寄　　書

ゼロの多い Array のメモリ縮約の一方法*

石　黒　美佐子**

1. まえがき

技術計算で，しばしば重要な問題となるゼロの値を多く含む array (Sparse Array) に対して，ゼロでない値だけを保存して，メモリを節約する方法の一つを，ここで紹介する．

2. 方　法

メモリには，array の non-zero value を保存し，三つのコントロール語と，array の non-zero value と array のインデックスを対応づける表を保存する．

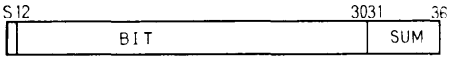
コントロール語には，次のものを入れる．

（1） メモリ上に実際にとられる array の大きさ．

（2） 使用上の array のインデックスの最大値．

（3） 現在保存されている non-zero value の数．

array と array インデックスの対応表は，文献 2) で述べられたことを，1 レベルインデックスに対して，応用したものである．すなわち，36 ビットの計算機を例にとれば，1 語を次のようなビット構成で使用し，



BIT 部の 1 ビットには，使用上の array の 1 エレメントが対応する．つまり，使用上の array インデックスの最大値を，30 で割った分だけの語数が必要である．0-ビットには zero エレメントが対応し，1-ビットには non-zero エレメントが対応する．SUM 部には，1-ビットがその語にいくつあるかを保存する．これにより使用上の array と実際の値との対応を，非常に効果的に行なうことができる．

array の non-zero value は，インデックスの小さい順に保存する．

三つのコントロール語と array と array インデックスの対応表，および non-zero value の保存状態は，計算進行中に array が参照されるたびに変えられる．

3. IBM 7044 による実験と評価

この試みは，もともとゼロの多い行列演算の array を，縮小することを意図して作成されたが，その有効性をメモリの点と，処理時間および使い方の点で次に述べたい．

メモリの点では，ルーチンが 456 語で作成されているので，**2.** で述べたメモリの使用法を考えあわせると，次の場合には，メモリを節約することができる．

$$MAX \geqq 3+[MAX/30]+COUNT+456$$

ここで MAX とは使用上の array のインデックスの最大値，COUNT は array の non-zero value の数である．たとえば，プログラムで仮想的に使用する array の大きさが 5,000 語で，そのうち，non-zero value の総数が 1,000 以下であるならば，必要とするメモリは

$$3+[5,000/30]+1,000+456=1,626$$

たかだか 1,626 語であり，3,374 語が節約される．

処理時間については，array のメモリ保存状態などが，計算実行中に必要に応じて変化するのでかなりかかるが，array の大部分の値が 0 であることを考えると，かなり少くてすむ．大ざっぱなところ，著者の作成したルーチンでは，zero-value の保存に対し，アセンブリ言語のステップで 10，non-zero value だと約 200 であり，取出しに対しては 50 ステップ要した．

このルーチンは，2 次元以上の array に対して有効性の増すが，1 次元に変換して使用しているので，この点でも手間がかかる．

参考文献

1) W. J Worlton: "The Administration of Secoundary Storage in Nuclear Codes", LA-3546-MS.

2) M. C. Wunderlich: "Sieving Procedures on a Digital Computer", J. ACM, 14 (Jan. 1967).

* Strage Compression for the Sparsely Populated Array, by Misako Ishiguro (Japan Atomic Energy Research Institute, Computing Center)

** 日本原子力研究所