe*F/2.)
```

図5 The Parallel Version of Subroutine Y2

```
     &     /((W-5.0/3.0*F/2.)**2-10.0/9.0*(F/2.)**2)
           UeT=SQRT(E)*UeT
C          ----
           ELSE
C          ----
!XOCL SPREAD DO/(QQ)                ⇦ ⑤
           DO 11 I=1,200
              XI  = FLOAT(I-1)*DXI + 0.5*DXI
              XI2 = XI*XI
           DO 21 J=1,200
              XJ  = FLOAT(J-1)*DXJ + 0.5*DXJ
              XJ2 = XJ*XJ
C
              IF( ( (XI2+XJ2) .GT. 25.0 ) .OR.
     &            ( XI2 .LT. (XJ2*(1.0-E)/(2.0*E)) ) ) GO TO 21
C
              UINTEG = ( W-1.0-ETAe*( XI2+XJ2-1.5 ) )
     &               / ( W-F*( XI2/2. + XJ2 ) )
     &               * EXP(- XI2 - XJ2) * XI
C
              UeT = UeT+UINTEG*DXI*DXJ
C
   21      CONTINUE
   11      CONTINUE
!XOCL END SPREAD SUM(UeT)           ⇦ ⑥
           UeT = 4.*UeT/SQRT(PI)
C          -----
           ENDIF
C          -----
C
CMY        WRITE(6,*) UeT
CMY        WRITE(6,*) HH,MAX,EP,ETA,BO,N,S,A,AKPARA2,Q
CMY        WRITE(6,*) E,TAU,ETAe,ETAi,IFLUIDe
CMY        STOP
C
           ENDIF
C          ==================================================
C
           Y2 = -2.*ZETA*(S-A*CO)/(1.+ZETA**2) * P
     &        + A/(4.*(ETA+1.)*EP*(1.+ZETA**2))
     &        * ( ( (1.0-sqrt(E))*(W-1.0) - AI1*DELTA )
     &          * ( (1.0-0.6*sqrt(E))*(W-1.0) - AI1*DELTA )
     &          / ( 1.0 + TAU - TAU*AIO - UeT )
     &          - (1.0-0.6*sqrt(E))*( (W-F)*(W-1.0) + ETAe*F )
     &          + AI2*DELTA**2/TAU ) * Y
C
           RETURN
           END
```

図5 The Parallel Version of Subroutine Y2 (continued)

表1 Dynamic Behavior of the Original Version of KBSHOOT

vectorize - total list

name	ex-count	v-cost	%	s-cost	%	v-leng	v-rate	v-effect	overhd
Y2	270000	.1273E12	86.1	.4170E13	99.5	397	99.7	26.3- 37.9	0
BESJO	54000000	.2053E11	13.9	.2053E11	0.5	1	0.0	1.0- 1.0	0
Y	27	11745837	0.0	11745837	0.0	1251	0.0	1.0- 1.0	0
MAIN	1	40827	0.0	40827	0.0	1251	0.0	1.0- 1.0	0
ZROOT	2	17963	0.0	17963	0.0	6	0.0	1.0- 1.0	0
TAGGER	1	11140	0.0	11140	0.0	228	0.0	1.0- 1.0	0
(total)	---------	.1479E12	100.0	.4190E13	100.0	---------	99.2	23.4- 32.1	------

表2 CPU and Elaps Time of the Parallel Version

	CPU 時間	VU 時間	経過時間	倍率
ｵﾘｼﾞﾅﾙ 版(ﾍﾞｸﾄﾙﾓｰﾄﾞ)	22m 19.66s	22m 01.79s	22m 27.96s	1.0
並列化版 (４ＰＥ)	29m 39.07s	22m 11.54s	7m 52.02s	2.8
並列化版 (１６PE)	41m 10.26s	23m 02.02s	2m 49.41s	7.9

(注意) 倍率はｵﾘｼﾞﾅﾙ CPU 時間/ 並列化版経過時間で求めた。