ハイパフォーマンス
コンピューティング　86−4
（２００１．５．２５）

並列分散計算によるBMI固有値問題解法

合田 憲人 [+]　　二方 克昌 [++]　　原 辰次 [+]
[+] 東京工業大学　　[++] 日本 IBM

概要

BMI(Bilinear Matrix Inequality) 固有値問題は、2 つのベクトル変数による双線形行列関数の最大固有値を最小化する解を求めることを目的とした数値最適化問題の 1 つである．本稿では，並列分散計算による BMI 固有値問題の ϵ 最適解求解法を提案するとともに，提案手法の PC クラスタおよび Grid 計算システム上での性能評価結果について述べる．提案手法を PC クラスタおよび Grid 計算システム上に実装し，性能評価を行った結果，逐次計算に比べて，128CPU 上の PC クラスタ上での提案手法による計算時間が約 1/91，地理的に分散された複数の計算機から構成される Grid 計算システム上での計算時間が約 1/4.8 に短縮される等，提案手法の有効性が確認された．

Parallel Algorithm for the BMI Eigenvalue Problem

Kento Aida[+]　　Yoshiaki Futakata[++]　　Shinji Hara[+]
[+]Tokyo Institute of Technology　　[++]IBM Japan
aida@dis.titech.ac.jp, {nihou,hara}@cyb.mei.titech.ac.jp

abstract

The BMI (Bilinear Matrix Inequality) Eigenvalue Problem is one of optimization problems and is to minimize the largest eigenvalue of a bilinear matrix function. This paper proposes a parallel algorithm to compute the ϵ-optimal solution of the BMI Eigenvalue Problem on parallel and distributed computing systems. The performance evaluation results of the proposed scheme on PC clusters and a Grid computing system showed that the proposed scheme reduced computation time of the BMI Eigenvalue problem to 1/91 of the sequential execution time on a PC cluster with 128CPUs and reduced that to 1/4.8 on a Grid computing system.

1　はじめに

BMI(Bilinear Matrix Inequality) 固有値問題は、双線形行列関数の最大固有値を最小化する解を求めることを目的とした数値最適化問題の 1 つである．ヘリコプターの旋回動作制御を初めとする多くの制御システムの解析・設計問題が，BMI 固有値問題の解を求めることにより解決できるため，本問題の解法は制御システムの設計・解析において重要な役割を持っている．本問題は NP 困難な問題であるが [1]，実用的な計算手法として，分枝限定法を用いて最適値の誤差が一定の範囲内に収まる解 (ϵ 最適解) を有限時間内に求める手法が提案されている [2, 3]．しかしこれらの従来手法では，実用的な制御問題を短時間で解くことはこれまで困難であり，計算時間の短縮が切望されていた．

本稿では，BMI 固有値問題の ϵ 最適解求解に要する計算時間を短縮するために，並列分散計算システム上での同問題の並列解法を提案するとともに，提案手法の PC クラスタおよび Grid 計算システム上での性能評価結果について述べる．一般に分枝限定法では，元の問題の解の探索空間を分割して子問題を生成し，各子問題毎に目的関数の下界値，上界値，解を計算する，という処理を繰り返す．提案手法では，BMI 固有値問題の ϵ 最適解を求める分枝限定法を Master-Worker 方式を用いて並列化する．具体的には，Master が複数の子問題をそれぞれ Worker に割り当て，各 Worker 上では，Master から割り当てられた子問題に関する計算，即ち各子問題の探索空間における下界値，上界値および解の計算を行う．また分枝限定法の並列計算では，Master から Worker に一度に割り当てられる計算量，即ち計算粒度がプログラム全体の計算時間に影響を与える．

本稿では，提案手法において，Worker に割り当てる計算粒度が全体の計算時間に与える影響についても評価を行う．

提案手法を PC クラスタおよび Grid 計算システム上に実装し，性能評価を行った結果，逐次計算に比べて，128CPU から成る PC クラスタ上での提案手法による計算時間が約 1/91，地理的に分散された複数の計算機から構成される Grid 計算システム上での計算時間が約 1/4.8 に短縮される等，提案手法の有効性が確認された．また，提案手法における Worker の計算粒度に関しては，提案手法を PC クラスタ上で実行する場合と Grid 計算システム上で実行する場合とでは，それぞれ異なる計算粒度を定義する必要があることが確認された．

2 BMI 固有値問題

本節では BMI 固有値問題を定義するとともに、その ϵ 最適解を求める分枝限定法について説明する。

2.1 BMI 固有値問題の定義

はじめに，対称行列 $F_{ij} = F_{ij}^T \in \mathcal{R}^{m\times m}(i = 0,\cdots,n_x, j = 0,\cdots,n_y)$ が与えられる時，ベクトル変数 $\mathbf{x}$，$\mathbf{y}$ に関する双線形行列関数 $F : \mathcal{R}^{n_x} \times \mathcal{R}^{n_y} \rightarrow \mathcal{R}^{m\times m}$ を (1) 式により定義する．

$$F(\mathbf{x},\mathbf{y}) := F_{00} + \sum_{i=1}^{n_x} x_i F_{i0} + \sum_{j=1}^{n_y} y_j F_{0j} + \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} x_i y_j F_{ij} \qquad (1)$$

ここで，変数 $\mathbf{x}$，$\mathbf{y}$ は，それぞれ $\mathbf{x} := (x_1,\cdots,x_{n_x})^T$，$\mathbf{y} := (y_1,\cdots,y_{n_y})^T$ である．

BMI 固有値問題は，$F(\mathbf{x},\mathbf{y})$ の最大固有値を最小化する $\mathbf{x}$，$\mathbf{y}$ を求める問題であり，(2) 式により定義される [2, 3]．

$$\Phi(B) := minimize\Lambda(\mathbf{x},\mathbf{y}),$$
$$\Lambda(\mathbf{x},\mathbf{y}) := \overline{\lambda}\{F(\mathbf{x},\mathbf{y})\},\ (\mathbf{x},\mathbf{y}) \in B \qquad (2)$$

ここで，$\overline{\lambda}\{F(\mathbf{x},\mathbf{y})\}$ は与えられた $\mathbf{x}$，$\mathbf{y}$ に対する行列 $F(\mathbf{x},\mathbf{y})$ の最大固有値を，B は $\mathbf{x}$，$\mathbf{y}$ の探索区間を表す．

2.2 分枝限定法による ϵ 最適解求解法

本 BMI 固有値問題では，以下に説明する分枝限定法を用いたアルゴリズムにより，$\Phi(B)$ を ϵ 近傍に大域収束させる解 (ϵ 最適解) を有限時間内に求めることができる [2, 3]．

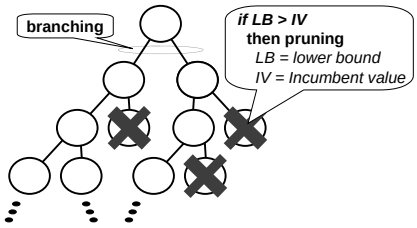

図 1: 探索ツリー

本アルゴリズムでは，以下の処理を $\Phi(B)$ の上限と下限の差が ϵ 未満になるまで繰り返す．(1) 変数 $\mathbf{x}$，$\mathbf{y}$ の解の探索空間を表す問題 (親問題) を 2 分割する (分枝操作)，(2) 分枝操作の結果生成された子問題について，下界値および上界値と，$\mathbf{x}$，$\mathbf{y}$ の解を求めるとともに，$\Phi(B)$ の最良の上界値，即ち暫定値を更新する，(3) 計算の終了した子問題について下界値と暫定値の比較を行い，下界値が暫定値よりも大きい場合は，この子問題について新たな解の探索を続けてもさらに良い解は得られないため，この子問題に関する探索は終了する (限定操作)，(4) 限定操作を受けずに残った子問題についてさらに解の探索を続けるため，その下界値が最小である子問題を選択し，(1) における親問題とする (選択操作)．図 1 は，上記の操作を繰り返すことにより生成される探索ツリーの例である．探索ツリーの根ノードは元の探索問題を意味し，子ノードは子問題に相当する．

3 BMI 固有値問題の並列解法

本節では，本稿が提案する BMI 固有値問題の ϵ 最適解を求める並列解法について述べる．本手法は，2 節で説明した分枝限定法を Master-Worker 方式に基づいて並列化したものである．Master は探索ツリーを管理するとともに，選択操作を行って選択されたノードをそれぞれ異なる Worker に割り当てる．Worker では，Master から割り当てられたノードに対する分枝操作と分枝操作の結果生成されたノードの計算を行い，結果を Master に返す．また，限定操作は Master と Worker の両方で行われる．

以下に，提案手法において主要な処理となる Master 上での選択操作，Worker 上での分枝操作，Master と Worker 上でそれぞれ実行される限定操作の 3

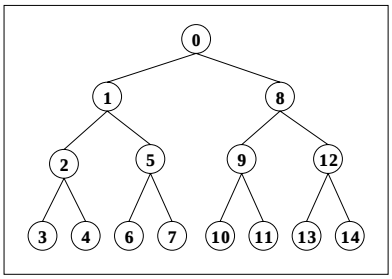

図 2: 分枝ツリー

つについてに述べる．

3.1 選択操作

Master 上で実行される選択操作では，毎回，探索ツリーの葉に相当するノードの中から下界値が最小であるノードを選択し，Worker に割り当てる．下界値が最小であるノードを選択するのは，以下の理由により探索の効率を向上させることができるためである．

1. あるノードが最小下界値を持つ場合，そのノードの下界値は元問題の最適値に関するその時点での最小下界値を意味する．従って，この元問題の最小下界値を更新するためには，最小下界値をもつノードに対して分枝操作を行う必要がある [4]．

2. あるノードが最小下界値を持つ場合，そのノードの下界値は他のノードの計算の結果得られる暫定値より大きくなることはない．従って，最小下界値を持つノードは他のノードの暫定値により限定操作を受けることがない [3]．

3.2 分枝操作

Worker 上では，Master から割り当てられた 1 個のノード (親問題) に対して，[3] の方法により分枝操作を適用して 2 つの子ノード (子問題) を生成する．この時，生成されたそれぞれの子ノードに対してさらに分枝操作を繰り返すことにより図 2 に示すようなツリーを生成できる．本稿では，全体の探索ツリーと区別するためにこのツリーを分枝ツリーと呼ぶ[1] ．図 2 の例は分枝操作を 3 回繰り返して生成された分枝ツリーである．

[1] 分枝ツリーは全体の探索ツリーのサブセットに相当する．

Worker は，生成された分枝ツリー上のノードを深さ優先順で探索し，各ノード毎に下界値，上界値，$\mathbf{x}$ と $\mathbf{y}$ の解を計算する．従って，Master からの 1 回のノード割り当てに対する Worker 上での計算時間，即ち計算粒度は，生成される分枝ツリーの大きさ，即ち Master から割り当てられたノードに適用される分枝操作の繰り返し回数により決定される．

ここで Worker 上での計算粒度に関しては，以下のトレードオフが存在する．提案手法のような分枝限定法の並列計算では，Worker 上でのタスク起動や Master と Worker 間の通信に起因するオーバーヘッド，および Worker 間の暫定値の更新頻度が全体の計算時間に影響を与える．前者のオーバーヘッドについては，これを減少させることによりプログラムの並列計算効率を向上させることができる．一方，後者の Worker 間での暫定値更新については，これをより頻繁に行うことにより各 Worker 上により最新の暫定値が通知されるため，Worker 内での分枝ツリーに対する限定操作を促進し，プログラム全体の計算量を減少させることができる．これら両者はともにプログラムの計算時間短縮に有効であるが，オーバーヘッドを削減するためには，Worker 上での計算粒度を増大することが必要であるのに対し，Worker 間の暫定値更新頻度を増大するためには，Worker 上での計算粒度を縮小する必要がある．後者の理由は，提案手法では Worker へ最新の暫定値の通知は Worker が Master からノードを割り当てられる際に 1 回だけ行われるためである．本稿では，4 節でこのトレードオフについての評価結果を示す．

3.3 限定操作

限定操作は，Master または Worker 上でそれぞれ実行され，(1) ノードの下界値が暫定値より大きい，(2) ノードの上界値と下界値の差が ϵ よりも小さい，という条件のうちのどちらかを満たすノードに関する探索を終了する．(1) については，この条件を満たすノードにさらに分枝操作を適用して探索を続けても暫定値より良い解は求められないため，このノードに関する探索を終了する．また (2) については，本手法で求める ϵ 最適解の条件は，暫定値と最小下界値との差が ϵ 未満であれば十分に満たされるため，(2) の条件を満たすノードについてさらに探索を続けて上界値と下界値の差を縮小する必要はない．従って，このノードに関する探索を終了する．

表 1: PC クラスタの構成

小規模クラスタ	node	(PentiumIII 500MHz x2CPU, 256MB memory) x 4nodes
	network	100Mbps ethernet
	software	linux 2.2.17, MPICH
Prosperous クラスタ	node	(PentiumIII 800MHz x2CPU, 256MB memory) x 132nodes
	network	100Mbps ethernet
	software	linux 2.2.16, MPICH

4 性能評価

本節では，提案手法の PC クラスタおよび Grid 計算システム上での性能評価について述べる．本性能評価では，提案手法を MPI[5] を用いて PC クラスタ上に実装するとともに Ninf[6, 7] を用いて Grid 計算システム上に実装し，実問題および性能評価のために用意した人工的な問題の解を計算した．

4.1 性能評価問題

性能評価に用いる BMI 固有値問題として，実問題であるヘリコプターの旋回動作制御問題 [8]，および性能評価のために変数 $\mathbf{x}$，$\mathbf{y}$ および行列 F_{ij} の大きさや値を人工的に生成した疑似問題を用意し，提案手法によりそれぞれの解を計算した．

ヘリコプターの旋回動作制御問題では，変数 $\mathbf{x}$, $\mathbf{y}$ の次元数，行列 F_{ij} の行列サイズ (m) はそれぞれ $n_x = 10$，$n_y = 2$，$m = 8$ となる．またアルゴリズムの終了条件を決定する ϵ は，$\epsilon = 10^{-4}$ とした．ただし実際の制御問題では，$\Phi(B)$ の暫定値 (Z) と最小下界値 (L) の相対的な差，即ち $(Z - L)/L$ が ϵ 未満になる解が求められれば実用的に十分であるため，本性能評価では，アルゴリズムの終了条件および 3.3 節で述べた限定操作の条件 (2) をこの相対的な差により判断している．

本稿が対象とする BMI 固有値問題では，(1) 子ノードの計算量および入力データサイズが F_{ij} の行列サイズに依存する，(2) 解の探索空間，即ち生成される子ノード数は $\mathbf{x}$，$\mathbf{y}$ の次元数に依存する，という性質を持つ．そこで，特に (1) の性質に注目し，子ノードの計算量がヘリコプターの旋回動作制御問題より大きな問題，即ち Worker 上での計算粒度が大きな疑似問題を人工的に生成し，性能評価を行った．これらの疑似問題では，問題サイズが n により表され，$n_x = n_y = n$，$m = 4n$ である．F_{ij} の要素は乱数により生成した．またヘリコプターの制御問題と同様に $\epsilon = 10^{-4}$ とした．本疑似問題に関する評価結果は，F_{ij} の要素が異なる 5 例の疑似問題を作成して計算した結果の平均値とする．

4.2 PC クラスタ上での性能評価

PC クラスタ上での性能評価では，提案手法のアルゴリズムを MPI(MPICH[5]) を用いて実装し，2 種類の PC クラスタ上で 4.1 節で説明した問題を計算した．表 1 に，本性能評価で用いた小規模クラスタおよび Prosperous クラスタの構成を示す．

4.2.1 小規模クラスタ上での性能評価結果

小規模クラスタ上でヘリコプターの旋回動作制御問題を計算した場合の計算時間を表 2 に示す．表中，Lv は Worker 上で Master から割り当てられた 1 つのノードに適用される分枝操作の繰り返し回数，即ち Worker 上での計算粒度を表している．また，() 内の数値は単一プロセッサ上での最短計算時間に対する計算時間の比，即ち計算速度を示している．ここで，本性能評価における単一プロセッサ上での計算では，逐次計算のために開発したプログラムを用いている．また，同様に疑似問題を小規模クラスタ上で計算した場合の計算速度を表 3 に示す．

表 2，3 より，全ての問題において並列計算による計算時間の短縮ができていることがわかる．特にヘリコプターの制御問題では，8CPU 上での計算時間が 1/6.95 に短縮できている．提案手法では 1 台の CPU が Master として動作するため，この結果はほぼ CPU 台数分の並列計算効果が得られているものといえる．

次に Worker の計算粒度については，全ての問題において $Lv = 1$ の場合が最も計算時間が短いことがわかる．この結果により，PC クラスタ上での計算では，Worker の計算粒度を増大して相対的なオーバーヘッドを削減するよりも，Worker の計算粒度を縮小して Worker 間の暫定値更新頻度を増大させるほうが全体の計算時間を短縮できることがわかる．

4.2.2 Prosperous クラスタ上での性能評価結果

次に，提案手法による並列計算効果のスケーラビリティを評価するために，より大規模な Prosperous クラスタ上でヘリコプターの制御問題および $n = 6$ と

表 2: 小規模クラスタ上でのヘリコプター制御問題の計算時間

	計算時間 [sec]			
CPU 数	$Lv=1$	$Lv=2$	$Lv=3$	$Lv=4$
1	3675 (1.00)	4228 (0.87)	4790 (0.77)	5504 (0.67)
4	1203 (3.05)	1381 (2.66)	1559 (2.36)	1794 (2.05)
8	529 (6.95)	593 (6.20)	601 (6.11)	670 (5.49)

表 3: 小規模クラスタ上での疑似問題 ($n=5$) の計算速度

	計算速度			
CPU 数	$Lv=1$	$Lv=2$	$Lv=3$	$Lv=4$
1	1.00	0.90	0.85	0.80
4	3.00	2.71	2.56	2.42
8	6.83	6.16	5.78	5.42

した場合の疑似問題 (1 例) を計算した結果を表 4 および表 5 に示す．なおここでの結果は，全て $Lv=1$ とした場合の結果である．

表 4，表 5 より，F_{ij} の行列サイズが 24 と比較的大きな疑似問題 ($n=6$) では，128CPU 上での計算時間が逐次計算に比べて約 1/91 と大きく短縮できていることがわかる．これに対して，F_{ij} の行列サイズが 8 と比較的小さいヘリコプターの制御問題では，計算時間に対する相対オーバーヘッドが疑似問題よりも大きいため，32CPU 程度で並列計算による速度向上が飽和している．

4.3 Grid 計算システム上での性能評価

Grid 計算システム上での性能評価では，提案手法のアルゴリズムを Ninf[6, 7] を用いて Grid 計算システム上に実装し，ヘリコプターの制御問題を計算した．

Ninf は，クライアントからサーバに対して計算の実行を依頼する RPC 型の Grid 計算システムである．Ninf による Grid 計算では，クライアント計算機 (Ninf クライアント) 上のプログラム中から Ninf executable と呼ばれるサーバ計算機 (Ninf サーバ) 上で実行されるプログラムが呼び出されることにより，Grid 計算が行われる．ここで，Ninf クライアントからの Ninf executable の呼び出し処理は Ninf-Call と呼ばれる．本性能評価では，Master のアルゴリズムを実行する Ninf クライアントプログラムと Worker のアルゴリズムを実行する Ninf executable をそれぞれ開発した．従って，Ninf クライアントが Master，Ninf サーバが Worker に相当し，Master から Worker へのノードの割り当ては，Ninf クライアント上での NinfCall により実現されている．

表 4: Prosperous クラスタ上でのヘリコプター制御問題の計算時間

CPU 数	計算時間 [sec]	計算速度
1	1144	1.00
4	375	3.05
8	161	7.11
16	78	14.7
32	43	26.6
64	53	21.6
128	42	27.2

表 5: Prosperous クラスタ上での疑似問題 ($n=6$) の計算時間

CPU 数	計算時間 [sec]	計算速度
1	23839	1.00
4	7757	3.07
8	3341	7.14
16	1592	15.0
32	797	29.9
64	427	55.8
128	261	91.3

本実験環境では，東京工業大学の大岡山キャンパスおよび長津田キャンパスに分散された Ninf サーバを用いて Grid 計算を行った．各キャンパスに設置された Ninf サーバおよび Ninf クライアントの仕様を表 6 に示す．Ninf サーバの数は 4 台であるが，サーバ A は 4CPU を搭載しているため，サーバ B，C，D と合わせて全部で 7CPU を Worker として使用する．大岡山キャンパスに設置された Ninf クライアントと同キャンパス内の Ninf サーバ (サーバ B，C，D) 間は 100Mbps のイーサネットにより接続されている．また，大岡山キャンパスと長津田キャンパス間は約 30km の距離があり，専用の光ファイバで接続されている．両キャンパス間のネットワークのバンド幅は 150Mbps であるが，長津田キャンパス内に 10Mbps イーサネットが存在するため，Ninf クライアントと Ninf サーバ (サーバ A) 間のネットワークの利用可能な最大バンド幅は 10Mbps となる．また，両マシン間の ping によるレイテンシは約 3.4msec である．

表 7 に，Grid 計算システム上でヘリコプターの旋回動作制御問題を計算した場合の計算時間および計算時間中に占める Master・Worker 間の通信時間の割合を示す．表中，Lv は，PC クラスタ上での性能評価と同様，Worker 上で Master から割り当てられた 1 つのノードに適用される分枝操作の繰り返し回数を意味する．また計算時間の欄の () 内の数値は，

表 6: Grid 計算システム上の Ninf クライアント・Ninf サーバの構成

計算機	ハードウェア	設置場所
サーバ A	PentiumIII Xeon 550MHz x4CPU, 512MB memory	長津田
サーバ B	PentiumIII 733MHz x1CPU, 512MB memory	大岡山
サーバ C	PentiumIII 533MHz x1CPU, 256MB memory	大岡山
サーバ D	PentiumIII 500MHz x1CPU, 256MB memory	大岡山
クライアント	PentiumIII 400MHz x1CPU, 256MB memory	大岡山

表 7: Grid 計算システム上でのヘリコプター制御問題の計算時間

Lv	計算時間	$\frac{\text{通信時間}}{\text{計算時間}}$[%]
$Lv=1$	1784(2.06)	0.948
$Lv=2$	1371(2.68)	0.671
$Lv=3$	866(4.24)	2.24
$Lv=4$	771(4.77)	2.31
$Lv=5$	769(4.78)	1.88
$Lv=6$	912(4.03)	1.07

サーバ D 上で提案手法の逐次計算プログラムを実行した場合の計算時間に対する計算時間の比，即ち計算速度を示している．

表 7 から，Ninf を用いた Grid 計算により，計算時間が短縮されていることがわかる．最も計算時間短縮ができた場合で，Grid 計算システム上の計算時間が逐次計算に比べて 1/4.78 に短縮できている．

次に Grid 計算システム上での Worker の計算粒度に関するトレードオフについては，表7 より，$Lv=5$ の時に最も計算時間を短縮できていることがわかる．これは，Grid 計算システム上での本問題の計算では，Worker の計算粒度を増大して相対オーバーヘッドを削減するほうが，Worker の計算粒度を縮小して Worker 間の暫定値更新頻度を増大させるよりも計算時間を短縮できることを意味している．

5 むすび

本稿では，BMI 固有値問題の ϵ 最適解求解のための並列解法を提案した．また，提案手法を PC クラスタおよび Grid 計算システム上に実装し，その有効性を評価した．

性能評価の結果，PC クラスタおよび Grid 計算システムの両システム上で，提案手法により BMI 固有値問題の計算時間を短縮でき，提案手法の有効性が確認された．また，提案手法では Worker 上での計算粒度についてトレードオフが存在するが，PC クラスタ上の計算では，Worker の計算粒度を増大して相対オーバーヘッドを削減するよりも，Worker の計算粒度を縮小して Worker 間の暫定値更新頻度を増大させるほうが有効であることが確認された．しかし，PC クラスタに比べて並列計算時に発生するオーバーヘッドが大きい Grid 計算システム上では，問題によっては，オーバーヘッドによる性能低下が大きく，Worker 上での計算粒度を増大して相対的なオーバーヘッドを削減するほうが有効であることが確認された．

謝辞 本研究を進めるにあたり，Prosperous クラスタの利用環境をご提供いただいた東京工業大学学術国際情報センターの松岡聡教授，プログラム開発に御協力いただいた京都大学大学院工学研究科の藤沢克樹助手をはじめ，本研究について御議論，御助言いただいた Ninf プロジェクトの皆様に感謝いたします．

参考文献

[1] O. Toker and H. Ozabay. On the NP-Hardness of solving bilinear matrix inequalities and simultaneous stabilization with static output feedback. In *Proc. American Control Conference*, pages 2525–2526, 1995.

[2] K. C. Goh, M. G. Safonov, and G. P. Papavassilopoulos. Global optimization approach for the bmi problem. In *Proc. The 33rd IEEE Conrerence on Decision and Control*, pages 2009–2014, 1994.

[3] H. Fujioka and K. Hoshijima. Bounds for the bmi eigenvalue problem. *Trans. of the Society of Instrument and Control Engineers*, 33(7):616–621, 1996.

[4] R. Horst, P. M. Pardalos, and N. V. Thoai, editors. *Introduction to Global Optimization*. Kluwer Acadimic Publishers, 1995.

[5] MPICH. http://www-unix.mcs.anl.gov/mpi/mpich/.

[6] S. Matsuoka, H. Nakada, M. Sato, and S. Sekiguchi. Design issues of Network Enabled Server Systems for the Grid. In *Grid Computing – GRID 2000, Lecture Notes in Computer Science 1971*, pages pp.4–17. Springer-Verlag, 2000.

[7] Ninf: A Global Computing Infrastructure. http://ninf.etl.go.jp/.

[8] L. H. Keel, S. P. Bhattachayya, and J. W. Howze. Robust contorl with structured perturbations. *IEEE Trans. on Auto. Contr.*, 33(1):68–78, 1988.