

# LU分解法（3）

東京大学情報基盤センター 准教授 片桐孝洋

2015年12月15日(火)16:50－18:35

| 全学ゼミ：スパコンプログラミング研究ゼミ



東京大学情報基盤センター  
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

# 講義日程 (全学ゼミ)

- ▶ 9月15日: ガイダンス
- + ~~9月22日 ※休日の講義日~~
  - ~~並列数値処理の基本演算(座学)~~
- 2 ~~9月29日~~
  - ~~非同期通信、ソフトウェア自動チューニング~~
- 3 ~~10月6日: スパコン利用開始~~
  - ~~ログイン作業、テストプログラム実行~~
- 4 ~~10月13日~~
  - ~~高性能演算技法1  
(ループアンローリング)~~
- 5 ~~10月20日~~
  - ~~高性能演算技法2  
(キャッシュブロック化)~~

レポートおよびコンテスト課題  
(締切:  
**2016年1月12日(火)24時厳守**)

- 5. ~~10月27日~~
  - ~~行列ベクトル積の並列化~~
- 6. ~~11月3日 ※休日の講義日~~
  - ~~べき乗法の並列化~~
- 7. ~~11月10日~~
  - ~~行列 行列積の並列化(1)~~
- 8. ~~12月1日~~
  - ~~※休講と指定日のため~~
  - ~~行列 行列積の並列化(2)~~
- 9. ~~12月8日~~
  - ~~LU分解法(1)~~
  - ~~レポート・コンテスト課題~~
- 10. ~~12月15日~~
  - ~~LU分解法(2)~~

# LU分解法の演習日程

---

## 1. 今週

- ▶ 講義 & 並列化の検討

## 2. 今週

- ▶ LU分解法並列化実習

## 3. 今週

- ▶ LU分解法並列化実習

# 講義の流れ

---

1. 並列化実習の続き
2. 並列化のヒント(その2)の説明

# LU分解並列化のヒント（2） C言語版

ほぼ解答が載っています

# LU分解部分(1)

```
▶ ib = n/numprocs;  
  istart = myid * ib;  
  iend = (myid+1)* ib;  
  
/* LU decomposition ----- */  
for (k=0; k<iend; k++) {  
    iddiagPE = k / ib;  
    if (iddiagPE == myid) { /* 枢軸列をもつPE */  
        dtemp = 1.0 / A[k][k];  
        枢軸列の計算と、buf[ ]へ枢軸列をコピー；  
        for (i=myid+1; i<numprocs; i++) { /* 枢軸列の転送 */  
            MPI_Send(&buf[...], ..., MPI_DOUBLE, i, k, MPI_COMM_WORLD);  
        }  
        istart = k+1; /* 担当範囲の縮小 */  
    } else { /* 枢軸列を持たないPE */  
        MPI_Recv(&buf[...], ..., MPI_DOUBLE, iddiagPE, k, MPI_COMM_WORLD, &istatus);  
    }  
}
```

# LU分解部分(2)

```
/* 共通消去部分 */
for (j=k+1; j<n; j++) {
    dtemp = buf[j];
    for (i=istart; i<iend; i++) {
        A[j][i] = A[j][i] - A[k][i]*dtemp;
    }
}

} /* End of k-loop ----- */

/* 前進消去にメッセージがかぶらないように同期 ----- */
MPI_Barrier(MPI_COMM_WORLD);
```

# 前進代入部分(1)

```
▶ istart = myid * ib; iend = (myid+1) * ib; /* 担当範囲の初期化 */
/* Forward substitution ----- */
for (k=0; k<n; k++)
    c[k] = 0.0; /* cの初期化 */

for (k=0; k<n; k+=ib) { /* 対角ブロック判定用ループ */
    if (k >= istart) { /* 担当するブロックがある */
        idiagPE = k / ib;
        if (myid != 0)
            /* 左隣りPEからデータを受け取る */
            MPI_Recv(&c[k], ib, MPI_DOUBLE, myid-1, k, MPI_COMM_WORLD, &istatus);
        if (myid == idiagPE) { /* 対角ブロックをもつPE*/
            /* 対角ブロックだけ先行計算し値を確定させる */
            for (kk=0; kk<ib; kk++) {
                c[k+kk] = b[k+kk] + c[k+kk];/* 途中結果が送られてくるため必要な変更点*/
                for (j=istart; j<istart+kk; j++)
                    c[k+kk] -= A[k+kk][j] * c[j];
            }
        }
    }
}
```

## 前進代入部分(2)

```
} else { /* 対角ブロックを持たないPE */
    /* 自分の所有範囲のデータのみ計算(まだ最終結果ではない) */
    for (kk=0; kk<ib; kk++)
        for (j=istart; j<iend; j++)
            c[k+kk] -= A[k+kk][j]*c[j];

    /* 右隣のPEに、自分の担当範囲のデータを用いた演算結果を送る */
    if (myid != numprocs-1)
        MPI_Send(&c[k], ib, MPI_DOUBLE, myid+1, k, MPI_COMM_WORLD);
}

} /* End of if(担当するブロックがある) ----- */
} /* End of k-loop ----- */
```

# LU分解並列化のヒント（2） Fortran言語版

ほぼ解答が載っています

# LU分解部分(1)

```
▶ ib = n/numprocs
  istart = myid * ib + 1
  iend = (myid+1)* ib
c --- LU decomposition -----
  do k=1, iend
    idiagPE = (k-1) / ib
c --- 枢軸列をもつPE
  if (idiagPE .eq. myid) then
    dtemp = 1.0 / A(k, k)
    枢軸列の計算
c --- 枢軸列の転送
  do i=myid+1, numprocs - 1
    call MPI_Send(A(k,k), ..., MPI_DOUBLE_PRECISION, i, k, MPI_COMM_WORLD, ierr )
  enddo
c --- 担当範囲の縮小
  istart = k + 1
  else
c --- 枢軸列を持たないPE
  call MPI_Recv(A(k,k)), ..., MPI_DOUBLE_PRECISION idiagPE, k, MPI_COMM_WORLD, istatus, ierr)
  endif
```

# LU分解部分(2)

c --- 共通消去部分

```
do j=istart, iend  
    dtemp = A( k,j )  
    do i=k+1, n  
        A(i ,j) = A(i ,j) - A(i ,k) * dtemp  
    enddo  
enddo
```

```
enddo
```

c --- End of k-loop -----

c --- 前進消去にメッセージがかぶらないように同期 -----  
call MPI\_Barrier(MPI\_COMM\_WORLD, ierr)

# 前進代入部分(1)

```
c --- 担当範囲の初期化  
    istart = myid * ib + 1  
    iend = (myid+1) * ib  
  
c --- Forward substitution -----  
  
c --- c の初期化  
    do k=1, n  
        c[k] = 0.0    enddo  
  
c --- 対角ブロック判定用ループ  
    do k=1, n, ib  
        if (k .le. istart) then  
            idiagPE = (k-1) / ib  
  
c --- 担当するブロックがある  
    if (myid .ne. 0) then  
  
c     --- 左隣りPEからデータを受け取る  
        call MPI_Recv(c(k), ib,  
  
&         MPI_DOUBLE_PRECISION,  
&         myid-1, k, MPI_COMM_WORLD,  
&         istatus, ierr)
```

```
if (myid .eq. idiagPE) then  
    --- 対角ブロックをもつPE  
    do kk=1, ib  
        --- 途中結果が送られてくるため必要な変更点  
        c(k+kk-1) = b(k+kk-1) + c(k+kk-1)  
        --- 対角ブロックだけ先行計算し値を確定させる  
        do j=istart, istart+kk-2  
            c(k+kk-1) = c(k+kk-1) - A(k+kk-1, j ) * c(j )  
        enddo  
    enddo
```

# 前進代入部分(2)

```
else
c --- 対角ブロックを持たないPE
do kk=1, ib
    do j=istart, iend-1
        c(k+kk-1) = c(k+kk-1) - A(k+kk-1, j ) * c( j )
    enddo
    enddo
c --- 自分の所有範囲のデータのみ計算(まだ最終結果ではない)
if (myid .ne. numprocs-1) then
c --- 右隣のPEに、自分の担当範囲のデータを用いた演算結果を送る
    call MPI_Send(c(k), ib, MPI_DOUBLE_PRECISION, myid+1,
&                      k, MPI_COMM_WORLD, ierr)
    endif
    endif
    endif
c --- End of if 担当するブロックがある -----
enddo
c --- End of k-loop -----
```

おわり

お疲れ様でした