計算機アーキテクチャ　115－7
設　計　自　動　化　78－7
（1995．12．14）

並列数値シミュレーション言語NSL
–分散配列ライブラリによる実装と性能評価–

川合 隆光　市川 周一　島田 俊夫

名古屋大学工学部 電子情報学科

概要

並列計算機のプログラミングの困難を克服するため，我々は，偏微分方程式用の高水準数値シミュレーション言語 NSL(Numerical Simulation Language) を開発中である．従来の NSL に，制御構造と反復解法のための構文拡張を行い，解法の記述能力を高めた．マルチ・ブロック法における通信を扱うために開発した分散配列ライブラリにおいて，通信と演算のオーバラップを実現した．本稿では，新たに導入した構文および，分散配列ライブラリによる NSL システムの実装と，評価結果について報告する．

NSL: Parallel Numerical Simulation Language
–Implementation Using Distributed Array Library and Performance Evaluation–

Takamitsu Kawai　Shuichi Ichikawa　Toshio Shimada

Department of Information Electronics, School of Engineering, Nagoya University

Abstract

To overcome difficulties of multiprocessor programming, we are developing a high-level numerical simulation language NSL (Numerical Simulation Language) for PDE problems. We introduced constructs to solve equations iteratively and to represent control flow for various numerical schemes. We added a facility for overlapping communication and computation to a distributed array library developed to handle communications arising in multi-block method. In this paper, we show the newly introduced constructs and the implementation of NSL system using the distributed array library and its performance evaluation.

1 はじめに

近年，数値シミュレーションの分野において並列計算機の利用が注目されている．しかしそのプログラム開発のスタイルは依然としてFORTRANやCに並列化プリミティブを挿入する方式であり，大規模なソフトウェアを作成する場合，膨大な工数を伴う複雑なプログラミングが要求される．とりわけ分散メモリ型並列計算機では，通信や同期，領域分割に注意してプログラミングを行う必要があるため，プログラム開発は一般に困難である．

この問題を解決するため，高水準の問題記述から，数値シミュレーションを行うプログラムを自動生成するシステムが考えられている．主なものに，DEQSOL [1] [2] (日立製作所)，DISTRAN [3] (慶應義塾大学)，//ELLPACK [4] (Purdue大学)，などがある．これらは偏微分方程式で表される分布系の物理問題を対象とし，高水準言語による問題記述から数値解を求めるFORTRANプログラムを生成する．

我々は，柔軟な領域形状記述と自動並列処理を特徴とする偏微分方程式用数値シミュレーション言語 NSL (Numerical Simulation Language)[5] を開発中である．

本稿では，新たに導入した反復解法と制御構造のための構文，および通信と演算のオーバラップを実現した分散配列ライブラリによる実装とその評価結果について報告する．

2 NSLシステムの概要

NSLシステムは，領域形状，初期条件，境界条件および偏微分方程式を記述したNSLソースファイルを処理し，ターゲット並列計算機上で動作するC言語によるSPMD(Single Program Multiple Data)プログラムを生成する．このプログラムは，Cコンパイラによってコンパイルされ，実行形式となる．同時に分散配列ライブラリ (後述) と呼ぶ専用の通信ライブラリがリンクされる．

2.1 対象とする問題と解法

NSLでは，基本的に時間に関して1階，空間に関して2階までの2次元偏微分方程式の初期値・境界値問題を対象としており．数値解は陽解法に基づく差分法により求める．また，本稿で述べる反復解法機能により，時間依存でない方程式も対象とする．

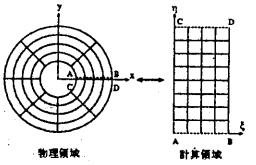

図 1: 境界適合法

差分法は通常，解を直交格子上で求めるので，矩形の物理領域に対しては容易に適用できる．しかし実用的な任意形状の境界を持つ物理領域の場合，曲線境界を階段状の直交格子で離散化すると境界条件を設定する際に精度の低下を招く．

このような問題を解決する有力な手段として，境界適合法 [6] [7] が用いられる．境界適合法は，複雑な曲線境界をもつ物理領域で成立する偏微分方程式を矩形の計算領域に写像して解く方法である (図1)．領域の写像のため偏微分方程式の変換が必要となり複雑な形になるが，これは数値的に解決できる．また，格子点が物理境界上に確実に置かれるため，境界条件を精度良く課すことができる．

物理領域の形状がさらに複雑になると，単一の矩形領域に写像するのが困難になる．そこで，複数の矩形領域 (ブロック) を接続し，物理形状に近いトポロジをもつ計算領域を構成する方法が採られる (図2)．このような方法は一般にマルチ・ブロック法 [8] とよばれる．

NSLでは境界適合法とマルチ・ブロック法における複雑領域の記述を柔軟に行うための構文が用意されており，多様な格子において問題を解くことができる．さらに，並列計算機においてこのような問題を解くとき，複雑なブロックの接続を指定すると発生するプロセッサ間通信も複雑になる．この通信は，専用に開発した分散配列ライブラリと呼ぶライブラリにより管理される．NSLが生成するプログラムは，このライブラリを呼び出すことにより必要な通信を行う．

3 改善点1:構文の拡張

3.1 反復解法構文の導入

従来のNSLでは，偏微分方程式の記述において，1階の時間微分項を持つ (連立) 偏微分方程式の陽解法をサポートしていた．

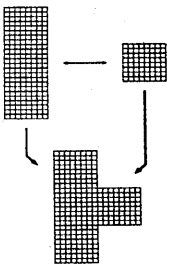

図 2: 複数ブロックの接続

例) 拡散方程式 (熱伝導など)

```
dt[T] = c*(dxx[T]+dyy[T]);
```

例)Navier-Stokes 方程式

```
dt[u] =
  -(u*dx[u]+v*dy[u])-dx[p]
  +(dxx[u]+dyy[v])/Re,
dt[v] =
  -(u*dx[v]+v*dy[v])-dy[p]
  +(dxx[u]+dyy[v])/Re,
dt[p] = -c*c*(dx[u]+dy[u]);
```

これらの例では，各方程式の

```
dt[]
```

の中にある変数が一つのタイムステップで同時に更新される．すなわち，バッファをタイムステップ 2 つ分用意し，交互にデータを書き換える．

$$\begin{cases} u^1 = f(u^0, v^0, p^0) \\ v^1 = g(u^0, v^0, p^0) \\ p^1 = h(u^0, v^0, p^0) \end{cases} \rightarrow \begin{cases} u^0 = f(u^1, v^1, p^1) \\ v^0 = g(u^1, v^1, p^1) \\ p^0 = h(u^1, v^1, p^1) \end{cases} \rightarrow \cdots$$

という処理の流れになる．

次に，時間微分項を含まない，

$$Lu = 0 \qquad (1)$$

の形の方程式を解くことを考える．

ただし，L は微分演算子である (例: $L = \frac{\partial}{\partial x^2} + \frac{\partial}{\partial y^2}$ など)．一般に (1) を解くには，連立 1 次方程式を解く必要がある．連立 1 次方程式を解くには LU 分解などの直接法と Jacobi 法などの反復法があるが，ここでは比較的実装が容易で並列性の高い反復法を考える．(1) は，$\frac{\partial u}{\partial t}$ が $t \rightarrow \infty$ において 0 に収束すれば，時間発展型の方程式

$$\frac{\partial u}{\partial t} = Lu$$

の解を，$t \rightarrow \infty$ まで追うことにより，等価的に解くことができる．しかし，この方法は収束が遅い．すなわち，$=0$ になるまでのタイムステップ数が多くかかる．そこで，より収束率の高い Gauss-Seidel 法や SOR(Successive Over Relaxation) 法などを導入する必要がある．しかし，通常の Gauss-Seidel 法や SOR 法のデータ依存性を正確に維持して並列化するには，例えばウェーブフロント法などを用いる必要があるが，マルチ・ブロック法に適用する場合，データ依存性の維持が困難になる．そこで，並列化に適した Schwarz の交替法 [9] と呼ばれる解法を用いる．これは，全体領域を，オーバラップ領域をもつ複数の部分領域に分け，その部分領域において通常の Gauss-Seidel 法や SOR 法を適用し，オーバラップ領域でデータ交換を行って計算を行う方法である．本来のデータ依存性を崩すため，収束性はある程度低下するが，並列処理できるという特徴は大きな利点である．Schwarz の交替法は multiplicative Schwarz 法と，additive Schwarz 法 [10] [11] の 2 つに分類される．multiplicative Schwarz 法は，本来，メモリの少ない計算機において大規模な問題を解くための技法の一つとして用いられた．まず一つの部分領域の暫定解をメモリにロードし，その部分領域の解を更新し，次に，隣接する部分領域の暫定解をロードし，今計算した部分領域の境界値を境界条件として解を更新する．しかし，一つの部分領域の計算を終えてから次の部分領域に移らなければならないため，並列処理のためには Red-Black オーダリング [12] 等の色分けが必要である．これに対し，additive Schwarz 法はすべてのブロックにおいて暫定解を求めてからデータ交換を行う．このため並列処理を行いやすい．今回は additive Schwarz 法を実装した．時間発展型の解法とは異なり，変数値を確保するバッファはタイムステップ 1 つ分のみ確保し，緩和計算をこのバッファ内で行う．

反復解法に対応する構文として，solve 文を導入し，時間微分の項を含まない方程式を記述できるようにした．次の例で，例えば,T は T について解くことを表す．

例) ラプラス方程式

```
solve[dxx[T]+dyy[T] = 0, T];
```

例)Navier-Stokes 方程式に現れるポアソン方程式

```
solve[
```

```
    dxx[p]+dyy[p] =
      -dx[u*u]-2*dy[u]*dx[v]-dy[v*v]
      +(dx[u]+dy[v])/delt, p
];
```

3.2 制御構文の導入

方程式を解く際，計算中の変数を参照して，それにより方程式のパラメータを変化させたり，処理の流れを変えたい場合がある．このようにより詳細に数値アルゴリズムを記述するためには，制御構造の導入が必要となる．このため，C言語に似た以下の制御構文を導入した．

- for 文
- while 文
- if 文
- do 文

また，ある方程式系を解くための数値解法は，逐次性を持っている場合がある．反復解法構文やスカラ変数への代入文など，一般の文を列挙することにより，逐次性を記述できるようにした．

これらにより，より柔軟なアルゴリズム記述が可能となる．

例)

```
do {
  dt[u] = dxx[u] + dyy[u];
} while (norm[u] < EPS);
```

例)

```
for (i = 0; i < 1000; i++) {
  umax = norm[u];
  if (umax > 1.2) f = f*0.9;
  if (umax < 1.0) f = 1.0;
  dt[u] = c*(dxx[u]+dyy[u]) + f;
}
```

4 改善点2:分散配列ライブラリの改善

4.1 通信と演算のオーバラップの実現

NSLが出力するプログラムでは，ブロック間で必要な通信を，独自に構成した分散配列ライブラリと呼ぶライブラリを呼び出すことにより行う．本ライブラリでは，ブロックの生成／消去，添字の管理，およびブロック間の矩形領域のデータ転送等を関数として持っており，呼び出し側では低レベルなメッセージ通信の詳細を管理する必要がない．現在の実装では，並列計算機 Cenju-3 [13] に提供されているメッセージパッシングライブラリである mini-MPI(MPI[14] のサブセット) の上位に構成されている．本ライブラリでは従来，通信と演算のオーバラップを行っていなかったのに対し，今回，これをオーバラップできるように拡張した．オーバラップは以下のように行う．

(1) ブロックの境界領域のデータに対して，non-blocking send/recv を発行する

(2) 境界を除く内部領域を計算する

(3) (1) で発行したメッセージを待つ

(4) メッセージを受信次第，境界領域を計算する

4.1.1 問題点と解決策

ブロック間でブロックの矩形領域のデータ転送を行う際，転置を伴うデータ転送が必要な場合がある．今回の実装に用いた Cenju-3 において提供されている mini-MPI では現在，ストライド転送を行うためのデータタイプ定義の機能 (プリミティブ `MPI_Type_vector` 等) はサポートされていない．また，仮に標準の MPI が実装されていたとしても，転置の方向によっては一旦別のバッファにコピーして転置し直さなければならないケースが存在する (図3)．本ライブラリではこれを統一的に扱うため，send や recv の際にメッセージ毎に固有の1次元のバッファを確保し，そこにデータをパックする際に，転置の操作を行う．また，nonblocking send/recv の際，MPI システムがメッセージの完了をユーザプログラムに通知するためのデータ領域も併せて確保している．

5 性能評価

今回導入した反復解法機能により，時間依存でない方程式を解く場合の収束性，および，通信と演算のオーバラップによる効果を評価した．評価は並列計算機 Cenju-3 で行った．

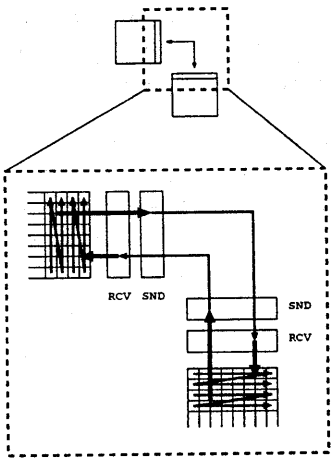

図 3: 転置を伴う転送

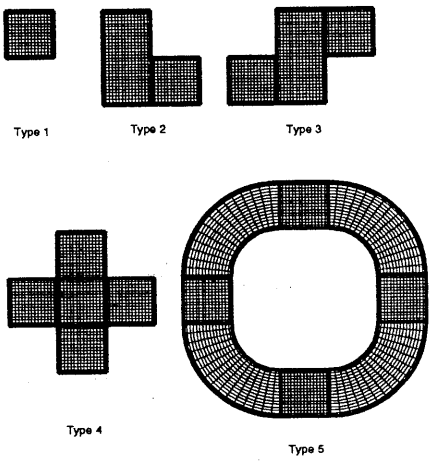

図 4: 領域形状

5.1 収束性

評価問題として，図4のような領域において，ラプラス方程式を反復解法 (solve 文) で解く場合を考える．

部分領域の解法として，

- Gauss-Seidel 法
- SOR 法

のそれぞれの場合について，収束性を評価した．各々のブロックは 4×4 に分割し 16 台のプロセッサに分散した．誤差の絶対値の最大値が一定基準 (10^{-6} とした) 以下になるまでの反復回数を図 5に示す．全般的に収束性は SOR, Gauss-Seidel の順に良いことがわかる．また，それぞれの領域について，時間発展型の解法 (dt[u] = ...) により求解を試みたが，いずれの場合もイテレーション数の上限 (1000 回とした) をに達しても誤差は 10^{-6}以下にならなかった．従って，反復解法機能は時間微分を含まない方程式に対して有効であるといえる．

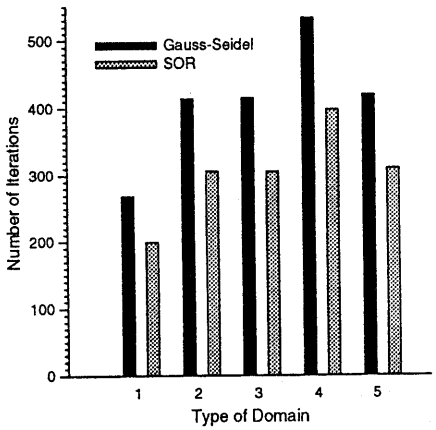

図 5: 反復回数

5.2 通信と演算のオーバラップによる効果

次に，通信と演算のオーバラップによる効果を見るため，問題サイズおよび，プロセッサ数を変化させた場合の性能向上率 P [%] を測定した (図 6)．P は次のように定義した．

$$P = \frac{T_{nonovl} - T_{ovl}}{T_{nonovl}} \times 100$$

ここで T_{nonovl}, T_{ovl} はそれぞれ，オーバラップを行わない場合，行った場合の 1 イテレーションの処理時間を表す．

定性的には，計算時間 T_{calc}, 通信時間 T_{comm} に対し，

$$T_{calc} \approx T_{comm}$$

ならばレイテンシが効果的に隠蔽されると考えられる．T_{calc} は格子点数が増大すると，また，T_{comm} はメッセージの個数及び長さが大きくなると増大する．図 6ではプロセッサ数が増加するにつれて性能向上率も増加する傾向が見られるが，さらに多くのプロセッサを用いるとメッセージに対する計算の割合が少なくなり，性能向上率は減少することが予想される．

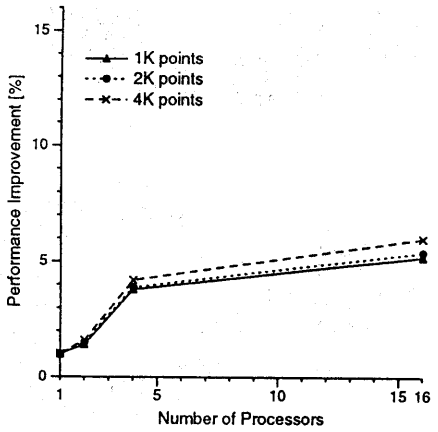

図 6: 性能向上率

6 おわりに

本稿では，偏微分方程式用並列数値シミュレーション言語NSLにおいて拡張した構文，および，通信と演算のオーバラップができるよう拡張した分散配列ライブラリによる実装と評価結果について述べた．今後の課題としては，

- Multigrid法などの，より高い収束率をもつ解法のサポート
- AMR(Adaptive Mesh Refinement)法などの，動的に演算負荷が変化する問題への対応
- 生成コードの最適化
- 3次元への拡張
- 解法の多様化

等が考えられる．

参考文献

[1] 佐川暢俊, 金野千里, 梅谷征雄: 数値シミュレーション用プログラミング言語 DEQSOL, 情報処理学会論文誌 Vol.30, No.1, pp.36–45, 1989.

[2] 大河内俊夫, 金野千里, 猪貝光祥: 数値シミュレーション向き高水準言語 DEQSOL の分散メモリ型並列計算機向けトランスレータに関する一考察, 並列処理シンポジウム JSPP'93, pp.39–46, 1993.

[3] 鈴木清弘, 山崎信行, 他: DISTRAN システムの並列計算機上への実装, 並列処理シンポジウム JSPP'91, pp.301–308, 1991.

[4] Weerawarana, S. et al.: Integrated Symbolic-Numeric Computing in //ELLPACK: Experiences and Plans, Technical Report CSD-TR-92-092, Department of Computer Sciences, Purdue University, 1992.

[5] 川合隆光, 市川周一, 島田俊夫: 数値シミュレーション用 NSL における並列処理手法, 情報処理学会研究報告 95-HPC-55, pp.1–8, 1995.

[6] Thompson, J.F. et al.: Numerical Grid Generation: Foundations and Applications, North Holland, 1985.

[7] Konno, C. et al.: The BF(Boundary-Fitted) Coordinate Transformation Technique of DEQSOL, SIAM, ISBN 0-89871-228-9, pp. 322–326, 1988.

[8] Rubbert, P.E. et al.: Patched Coordinate Systems, Numerical Grid Generation, Ed. Thompson, J.F., North Holland, 1982.

[9] Courant, R. et al.: Methods of Mathematical Physics, Vol 2. Willey, 1962.

[10] Brakkee, E. et al.: A domain decomposition method for the advection-diffusion equation, Report DUT-TWI-94-08, Delft University of Technology, 1994.

[11] Hackbusch, W: Parallel Algorithms for Partial Differential Equations, Vieweg, 1991.

[12] Zhu, J.: SOLVING PARTIAL DIFFERENTIAL EQUATIONS ON PARALLEL COMPUTERS, World Scientific Publishing, 1994.

[13] 広瀬哲也, 他: 並列コンピュータ Cenju-3 のアーキテクチャ, 情報処理学会研究報告 94-ARC-107, pp.121-128, 1994.

[14] Message-Passing Interface Forum: MPI: A Message-Passing Interface, 1994.