数理解析研究所講究録
1499 巻 2006 年 202-207

任意精度数値計算法に基づく常微分方程式の漸近安定性解析とその並列化

静岡理工科大学　幸谷智紀・大楫弘順 (Tomonori Kouya and Kojune Ohsugi)
Shizuoka Institute of Science and Technology

1. 初めに

本稿では，非線型の多次元自励系常微分方程式が平衡点を持ち，その近傍で漸近安定性 (Asymptotical Stability) を持つかどうかを高速かつ高精度に調べる方法について議論する。

細胞内及び細胞間における包括的な蛋白質と mRNA(messanger RNA) との促進・抑制関係を記述した Segment Polarity Network(SPN) モデル [4] や，簡易化・仮想化したモデル [7] は，多次元常微分方程式系として表現されるもので，

- 一定時間 (t_c) 経過後には定常状態になる
- 一定数以下の細胞が死滅しても再生し，再度この定常状態に戻る

ことも期待される。所謂 "Robust"な常微分方程式とは，これらの性質を満足するようにパラメータを選んで構築されたものである。従って，通常は Robust な系になるよう，常微分方程式を数値計算し，パラメータを虱潰しに調べる必要がある。

しかし，常微分方程式の数値計算を行ったところで，それが漸近安定な平衡点を持つかどうかの確証を得ることはできない。近似的な数値計算を積み重ねることで，平衡点の存在とその近傍における漸近安定性を示すことが出来れば，数学的にはあやふやな Robust という概念を用いずに済むことになる。そこでは，打切り誤差と丸め誤差を任意に設定できるようなアルゴリズムとソフトウェアが求められる。

我々は，仮想化モデル [7] に対して，平衡点の導出とその近傍における漸近安定性を示し得ることを既に述べた [5] が，同時に膨大な時間が必要であることもそこでは判明している。本稿ではその原因と，PC cluster を用いた高速化について述べる。

2. 常微分方程式の漸近安定性解析

対象となる N 次元自励系常微分方程式の初期値問題を

$$\begin{cases} \frac{d\mathbf{Y}}{dt} &= \mathbf{F}(\mathbf{Y}) \\ \mathbf{Y}(t_0) &= \mathbf{Y}_0 \end{cases} \tag{1}$$

とする。ここで $\mathbf{Y}_0, \mathbf{F}(\mathbf{Y}) \in \mathbb{R}^N$ である。また，$\mathbf{F}(\mathbf{Y})$ は非線型関数である。この常微分方程式の平衡点 $\mathbf{Y}^*$ は，非線型方程式

$$\mathbf{F}(\mathbf{Y}) = 0 \tag{2}$$

の解である。この平衡点の近傍で (1) が漸近安定であるとは，$\mathbf{Y}^*$ における関数 $\mathbf{F}$ の Jacobi 行列

$$\frac{\partial \mathbf{F}}{\partial \mathbf{Y}}(\mathbf{Y}^*) = \begin{bmatrix} \frac{\partial F_1}{\partial Y_1}(\mathbf{Y}^*) & \cdots & \frac{\partial F_1}{\partial Y_N}(\mathbf{Y}^*) \\ & \cdots & \\ \frac{\partial F_N}{\partial Y_1}(\mathbf{Y}^*) & \cdots & \frac{\partial F_N}{\partial Y_N}(\mathbf{Y}^*) \end{bmatrix}$$

の全ての固有値の実部が負になること，即ち

$$\Re\left(\lambda_i\left(\frac{\partial \mathbf{F}}{\partial \mathbf{Y}}(\mathbf{Y}^*)\right)\right) < 0 \quad (i = 1, 2, ..., N) \tag{3}$$

を満足することをいう [2]。

(1) の漸近安定性のみを確認したい場合は，適切な範囲に初期値 $\mathbf{Y}_0$ が存在していれば，(1) の解 $\mathbf{Y}(t)$ について

$$\lim_{t \to \infty} \mathbf{Y}(t) = \mathbf{Y}^*$$

が期待される。即ち，非線型方程式 (2) を直接解いて得た数値解と，常微分方程式 (1) の数値解との比較を行い，両者が極めて近くなっていることを確認して $\mathbf{Y}^*$ の存在を数値的に推定できることになる。

よって，漸近安定性を数値計算によって確認 (or 推定) するためには次のステップを踏むことになる。

(a) 適当な時間の閾値 t_c を決め，(1) の数値解 $\mathbf{Y}(t_c)$ を求める。

(b) 非線型方程式 (2) を解き，数値解 $\mathbf{Y}^*$ を求める。

(c) $\|\mathbf{Y}^* - \mathbf{Y}(t_c)\| < \varepsilon\|\mathbf{Y}^*\|$ であることを確認する。

(d) $\frac{\partial \mathbf{F}}{\partial \mathbf{Y}}(\mathbf{Y}^*)$ の全ての固有値の実部が負であることを確認する。まず Gerschgorin の定理に基づくチェックを行い，それで分からなければ固有値を計算する。

これで分かるのは，1 つの平衡点 $\mathbf{Y}^*$ が存在して，これは漸近安定点である，ということだけである。平衡点が興味のある範囲内で本当に唯一か，他に平衡点がないのかどうかは分からない。よって，せめて計算結果に対しては，経験的な誤差推定法 [11] に基づいて，打切り誤差と丸め誤差を推定できるようにする必要がある。具体的には，打切り誤差については補外法を用いて，丸め誤差については多倍長浮動小数点数 [13, 14, 12] を用いて，任意精度計算で実現できる。

しかし，次元数 N が増えると計算時間が膨大なものになるので，(a)〜(d) のうち特に計算量の多い所を並列化して計算時間を短縮しなければならない。

3. 数値計算の並列化について

漸近安定性解析のための数値計算 (a)〜(d) のうち，並列化によって計算時間の短縮を図ることが出来ると期待される部分は (a), (b), (d) である。但し，今回は (d) において固有値そのものの数値計算は不要な仮想化モデルについてのみ計算を行ったため，(a), (b) 及び (d) の Jacobi 行列計算の並列化についてのみ考えることにする。ベンチマークに使用した環境は次の通りである。

使用ハードウェア

cs-pccluster2 Pentium IV 2.8GHz, Vine Linux 2.6, 11 nodes

VTPCC Dual Xeon 3.0GHz, Redhat 8.0, 8 nodes(max 16PEs)

使用ソフトウェア

- GMP 4.1.4, MPFR 2.1.1, BNCpack[12]
- MPICH2 1.0.1(cs-pccluster2), MPICH 1.2.5(VTPCC)
- gcc-3.4.3(cs-pccluster2), gcc-3.2(VTPCC)

3.1 常微分方程式の数値計算

今回対象とする常微分方程式 (1) は漸近安定性が期待されるが故に解析を行うものである。従って，ある時刻 $t_c \in \mathbb{R}$ を超えた $t > t_c$ においては，解 $\mathbf{Y}(t)$ の挙動は至極おとなしいものとなる。このような安定した問題に対しては，ステップ幅制御を用いた陽的解法を，IEEE754 倍精度で計算すれば十分な精度を得ることが出来る。今回は打切り誤差を任意に設定できるよう，補外法に基づいた GBS アルゴリズム [1] を使用する。

大次元になった場合はベクトル単位で各 PE に分割して並列化することも可能であるが，今回対象とする非線型問題では 1 ステップ進むごとに，Allgather を行って $\mathbf{Y}$ を全ての PE において集結させ，関数 $\mathbf{F}(\mathbf{Y})$ を呼び出す必要があり，ネットワークの遅い分散メモリ環境では並列化の効果はあまり期待できず，比較的小さい次元数では並列計算の効果はなく，むしろ計算時間は増大する [5, 8]。逆に多倍長計算では低い次元数でもそれなりに効果があるが，前述したようにこれ程の精度は必要ない。また，ステップ幅制御機構のあるソルバを使用したにもかかわらず，0.1 程度の初期ステップ幅を与えておけば，殆どの部分でこのステップ幅での計算が実行されることも判明している。

よって，この部分 (a) においては (b) 以降の計算を実施する前に，あらかじめ並列化せずに IEEE754 倍精度で計算しておく方が効率的である。但し，より大次元の場合にも対応でき，後述する Jacobi 行列の並列計算にも利用できるように，並列化した関数 $\mathbf{F}(\mathbf{Y})$ は用意し，少なくとも (b) 以降の計算で使用される Jacobi 行列が常微分方程式の $\mathbf{F}(\mathbf{Y})$ と同じものを用いて計算されたものである，という証明を行えるようにすると共に，より高速な Jacobi 行列の評価が可能なようにしておく。

3.2 Jacobi 行列の数値計算

Jacobi 行列の計算は最も精度が要求される部分であるため，精度を任意に設定できる計算法が必要である。従って，ここでも各要素の偏微分 $\partial F_i/\partial Y_j$ の計算には中点公式 (3 点公式) を初期系列とする補外法 [9, 10] を使用し，全て多倍長計算で行うことにする。

初期系列の計算は，まず $\mathbf{Y}_c = [Y_1^{(c)} \cdots Y_N^{(c)}]^T$ の第 i 成分を基準に適切な δ_i を Romberg 数列に乗じて

$$\mathbf{Y}_c^{i\pm} = \left[\cdots\ Y_{i-1}^{(c)}\ Y_i^{(c)} \pm 2^{-p}\delta_i\ Y_{i+1}^{(c)}\ \cdots\right]^T$$

とし，中心差分を用いた近似式

$$\mathbf{J}_i^{p,1} = (2^p\delta_i)^{-1}\left(2^{-1}\mathbf{F}(\mathbf{Y}_c^{i+}) - 2^{-1}\mathbf{F}(\mathbf{Y}_c^{i-})\right)$$

で計算する。今回は任意の i に対して $\delta_i = 1$ と設定した。

補外計算では以下の表の初期系列より右の下三角成分を求める。

初期系列				
$\mathbf{J}_i^{1,1}$				
$\mathbf{J}_i^{2,1}$	$\mathbf{J}_i^{2,2}$			
$\vdots$	$\vdots$	$\ddots$		
$\mathbf{J}_i^{m-1,1}$	$\mathbf{J}_i^{m-1,2}$	$\cdots$	$\mathbf{J}_i^{m-1,m-1}$	
$\mathbf{J}_i^{m,1}$	$\mathbf{J}_i^{m,2}$	$\cdots$	$\mathbf{J}_i^{m,m-1}$	$\mathbf{J}_i^{m,m}$

ここで各段の計算は，

$$\mathbf{J}_i^{p,q} := \mathbf{J}_i^{p,q-1} + (4^{p-1}-1)^{-1}\left(\mathbf{J}_i^{p,q-1} - \mathbf{J}_i^{p-1,q-1}\right)$$

として行う。

この際，行列の列ごとに補外計算を行うため，収束判定については $\mathbf{J}_i^{p,q} = [\cdots J_{ij}^{p,q} \cdots]$ の各要素 $J_{ij}^{p,q}$ ごとに行う必要がある。この収束判定には，福井・永坂が提案した丸め誤差限界の評価値 $\varepsilon_{FN,j}$ と，あらかじめユーザから与えられた相対許容度 ε_R と絶対許容度 ε_A を用いて行う。

福井・永坂 [9] は，補外計算を用いた数値微分における初期系列の丸め誤差限界評価値を，関数の評価回数，関数評価点，数値微分公式の係数，浮動小数点数の基数と計算桁数のみで設定した。福井 [10] は補外計算によっても丸め誤差の増大率は 2 倍以内に収まることを証明している。よって我々は補外計算における丸め誤差限界評価値 $\varepsilon_{FN,j}$ を

$$\varepsilon_{FN,j} = \frac{\max\left(\left|F_j(\mathbf{Y}_c^{i+})\right|, \left|F_j(\mathbf{Y}_c^{i-})\right|\right)\varepsilon_M}{2^p\delta_i}$$

とした。ここで ε_M はマシンイプシロンである。

これらの $\varepsilon_{FN,j}, \varepsilon_R, \varepsilon_A$ を用いて，

$$\left|J_{ij}^{p,q} - J_{ij}^{p,q-1}\right| \leq \max\left(\varepsilon_{FN,j}, \varepsilon_R \left|J_{ij}^{p,q-1}\right| + \varepsilon_A\right)$$

を満足していれば，その k 番目の成分は収束したと判断する。もしこの基準を満足したときには，その j 番目の成分の計算を飛ばして列単位の補外計算を続ける。従って，補外計算が停止するのは全ての列成分が収束した時となる。

この収束判定により，あくまで列単位の計算を行いつつ，成分ごとのきめ細やかな収束判定を行うことができるため，最大 m 段までの補外を実施したとしても，全ての要素を計算するためには最大 $2mN$ 回の関数 **F(Y)** 呼び出しで済む。もしこれを成分単位で行おうとすると，関数評価回数は $2mN^2$ に激増する。これによって，全体の計算時間の大幅な縮減が期待できる。実際，VTPCC と cs-pccluster2 において，1PE のみを用いて仮想化モデル ($N = 200$) の Jacobi 行列を 10 進 50 桁で計算したところ，表 1 のように，約 100 倍の効果があることが判明している。

表 1: Jacobi 行列の計算時間 (単位:sec)

	VTPCC	cs-pccluster2
Elementwise eval.	2270.9	2361.4
Columnwise eval.	24.2	28.7

さらに今回は，並列分散化した **F(Y)** を用い，図 1 のように Jacobi 行列の計算を行単位に分割し，各 PE で並列計算させることで，より大次元の問題に適用できるようにした。

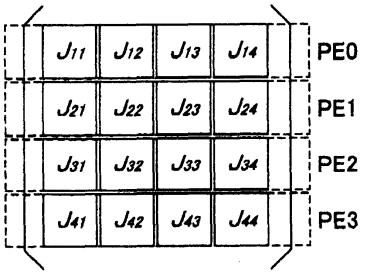

図 1: Jacobi 行列計算の並列化

但し，実際の並列計算では関数呼び出しの際には必ず集団通信が必要となるため，比較的小さい次元数や桁数の計算では計算時間のボトルネックが生じやすい。これについては後述する。

3.3 非線型方程式の数値計算

非線型方程式 (2) の解 $\mathbf{Y}^*$ を求める計算は，Newton 法及びその変種の反復アルゴリズムを用いることで実現できる。平衡点のみを高速に計算するのであれば

- 反復法の初期値に $\mathbf{Y}(t_c)$ を採用する
- なるべく Jacobi 行列を用いずに済む変種法 (準 Newton 法や Regula-Falsi 法) を使用する
- 反復計算の中に現れる連立一次方程式を Krylov 部分解法のように，並列化が容易なアルゴリズムで計算する [3]

という高速化手法を取るべきである。しかし，「平衡点探索ツール」として本計算を考えると，これとは逆に

- 反復法の初期値に，(1) の初期値 $\mathbf{Y}_0$ を採用し，$\mathbf{Y}(t_c)$ とはかけ離れた平衡点に収束しないことを確認する
- あえて馬鹿正直に Jacobi 行列を用いた Newton 法を用い，$\mathbf{Y}(t_c)$ に近い平衡点に収束することをもって，Jacobi 行列の正しさを確認する

というやり方もある。今回はこの後者の考え方を用い，高速化は並列化した Krylov 部分解法 (GPBiCG 法 [6] を採用) のみで行うことにした。従って，(2) を解くための反復計算回数そのものは少なくなっているものの，Jacobi 行列の評価回数は増えているため，全体の計算時間はかなり増大している。

4. 仮想化した細胞間再生モデル

今回は，仮想化した細胞間再生モデル [7] についてのみ，安定性解析を行った。ここではこのモデルの考え方を数値例も含めて説明する。

円環状に連なった n 個の細胞にそれぞれにおいて蛋白質グループ $\mathbf{x}_i(t), \mathbf{y}_i(t)$ が存在しているものとする。これらの蛋白質グループの各要素は，他の要素によって生成を促進されたり抑制されたりする。もしある蛋白質 $z(t)$ が，別の複数の蛋白質の働きによって生成が促進される場合，これらを線型結合した $u(t)$ によって

$$\frac{d}{dt}z(t) = \frac{(u(t))^q}{b_u + (u(t))^q} - a_z z(t) \tag{4}$$

と表現される。逆に，生成が抑制される場合は，抑制効果のある要素の線型結合 $v(t)$ を用いて

$$\frac{d}{dt}z(t) = \frac{1}{b_v + (v(t))^p} - a_z z(t) \tag{5}$$

と表現される。ここで，$a_z z$ は自壊する量，各パラメータ $a_z, b_u, b_v \in \mathbb{R}, p, q \in \mathbf{N}$ は正定数である。

このような促進・抑制効果をまとめて記述すると (6) 式のようになる。(1) の次元数でいえば，$N = 2n^2$ となる。

$$\frac{d}{dt}\begin{bmatrix}\vdots\\ \mathbf{x}_i(t)\\ \\ \mathbf{y}_i(t)\\ \vdots\end{bmatrix} = \begin{bmatrix}\vdots\\ \dfrac{\left(h_j^{(i)}(t)\right)^{p_x}}{b_x+\left(h_j^{(i)}(t)\right)^{p_x}} - a_x x_j^{(i)}(t)\\ \vdots\\ \dfrac{\left(y_j^{(i)}(t)\right)^{p_y}+c_f\left(f_j^{(i)}(t)\right)^{p_f}}{b_y+\left(y_j^{(i)}(t)\right)^{p_y}+c_f\left(f_j^{(i)}(t)\right)^{p_f}+c_g\left(g_j^{(i)}(t)\right)^{p_g}+c_h\left(h_j^{(i)}(t)\right)^{p_h}} - a_y y_j^{(i)}(t)\\ \vdots\end{bmatrix} \quad (6)$$

(正定数：$a_x, a_y, b_x, b_y, c_f, c_g, c_h \in \mathbb{R}$, $p_x, p_y, p_f, p_g, p_h \in \mathbb{N}$)

ここで，$f_j^{(i)}(t), g_j^{(i)}(t), h_j^{(i)}(t)$ は各蛋白質成分の線型結合で表現される関数である。今回は図 2($n = 5$ の場合) に示されるような促進・抑制関係を使用するので，これらの関数は

$$\begin{aligned} f_j^{(i)}(t) &= x_{j-2}^{(i-1)}(t) + x_{j+2}^{(i-1)}(t) + x_{j-2}^{(i+1)}(t) + x_{j+2}^{(i+1)}(t) \\ g_j^{(i)}(t) &= \sum_{k=1,k\neq j}^{n} y_k^{(i)}(t) \\ h_j^{(i)}(t) &= y_j^{(i-1)}(t) + y_j^{(i+1)}(t) \end{aligned}$$

となる。但し，1 番目と n 番目の細胞が繋がって円環状になっているため，この添字は

$$\begin{aligned} i-1<1 \quad &\text{の時は} \quad (i-1)+n \\ i+1>n \quad &\text{の時は} \quad (i+1)-n \\ j-2<1 \quad &\text{の時は} \quad (j-2)+n \\ j+2>n \quad &\text{の時は} \quad (j+2)-n \end{aligned}$$

となる。

このモデルに必要とされる性質は次のようになる。

1. 十分時間が経過した時点 ($t > t_c$) において

$$\frac{y_i^{(i)}(t)}{y_j^{(i)}(t)} \geq 10 \; (j \neq i)$$

となる。

2. $t_d > t_c$ において，一定数以下の細胞が死滅 ($\mathbf{x}_i(t_d) = 0, \mathbf{y}_i(t_d) = 0$) しても再生する (上記の条件を満足する)。

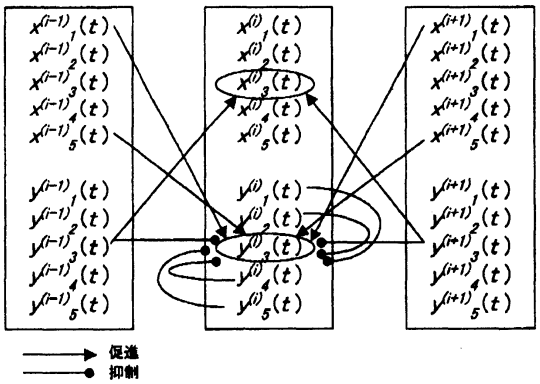

図 2: $n = 5$ の場合の促進・抑制関係図

特に 2 の性質は重要であり，これが「細胞間再生モデル」という名前の由来となっている。一部の細胞が死滅すると，その部分はどの細胞にもなりうるオールマイティな幹細胞に入れ替わる。一定時間経つとこの幹細胞が元の細胞の性質を持つようになる。数学的には i 番目の細胞ならば，蛋白質 $y_i^{(i)}(t)$ がその他の $\mathbf{y}_i(t)$ の成分に比べて 10 倍以上優越するようになることを意味する。

今回もパラメータとしては大槻ら [7] が使用した

$$\begin{aligned} b_x &= 1000, a_x = 10, p_x = 3 \\ b_y &= 10^{-9}, a_y = 10^{-1}, p_y = 1 \\ c_f &= 10^{-1}, p_f = 3, c_g = 1, p_g = 3 \\ c_h &= 10^{-9}, p_h = 2 \end{aligned}$$

を用いる。安定化の後は，細胞の死滅があっても再

生してこの値に戻ることを考えると，これが平衡点で，その周囲ではかなり強い漸近安定性を持つと考えられる。

5. 仮想化した細胞間再生モデルを用いた数値実験

$N = 50, 200 (n = 5, 10)$ 次元までの，図2のような促進・抑制関係を持つ仮想化モデルの漸近安定性は既に確認済みである [5] が，今回並列化を行うことによって，$N = 800, 1800 (n = 20, 30)$ 次元の漸近安定性を確認することが出来た。なお，(b)〜(d) までの全ての計算は 10 進 50 桁相当 (2 進 167bits) の多倍長計算で行っている。

5.1 漸近安定性の確認

前述した仮想化モデルの促進・抑制関係を用いた場合は，今回調査した 200, 800, 1800 次元全てにおいて，平衡点における Jacobi 行列は対角優位非対称行列となり，対角成分は −10 もしくは −0.0990 のどちらである。例として $N = 8 (n = 2)$ の時の Jacobi 行列と平衡点を図3に示す。

図3の平衡点は Newton 法で4回反復した後に得られたものであるが，他の次元数においては6回の反復で収束することが確認できた。また，これらの平衡点のユークリッドノルム値 $\|\mathbf{Y}^*\|_2$ と，$t = 100$ における常微分方程式の数値解のノルム値 $\|\mathbf{Y}(100)\|_2$ はかなり近いものであることが確認できた (表2)。

表 2: $\mathbf{Y}^*$ と $\mathbf{Y}(100)$(上)，Gerschgorin disc(中, 下)

N	$\|\mathbf{Y}^*\|_2$	$\|\mathbf{Y}(100)\|_2$
200	3.16236e + 1	3.16242e + 1
800	4.47224e + 1	4.47230e + 1
1800	5.47736e + 1	5.47732e + 1

N	max radius for −10
200	$\|\lambda - (-1.00e + 1)\| \leq 1.50e - 1$
800	$\|\lambda - (-1.00e + 1)\| \leq 1.50e - 1$
1800	$\|\lambda - (-1.00e + 1)\| \leq 1.50e - 1$

N	max radius for −0.0990
200	$\|\lambda - (-9.90e - 2)\| \leq 1.23e - 5$
800	$\|\lambda - (-9.90e - 2)\| \leq 1.26e - 5$
1800	$\|\lambda - (-9.90e - 2)\| \leq 1.29e - 5$

更に，各対角成分ごとに Gerschgorin disc の半径を計算してみると，最大でも 0.15 もしくは 1.23×10^{-5} 〜1.29×10^{-5} であり，固有値を計算するまでもなく，その実数部は負になることが判明した。

5.2 並列化による計算時間の縮減効果

1(b)〜(d) の総計算時間は，表3のようになった。×印は実行が不可能であったことを示す。

以下，計算時間の内訳を，$N = 800 (n = 20)$ の場合に限って，詳細に見ることにする。まず，Jacobi 行列の並列化の効果を4に示す。

表 3: (b)〜(d) 全体の最短計算時間 (単位:秒)

N	VTPCC	cs-pccluster2
200	78.0 (5PEs)	187.7(2PEs)
800	672.6(20PEs)	2002.4(10PEs)
1800	2832.0(10PEs)	×

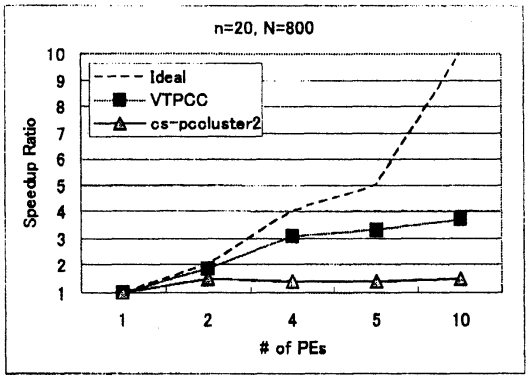

図 4: Jacobi 行列の並列化の効率

理想的には PE 数に比例して性能向上があるところが，最も効率の良い VTPCC でも 4PE では 3 倍程度に留まっており，最も性能向上が見られる 20PE でも 4 倍程度までしか上がらないことが実験から判明している。これは各 PE で分散して $\mathbf{F}(\mathbf{Y})$ の評価をしている所で必ず一度集団通信 Allgather を実行していることが響いていると思われる。逆に言えば，現状の実装ではこの集団通信の実行回数を減らす必要があると言える。

6. 結論と今後の課題

漸近安定性解析を行うために必要な数値計算の主要部分，特に Jacobi 行列の並列計算によって大幅な計算時間の縮減を達成することが出来，1800 次元の非線型常微分方程式の漸近安定性を確認することが可能となった。

しかし，現時点の並列化はまだ改善の余地がある。特に

- Jacobi 行列の sparsity を考慮していない
- 並列化した Jacobi 行列計算の集団通信回数が多い

という部分が残っており，ここを改善することで (b)〜(d) 全体の計算時間を減らすことが可能となる。今後の課題としては，さらに Jacobi 行列の計算効率化を図ることで，より大規模な常微分方程式の漸近安定性を示すことが挙げられる。

$$\frac{\partial \mathbf{F}}{\partial \mathbf{Y}}(\mathbf{Y}^*) = \begin{bmatrix} -1.0e+1 & 0 & 0 & 0 & 0 & 0 & -4.1e-47 & 0 \\ 0 & -1.0e+1 & 0 & 0 & 0 & 0 & 0 & 3.0e-2 \\ 0 & 0 & -1.0e-1 & 5.2e-38 & 1.5e-12 & 0 & 0 & 0 \\ 0 & 0 & -6.4e-39 & -9.9e-2 & 0 & -2.9e-49 & 0 & -1.7e-48 \\ 0 & 0 & 3.0e-2 & 0 & -1.0e+1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4.1e-47 & 0 & -1.0e+1 & 0 & 0 \\ -2.9e-49 & 0 & -1.7e-48 & 0 & 0 & 0 & -9.9e-2 & -6.4e-39 \\ 0 & 1.5e-12 & 0 & 0 & 0 & 0 & 5.2e-38 & -1.0e-1 \end{bmatrix}$$

ここで $\mathbf{Y}^* = \begin{bmatrix} -4.852948038257050094482194114077835149 07e-48 \\ 8.888888888592725715655173385939022759 69e-2 \\ 9.999999990004492905027271143747014181 3 \\ 2.117204939929691746348329677595936526331e-35 \\ 8.888888885927257156551733859390227596 9e-2 \\ -4.852948038257050094482194114077835149 07e-48 \\ 2.117204939929691746348329677595936526331e-35 \\ 9.999999990004492905027271143747014181 3 \end{bmatrix}$

図 3: $N = 8(n = 2)$ の Jacobi 行列と平衡点 $\mathbf{Y}^*$

謝辞

cs-pccluster2 は静岡理工科大学学内研究費の補助によって構築された。VTPCC は名古屋大学情報科学研究科三井斌友研究室所有のものを使用させて頂いた。関係者各位に感謝する。

参考文献

[1] E.Hairer, S.P.Nørsett, G.Wanner, Solving Ordinary Differential Equations I (2nd ed.), Springer-Verlag, 1993.

[2] M.W.Hirsch, S.Smale, 力学系入門, 岩波書店, 1976.

[3] C.T.Kelley, Solving Nonlinear Equations with Newton's Method, SIAM, 2003.

[4] G. von Dassow et al., "The segment polarity network is robust developmental module", Nature Vol.406, pp.188-192, 2000.

[5] 幸谷智紀・大槻弘順, 仮想化した細胞間モデルの安定性解析, HOKKE2005, 2005.

[6] 幸谷智紀, Windows を用いた PC cluster 上における並列多倍長数値計算ライブラリ MPIBNC-pack の性能評価, HPCS2005 ポスターセッション, 2005.

[7] 大槻弘順・他, 単純化した相互作用ルールによる擬似細胞のパターン形成と再生, 静岡理工科大学紀要 Vol.11, 2003.

[8] 幸谷智紀・大槻弘順, Segment Polarity Network Model に基づく細胞間再生モデル, 研究集会「常微分方程式とその周辺」, 2005.

[9] 永坂秀子・福井義成, "数値微分の誤差", 情報処理学会論文誌, vol.22, No.5, pp.411-416.

[10] 福井義成, "数値微分における補外法の誤差", 日本応用数理学会論文誌 Vol.15, No.4, pp.521-535, 2005.

[11] 幸谷智紀, 数値計算における誤差について–数値微分を例に–, `http://na-inet.jp/na/na_error_diff.pdf`

[12] BNCpack, `http://na-inet.jp/na/bnc/`

[13] GMP, `http://swox.com/gmp/`

[14] MPFR, `http://www.mpfr.org/`