SSI: 大規模シミュレーション向け基盤ソフトウェアの概要

西　田　　　晃 †,††

科学技術の様々な分野において, 数値シミュレーションは有力な研究手法として, その重要性を増している. マルチスケールな現象を扱う上で, 大規模な計算資源の利用を前提としたスケーラブルな並列数値演算ライブラリの果たす役割は大きい. しかしながら国内においては, 専用のルーチンを独自に開発するか, またはベンダが提供するライブラリを利用することが多く, 知的共有資産としての汎用のライブラリが開発される例が少なかった. 本プロジェクトでは, 従来それぞれの分野において別個に進められてきた並列アルゴリズムや実装に関する研究をもとに, 理論的な背景を考慮した標準的なソフトウェア基盤を整備することを目指している.

SSI: Overview of Simulation Software Infrastructure for Large Scale Scientific Applications

Akira Nishida †,††

Recent progress of science and technology has made numerical simulation an important approach for studies in various fields. Although scalable and high performance numerical libraries on large scale computing resources are indispensable tools for handling various multiscale phenomina, few projects for developing general purpose numerical libraries have been reported in Japan. The object of this project is the equipment of a basic library of solutions and algorithms required for large scale scientific simulations, which have been developed separately in each fields, and the construction of a standard software infrastructure for scalable scientific computing.

1. 研究の背景と目的

近年, 科学技術の様々な分野において, 数値シミュレーションは理論・実験に代わる有力な手法として, その重要性が高まっている. ミクロからマクロに至る様々な現象を扱う上で, 大規模な計算資源の利用を前提としたスケーラブルな並列数値演算ライブラリの果たす役割は大きい. しかしながら, わが国においても現在多数の科学技術アプリケーションが開発されているものの, それらの基礎となる数値演算ライブラリについては, 専用のルーチンを独自に開発するか, またはベンダが提供するライブラリを利用することが多く, 知的共有資産としての汎用のライブラリが開発される例は少ない.

高速ネットワーク環境の発展に伴い, 今後は大規模アプリケーションについても, PC から演算サーバ, スーパーコンピュータに至る多様な資源をネットワーク上で共有し, オープンな環境で研究開発を行う機会が増えてくるものと予想される. このような環境のもとで, 多重化するメモリ階層を考慮したスケーラブルな並列ライブラリの必要性は増大している. そこで, 本研究では従来それぞれの分野において別個に進められてきた並列アルゴリズムや実装に関する研究をもとに, 理論的な背景を考慮した標準的なソフトウェア基盤を整備することを目指し, 以下の内容について研究を行うことを予定している.

† 東京大学大学院情報理工学系研究科コンピュータ科学専攻
Department of Computer Science, the University of Tokyo
†† 科学技術振興機構 CREST
Japan Science and Technology Agency

2. 主な研究項目

本研究の目的は, 従来それぞれの分野において別個に進められてきた並列アルゴリズムや実装に関する知見をもとに, 理論的な背景を考慮した標準的なソフトウェア基盤を整備することにある. 現在様々な応用ソフトウェア開発プロジェクトが並行して実施されているが, 基盤となる共通の汎用ルーチンの開発・普及までを行うことは稀であり, それぞれの用途に適したライブラリを独自に作成するか, または国外で開発された汎用ルーチンを用いるのが普通である. 本研究では, 多様化する計算環境での利用に十分耐え得る頑健かつ高性能な基本演算ライブラリを構築するとともに, 国内の需要に応え得るレベルの高い成果を挙げることを目指している.

本研究では, 大規模ソフトウェア開発の方法論に関して, 数値アプリケーション開発の観点から国内外の研究

動向について詳細な調査研究を実施するとともに, これら関連分野の研究者との研究交流, および国内における科学技術ソフトウェア開発上の問題点を解決するための方策の提案とその具体化を主要目的のひとつとしている．この結果をもとに, 国内において開発されているアプリケーションの調査及び関連分野の研究者との協力により, 国内の科学技術アプリケーション開発において必要と考えられる共通基盤ソフトウェアの内容とその構成の詳細を決定することとした．実際の研究開発においては, 仕様の策定, プロトタイプ実装, 評価用ハードウェアの選定, 性能評価, 公開及び配布など, 各段階で詳細な検討が必要であり, 十分な人的, 物的資源を用意する必要がある．本研究では, 主に固有値解法，連立一次方程式解法，高速関数変換の三分野を想定し, 分野によって最適と思われる研究者への協力を得るとともに, 常に最新の開発環境を提供できるよう, 多様なアーキテクチャから構成される計算機群を導入していく予定である．

以下では主要な研究項目と現時点での主な研究成果について述べる．最終的には, これらを基礎として, 多くの研究者による多彩な研究成果を取り込んでいく予定である．

3. 計算手法, アルゴリズムに関する研究

並列ソフトウェア基盤の設計に当たっては, 常にその計算手法及び実装手法について, 最新の研究成果をもとに見直しを行っていく必要がある．本研究では, 各分野の分担者が中心となり, 研究集会等を積極的に開催することにより, 基礎的な研究の振興を図るとともに, 研究設備を共同研究者に広く提供し, 最新の計算機環境を利用した研究活動を通じて, 完成度の高い基盤ソフトウェアの構築を目指している．

3.1 固有値解法

物性物理や量子化学などの物質科学分野では, 密度汎関数法や第一原理計算を用いた大規模なシミュレーションが必要となることが多い．これらの計算において, 高速な固有値解法に対する需要は大きく, 両分野の研究者が密接に協力することにより, 実用性の高い汎用ソフトウェアを開発することができるものと期待される．本研究グループでは, QR 法などの標準的な固有値解法とともに, 大規模問題において少数の固有値・固有ベクトルを効率的に求める手法についても, 実際の応用に耐え得る高速かつスケーラブルな固有値計算アルゴリズムの設計及び実装を目指している．

大規模疎行列の固有値を数値的に求める場合, いくつかの解法を考えることができ, 主なものとしては, Lanczos 法やその非対称問題への拡張である Arnoldi 法, あるいは量子化学計算で利用されることの多い Davidson 法や, その一種である Jacobi- Davidson 法[17),18)] などの反復解法を挙げることができる．

しかしながら, 実対称行列 A, B に関する一般化固有値問題

$$Ax = \lambda Bx \tag{1}$$

の最小固有値, あるいはこれと同値な問題

$$Bx = \mu Ax, \quad \mu = 1/\lambda \tag{2}$$

の最大固有値を求める場合, この問題は Rayleigh 商

$$\mu(x) = \frac{x^T Bx}{x^T Ax} \tag{3}$$

の極値問題に帰着することができ, 最急勾配方向が

$$\nabla\mu(x) \equiv g(x) = \frac{2(Bx - \mu Ax)}{x^T Ax} \tag{4}$$

であることから, 適当な係数 α_i を用いて

$$x_{i+1} = x_i + \alpha_i p_i, \quad p_i = -g_i \tag{5}$$

より最急上昇法を, また修正方向として

$$p_i = -g_i + \beta_{i-1}p_{i-1}, \quad \beta_{i-1} = \frac{g_i^T g_i}{g_{i-1}^T g_{i-1}} \tag{6}$$

などを用いることにより, 共役勾配法を導くことができる．この手法は 1951 年に Hestenes らにより提案され, Fletcher らが発展させたものであるが[3),8)], 1980 年代以降の Knyazev らの研究[12),13)] により, 代数的マルチグリッド法などの適切な前処理と組み合わせることによって, Lanczos 系の解法と比較してより高速に固有値を計算できることが分かってきた (図 1)[16)].

本手法は比較的新しい解法であり, 収束性など, 明らかになっていない部分も多い．本プロジェクトでは, 大規模固有値解法に対する有力な解法の一つとして評価していくとともに, 非対称問題への拡張や効率的な並列実装手法について, 今後検討を進めていきたいと考えている．

3.2 連立一次方程式解法

流体シミュレーションや構造解析など, 偏微分方程式の数値解が必要となる分野においては, 離散化の過程で大規模な連立一次方程式を扱うことから, 効率的なアルゴリズムに対する需要は大きい．本研究では, これらの応用分野から派生する連立一次方程式の反復解法及びその前処理手法を中心に, 極めて高並列な環境での使用にも耐え得るスケーラブルかつ信頼性の高いアルゴリズムの設計及び実装を行っていく予定である．

近年，大規模連立一次方程式の反復解法として, マルチレベルな解法が多く研究されている．そのような手法のひとつとして, 代数的マルチグリッド（AMG）法を挙げることができる[5)].

一般にマルチグリッド法は

- 並列性に富む
- 逐次計算の場合，問題行列の大きさを $n \times n$ とすると, うまく機能した場合収束までの計算量は $O(n)$ である

といった性質を持つ．AMG 法は幾何的マルチグリッド法と同様の原理にもとづいているが, 問題行列のみを利用して誤差の高周波数成分の取り除かれた粗いレベルの行列を生成し, 幾何的マルチグリッド法と同等の計算量で高速に問題を解くことができ,

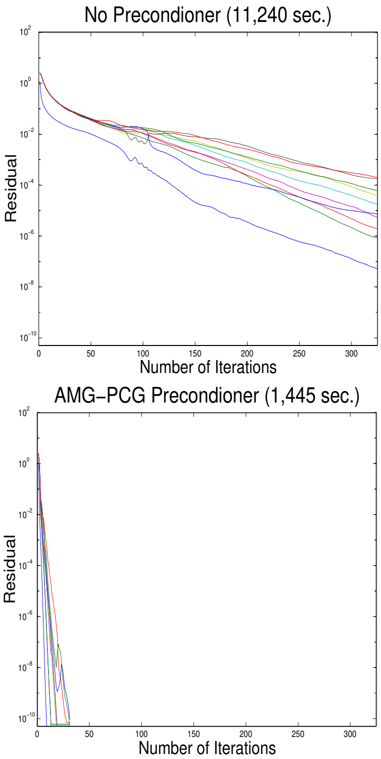

図 1　100^3 元 7 点 3 次元 Laplace 行列（非零要素数 6,940,000）の固有値問題における共役勾配法前処理と収束率の関係（下図は AMG 前処理を用いた場合.）

- 不規則な疎行列に対しても適用可能である
- 異方性問題に対しても有効である

利点を持つ．初年度において，我々は AMG 法のうち最も有力な解法のひとつである smoothed aggregation に基づく algebraic multigrid 法[24] を前処理として用いた共役勾配法（SA-AMGCG 法）の効率的な並列実装手法を提案するとともに，既存のワークステーションクラスタ上で最大 1562 万次元（250 × 250 × 250）の 3 次元ポアソン方程式を解き，ICCG 法（Localized ILU 前処理付共役勾配法）との比較から，大規模な問題になるほどより高速に解けることを示した（図 2）[6]．

AMG 法は汎用の前処理手法としては現時点で最も有効なアルゴリズムであり，スケーラビリティに関しても優れた特性を持つ．本プロジェクトにおいてもより詳細な研究を行っていくとともに，強力な前処理手法の一つとして早期にライブラリとして公開することを検討している．

3.3　高速関数変換

気象予測や地球流体解析など，球面上の流体方程式を扱う必要のある分野では，スペクトル法などの高速積分変換を用いた高精度な解法が利用されている．本研究では，これらの分野の研究者との協力により，アルゴリズムレベルにおいて並列度の高い解法の研究及び設計を行うとともに，高性能な並列高速フーリエ変換ライブラリを中心に，実際の計算環境において高い実効性能を得ることのできる高速関数変換ソフトウェアの開発を行っていく予定である．現段階では，分散共有メモリノード内での高効率な高速フーリエ変換ライブラリの開発を中心に行っている．

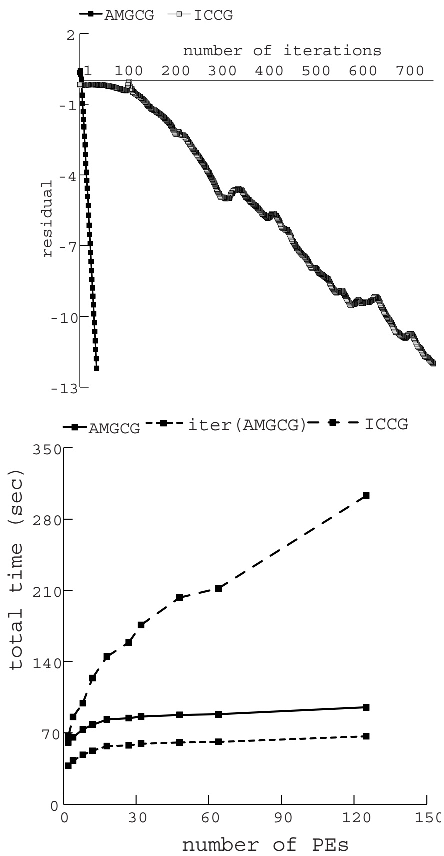

図 2　250^3 元 3 次元 Poisson 問題を解いた場合の AMGCG 法，ICCG 法の収束特性 (Sun Blade 1000 125PE を使用) と同クラスタ上での計算時間（iter は反復解法部の計算時間を示す.）

近年，並列アーキテクチャ技術の発展により，高性能なマイクロプロセッサを複数搭載した共有メモリ型並列計算機が比較的容易に利用できるようになってきた．主記憶を分散配置するネットワーク結合型の分散共有メモリアーキテクチャは，バス結合型のアーキテクチャと比較してより大規模な構成を組むことができるが，主記憶に対するアクセス時間はそのプロセッサと同じノード内であるか他のノードにあるかで不均等になる．一般に大規模な計算では各プロセッサのメモリアクセスに局所性がある場合が多く，このような場合には各ノードの主記憶へのデータ配置方式を工夫することによって，より多くのメモリアクセスをノード内に限定することができる．ノード内のメモリアクセスの割合が多ければ，プロセッ

サ数が増えても高いスケーラビリティを持つ可能性が高くなり，高い拡張性が必要となる大規模科学技術計算に適している.

そこで, 初年度においては, 分散共有メモリ型アーキテクチャの NEC Itanium ccNUMA サーバを利用し，並列 FFT アルゴリズムのデータ配置方法とその最適化手法について，実機上で様々なデータを元に検討した. キャッシュを有効利用する従来のアルゴリズムにおいては, キャッシュ内で行われている FFT アルゴリズムに Stockham FFT 等 bit-reverse 処理を要しないアルゴリズムが用いられることが多いが，我々はこの bit-reverse 処理を主記憶アクセス時間に隠蔽することで, in-place アルゴリズムを用いることができるような手法を提案し，実際にほぼ隠蔽可能であることを確認した. これにより, 2^{24} 点 FFT の計算において, 8PE 上でピーク性能の 13.3% に相当する 3.12Gflops の性能を引き出すことに成功した[19].

4. ハードウェア, システム技術に関する研究

近年の情報技術の発展に伴い, 計算機の利用形態は著しい変化を遂げつつある. 本研究では, これらの動向を踏まえ, 今後国内で普及すると思われる利用形態を想定しつつ, これらの上で高い性能を発揮することのできる並列ソフトウェア基盤を設計, 構築していくため, 常に最新の開発環境を整備することを目指している.

初年度においては, 共通開発環境として共有メモリ型並列計算機 (SGI Altix 3700), 小型ベクトル計算機 (NEC SX-6i), PC クラスタ (dual Intel Xeon プロセッサ搭載サーバ 16 台) を導入するとともに, 10Gbps ネットワークに対応する高性能ルータ (Cisco C6509) によりこれらを接続し, 基礎的な開発環境を整備した.

ハードウェア技術に関しては, 省電力化, 高密度化の進展とともに, 従来と比較してより高並列なプログラミング環境が中心的になっていくものと思われる一方, プロセッサの速度と比較してメモリレイテンシが相対的により増大していくことから, チップ内でのマルチスレッド処理もより一般的になっていくものと考えられる. したがって, 前節で述べた並列ライブラリの重要性は, より大きくなっていくものと考えられる.

このような環境では, 分散したノード間の通信スケジュールやプロセスの配置を最適化することが重要となる. 我々のグループでは, 通信最適化ライブラリに関する研究の一環として, まずスケジューリングの影響が大きいと考えられる木構造ネットワークにおける集合通信の最適化について研究を行った．不均一なネットワークを考えた場合, 最適スケジューリングの計算は NP 困難な問題となるが, ここでは最適解の探索に深さ優先探索による分枝限定法を用い, 木構造の対称性から来る冗長性を同型判定アルゴリズムにより省くことにより, 短時間で解けることを示した. さらに, ここで求めた最適解を用いて集合通信を行うことにより, 汎用のメッセージ通信ライブラリ (MPICH-Score, LAM/MPI) と比較して大幅に通信時間が削減される場合があることが分かった[10]. 今後, より大規模な環境に対して適用できるようこの手法を拡張することにより, 低レベルな通信の最適化が可能になるものと期待される.

5. ソフトウェア技術に関する研究

本研究の実施に当たっては, 関連分野の多様な需要に応えつつ, 汎用性が高く, かつ使いやすいソフトウェア基盤の提供を目的としている点を踏まえ, ソフトウェア技術の方向性や, 関連応用分野における研究開発の現状に関する十分な調査が必要であると考えられる.

本プロジェクトでは, 数値解析に関して理論的な知識を持つ研究者を中心にメンバーを構成しており, 一方で応用分野の研究者を主な利用層として想定している. このため実装に用いる言語としては, できるだけ既存の逐次言語に近く, 並列化が容易なものが望ましいが, 一方で性能上の要求は厳しく, 大規模な並列環境で高い性能が得られるものでなければならない.

一方, インタフェースの設計においては, 利用者がどこまで柔軟に操作を記述できるかが重要な問題となる. このような処理を効率よく記述する並列ライブラリとして, 線形解法を実装した PETSc[2] や Aztec/Trilinos[7),23] などを挙げることができる. これらはオブジェクト指向に基づくプログラミングインタフェースを採用しており, ベクトル, 行列などから構成されるオブジェクトに対する基本的な操作を組み合わせることにより, 複雑な機能を持つライブラリを構築している.

このようなインタフェースの採用は, データに対する操作の詳細を隠蔽する上で効果的であるが, オブジェクトに対する操作は規定されたものに限られる. 共有メモリ環境に関しては対応する並列ライブラリをある程度機械的に適用することが可能であるが, 分散メモリ環境を利用する場合はライブラリで規定されたデータ分散方式を用いる必要があり, 一般に API を用いずに明示的に指定することはできない. これらを記述できるライブラリを新規に開発することも可能であるが, 複雑な構文を解釈するためのプリプロセッサが必要となり, 現実的でない. したがって, 応用分野での利用に適したライブラリとするためには, 初期の段階から高度に完備した命令群を定義する必要がある. なお, この場合 Aztec など C++ で実装されたライブラリが実現しているように, 演算子の多重定義機能を持つ言語 (C++, Fortran 9x など) を使用することにより記述の柔軟性を与えることができるが, 並列処理に関する明示的な指示は依然として困難である[1].

5.1 データ分散と記述方式

一方, MPI (Message Passing Interface) ライブラリを用いて記述されたプログラムや HPF (High Performance Fortran) などの並列言語で記述されたプログラムからライブラリを呼び出す場合には, 呼び出し側においてもデータの分散方式を容易に指定することができるため, これらの問題点を回避することができる.

以下では, 既存の分散メモリ向けプログラミングインタフェースについて簡単にまとめておく[22].

5.1.1 HPF

HPF (High Performance Fortran) は, 1991 年に Rice 大学の Ken Kennedy 教授のもとで仕様策定が開始され, 1993 年に HPF 1.0[9] が, また 1997 年にこれを整理した HPF 2.0[14] がまとめられた. HPF では, 利用者がデータ分散を明示的に指示するための最小限の構文を Fortran 言語に付加し, それ以外の作業を処理系に任せることを目指している. プログラムは逐次言語と同様のシングルストリームのセマンティクスで記述され, データは大域的に扱われるため, 利用者は明示的な通信管理やデータ分散を行わなくてよい. 国内においては, 地球シミュレータプロジェクトを中心として, 主要な HPF ユーザにより性能と実用性を向上させるための独自の仕様拡張である HPF/JA 1.0[11] が策定され, PC クラスタ向けの HPF 開発環境の整備も地球シミュレータセンター及び HPF 推進協議会を中心として進められている. MPI 等の他の並列化インタフェースとの共存にも対応しており, 逐次ライブラリの並列化の容易さを考慮すると, 上位プログラムの記述言語として HPF を想定することも選択肢として妥当であると思われる.

5.1.2 MPI

MPI (Message Passing Interface)[4] は, SPMD (Single Program Multiple Data Stream) 形式に基いて利用者がプロセッサ間のデータ転送を明示的に指定するためのライブラリである. Argonne 国立研究所が開発した仮想デバイス上への参照実装である MPICH[25] をもとに多くの商用計算機に移植されており, 分散メモリ向けのプログラミングインタフェースとして最も広く用いられている.

5.1.3 Co-Array Fortran

Co-Array Fortran[20] は Cray 社が開発したデータ並列型の言語であり, Fortran 95 にデータ分散を記述するための最小限の拡張を行っている. MPI と同様に SPMD 型のモデルを採用しており, image と呼ばれる独立したメモリイメージを持つ複数の実行ストリームがプログラムを重複して実行する. どのストリーム上のデータであるかは, データに co-array と呼ばれる配列を付加することにより指定できるようになっている. 各プログラムは非同期に実行され, ストリーム間でのデータ参照の際には明示的な同期が必要となる.

5.1.4 Global Arrays

Global Arrays[15] は仮想的な共有メモリへのアクセスをライブラリ呼び出しの形で実現するインタフェースである. グローバル配列の宣言, グローバル配列へのアクセス, データ一致制御などはライブラリを呼び出す形で実現されている.

この他, ソフトウェア分散共有メモリなども含めることができる. 一般に, MPI のようにライブラリ呼び出しにより通信を実現する方式は移植性が高く, コンパイラへの負担は軽い. 同様に, Co-Array Fortran や Global Arrays は計算マッピングをユーザが指定するため, HPF より軽量なインタフェースとなっている. Co-Array Fortran は次期の Fortran 規格への追加が検討されているが, 専用の処理系が必要であり, 現時点では移植性に乏しい. 現段階では, 新規のライブラリに関して MPI と共に HPF を開発言語として使用することとし, MPI プログラムからの呼び出しにも対応させることにより, 効率的に作業を進めることを予定している. これらと合わせて, 最適化通信ライブラリなどを統合していく予定である.

インタフェースに関しては, オブジェクト指向言語を用いた簡潔なものと, 並列プログラムからの呼び出しにも対応した自由度の高いものの両者を用意することが望ましい. 本プロジェクトでは, 並列化インタフェースを含む多様な言語で書かれたプログラムに対応するとともに, まもなく仕様が確定する Fortran 2003[21] などを用いた標準的なインタフェースを開発することを検討している.

6. 研究成果の配布, 普及活動

本研究の成果については, 開発段階においてもネットワークを通じて広く一般の研究者に配布し, フィードバックをもとに, より汎用性の高いソフトウェアとしていく予定である. 特に応用ソフトウェア開発プロジェクトとの共同研究を促進し, 大規模問題等への適用に耐え得る頑健なソフトウェア基盤を最終的に提供することを目指している.

7. む す び

計算機の能力は, 今後も飛躍的に発展していくものと期待され, それに伴って, 計算機を利用した数値シミュレーションはより広い分野で活用されるようになっていくものと予想される. 本研究は, 従来国外の研究成果に依存してきた大規模並列アプリケーション向けのソフトウェア基盤を国内で実現する上で, さきがけとなるものであり, 関連応用分野の基盤を提供するとともに, 今後の研究のモデルケースになることを期待している. 本研究の成果物は, 国内外での長期間の利用に耐えるものとして設計され, またその内容についても, 十分な理論的検討が行われた完成度の高いものにしていく予定である. このような性格の研究は国内においても過去に例が少なく, 我が国における知的財産の形成に資するものと思われる .

謝辞　本報告をまとめるにあたり, 活発にご議論頂きましたプロジェクトメンバーの諸氏に感謝致します. な

お，本研究は科学技術振興機構戦略的創造研究推進事業(研究領域「シミュレーション技術の革新と実用化基盤の構築」) CREST プロジェクト「大規模シミュレーション向け基盤ソフトウェアの開発」(平成 14-19 年度) によるものである．

参 考 文 献

1) S. Balay, W. Gropp, L. McInnes, and B. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, 1997.

2) S.Balay, W.D. Gropp, L.C. McInnes, and B.F. Smith, *PETSc Users Manual*, Tech. Rep. ANL-95/11 - Revision 2.1.1, 2001.

3) W. W. Bradbury and R. Fletcher, *New Iterative Method for Solution of the Eigenproblem*, Numer. Math., 9 (1966), pp. 259–267.

4) L.Clarke, I.Glendinning, and R.Hempel, *The MPI Message Passing Interface Standard*, tech. rep., March 94.

5) A.J. Cleary, R.D. Falgout, V.E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge, *Robustness and scalability of algebraic multigrid*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1886–1908.

6) A.Fujii, A.Nishida, and Y.Oyanagi, *Parallel AMG Algorithm by Domain Decomposition*, IPSJ Transactions, 44 (2003), pp. 9–17.

7) M. Heroux, R. Bartlett, V. H. R. Hoekstra, J.Hu, T.Kolda, R.Lehoucq, K.Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams, *An Overview of Trilinos*, Tech. Rep. SAND2003-2927, Sandia National Laboratories, 2003.

8) M. R. Hestenes and W. Karush, *A method of gradients for the caluculation of the characteristic roots and vectors of a real symmetric matrix*, J. Res. Natl. Inst. Stand. Technol., 47 (1951), pp. 45–61.

9) High Performance Fortran Forum, *High Performance Fortran language specification, version 1.0*, Tech. Rep. CRPC-TR92225, Houston, Tex., 1993.

10) Y. Hourai, A. Nishida, and Y. Oyanagi, *Optimal Broadcast Scheduling on Tree-structured Networks*, IPSJ Transactions, 45 (2004), pp. 100–108.

11) Japan Association of High Performance Fortran, *HPF/JA Language Specification Version 1.0*, 1999.

12) A.V. Knyazev, *Preconditioned eigensolvers—an oxymoron?*, Electron. Trans. Numer. Anal., 7 (1998), pp. 104–123 (electronic). Large scale eigenvalue problems (Argonne, IL, 1997).

13) ———, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.

14) J. Merlin and B. Chapman, *High Performance Fortran 2.0*, in Sommerschule uber Moderne Programmiersprachen und Programmiermodelle, Technical University of Hamburg-Harburg, 1997.

15) J. Nieplocha, R. J. Harrison, and R. J. Littlefield, *Global Arrays: A Nonuniform Memory Access Programming Model for High-Performance Computers*, J. Supercomputing, 10 (1996), pp. 197–220.

16) A. Nishida, *Performance Evaluation of Parallel AMG Preconditioned Conjugate Gradien t Methods for Large Scale Eigenproblems*, IPSJ SIG Notes, 2004 (2004), pp. 205–210.

17) A. Nishida and Y. Oyanagi, *A Survey of the Jacobi-Davidson Method and its Characteristics for Larg e-scale Eigenvalue Problems*, IPSJ Transactions, 41 (2000), pp. 101–106.

18) ———, *A Parallel Implementation of the Jacobi-Davidson Method using OpenMP and its Evaluation on Shared Memory Architectures*, Proceedings of Joint Symposium on Parallel Processing 2002, (2002), pp. 79–86.

19) A. Nukada, A. Nishida, and Y. Oyanagi, *Parallel Implementation of FFT Algorithm on Distributed Shared Memory Architecture and its Optimization*, IPSJ Transactions, 44 (2003), pp. 1–8.

20) R. Numrich and J. Reid, *Co-Array Fortran for parallel programming*, 1998.

21) J. Reid, *Fortran 2000 CD ballot and WG5's response*, ACM FORTRAN Forum, 22 (2003), pp. 13–17.

22) Research Institute of Advanced Information Technology, *2002 Annual Report*, tech. rep., Japan Information Processing Development Center, 2003.

23) V.R. Tuminaro, *Official Aztec User's Guide*.

24) P. Vanek, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numerische Mathematik, 88 (2001), pp. 559–579.

25) G. William and E. Lusk, *User's guide for mpich, a portable implementation of mpi*.