精度保証付き数値計算レポート (演習5-2)

600p009-0 岡部 貴弘

2001年 2月 5日（月）

1 課題

Scilab の ATLAS(or 最適化)BLAS による高速化を行なえ．

2 解答

2.1 はじめに

谷口君が行列の演算において高速化に成功したのを受け，又 LU 分解関数が高速化されれば連立一次方程式も高速化されるであろうと思ったので，自分は LU 分解と QR 分解の高速化を試みた。以下，既に ATLAS によって最適化された Scilab である事を前提に話を進めていく．

2.2 環境

実行環境は以下の通りである．

OS	Vine Linux 2.0
CPU	Mobile Intel Celeron 500Mz
数値計算ソフト	Scilab2.5

2.3 なぜ高速化されないのか

高速化されなかった原因として，今までの Scilab における LU 分解は

```
scilab-2.5/routines/interf/matlu.f
```

に組み込まれている

```
scilab-2.5/routines/control/dgefa.f
```

という fortran のプログラムで定義されたもので計算されていた．このサブルーチンは linpack を用いているため，高速化されなかった．

そこで様々なファイルを調べた結果，

```
scilab-2.5/routines/lapack/dgetrf.f
```

というファイルが LAPACK を用いた LU 分解をすること見つけた．つまり，Scilab における LU 分解は，いくら ATLAS で高速化しようとしても実際は linpack によるサブルーチンによって計算されるため，高速化しないのは当然である．そこで LU 分解する際に control/dgefa.f が呼び出されるわけだが，このファイルが呼び出されると自動的に lapack/dgetrf.f が呼び出されるようにすればよい，と考えた．又，LU 分解が高速化されれば，自動的に連立一次方程式も高速化されることになる．

2.4 実装

以上のような原因をふまえて，柏木研 M1 の吉田君と共に実際にプログラムを組み変えてみた．
まず，scilab-2.5/routines/control/dgefa.f の中身を見てみると

dgefa.f

```
    subroutine dgefa(a,lda,n,ipvt,info)
    integer lda,n,ipvt(*),info
    double precision a(lda,*)
c!purpose
c
c     dgefa factors a double precision matrix by gaussian elimination.

          ～～～～～～～～～～～～～ 中略 ～～～～～～～～～～～～～
  70 continue
     ipvt(n) = n
     if (a(n,n) .eq. 0.0d+0) info = n
     return
     end
```

となっている．ここで，このファイルが呼び出された時に自動的に scilab-2.5/ routines/lapack/dgetrf.f が呼び出されるようにするためには以下のように dgefa.f を書き換えた．

dgefa.f(改良版)

```
    subroutine dgefa(a,lda,n,ipvt,info)
    integer lda,n,ipvt(*),info
    double precision a(lda,*)
    call dgetrf(n,n,a,lda,iput,info)
    return
    end
```

とした．つまり，dgefa.f の中に dgetrf.f を call という命令で自動的に呼び出されるように組み込んだ．ここで重要なのは dgefa. と dgetrf.f との引数同士を合わせることである．このファイルを保存した後，./configure の地点から再度 Makefile し直した．
その後， A を 1000×1000 の乱数行列として，高速化前と高速化後の実行速度を比べて見た所，高速化したことにはした．

しかし，ここで問題が発生した．
例えば適当な 2×2 行列や 5×5 行列の LU 分解の解を，ATLAS 組み込み前と ATLAS 組み込み後とで比べた場合，実行速度自体は早くなるのだが，肝心の解が正しくないという結果がでてしまった．
そこで，dgefa.f 内に命令を付け足しして以下のように組み変えた．

dgefa.f(改良版の改良版)

```
subroutine dgefa(a,lda,n,ipvt,info)
integer lda,n,ipvt(*),info
double precision a(lda,*)

integer kb, k, i

call dgetrf(n,n,a,lda,ipvt,info)

do 12 kb = 1, n
c do 12 kb = n, 1, -1
k = n+1-kb
 if(k.ne.n) then
call dscal(n-k,-1.0d+0,a(k+1,k),1)
 endif
i = ipvt(k)
 if (i.ne.k .and. k.ne.0) then
call dswap(k-1,a(i,1),lda,a(k,1),lda)
 endif
  12  continue

if (info.lt.0) then
  info = -info
endif

return
end
```

ここで重要なのは，右端に揃えるのではなく少し間隔を開けてプログラムを書くことである (これに気付かなかったためになかなかコンパイルできず，かなり苦労した)．

2.5 実行結果

A を 1000×1000 の乱数行列として，LU 分解における Scilab の高速化前と高速化後の実行速度を比べて見た．

ATLAS 組み込み前	ATLAS 組み込み後
17.73 秒 (sec)	5.03(sec)

ATLAS 組み込み前と ATLAS 組み込み後の 5×5 行列の解の比較は以下の通りである．ここでは

$$p * a \ - \ l * u \ = 0 \tag{1}$$

となればよいわけで，実際にそのようになったのでこのプログラムは正しいことが分かった．

dgefa.f(ATLAS組み込み前)

```
-->a=[1 2 3 4 5 ; 2 3 4 5 6; 3 4 5 6 7;4 5 6 7 8;5 6 7 8 9]
 a  =

!   1.    2.    3.    4.    5. !
!   2.    3.    4.    5.    6. !
!   3.    4.    5.    6.    7. !
!   4.    5.    6.    7.    8. !
!   5.    6.    7.    8.    9. !

-->[l u p]=lu(a)
 p  =

!   0.    0.    0.    0.    1. !
!   1.    0.    0.    0.    0. !
!   0.    0.    1.    0.    0. !
!   0.    1.    0.    0.    0. !
!   0.    0.    0.    1.    0. !
 u  =

!   5.    6.     7.     8.     9.  !
!   0.    0.8    1.6    2.4    3.2 !
!   0.    0.     0.     0.     0.  !
!   0.    0.     0.     0.     0.  !
!   0.    0.     0.     0.     0.  !
 l  =

!   1.     0.      0.           0.    0. !
!   0.2    1.      0.           0.    0. !
!   0.6    0.5     1.           0.    0. !
!   0.4    0.75    0.5869865    1.    0. !
!   0.8    0.25    0.5          0.    1. !

-->p*a-l*u
 ans  =

!   0.           0.    0.    0.    0. !
!   0.           0.    0.    0.    0. !
! - 4.441E-16    0.    0.    0.    0. !
!   0.           0.    0.    0.    0. !
!   0.           0.    0.    0.    0. !
```

dgefa.f(ATLAS組み込み後)

```
-->[l u p]=lu(a)
 p  =

!   0.    0.    0.    0.    1. !
!   1.    0.    0.    0.    0. !
!   0.    0.    1.    0.    0. !
!   0.    1.    0.    0.    0. !
!   0.    0.    0.    1.    0. !
 u  =

!   5.    6.     7.     8.     9.  !
!   0.    0.8    1.6    2.4    3.2 !
!   0.    0.     0.     0.     0.  !
!   0.    0.     0.     0.     0.  !
!   0.    0.     0.     0.     0.  !
 l  =

!   1.     0.      0.           0.    0. !
!   0.2    1.      0.           0.    0. !
!   0.6    0.5     1.           0.    0. !
!   0.4    0.75    0.3695998    1.    0. !
!   0.8    0.25    0.5          0.    1. !

-->p*a-l*u
 ans  =

!   0.           0.    0.    0.    0. !
!   0.           0.    0.    0.    0. !
! - 4.441E-16    0.    0.    0.    0. !
!   0.           0.    0.    0.    0. !
!   0.           0.    0.    0.    0. !
```

2.6 謝辞

情報提供者：柏木先生
協力者：柏木研 M1 吉田君

参考文献

[1] 大石進一著：”Linux 数値計算ツール”，コロナ社 (2000)