日本オペレーションズ・リサーチ学会
2021年 春季研究発表会

OpenMP による自動微分ソフトウェアの高速化

01308414　関西大学　　*檀 寛成　DAN Hiroshige
非会員　関西大学大学院　野口将嗣　NOGUCHI Masashi

1. はじめに

我々のグループでは,（任意精度計算可能な）非線形最適化ソフトウェア “NOA” (Nonlinear Optimizer with Arbitrary precision arithmetic) を開発している [3, 5]．本研究では，NOA に含まれる自動微分ソフトウェア noa_ad の高速化を試みる．

非線形最適化問題を解く際には，目的関数や制約条件に現れる関数の値や，これらの勾配（やヘッセ行列）を計算する必要がある．これらの計算には自動微分を用いるのが一般的であり，我々も自動微分ソフトウェア noa_ad を実装した．

noa_ad では，これまで，単一のスレッドで微分値等の計算を行っていた．しかし，異なる関数の微分値計算を複数のスレッドで並列に実行すれば，計算時間を短縮することが可能なはずである．昨今の CPU はマルチコア化しており，かつ並列計算プログラミングを可能とするフレームワークも多く提案されているので，並列計算を簡易に実現できる環境が整っている．

このような状況を踏まえ，本研究では，スレッド並列化手法として最も普及している方法の一つである OpenMP [4] を用い，noa_ad を高速化する．

2. noa_ad の実装の概要

一般に，大規模な（非線形）最適化問題においては，変数や制約条件が添字付けられていることが多い．noa_ad では，このような状況をうまく扱うことができるような実装を行っている [3]．

自動微分により勾配等を計算するためには，関数値を計算するための計算グラフ（と等価なデータ構造）が必要になる．noa_ad では，一旦，添字を抽象化したまま対象となる式の計算グラフを構成し（図 1），そののちに添字を展開する（図 2）という実装を行っている．

3. OpenMP を用いた noa_ad の並列化

本研究では，従来の noa_ad に OpenMP を用いるような修正を加え，勾配の計算を並列に実行できるようにした．具体的には，図 2 のように添字が展開された個々の計算グラフ（個々の関数に対応）に対して自動微分計算を行うようにした（下記イメージ参照）．

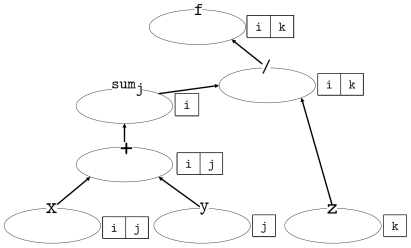

図 1: 添字を抽象化したままの計算グラフ
($f_{ik} = \sum_{j \in J}(x_{ij} + y_j)/z_k \ (i \in I, k \in K)$)

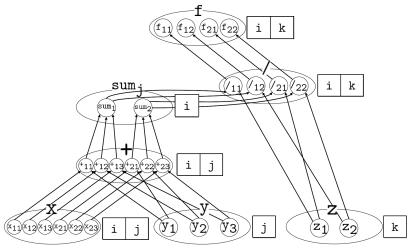

図 2: 添字を展開した計算グラフ
($I := \{1,2\}, J := \{1,2,3\}, K := \{1,2\}$)

OpenMP による並列化（イメージ）

```
#pragma omp parallel for  // OpenMP の命令
  for(int i = 0; i < size; ++i){
    関数 f(i) の勾配の計算; // 並列に実行される
  }
```

ただし，各スレッドのメモリアクセスは（特に指示をしなければ）排他的に行われるわけではないため，異なるスレッドがメモリ上の共通領域に値を書き込むことがあると，計算は正しく行われない．従来の noa_ad の実装では，各関数の勾配を計算する際にいくつかの箇所で共通領域への書き込みが発生していたため（単一スレッドでは正しく動作する），それを回避する修正を行った[1]．

[1]現在の実装は試行的な部分があり，特定の条件を満たす関数では微分値を正しく計算できないことがわかっている．今後修正の予定である．

4. 数値実験

ここでは，The COCONUT Benchmark [1] に含まれる非線形最適化問題のうちの 29 題に対し，目的関数と制約条件を構成する関数の勾配を 100 回計算する時間を，並列計算に用いるスレッド数を変化させながら測定した．

表 1: 計算環境

OS	Ubuntu 18.04 LTS
CPU	Intel Core i9-9900K（16 スレッド）
Memory	64GB
Compiler	Intel C++ コンパイラ 19.0.5

表 2: 計算結果

Name	#func	#var	(♢)	(♠1)	(♠8)
aug2d	9997	20400	2.93e-4	4.45	2.61
aug2dc	10195	20400	2.90e-4	4.46	2.61
aug2dcqp	30595	20400	1.29e-4	6.01	3.12
aug2dqp	30595	20400	1.29e-4	5.92	3.08
broydn3d	10001	10000	3.00e-4	3.31	0.91
curly10	10001	20000	6.50e-4	4.21	2.12
cvxbqp1	20001	10000	1.50e-4	2.50	1.86
dtoc3	9999	14999	3.33e-4	3.57	2.13
dtoc4	9999	14999	3.33e-4	4.64	2.31
dtoc6	5001	10001	5.00e-4	1.84	1.32
hager1	5002	10001	4.00e-4	1.39	0.61
hager2	5001	10001	5.00e-4	2.55	1.86
hager3	5001	10001	5.00e-4	3.16	2.51
hues-mod	10003	10000	4.00e-4	5.42	5.16
liarwhd	1	10000	1.00	1.91	1.92
mancino	10101	10200	2.94e-4	4.86	1.63
ncvxbqp1	20001	10000	1.50e-4	2.45	1.81
nondia	1	10000	1.00	1.21	1.21
orthregc	5001	10005	8.99e-4	3.83	1.52
orthregd	5001	10003	7.00e-4	4.12	1.92
quartc	1	10000	1.00	0.60	0.61
reading1	25005	10002	2.00e-4	8.12	4.28
reading2	30007	15003	1.56e-4	6.13	3.62
sinquad	1	10000	1.00	1.96	2.26
sipow1	10001	2	1.00	4.01	1.21
sipow1m	10001	2	1.00	4.69	1.41
sipow2	5002	2	1.00	1.80	0.50
sosqp1	50002	20000	1.00e-4	4.61	2.32
sosqp2	50002	20000	1.00e-4	4.85	2.34

(♢): ヤコビ行列の非零要素の密度，
(♠1) / (♠8): 1 スレッド / 8 スレッドでの計算時間（秒）

表 2 に，実験対象とした各問題の規模，ならびに 1 スレッド/ 8 スレッド使用時の計算時間を示した．8 スレッドを利用した場合，1 スレッドの場合に比べ，平均すると計算時間を 39.4% 削減することができた．一方で，8 スレッドにすると計算時間が長くなるケースも認められた．これは，関数の数が極端に少なく，並列化処理のオーバーヘッドの影響が大きいケースであると考えられる[2]．また，計算時間の削減率が小さい例もあった (`hues-mod`)．今後理由を検討したい．

[2]条件分岐により，このようなケースでは並列計算を回避することが可能である．

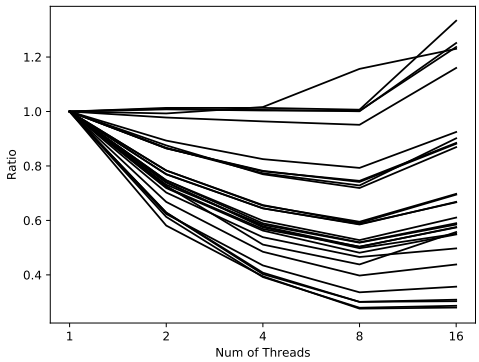

図 3: 計算時間の変化

図 3 には，スレッド数が 1 のときの計算時間を 1 とし，スレッド数を 2, 4, 8, 16 と変化させたときの計算時間の変化を示した．図 3 からは，ほとんどの問題で，8 スレッドのときよりも 16 スレッドの場合の方が計算時間が長くなることがわかる．これは，演算性能に比べ，メインメモリとキャッシュメモリ間のデータ転送能力が低いことが原因であると考えられる．[2] では「現在の経験的な性能として，OpenMP での並列化は，8 スレッド並列以下の実行に向いている」と指摘されているが，本結果もこの指摘に沿うものとなっている．

5. おわりに

本研究では，並列計算環境 OpenMP を用いて，我々のグループで従来より開発している自動微分ソフトウエア noa_ad での微分値計算を並列化した．また，数値実験により微分値計算を一定程度高速化できていることが確認できた．

なお，本研究は JSPS 科研費 18K11185 の助成を受けたものです．

参考文献

[1] The COCONUT Benchmark, `https://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Benchmark.html`（2021 年 1 月 7 日確認）．
[2] 片桐 孝洋，並列プログラミング入門 サンプルプログラムで学ぶ OpenMP と OpenACC，東京大学出版会，2015.
[3] 野口 将嗣，檀 寛成，添字を扱うことができる自動微分ソフトウェアの設計と実装，日本オペレーションズ・リサーチ学会 2018 年春季研究発表会 アブストラクト集，172-173.
[4] OpenMP, `https://www.openmp.org`（2021 年 1 月 7 日確認）．
[5] 崎山 皓瑛，檀 寛成，非線形最適化問題に対する任意精度計算可能な内点法の性能評価，日本オペレーションズ・リサーチ学会 2019 年秋季研究発表会 アブストラクト集，54-55.