FrontISTRの
線形ソルバーと前処理

合同会社PExProCS 後藤和哉

2016年3月18日 **第26回FrontISTR研究会** ＠東京大学生産技術研究所

ねらい

- FrontISTRの線形ソルバーを使いこなす
- 計算を高速化するための指針を得る

本日の内容

- 線形ソルバーの概略
- FrontISRの線形ソルバーと前処理
- 各線形ソルバー・前処理の特長
- 問題の種類による行列の性質と求解パターン
- !SOLVERの設定

線形ソルバーの概略

線形ソルバーとは

連立一次方程式（行列方程式）の解法

- $\mathbf{Ax} = \mathbf{b}$ を解く

多くのFEM構造解析で必要

- 一部の解析を除く（陽解法による動解析など）

FEM解析の計算時間の大部分を占める

- 線形ソルバーの高速化が全体の高速化につながる

解法の種類

直接法

- 直接、解を求める
  - 反復回数１の反復法とも考えられる

反復法

- 反復的に解を求める

直接法（ガウスの消去法）

下三角行列と上三角行列の積に分解

- A＝LU （対称の場合は LL^T or LDL^T ）と分解
- 分解が計算の大部分を占める

前進消去、後退代入で求解

- Ly=b
- Ux=y
- 求解は速い

疎行列の直接法

Aが疎行列の場合、L+UがAよりも密になる

- 新たに発生する非ゼロ成分を「フィルイン」と呼ぶ

計算量・メモリ使用量はフィルインの量で決まる

- フィルインが少なければ、より大きな問題に適用可能

反復法

定常反復法

- Jacobi法、Gauss-Seidel法、SOR法など
- 主に、クリロフ部分空間法の前処理として使われる

非定常反復法（クリロフ部分空間法）

- CG法、GMRES法、BiCGSTAB法など
- 定常反復法よりも収束が速いとされる
- 単に「反復法」というと、こちらを指すことが多い

前処理

反復法の収束性改善のための工夫

- Ax＝b を解く代わりに $M^{-1}Ax=M^{-1}b$ を解く

様々な前処理がある

- 対角スケーリング
- Jacobi, Gauss-Seidel, SORなど
- 直接法の分解を不完全に行うもの（ILU(k), IC(k)など）
- マルチグリッド法にもとづくもの（AMGなど）

FrontISTRの線形ソルバーと前処理

３自由度用のみ解説します

直接法

組み込みの直接法

- シリアル計算にのみ対応

Intel MKLに含まれる直接法 PARDISO

- シリアル計算にのみ対応（スレッド並列は可能）
- 接触解析でのみ利用可能

外部の並列直接法ライブラリ MUMPS

- 並列計算にも利用可能
- FrontISTRのすべての解析で利用可能

反復法

CG法

- 対称正定値行列向け
- ほとんどの構造解析では「反復法」と言えばこちら

GMRES法、BiCGSTAB法、GPBiCG法

- 非対称行列に対応
- 摩擦を含む接触解析ではこちらを利用
  - ただし、接触解析への対応状況は「試験的」

前処理（1/2）

対角スケーリング

- 行列の対角成分を 1 にする前処理
- もっとも単純かつ軽量

SSOR

- 対称逐次過緩和前処理
- 比較的軽量

前処理（2/2）

ILU(k), k=0~2

- 不完全LU分解前処理
- 考慮するフィルインのレベルに応じて重くなる

外部ライブラリMLによるAMG前処理

- 代数マルチグリッド前処理
- 計算は重いが、特に大規模問題では非常に強力

修正済み

各解法・前処理の並列化状況

解法	前処理	MPI	OpenMP
DIRECT	-	×	×
DIRECTmkl	-	×	○
MUMPS	-	○	BLASによる
CG GMRES BiCGSTAB GPBiCG	Diag. Scaling	○	○
	SSOR	○	○
	ILU(0)	○	×
	ILU(1)	○	×
	ILU(2)	○	×
	ML	○	△

その他の機能

- 反復法の収束履歴
- 計算時間
- 行列のダンプ
- ベクトル機向け最適化
- MPC条件の処理方法の選択
- 条件数の推定

各解法・前処理の特長

直接法の長所

ロバスト

- 決まった演算数で解が得られる

求解を繰り返す場合に有利

- 分解は時間がかかるが、求解は一瞬

フィルインが少ない場合は高速

- 小規模問題や薄肉形状など

直接法の短所

大規模問題では使えない

- 計算量・メモリ使用量が爆発
- ただし、フィルインの量による

並列化してもあまり速くならない

- 「より多くのメモリを使うため」の並列化

フィルインが多い場合は遅い

- 大規模問題や厚肉形状など

フィルインの多寡は何で決まるか

行列の非ゼロ成分のプロファイルで決まる

≒ メッシュの構造で決まる

≒ モデルの形状で決まる

- 厚肉形状：フィルインが多い
- 薄肉形状：フィルインが少ない

反復法の長所

少ない演算で解が得られることがある

- 行列が解きやすい場合

必要な精度で計算を打ち切ることが可能

- 低い精度でよければ、その分、早く終わる

大規模問題でも使える

- 計算量・メモリ使用量の増大が緩やか
- 並列化効率が得やすい

反復法の短所

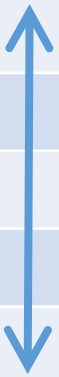
解が得られない（収束しない）ことがある

- 行列が解き難い場合
- 前処理が収束性を左右する

前処理の比較

前処理	セットアップ	１反復あたりの計算	収束性改善の効果
対角スケーリング	不要	軽い	弱
SSOR	軽微	行列ベクトル積程度	中
ILU(k)	kに応じて重くなる	行列ベクトル積の定数倍程度（kに応じて重くなる）	中〜強
AMG	重い	行列ベクトル積の定数倍程度	強

解法・前処理の比較

解法・前処理	セットアップ	求解	ロバスト性
CG法＋Diag	軽い	遅い	低い
CG法＋SSOR			
CG法＋ILU(0)			
CG法＋ILU(1)			
CG法＋ILU(2)			
CG法＋AMG			
直接法	重い	速い	高い

問題の種類による
行列の性質と求解パターン

行列の性質は何で決まるか

条件数：最大固有値と最小固有値の比

- 形状
- 静解析か動解析か
- その他
  - 問題の規模、要素のアスペクト比、物性のばらつき、…

形状

厚肉

- 固有値分布が比較的小さい
- 条件数が小さい
- 解きやすい

薄肉

- 固有値分布が比較的大きい
- 条件数が大きい
- 解き難い

静解析と動解析

静解析

- $Ku = f$ を解く
- 剛性行列の性質で解きやすさが決まる

動解析

- $M\ddot{u} + C\dot{u} + Ku = f$ を時間方向に離散化して解く
- ニューマークβ法の場合
  $\hat{K}u = \hat{f}$ $\left(\hat{K} = \frac{1}{\beta \Delta t^2} M + \frac{\gamma}{\beta \Delta t} C + K,\ \hat{f} = \text{省略}\right)$ を解く
- 剛性行列に質量行列を加えた行列を解く
  - 静解析よりも解きやすい

求解パターン

静解析/動解析	線形/非線形	求解	行列
静解析	線形	１回だけ	―
	非線形	繰り返し	数値が変化
動解析	線形	繰り返し	数値が一定
	非線形	繰り返し	数値が変化

おすすめ設定

形状	静・動	線形・非線形	規模	解法
薄肉	静・動	線形・非線形	小～大	直接法
			超大	反＋AMG
厚肉	静	線形	小	直接法
			中～大	反＋強
		非線形	小	直接法
			中～大	反＋強
	動	線形	小～中	直接法
			大	反＋AMG
		非線形	小～大	反＋軽
			超大	反＋AMG

小：～数万$_{dof}$　中：～数十万$_{dof}$　大：～数百万$_{dof}$　超大：数千万$_{dof}$ ～

!SOLVERの設定

FrontISTR_user_manual_Ver36 を見ながら

パラメータ：METHOD

METHOD = 解法（CG、BiCGSTAB、GMRES、GPBiCG、DIRECT、DIRECTmkl、MUMPS）

DIRECT：接触解析以外での直接法（逐次処理のみ）

DIRECTmkl：接触解析における Intel MKL による直接法（逐次処理のみ）

MUMPS：並列直接法パッケージ MUMPS による直接法

直接法を選択したとき、データ行は無視される。

1、2 自由度問題では、CG、DIRECT、MUMPS のみ有効

シェル要素は、DIRECT、MUMPS のみ有効

3 自由度用の反復法は OpenMP によるスレッド並列が利用可能

よく使うのはCGとMUMPS

パラメータ：PRECOND

直接法の時は設定不要

PRECOND　＝　反復法の前処理手法（1、2、3、5、10、11、12）

1, 2：（Block) SSOR　（3 自由度用のみマルチカラーオーダリング付き）

3：（Block) Diagonal Scaling

5：マルチグリッド前処理パッケージ ML による AMG (試験的)

10: Block ILU(0)

11: Block ILU(1)

12: Block ILU(2)

10、11、12 は 3 自由度問題でのみ利用可能

OpenMP によるスレッド並列時は SSOR または Diagonal Scaling を推奨

わからない時は SSOR か ML を推奨

パラメータ：ITERLOG, TIMELOG 他

ITERLOG　＝　反復法ソルバー収束履歴出力の有無（ YES/NO ）（デフォルト：NO）

TIMELOG　＝　ソルバー計算時間出力の有無（ YES/NO ）（デフォルト：NO）

USEJAD　＝　ベクトル機向けオーダリングの有無（ YES/NO ）（デフォルト：NO）

3 自由度問題で反復法使用時のみ有効

SCALING　＝　行列の対角成分を 1 とするスケーリングの有無（ YES/NO ）（デフォルト：NO）

3 自由度問題で反復法使用時のみ有効

パラメータ：DUMPTYPE, DUMPEXIT

DUMPTYPE　=　行列ダンプ型式（NONE、MM、CSR、BSR）（主にデバッグ用）

NONE：ダンプしない（デフォルト）

MM：マトリックスマーケット型式

CSR：Compressed Sparse Row（CSR）型式

BSR：Blocked CSR 型式

DUMPEXIT　=　行列ダンプ直後のプログラム終了（ YES/NO ）（デフォルト：NO）

並列の時は領域ごとにローカルな行列を出力
（グローバル対応は未実装）

パラメータ：MPCMETHOD 他

MPCMETHOD　＝　多点拘束条件の処理手法（1､2､3）

1：　ペナルティ法

2：　MPC-CG 法

3：　陽的自由度消去法（デフォルト）

3 自由度問題で反復法使用時のみ有効

（直接法および反復法（１，２自由度）では常にペナルティ法、反復法（4，６自由度）では常に MPC-CG 法が採用される）

ESTCOND　＝　条件数推定の頻度（試験的）

指定された反復ごと、および、反復終了時に条件数推定を実施

0 の場合は推定を行わない

データ行①

数万回で収束しない場合は
設定を見直した方がいい

推奨は 1

（2 行目）NIER, iterPREmax, NREST, NCOLOR_IN

変数名	属 性	内　　容
NIER	I	反復回数　（デフォルト：100）
iterPREmax	I	Additive Schwarz による前処理の繰り返し数　（デフォルト：1） （推奨値は、逐次計算、前処理に対角スケーリングを用いる場合、および、MPC を含むモデルの計算では 1、その他の並列計算では 2）
NREST	I	クリロフ部分空間数（デフォルト：10） （解法として GMRES を選択したときのみ有効）
NCOLOR_IN	I	マルチカラーオーダリングにおける目標色数（デフォルト：10） （OpenMP のスレッド数が 2 以上の時のみ有効）

推奨は10～100
（規模による）

データ行②

1e-6 程度でも良い

（3 行目）RESID, SIGMA_DIAG, SIGMA

変数名	属 性	内　　容
RESID	R	打ち切り誤差　（デフォルト値：1.0e-8）
SIGMA_DIAG	R	前処理行列計算時に対角成分にかける倍率（デフォルト値： 1.0）
SIGMA	R	未使用（デフォルト値： 0.0）

ILU(k)で発散する場合、
1.2ぐらいまで徐々に増やすと
収束する場合がある

!SOLVERの設定例

1: !SOLVER, METHOD=CG, PRECOND=1, ITERLOG=YES, TIMELOG=YES

2: 10000, 2, 10, 100

3: 1.0e-8, 1.0, 0.0

本日の内容

- 線形ソルバーの概略
- FrontISRの線形ソルバーと前処理
- 各線形ソルバー・前処理の特長
- 問題の種類による行列の性質と求解パターン
- !SOLVERの設定