社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－ARC－162 （17）
2005－HPC－101 （17）
2005／3／8

RAO-SS: Autopilotを用いた疎行列ソルバにおける実行時自動チューニング機構

石　井　良　規[†]　片　桐　孝　洋[†,††]　本　多　弘　樹[†]

本稿では，疎行列ライブラリ用の行列の非疎度によるパラメータ自動選択手法を提案する．さらに，提案手法を利用してライブラリ実行時に適するパラメータを自動選択し高速化する機構である，RAO-SS (Run-time Auto-tuning Optimizer for Sparse Solvers) を提案し，その性能評価を行った．RAO-SSは，実行時に自動チューニングを行うことを支援する目的で，ミドルウェアのAutopilotを用いて実装されている．また，適用対象の数値計算ライブラリはSuperLUである．性能評価の結果，提案手法をSuperLUに適用した場合，SuperLUのデフォルトオーダリングパラメータの実行時間に対して平均1.2倍，最大で3.6倍の速度向上を達成した．またRAO-SSにおいては，Autopilotのソフトウェアオーバーヘッドは無視できるほど小さいことが明らかになった．

RAO-SS: A Run-time Auto-Tuning Facility for Sparse Solvers with Autopilot

Yoshinori Ishii[†]　Takahiro Katagiri[†,††]　Hiroki Honda[†]

In this report, a method of automatic parameter selection for the sparsity of input matrix is proposed. In addition, we propose and evaluate RAO-SS (Run-time Auto-tuning Optimizer for Sparse Solvers), which is an auto-tuning facility by using the proposed method in run-time for the library. RAO-SS is implemented by using Autopilot, which is middle-ware to support run-time auto-tuning. The target numerical library is SuperLU. The result of performance evaluation indicated that: (1) the speedup factors of 1.2 for average and 3.6 for maximum to the default execution were obtained; (2) the software overhead of Autopilot can be ignored in RAO-SS.

1. はじめに

近年，気象シミュレーションやロケットの軌道予測シミュレーションなど様々な分野で，行列演算の高速化が必要とされている．この行列計算の高速化のために，高性能な数値計算ライブラリを利用することは効果的な手段である．数値計算ライブラリは，密行列ソルバ，および疎行列ソルバに分類できる．密行列ソルバに対して疎行列ソルバが有する特筆すべき特徴は，行列の数値特性，行列の零要素の位置などによって，高速化手法が異なる点である．したがって一般的に，疎行列ソルバを利用するには，密行列ソルバよりも複雑なパラメータ設定を必要とする．このことからユーザにとって，適するライブラリの選定やパラメータ設定に関し負担がかかるという問題がある．また誤ったライブラリ選択，誤った実行パラメータ設定をしてしまうと，計算性能が劣化するとともに計算が終了しないという，密行列ソルバではあまり生じていなかった問題も生じる．

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
†† 科学技術振興機構さきがけ
PRESTO, JST

密行列ソルバや疎行列ソルバに関係なく，現在ほとんどの数値計算ライブラリでは，性能パラメータに対して固定値（デフォルト値）が設定されて公開されている．このことから，計算機環境や入力行列特性に応じた最適なパラメータが設定されない問題があることが知られている．

これらの問題を考慮し，自動的に適切なライブラリを選択，および自動的に実行パラメータの設定を行うライブラリ構成法の研究が行われている．特に数値計算ライブラリにおいては，パラメータ設定に関する自動チューニング技術が活発に研究されている．

数値計算ライブラリにおける自動チューニングにおいて，その適用タイミングから分類すると，以下の二つの型が知られている[8]．一つは，ライブラリを新しいマシン環境にインストールする時に行うインストール型である．もう一つは，ライブラリを実行する時に行う実行時型である．本稿では疎行列ソルバの特性を考慮し，実行時型の自動チューニングに焦点を当てる．

実行時型のソフトウェアでは，Autopilot[1]，ActiveHarmony[7]という基盤ソフトウェア（ミドルウェア）が知られている．インストール時と実行時型を融合したライブラリとしては，I-Libが知られている．また片桐らのFIBERでは，上記の2種のチューニング型に加え，新たに

ユーザによるパラメータ設定後に自動チューニングを行う実行起動前型を導入した[5].

そこで我々は，ライブラリ実行時に適するパラメータを自動的に選択し，ライブラリを高速化する機構である RAO-SS (Run-time Auto-tuning Optimizer for Sparse Solvers) を提案する[6]．RAO-SS は，実行時に自動チューニングを行うことを支援する目的で，ミドルウェアの Autopilot を用いて実装されている．本稿では，RAO-SS の自動チューニングによる性能向上の効果検証を行うことを目的とした．

本稿の構成を以下に示す．2 章で，Autopilot について説明する．3 章で，行列演算に用いるソルバである SuperLU について説明する．4 章では，提案する RAO-SS の概要，および実装の詳細を述べ，5 章で性能評価をする．最後にまとめと今後の課題を述べる．

2. Autopilot の説明

2.1 概要

Autopilot とは，複雑な計算機管理ポリシーを記述するために設計されたミドルウェアである．たとえば，分散並列計算機の性能を最大化するためには，お互いに衝突するような要求がなされる．例えば，レイテンシの最小化とハンド幅の最大化である．このような複雑な計算機管理ポリシーを記述するために，Autopilot では，Fuzzy Logic をミドルウェアの機能として導入している[1]．これら Autopilot の基本機能を，数値計算ライブラリ実行時に自動チューニングを行うための支援ソフトウェアとして用いたことが，RAO-SS の大きな特徴である．Autopilot はデーモンとしてシステムに常駐するので，自動チューニング機能付き GRID ポータル[3]などに容易に組み込むことが可能である．

Autopilot を用いて，逐次，並列プロセッサ上でプログラムを実行すれば，遠隔のプロセスで同時実行される内部データにアクセスしたり，修正したりすることが可能となる．Autopilot の特筆すべき機能は，幅広いアプリケーションをサポートする機能，すなわち適応性があり相互に作用するプログラム操作に対してサポートできる機能である．Autopilot は，アプリケーションに対して適応性のある制御やリソースマネジメントポリシーを実行するために，パフォーマンス Sensor や Decision Procedure，そしてポリシー Actuator などの命令セットを与える．

Autopilot は，データの受け渡しを行うための機能をサポートしている．それは，データを送る Sensor，データを受け取る Actuator である．また，Sensor からデータを受け取り最適なパラメータを選択する部分である Decision Procedure は，本稿で提案する RAO-SS の実装に関して重要な意味を持つ．さらに，Autopilot の機能の一つである Fuzzy Logic は，数値計算ライブラリ開発者である Autopilot ユーザに判断できないあやふやな部分，例えば行列を解く際の最適なパラメータ選択処理などを，自動チューニング機能として実装する際に活用できる．

2.2 Decision Procedure

Autopilot の Decision Procedure は，アプリケーションやシステムの振る舞いに基づいて，動的に定義したポリシーを選択してくれる．Autopilot の Decision Procedure のインフラは，並列環境や広範囲の分散システム上でアプリケーションの適応性のある制御や，システムリソースマネジメントポリシーを実現するための要素を与える．

図 1 は，Autopilot の機能の一つである Fuzzy Logic における Decision Procedure の例である．まず，システムやアプリケーションから Sensor を通じてデータの送信を行う．Sensor は，Fuzzy Logic Rules Base へデータを送信し，ここで定義したポリシーを基に Decision Procedure で決定する．Decision Procedure で決定したデータを，今度は Actuator が受信しシステムやアプリケーションへ送信する．

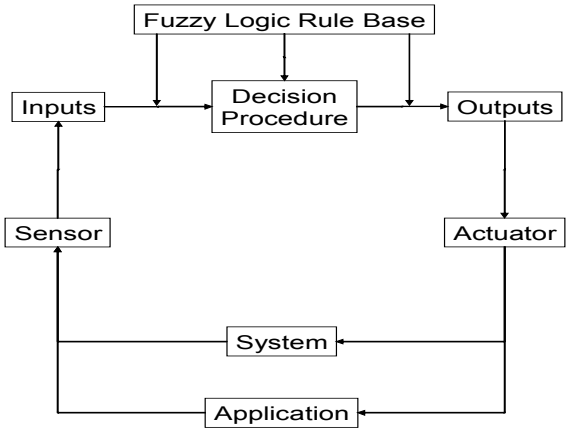

図 1　Autopilot Fuzzy Logic Decision Procedure の構成

3．SuperLU の説明

3. 1 概要

SuperLU[2]は，大規模で疎な，非対称行列を係数とする連立一次方程式を，直接解法（LU 分解法）で高速に解くライブラリである．SuperLU のライブラリルーチンは，前進消去法，後退代入法，partial pivoting（部分枢軸選択法）を行い，行列の LU 分解を行う．

また SuperLU のパッケージは，逐次計算機用，shared memory 並列計算機用，および distributed memory 並列計算機用の３種類ある．

本研究では，逐次用の SuperLU Ver3.0 を用いた．

3.2 オーダリングパラメータ

SuperLU で使われている，性能に関するパラメータは以下の通りである．

(1) オーダリングパラメータ

(1.1) Approximate minimum degree column ordering （デフォルト） 略：COLAMD

(1.2) Natural ordering 略：NATURAL

(1.3) Minimum degree ordering on structure of A^T+A 略：AT＋A

(1.4) Minimum degree ordering on structure of A^T*A 略：AT×A

(2) パネルサイズ

(3) 緩和パラメータ

(4) スーパーノードの最小数

(5) 最小の 2 次元ブロッキングの行次元

(6) 最小の 2 次元ブロッキングの列次元

このうち，本稿で自動チューニングの対象となるパラメータは，(1)オーダリングパラメータである．オーダリングの手法としては，三角化 Markowitz の方法，Tewarson の方法等）や帯幅縮小（RCM 法，最小次数順序法等），ブロック化（Stewart の方法，dissection 分割法）など，多くのアルゴリズムが知られている [4]．

入力される行列の零要素数，非疎度の割合などによって最適なオーダリングパラメータは異なるので，RAO-SS ではこのオーダリングパラメータをライブラリ実行時にチューニングすることを，主な自動チューニング項目とした．

4. RAO-SS の提案

4.1 概要

RAO-SS は，ライブラリ実行時に適するパラメータを自動的に選択し，ライブラリを高速化する機構である．実装には Autopilot の Fuzzy Logic が用いられている．ここで，適用対象の数値計算ライブラリは，SuperLU である．

4.2 設計方針

RAO-SS は，以下の設計方針のもとに設計されている．

(1) 入力行列を実行時に高速に抽出するため，特徴量として非疎度を用いる．

(2) パラメータ選択処理を容易に実装するため，Autopilot の Decision プロセスを利用する．

(3) レガシーなコードやオブジェクトに対して再コンパイルなしに RAO-SS 機構を付加するために，Autopilot をミドルウェアとして利用する．

(4) システムスループットの向上のために高速化するのではなく，個々のジョブに対するデフォルトオーダリングパラメータでの実行に対し，極端な速度低下を防ぐことを目的とする．

(2)，(3)の設計方針は Autopilot の基本機能から活用できることがわかる．

図 2 に本稿で提案する RAO-SS のシステム構成を示す．まず，ユーザが対象の行列を入力するとその入力行列の非疎度を求める．非疎度の計算方法は，以下の式(1)を利用して求める．

非疎度 ＝ 行列の非零要素数／(N＊N) (1)

N は行列の次元数である．この非疎度は，システムに常駐する Autopilot の入力デーモン（Sensor）に渡される．常駐プロセスは，非疎度の値から高速化されると判断されるパラメータを，Autopilot の Decision プロセスで決定する．選ばれたパラメータは，出力デーモンを通して疎行列ダイレクトソルバである SuperLU を呼び出すユーザプロセスが受け取る．

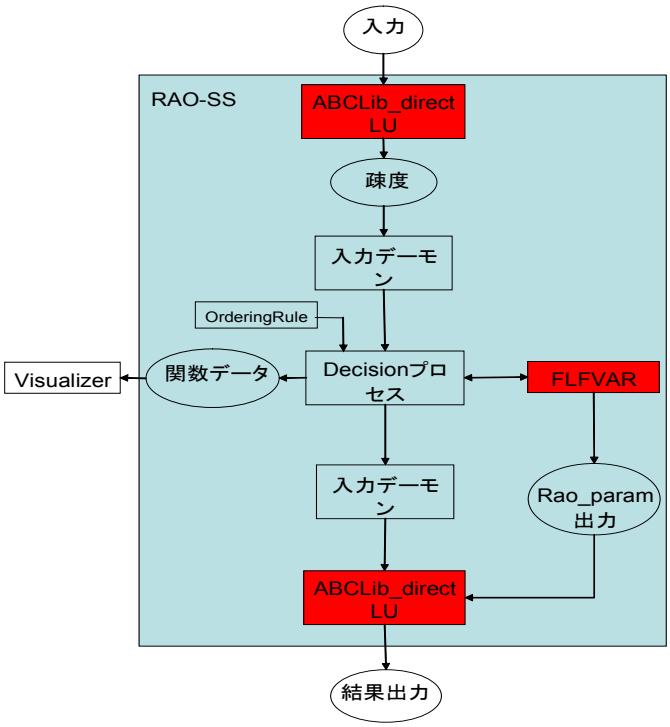

図 2 RAO-SS システム構成図

4.3 パラメータ選択手法

行列の非疎度を式(1)で計算し，その非疎度の割合によって SuperLU で使われているオーダリングパラメータの自動選択を RAO-SS で行う．非疎度によって最適なオーダリングパラメータが異なるが，選び方によっては，きわめて多くのメモリが動的に確保されシステムがダウンすることも

あるので注意が必要である．なお本稿では，誤ったパラメータを設定した場合に，自動的にそのプロセスを消滅し再実行する機能は考慮しない．

本稿で用いる疎行列の非疎度の割合は，0.0001%～10%程度である．

図 3 は，行列の非疎度によるオーダリングパラメータ自動選択のポリシーを示した図である．縦軸真値が 1 に近いほうのオーダリングパラメータを選ぶ．なお選択すべきオーダリングパラメータが重複する部分は，どちらのオーダリングパラメータを用いても良いが，実行時に一つに決められる．この指定が，Fuzzy 機能がもつ特徴である．

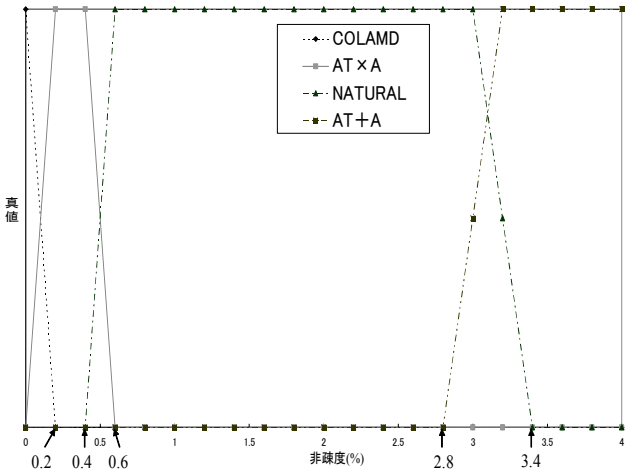

図 3　非疎度によるオーダリングパラメータ選択ポリシー

4.4 実装の詳細

RAO-SS を実装するに当たって，SuperLU と Autopilot のインストールを前提としている．SuperLU は，3 章でも述べたとおり逐次版を用い，また Autopilot をインストールするに当たって Globus もインストールを行わなければならない．

まず，Autopilot の Fuzzy Logic ライブラリに提案した非疎度によるオーダリングパラメータ設定機能を実現する．また，図 3 の設計は，後述の実験行列を用いて経験的に最適化したものである．すなわち，経験的に有効となるパラメータ設定手法（ヒューリスティック法）を RAO-SS では採用した．

Autopilot のメインプログラムに入力行列の読み込み部分，非疎度の計算部分，非疎度によるオーダリングパラメータ選択部分を，RAO-SS の機能として組み込んだ．

5. 性能評価

5.1 測定環境

この章では，RAO-SS および SuperLU（以下 SLU）の性能評価を行う．測定には，PC クラスタを用いた．PC クラスタのマシンスペックは，OS：RedHat Linux7.8，CPU：Intel Pentium4 （1.8GHz），Memory：512MB，コンパイラ：gcc ver.2.96 である．Autopilot のバージョンは 2.4 である．

また，性能評価に利用した行列は全部で 32 種類である．全て非対称正方行列である．これらの行列は，University of Florida Sparse Matrix Collection から入手した．

5.2 最適パラメータの調査

まず，Pentium4 上で SLU を用いてオーダリングパラメータ（以下 OP）を手動で変えて性能評価を行った．デフォルト OP に比べて最適な OP の実行時間が，どれほど向上しているかを図 4，5 に示した．

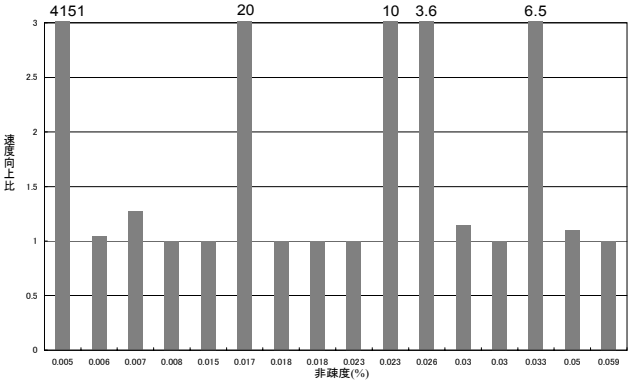

図 4　デフォルト OP に対する最適 OP の速度向上比
（非疎度 0.005～0.059）

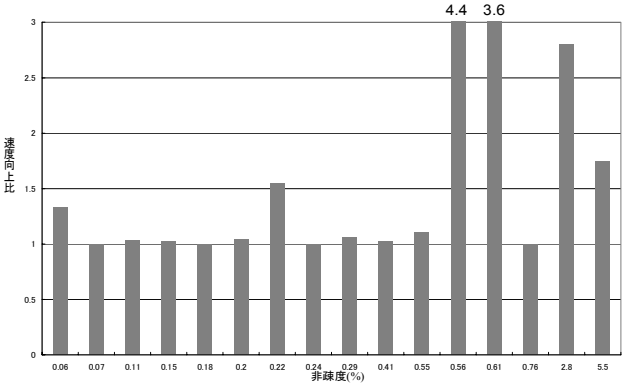

図 5　デフォルト OP に対する最適 OP の速度向上比
（非疎度 0.060～5.5）

図 4，5 を見てみると，縦軸（速度向上比）が 0～3 で表示してあるが，最高で 4000 倍以上，その他にも 20 倍程度の

速度向上が得られた．ここで，速度向上比を求める式は以下の式(2)のとおりである．

速度向上比 = デフォルト OP の実行時間 ／
最適な OP の実行時間　　(2)

速度向上比が 1 を超えているほど，SLU のデフォルト OP に比べて速度が向上しているということになる．すなわち，自動チューニングによる効果が期待できる．

5.3 提案手法を組み込んだ SuperLU の性能評価

ここでは，行列の非疎度を計算して得た情報によりパラメータ選択を行う提案手法を，SLU に組み込んだ場合の性能を評価する．提案手法の利用により，必ずしも最適な OP が選択されるとは限らないが，最適な OP に比べて自動選択された OP が，どれほど最適性能に近づけているか評価することを目的とした．

図 6，7 に，デフォルト OP に対する速度向上比を示す．式(2)と同様に，デフォルト OP の実行時間と非疎度によるパラメータ自動選択を適応し測定した時間との速度比の式は，以下の式(3)である．

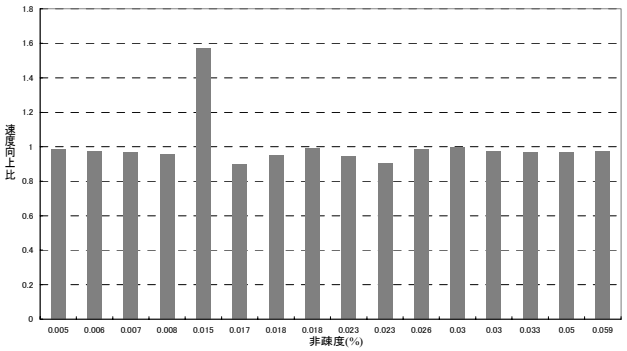

図 6　デフォルト OP に対する SLU'の速度向上比
（非疎度 0.005～0.059）

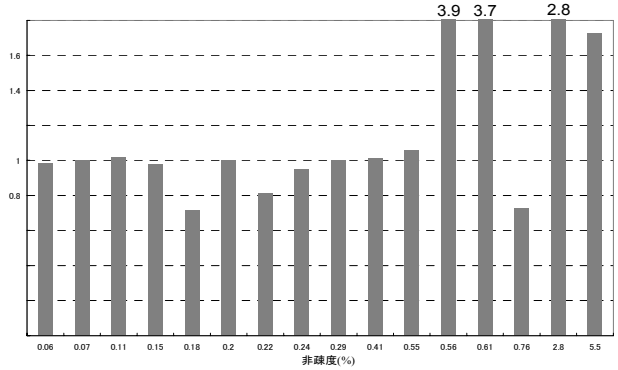

図 7　デフォルト OP に対する SLU'の速度向上比
（非疎度 0.060～5.5）

速度比 = デフォルト OP の実行時間 ／
非疎度によるパラメータ自動選択での実行時間　　(3)

これより先では，SuperLU に提案する非疎度によるパラメータ選択法を付加したソフトウェアを SLU'と表記し，SuperLU オリジナルのソフトウェアを SLU と表記して区別する．

5.4 RAO-SS による性能評価

ここでは，提案した RAO-SS の性能評価を行う．性能評価の目的は以下の二つである．

まず，SuperLU のデフォルト OP での実行時間と比べて，どれほど性能がでているか評価することである．

次に，SLU'の実行時間と比べて，Autopilot を用いるとどれほど性能が劣化するのか，すなわち Autopilot のソフトウェアオーバーヘッドの検証である．

図 8，9 はデフォルト OP と RAO-SS の実行時間を比較した図である．式(3)と同様に，速度比を求める式は以下の式(4)，(5)である．

速度比 = SuperLU 単体でデフォルト OP の実行時間 ／
RAO-SS を用いたパラメータ選択方針での実行時間　(4)

速度比 = SLU'の実行時間 ／
RAO-SS を用いたパラメータ選択方針での実行時間　(5)

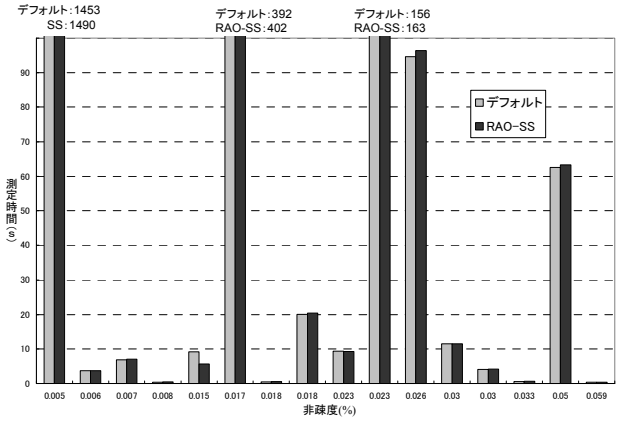

図 8　RAO-SS とデフォルトの実行時間（非疎度 0.005～0.059）

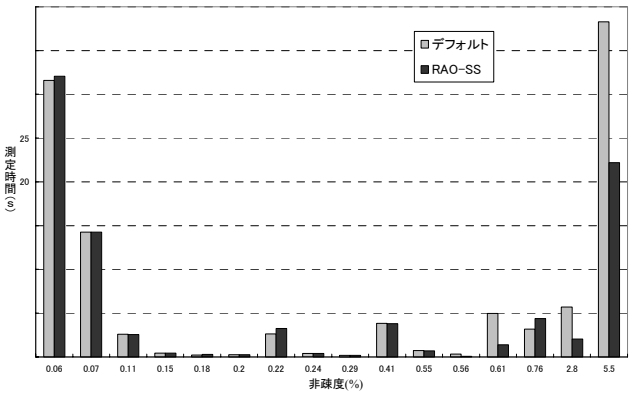

図 9　RAO-SS とデフォルトの実行時間（非疎度 0.060～5.5）

5.5 考察

SuperLU の最適パラメータの考察

SLU のデフォルト OP の実行時間に比べ，最適な OP での実行時間は，最大で 4000 倍以上速くなる例が見られた．その行列は，電子回路に関する行列である．他の電子回路に関する行列も，デフォルト OP に比べて 20 倍近く速くなる例もあった．したがって，電子回路に関する行列特性から，デフォルト OP での実行は性能劣化もたらすと考えられる．また，SLU'では速度比が 0.9～1.0 のものが多数見られた．この性能劣化の原因としては，実行時間が小さいものが多く，かつ測定回数が全ての行列で 1 度しか行っていないため，誤差が混入したと考えられる．また，提案手法の実装によるオーバーヘッドも要因として考えられる．

RAO-SS の考察

RAO-SS では，デフォルト OP に対する速度向上比が SLU'と同程度になった．速度比が 0.9～1 となり性能劣化する原因としては，SLU'での理由と同様に，測定誤差，および Autopilot に非疎度による選択機構を組み込んだことによるシステムオーバーヘッドが考えられる．

6. まとめと今後の課題

本稿では，RAO-SS での自動チューニング効果検証と SuperLU に非疎度を取り入れた場合の効果検証を行った．

SuperLU のデフォルトのパラメータ実行に比べ，RAO-SS により，以下のような速度向上が期待できる．

- 全速度比の合計 = 39.5
- 全行列の種類 = 32
- 平均速度向上比 = 39.5 / 32 = 1.23

すなわち RAO-SS を用いると，SuperLU のデフォルトパラメータの実行時間に比べ，平均で 1.23 倍，最大で 3.6 倍の速度向上が得られた．

また，この速度向上から，SuperLU に提案手法を組み込んだソフトウェアと RAO-SS は，同程度の速度向上が得られたといえる．すなわち Autopilot のソフトウェアオーバーヘッドは十分に小さく，無視できる程度である．

今後の課題を以下に列挙する．

- **より効果的な手法の開発**：手動で SuperLU のパラメータを変えてチューニングした場合に比べ，現在の提案手法は必ずしも最適なパラメータが自動選択されていない．手動チューニングでは，最大で 4000 倍以上もの速度向上が得られた例があった．したがって今後の課題として，より最適なオーダリングパラメータ選択できるヒューリスティック手法の開発が必要である．
- **多くのテスト行列を用いた評価**：提案した Fuzzy 機能によるパラメータ選択ポリシーは，32 種のテスト行列に対して効果的になるように設計した．したがって，より多くのテスト行列を用いた評価が必要である．
- **動的に変更できるポリシー指定機構の開発**：動的に選択ポリシーを設定できる手法が構築できれば，より精度の高いパラメータ選択機構が構築できるかもしれない．動的にパラメータ選択ポリシーが変化する機構の研究も興味深いテーマである．さらに本稿では実装および評価までに至らなかった「行列の対角要素からの距離」[9]情報を取り入れて，非疎度情報と組み合わせ評価するという方式も考えられる．このことにより，さらに細かいパラメータ選択方針が設定できるだろう．また，Autopilot 自体がもつ，Fuzzy 機能によるパラメータ最適化機能をさらに活用したパラメータ選択方式と比較することも重要な課題である．
- **GRID ポータルへの適用と評価**：Autopilot は Globus 上で動作する．したがって RAO-SS は，Globus が動作する GRID 環境への適用が容易である．RAO-SS を GRID 上で性能評価することも重要な課題である．

参考文献

1) Ruth Aydt and Randy Ribler: Autopilot User's Manual（2003）
2) James W.D，John R.G，Xiaoye S.Li: SuperLU User's Manual (2003)
3) 直野健，今村俊幸，恵木正史:GRID コンピューティング環境における行列ライブラリ向け性能保証方式の検討，情報処理学会論文誌，コンピューティングシステム，vol.45，No.SIG 6 (ACS 6)，pp105-112 (2004)
4) 西出隆二，片桐孝洋，金田康正:ブロック幅を動的決定する疎行列連立一次方程式の直接解法，情報処理学会研究報告，2001-HPCS-89 (2001)
5) 片桐孝洋，吉瀬謙二，本多弘樹，弓場敏嗣:FIBER：汎用的な自動チューニング機能の付加を支援するソフトウェア構成方式，情報処理学会研究報告，2003-HPC-94，pp1-6 (2003)
6) 石井良規，片桐孝洋，本多弘樹:RAO-SS：疎行列ソルバにおける実行時自動チューニング機構，HPCS2005 ポスター論文集, p2 (2005)
7) Tapus,C.，Chung,I-H. and Hollingsworth,J.K.，Active Harmony: Towards Automated Performance Tuning，Proceedings of SC2002 (2002)
8) 片桐孝洋：ソフトウェア自動チューニング：数値計算ソフトウェアへの適用とその可能性，慧文社 (2004)
9) 石井良規，片桐孝洋，本多弘樹，弓場敏嗣：Autopilot を用いた疎行列ソルバにおける実行時自動チューニング機構の設計，電子情報通信学会総合大会，東京工業大学大岡山キャンパス，D-3-9，pp.28 (2004)