P2-1 Lis：a Library of Iterative Solvers for linear systems

小武守 恒（JST／東京大学） 藤井 昭宏（工学院大学）
長谷川 秀彦（筑波大学） 西田 晃（東京大学）

☆ ソフトウェアライブラリ公開中 ☆

http://ssi.is.s.u-tokyo.ac.jp/lis/

Lis (a Library of Iterative Solvers for linear systems)

- C言語で記述された線型方程式反復解法ライブラリ
- AMG前処理ルーチンを含むLis-AMGでは、Fortran 90も使用
- 現バージョンではC言語からの呼び出しにのみ対応
- 並列計算には、OpenMP または MPI-1 を使用

特徴

- 色々な解法と前処理を組み合わせて使える
- 様々な格納形式の入力を受け付ける
- 逐次環境から共有メモリまたは分散メモリ環境へのプログラム移行はソースコードの変更なし（あるいはごくわずかの変更）
- 反復解法、前処理の選択は、コマンドラインからも選択可能

実装されている解法・前処理・格納形式

格納形式

Point	Compressed Row Storage	(CRS)
	Compressed Column Storage	(CCS)
	Modified Compressed Sparse Row	(MSR)
	Diagonal	(DIA)
	Ellpack-Itpack generalized diagonal	(ELL)
	Jagged Diagonal	(JDS)
	Dense	(DNS)
	Coordinate	(COO)
Block	Block Sparse Row	(BSR)
	Block Sparse Column	(BSC)
	Variable Block Row	(VBR)

解法

対称	CG
非対称	BiCG
	CGS
	BiCGSTAB
	BiCGSTAB(l)
	GPBiCG
	Orthomin(m)
	GMRES(m)
	QMR
	Jacobi
	Gauss-Seidel
	SOR

前処理

- Jacobi
- ILU
- SSOR
- Hybrid

  前処理行列をMとする。反復過程で解かなければならないMz=vの代わりにAz=vをSOR法などで近似的に解く
- SA-AMG

  問題行列AからAと似た性質を持つサイズの小さい行列A'を生成する（構築部）。その後、サイズの小さい問題A'x'=b'とAx=bを交互に緩和法を適用しながら効率よく解く（解法部）。
- I+S型

  単位行列と係数行列A=(a_{ij})のi+1の部分の符号を逆にしたもの

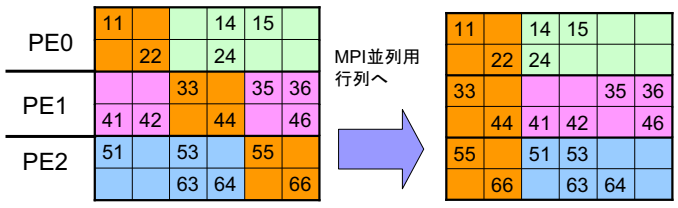
$$S=\begin{cases}-a_{ij} & j=i+1\\ 0 & else\end{cases}$$

- SAINV

  近似逆行列

MPIによる並列化

- 行ブロック分割（CCS,BSCは列ブロック分割）
- 通信テーブルの作成
  - 通信テーブルはGeoFEMを参考
  - 行列作成時にライブラリ側で自動生成



通信テーブル		PE0	PE1	PE2
neibpetot	通信を行うPE数	2	2	2
beibpe	PE番号	1,2	0,2	0,1
export_index	export_nodeの開始位置	0,2,3	0,1,3	0.1.3
export_node	送信する要素の位置	0,1,0	1,0,1	0,0,1
import_index	import_nodeの開始位置	0,1,2	0,2,4	0,1,2
import_node	受信する要素の位置	2,3	2,3,4,5	2,3,4

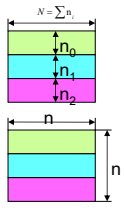
求解までの流れ

- 初期化
- 行列作成
  - 直接作成
  - ファイルから
    - MatrixMarket フォーマット
- 初期ベクトル、右辺ベクトル作成
- パラメータ設定
  - 解法、前処理等
  - コマンドラインからの指定も可
- 求解
- 終了処理

```
int main(int argc, char* argv[]) {
  LIS_MATRIX   A;
  LIS_VECTOR   b,x,u;
  int          n,gn,nnz,options[LIS_OPTIONS_LEN],*index,*ptr;
  LIS_SCALAR   params[LIS_PARAMS_LEN],*value;
  LIS_SCALAR   status[LIS_STATUS_LEN];

  lis_initialize(argc, argv, NULL, options, params);

  /******* ユーザーが行列を直接作成する場合 ********/
  lis_matrix_create(0,gn,LIS_COMM_WORLD,&A);
  /* CRSに必要な配列 ptr,index,valueをmalloc */
  lis_matrix_get_range(A,&is,&ie);
  for(i=is;i<ie;i++) {  /* ptr,index,valueに値を代入 */ }
  lis_matrix_set_crs(nnz,ptr,index,value,A);
  lis_matrix_assemble(&A,LIS_MATRIX_CRS);

 /******* ファイルから行列、ベクトルを読み込む場合 ********/
  lis_input(&A,&b,&x,argv[1],LIS_COMM_WORLD,LIS_MATRIX_CRS);

  n  = A->n; gn = A->gn;
  lis_vector_create(0,gn,LIS_COMM_WORLD,&u);
  lis_vector_duplicate(u,&b); lis_vector_set_all(1.0,u);
  lis_matvec(A,u,b);
  lis_vector_duplicate(u,&x); lis_vector_set_all(0.0,x);
  lis_solve(A,b,x,params,options,status);
  lis_finalize();
}
```

行列の作成

- lis_matrix_create(int local_n, int global_n, LIS_Comm comm, LIS_MATRIX *Amat)
- 2通りの作成方法
  - 部分行列の大きさを指定
    - lis_matrix_create(ni,0,comm,&A);
    - 各プロセッサにnp x Nの部分行列を作成
  - 全体行列の大きさを指定
    - lis_matrix_create(0,n,comm,&A);
    - 各プロセッサにmp x nの部分行列を作成



配列の関連付け

- lis_matrix_set_crs(int nnz, int *ptr, int *index, LIS_SCALAR *value, LIS_MATRIX A)
- 行列の列番号は全体行列の番号

数値実験

- 環境

	SGI Altix3700	PC Cluster
CPU	Itenium2 1.3GHz	Xeon 2.8GHz
メモリ /node	2GB	1GB
通信	NUMAlink	1000Base-T
コンパイラ	ICC Ver8.1	ICC Ver9.0

- 行列
  - 3次元ポアソン方程式を有限要素法で離散化
  - 次数n 1,000,000
  - 非零要素数nnz 26,463,592
  - メモリ量(CRS) 304MB

求解

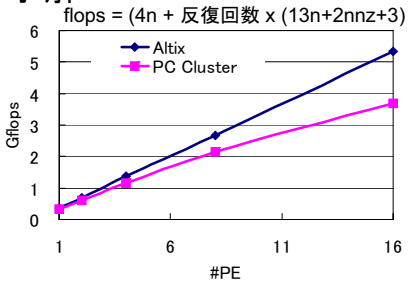
- Ax=bを反復法で解く
  - 解法：CG法
  - 前処理：Jacobi
  - 右辺bは解xの要素すべてが1となるように設定
  - 収束判定基準：10^{-12}
  - 格納形式：CRS
- 結果
  - Altix：ほぼ理想的な速度向上
  - PC Cluster：台数増加に伴い通信時間がネックになっている

行列ベクトル積y=Ax

- Petscとの比較を行う
- 結果
  - Petscと同程度の性能（Lisのほうがやや良い）
  - 行列ベクトル積は求解の実行時間の約6割を占める

求解

flops = (4n + 反復回数 x (13n+2nnz+3) / 実行時間



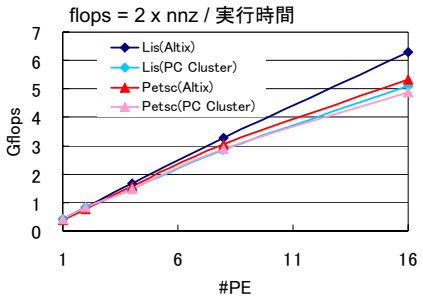
反復回数：100

実行時間（秒）[カッコ内は速度向上比]

	1	2	4	8	16
Altix	18.82 (1.00)	9.40 (2.00)	4.77 (3.95)	2.48 (7.58)	1.23 (15.25)
PC Cluster	20.57 (1.00)	10.76 (1.91)	5.73 (3.58)	3.07 (6.70)	1.79 (11.50)

行列ベクトル積

flops = 2 x nnz / 実行時間



実行時間（秒）[カッコ内は速度向上比]

	1	2	4	8	16
Lis(Altix)	0.123 (1.00)	0.063 (1.97)	0.032 (3.89)	0.016 (7.66)	0.008 (14.67)
Lis(PC)	0.124 (1.00)	0.064 (1.95)	0.036 (3.48)	0.019 (6.71)	0.010 (11.99)
Petsc(Altix)	0.133 (1.00)	0.068 (1.97)	0.034 (3.97)	0.017 (7.71)	0.010 (13.46)
Petsc(PC)	0.126 (1.00)	0.064 (1.97)	0.036 (3.51)	0.018 (6.91)	0.011 (11.62)