# 常微分方程式数値解法の並列アルゴリズムとパブリックドメインソフトウェアパッケージPVM

川　崎　拓　弥*・奈　良　　久**

## Parallel Algorithms for the Numerical Solution of the Ordinary Differential Equation and the PDS Package of the Pvm

Takumi KAWASAKI and Hisashi NARA

Abstract

The VLSI revolution is stimulating research in the fields of parallel processing and multiprocessor system. Multiprocessor systems are now available for very fast computer used in the scientific numerical computing. However, conventional algorithms of numerical analysis are suitably constructed only for monoprocessor computers. Parallel algorithms for the multiprocessor system have not yet thoroughly investigated, except for parallel algorithms for some matrix computations. In this paper, we propose a parallel algorithm for the numerical solution of the ordinary differential equations used in various fields. This parallel algorithm is based on the predictor formula and its iterative procedure. Simulations have been carried out and it is proved that this parallel algorithm is reliable and offers a faster computing times than the existing algorithms such as Runge-Kutta method. Moreover, we describe how to apply the parallel algorithm by using a software package of PVM(Parallel Virtual Machine) which provide us with a virtual, parallel computer.

キーワード：常微分方程式(ODEs)，並列アルゴリズム(Parallel Algorithm)，複区分予測子法，PVM(Parallel Virtual Machine)

## 1. はじめに

近年の VLSI 技術のめざましい発展に伴い，並列処理計算機の研究が盛んに行われており，高速演算が必要である科学用数値計算の分野で大いに期待されている。現在実用化されている科学計算用の高速計算機は更なる高速演算に向けて様々な努力が続けられているが，スピード向上の原理的限界に近づいていると考えられている。VLSI 技術の発展により単一プロセッサの性能は著しく向上したが，これを多数用いる並列処理システムが，瞬間風速的には科学計算用高速計算機には及ばないが，総体的には単位時間にはるかに莫大な計算量を消化できるので，注目を浴びている。

実際，並列プロセッサを用いた様々な並列処理計算機システムが多数提案されている。しかしながら，このような並列処理計算機に対する数値計算アルゴリズムについては，行列演算などのベクトル演算以外については未だ十分には研究されていない。特に，高次代数方程式，常微分方程式や偏微分方程式などの数値解析手法は逐次近似のアルゴリズムを用いており，この種の並列処理アルゴリズムに関しては従来ほとんど研究されていなかった。本報告では，初期値問題として科学の分野のみならず広く用いられている常微分方程式を取り上げ，その並列数値計算アルゴリズムを示すとともに，従来のアルゴリズム[1)-3)]との比較検討した結果について述べる。

また，仮想的な並列計算機を作り出す事ができるフリーソフトウエア「PVM(Parallel Virtual Machine)」[4)-6)]について一般的な説明を加え，今後どのような構成で使用するのか，並列アルゴリズムを PVM にどのようにして適用するのか，またその時の問題点について述べる。

平成 12 年 12 月 21 日受理

*　八戸工業大学大学院　電気電子工学専攻
博士前期課程 1 年
** 同上　及び　八戸工業大学　システム情報工学科　教授

## 2. 一般的な常微分方程式の解法

常微分方程式は，熱伝導，力学系あるいは人口の増加などの様々な分野の数学的モデルとして用いられている。特に，$y_0 = y(x_0)$ の条件の下に1階常微分方程式

$$\frac{dy}{dx} = f(x,y) \quad \cdots\cdots (1)$$

を満足する特殊解 $y(x)$ を求める初期値問題は各種の制御系において用いられ，その数値解法の高精度化と高速化が必要とされる。以下従来用いられてきた各種の逐次型の方法についてまとめておく。

### 2. 1 Euler 法

Euler 法は(1)式を形式的に積分して得られる

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(x,y)dx \quad \cdots\cdots (2)$$

において，$f(x,y)$ を区間 $(x_n, x_{n+1})$ で十分滑らかだとし

て，積分記号の外に出し，

$$y_{n+1} = y_n + hf(x_n, y_n) \quad \cdots\cdots (3)$$

で近似する。もちろん

$$y_n = y(x_n)$$
$$y_0 = y(x_n)$$
$$h = (x_{n+1} - x_n)$$

である。

### 2. 2　Taylor 展開法

(2)式に Taylor 展開法を用い k 次以上の微分項を無視すれば，

$$y_{n+1} = y_n + hf(x_n, y_n) + \frac{h^2}{2} f'(x_x, y_n) + \cdots$$
$$\cdots + \frac{h^k}{k!} f^{(k-1)}(x_n, y_n) \quad \cdots\cdots (4)$$

が得られる。(4)式において k=1 とすれば Euler 法に一致する。

### 2. 3　Runge-Kutta 法

(4)式は $f', f'', \ldots, f^{(k-1)}$ を含むために実際には非常に複雑である。Runge-Kutta 法は Taylor 展開(4)式の係数と一致するように考案された方法である。４次の Runge-Kutta 法は

$$y_{(n+1)} = y_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \quad \cdots\cdots (5)$$

$$\left.\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\ k_3 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \\ k_4 &= hf(x_n + h, y_n + k_3) \end{aligned}\right\} \quad \cdots\cdots (6)$$

で与えられ，現在でもよく用いられている解法である。

## 3. 並列数値計算アルゴリズム

並列処理計算システムは様々な構成が考えられているが，ここでは N 個のプロセッサと共有メモリから構成されたシステムを考える。

等間隔 h で並んだ分点 $x_1, x_2, \ldots, x_N$ における N 個の初期近似解 $y_1^{(0)}, y_2^{(0)}, \ldots, y_N^{(0)}$ が何らかの方法(例えば前述の Runge-Kutta 法)で求まっているものとする。(2)式はパラメータ M を導入することにより

$$y_{n+1} = y_{n-M} + \int_{x_{n-M}}^{x_{n+1}} f(x, y)dx \quad \cdots\cdots (7)$$

と書ける。ここで M は正の整数または０である。

(7)式の $f(x, y)$ に対して Newton の後退補間多項式を用いると(7)式は，

$$y_{n+1} = y_{n-M} + \int_{x_{n-M}}^{x_{n+1}} [f_n + \frac{\nabla f_n}{h}(x - x_n) + \frac{\nabla^2 f_n}{2h^2}(x - x_n)(x - x_{n-1})x$$
$$+ \cdots + \frac{\nabla^k f_n}{k!h^k}(x - x_n)\cdots(x - x_{n-k+1})]dx \quad \cdots\cdots (8)$$

となる。(8)式をさらに変形すると次式が求まる。

$$y_{n+1} = y_{n-M} + \sum_{k=0}^{K} hP_k' \nabla^k f_n \quad \cdots\cdots (9)$$

$$P_k' = \frac{1}{k!}\int_{-M}^{1} u(u+1)\cdots(u+k-1)du \quad \cdots\cdots (10)$$

$$P_0' = 1$$

$$\nabla^k f_n = f_n - \binom{k}{1} f_{n-1} + \binom{k}{2} f_{n-2} - \cdots + (-1)^k f_{n-k} \quad \cdots\cdots (11)$$

$$f_n = f(x_n, y_n)$$

(9)式は更に，

$$y_{n+1} = y_{n-M} + \sum_{k=0}^{K} hP_k f_{n-k} \quad \cdots\cdots (12)$$

$$P_k = \frac{1}{k!}\int_{-M}^{1} u(u+1)\cdots(u+k-1)du \quad \cdots\cdots (13)$$

となり，(12)式を複区分予測公式と呼んでいる。

関数 $f(x, y)$ を k 次の Newton の前進補間多項式で近似すると，(7)式は，

$$y_{n+1} = y_{n-M} + \int_{x_{n-M}}^{x_{n+1}} [f_n + \frac{\Delta f_n}{h}(x - x_n) + \frac{\Delta^2 f_n}{2h^2}(x - x_n)(x - x_{n+1})$$
$$+ \cdots + \frac{\Delta^k f_n}{k!h^k}(x - x_n)\cdots(x - x_{n+k-1})]dx \quad \cdots\cdots (14)$$

となる。M=3，K=2 の場合について(11)式と(14)式を書き下すと，

$$y_{n+1} = y_{n-3} - \frac{4h}{3}(2f_n - f_{n-1} + 2f_{n-2}) \quad \cdots\cdots (15)$$

となり，Milne の予測公式と一致する。

また，(9)式から(15)式を導出した手法とほとんど同じ手法を適用すると，(15)式に対応する式

$$y_{n-1} = y_{n+3} - \frac{4h}{3}(2f_n - f_{n+1} + 2f_{n+2}) \quad \cdots (16)$$

が得られる。(15)式と(16)式は M－n の値が正になるか負になるかに従って使い分ける。

(15)･(16)式の複区予測公式を前もって求めておいた初期近似解 $\{y_n^{(0)}\}$ と第１次の近似解 $\{y_n^{(1)}\}$ が求まる。$y_n^{(1)}$ を求める場合，(15)式の右辺は $y_{n-2}^{(0)}$ と $y_{n-1}^{(0)} \equiv f_{n-1}(x_{n-1}, y_{n-1}^{(0)})$，$y_{n-2}^{(0)} \equiv f_{n-2}(x_{n-2}, y_{n-2}^{(0)})$ および $y_{n-3}^{(0)} \equiv f_{n-3}(x_{n-3}, y_{n-3}^{(0)})$ から成っている。同様に $y_{n+1}^{(1)}$ は，$y_{n-3}^{(0)}$，$y_n^{(0)}$，$y_{n-1}^{(0)}$ および $y_{n-2}^{(0)}$ から(15)式に従って計算すればよい。$\{y_n^{(0)}\}$ はすべて共有メモリに保

存されているから，結局 $y_n^{(0)}$ と $y_{n+1}^{(1)}$ は 2 個のプロセッサがあれば，並列に出来ることになる。同様の考察から n 個のプロセッサがあれば $\{y_n^{(1)}\}$ はすべて並列に計算できる。ただし，刻み幅 h は $y_n^{(1)}$ に対しては nh と取る。

## 4. 各並列アルゴリズムの処理速度

実際の実行速度というのは，関数 $f(x,y)$ の計算処理やオーバーヘッドによる遅延時間など様々な要因を含んでいるが，実行速度の大部分が関数 $f(x,y)$ の計算処理の時間を占めるために，簡単のために，従来の逐次処理計算で引用される $f(x,y)$ の回数と，並列処理で 1 つのプロセッサにおいて引用される関数 $f(x,y)$ の回数の比を実行速度として処理速度を求めた。シュミレーションで得られた処理速度を Table1 に示す。Table.1 では，逐次処理での誤差が $10^{-6}$ 以下に収束するまでの逐次処理の関数 $f(x,y)$ の計算回数と，並列処理での関数 $f(x,y)$ の計算回数の比率を求める事で何倍早くなったかを求めた。実際にこの値がどの程度信用できるかについては，後に述べる PVM を使用して検証していくことになる。

Table.1 各種シュミレーションによる並列アルゴリズムの処理速度

| 初期値問題 | | | | | 処理速度 | | |
|---|---|---|---|---|---|---|---|
| $f(x,y)$ | Initialvalue | $y(x)$ | h | N | Parallel Euler | Parallel Adams | Parallel Runge |
| $y' = -xy$ | $y(0)=1$ | $\exp(-x^2/2)$ | 0.01 | 100 | 7.1 | 8.5 | 12.2 |
| $y' = -1/2y$ | $y(0)=1$ | $sqrt(1-x)$ | 0.01 | 99 | 2.4 | 2.6 | 5.8 |
| $y' = -y\sin x$ | $y(0)=1$ | $\exp(\cos x - 1)$ | 0.068 | 100 | 6.8 | 7.2 | 7.5 |

## 5. 数値計算結果

3 節で示した並列アルゴリズムを用いて，ノイマン型計算機上でシミュレーションを行った数値計算の結果について述べる。シミュレーションプログラムは C 言語を用いて作成した。以下，「並列 Euler 法」などの用語は，初期近似値 $\{y_i^{(0)}\}$ を Euler 法で求め，3 節で提案した並列アルゴリズムに従って得られた結果を示す。

収束の判定は

$$\varepsilon_n^{(i+1)} = \left| \frac{y_n - y_n^{(i+1)}}{y_n} \right| \quad \cdots\cdots(17)$$

によって行う。実用的には，真値は未知であるので(17)式の相対誤差を

$$\varepsilon_n^{(i+1)} = \left| \frac{y_n^{(i+1)} - y_n^{(i)}}{y_n^{(i+1)}} \right| \quad \cdots\cdots(18)$$

としてある。必要ならば $\varepsilon_n^{(i+1)}$ の分布変化を収束の判定に用いればよい。

### 5. 1 区間相対誤差分布

まず始めに，$y' = -1/2y, y(0) = 1$(Euler 法)の数値解法の例を考える。この微分方程式の解は，当然ながら，$y = \exp(-x^2/2)$ である。

Fig.1 は，従来の Euler 法に対する相対誤差分布である。

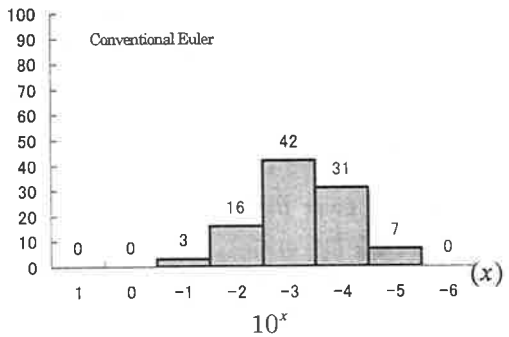
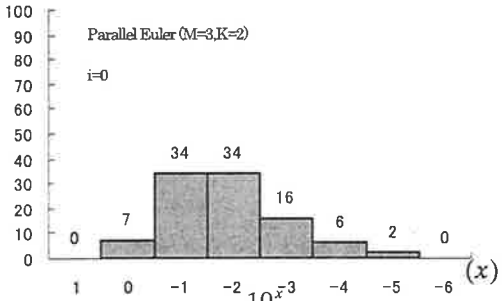


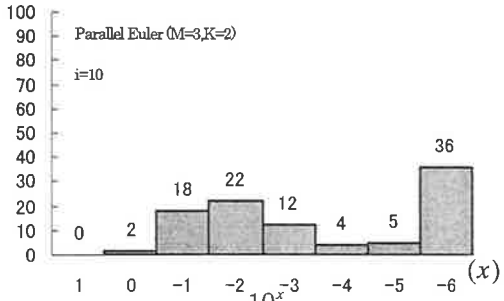

Fig.1 従来の Euler 法による区間相対誤差分布

従来の Euler 法では，$10^{-2} \sim 10^{-4}$ の範囲に相対誤差が多く分布しており，実用的な解としては使い物にならない。

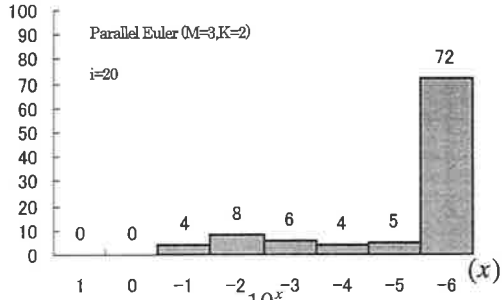
この微分方程式に，本論分のアルゴリズムを適用した結果を Fig.2 に示す。



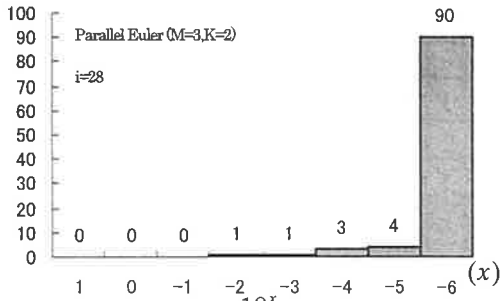

(a) Parallel Euler $\{y_n^{(0)}\}$




(b) Parallel Euler $\{y_n^{(10)}\}$




(c) Parallel Euler $\{y_n^{(20)}\}$




(d) Parallel Euler $\{y_n^{(28)}\}$

Fig.2　$y' = -\dfrac{1}{2y}, y(0) = 1, \{$解は $y(x) = sqr(1-x)\}$,

$h = 0.01, N = 99$ の各種数値解に対する区間相対誤差分布

Fig.2(a)は，逐次近似を行うための初期近似解の誤差分布図である。$\{y_n^{(0)}\}$に対して刻み幅を nh と取っているため，$\{y_n^{(0)}\}$は

$$y_n^{(0)} = y(0) - nh\{1/2y(0)\} \cdots\cdots\cdots\cdots\cdots\cdots (19)$$

となり，従来の Euler 法よりも更に誤差が大きくなっていることに注意したい。

しかし，この初期近似解から出発しても第 10 近似解 $\{y_n^{(10)}\}$ の相対誤差は$10^{-1}$～$10^{-6}$と改善され(Fig.2(b))，第 20 近似解$\{y_n^{(20)}\}$で$10^{-2}$～$10^{-6}$ の範囲になり，第 28 近似解 $\{y_n^{(28)}\}$ で誤差が収束し(Fig.2(c))，$10^{-6}$に相対誤差が集中してきて，かなり精度がよくなっていくのがわかる(Fig.2(d))。

## 5. 2　空間相対誤差分布

次に並列 Euler 法と並列 Adams 法・並列 Runge-Kutta 法による収束の違いについて調べる。ここで Adams 法とは，改良 Euler 法とも呼ばれ，(3)式の代わりに，

$$y_{n+1} = y_n + (h/2)[f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \cdots\cdots (20)$$

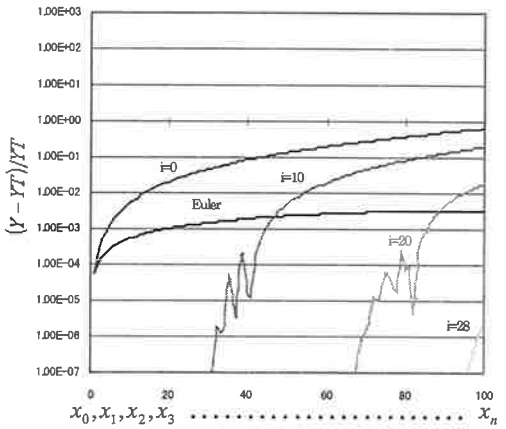
を用いる方法である。

Fig.3 は，$y' = -xy, y(0) = 1$ に対する各分点での相対誤差を表した空間相対誤差分布である。Fig.2 より各種解法による収束の違いについて述べる。

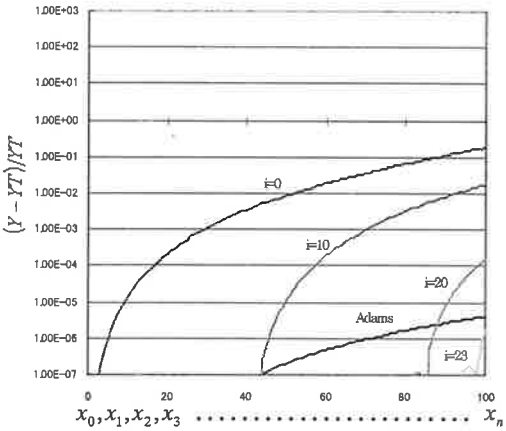
並列 Euler 法では，i=28 で全点において$10^{-6}$以下，並列改良 Euler 法(並列 Adams 法)では i=23 で$10^{-6}$以下，並列 Runge-Kutta 法では，i=16 で$10^{-7}$の分布を示す。

初期近似解の精度は，当然かなり悪いが，本アルゴリズムを適用すると，相対誤差分布が急速に$10^{-6}$以下に集中するようになり，かなりの精度を得る事がわかる。

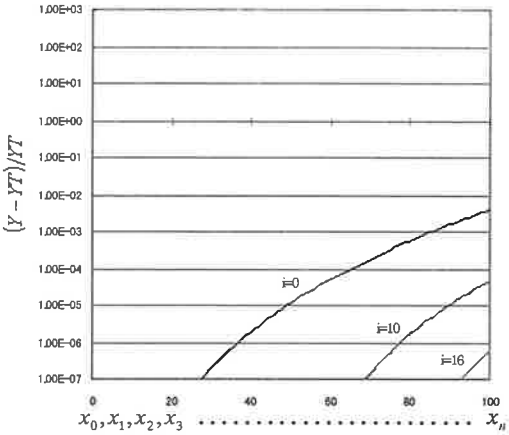
計算速度は，並列 Euler 法と比べ，並列改良 Euler 法(並列 Adams 法)で 1．2 倍，並列 Runge-Kutta 法では 1．8 倍早く計算できる事がわかる．これは，並列改良 Euler




(a) Parallel Euler




(b) Parallel Adams

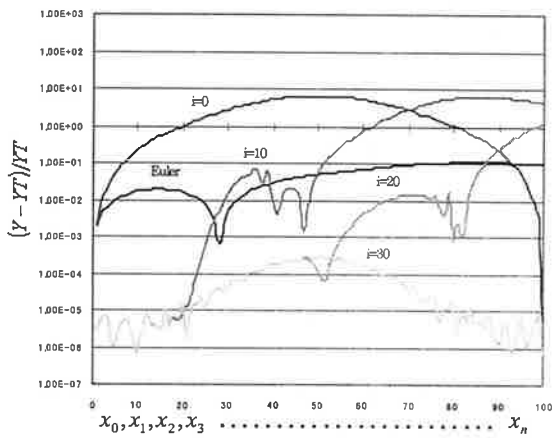



(c) Parallel Runge-Kutta

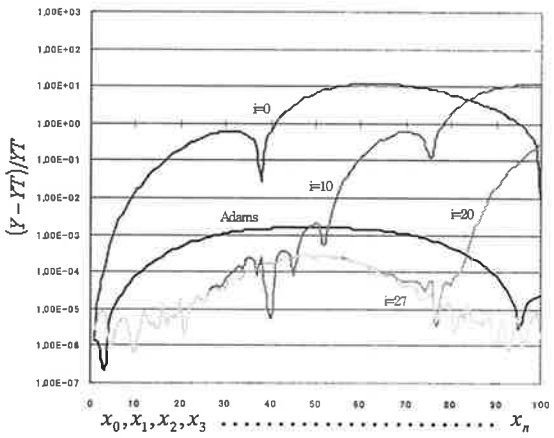
Fig.3

$y' = -xy, y(0) = 1, \{$解は $y(x) = \exp(-x^2/2)\},$

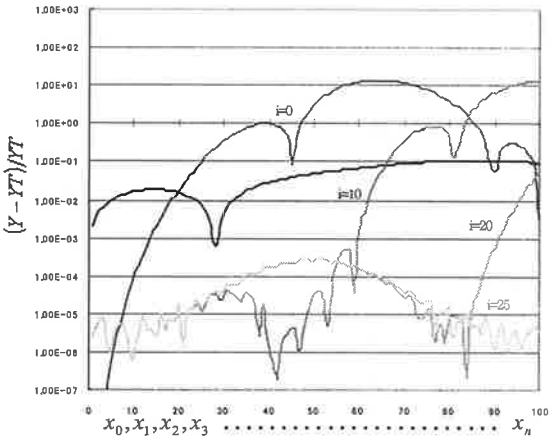
$h = 0.01, M = 3, K = 2$ に対する空間相対誤差分布




(a) Parallel Euler




(b) Parallel Adams




(c) Parallel Runge-Kutta

Fig.4

$y' = -y \sin x, y(0) = 1, \{$解は $y(x) = \exp(\cos x - 1)\},$

$h = 0.0628, M = 3, K = 2$ に対する空間相対誤差分布

法(並列 Adams 法)，並列 Runge-Kutta 法の初期近似解に対する相対誤差が小さいことに起因していると考えられる。

Fig.4 は，$y'=-y\sin x, y(0)=1$ に対する各分点での相対誤差を表した空間相対誤差分布である。Fig.4 はまた，各種解法による収束の違いを示したものでもある。

並列 Euler 法，並列改良 Euler 法(並列 Adams 法)，並列 Runge-Kutta 法ともに$\{y_n^{(0)}\}$にあまり違いが見られない．このような場合には並列逐次近似回数も両者あまり違わなくなる。

また，単純には相対誤差分布は右上がりの曲線と期待されるが，この例では，相対誤差が複雑な分布を示し，現在のところその理由は明らかではない。

## 6. PVM

実際の並列計算機を用いて複区分予測子法等を用いた並列アルゴリズムを実行すべきであるが，あいにく，並列計算機は高価で，すぐ利用できない環境にある。そこで，仮想的な並列計算機を作り出すことが出来る PVM[4)-6)]を利用しようと考えた。

PVM とは Parallel Virtual Machine の略であり，アメリカのオークリッジ国立研究所(Oak Ridge National Laboratory)と，エモリー大学(Emory University)らの研究者たちによる異機種分散計算の研究プロジェクトから生まれた。

PVM は，ネットワークに接続された異機種 UNIX コピュータ群を，仮想的に単一の並列コンピュータとして利用する事を可能にするソフトウェアシステムである。これによって多数のコンピュータの持つ計算パワーを，1 つの大規模計算問題に結集して処理を行う事ができる。

この PVM は，Web 上でフリーウェアとして公開されており，先端科学分野における大規模計算のためのソフトウェアとして世界中で利用されている。また，FreeUnix 系の OS(Linux･FreeBSD･NetBSD･OpenBSD 等)で動作させることができるので，低コストで仮想的な分散メモリ型並列計算機を構築することができる。

PVM を構成するためには，master 側の計算機と slave 側の計算機が r-認証(rsh,rlogin)でお互いに通信できるように設定するか，または r-認証ではなくパスワードの入力で認証を行いたい場合も，ホスト設定ファイルに記述する。正しく設定する事で PVM 間の通信が可能となる。

PVM は基本的に C 言語（あるいは Fortran）でプログラムを記述するが，pvm_initsend()，･pvm_recv()という関数をプログラムに記述する事で master－slave 間，あるいは，slave－slave 間のデータ交換を行う。

そしてそのプログラムの実行は，各計算機で実行形式ファイルが置かれているディレクトリのパスを指定し，master プログラムから slave プログラムを起動することによって可能になる。必要に応じて master－slave 間及び slave－slave 間でデータ交換を用いながら並列処理を行うのである。PVM の master，slave 計算機へのインストールと，各種サンプルプログラムによる動作確認は完了している。

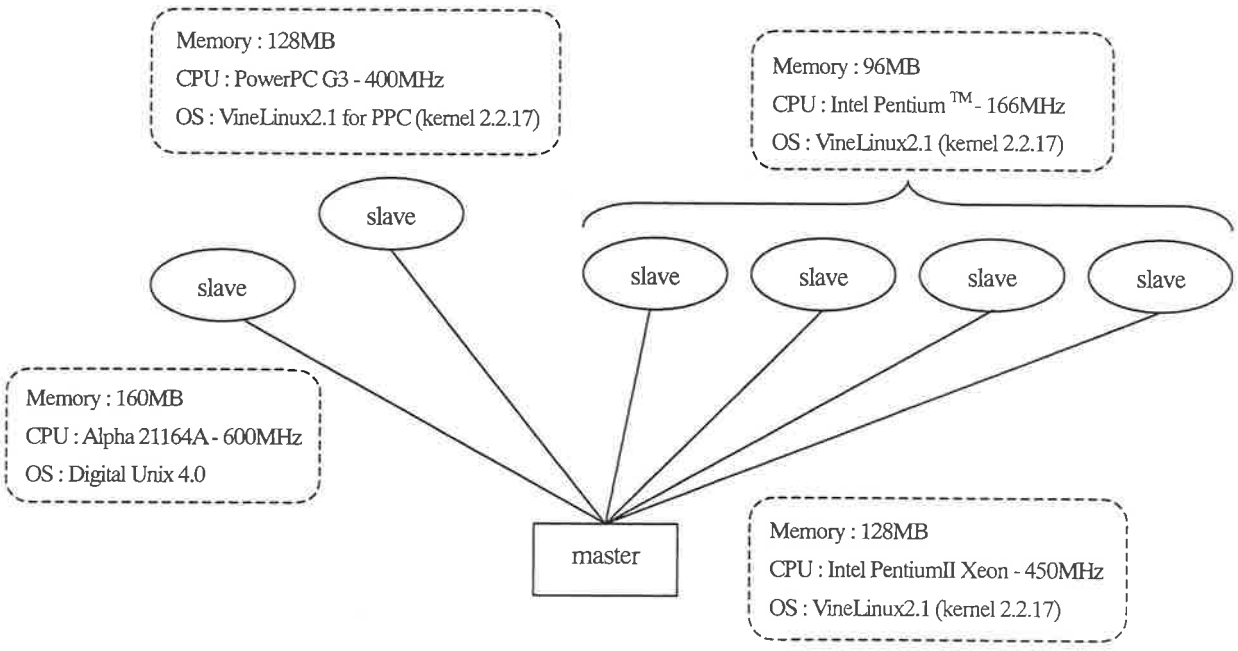



Fig.5　　PVM 環境の概略図

今後，次のようにして複区分予測子法を用いた並列アルゴリズムを検証していきたいと考えている。

(ⅰ) master(Host)で，まず初期近似解を求める。

(ⅱ) master が slave(Client)に必要な初期近似解のデータを必要な分だけ送る。

(ⅲ) slave は受け取った初期近似解のデータを(15)･(16)式の複区分予測子法に適用して，新しい近似解を求める。

(ⅳ) slave で求めた近似解のデータを，更に必要とする slave に送る。

(ⅴ) 収束条件(18)式を満たすまで(ⅲ)～(ⅳ)手順を繰り返し，条件を満足したら master にデータを送り，答えを出力する。

Fig.5 に現在の PVM 環境の概略図を示す。

現実的には，現在,研究室では Fig.5 のように 6 台の計算機を slave として設定できるのだが，分割数 N を例えば 100 個(あるいは 99 個)にした場合に 6 個の slave でどのようにしてデータを扱うか，また Fig.5 のように slave の処理能力に差がある場合，それぞれの slave に効率的に計算させるために，処理能力に応じてどのようにデータ量を割り当てるか，という問題がある。

また，PVM を学内 LAN 上でつなげるだけでなく，インターネット上で実現してみようと計画している。東北大学及び北陸先端科学技術大学院大学の計算機をそれぞれ借用できる事になったので，それぞれの計算機を slave として PVM を動作実験してみたいと考えている。

八戸工業大学－東北大学間，八戸工業大学－北陸先端科学技術大学院大学間の通信には，両大学間の地理的位置関係により，物理的にミリ秒単位の時間がかかる（レイテンシーlatency)。このレイテンシーのほかに，通信線の容量，実際に選択された経路，その他の理由によるレイテンシーが加わる。レイテンシーの実体の解明と，その改善も重要な研究テーマとなるであろう。

更に，八戸工業大学はギガビットネットワーク JGN に参加しており，ギガビットネットワークを用いた PVM の利用も１つの興味ある研究プロジェクトとして考えている。

## 7．まとめ

VLSI 技術の発展により，同一プロセッサを多数用いる並列処理計算機システムは高速演算が必要とされている科学数値計算の分野で大いに期待されている。本報告では，予測子法と逐次計算手法に基づいた常微分方程式の並列計算アルゴリズムを提案し，各種のシミュレーションを行い，この並列計算アルゴリズムの検討を行った。その結果，この並列計算アルゴリズムは従来の計算アルゴリズムに比べかなりの高速化が実現されることと，数値解の精度も比較的高いことがわかった。今後は，PVM を使用して，シミュレーションで得られた処理速度が実際に正しいかどうか，検証していく予定である。また，数値計算における誤差分布の振る舞いは複雑であり，誤差伝播に関するより正確な解析は今後の問題である。

## 8．参考文献


1) 奈良久･早川美徳･阿部亨；“*数値計算法*”，朝倉書店(1991)

2) 玄光男；“*数値計算とデータ構造*”，共立出版(1994)

3) 小沢一文；“*数値計算法*”，共立出版(1987)

4) 湯淺太一･安村通晃･中田登志之；“初めての並列プログラミング”，共立出版(1999)

5) Al Geist；“*PVM 3 USER'S GUIDE AND REFERENCE MANUAL*” http://www.netlib.org/pvm3/book/pvm-book.html (1993)

6) 村田英明；“*PVM3 ユーザーズガイド&リファレンスマニュアル日本語版*” http://phase.etl.go.jp (1995)