1. 研究分野

PDES，可逆計算

2. 目的

逆計算を用いるPDESの性能評価

3. 背景

現在，楽観的なPDESをロールバックを可能にするために逆計算が用いられている．逆計算を可能にする方法は2種類あり，incremental state savingと逆実行である．ここでは，2つの方法に基づいて作成されたコードを体系的に評価する．

4. アプローチ

手書きのC++コードとBackstrokeとJanusから自動生成されたC++コードをそれぞれ比較する．行列の乗算は個々のステップが非可逆であるため，JanusのコードはLDU分解を使用し，生成された三角行列との乗算を行うことで可逆な行列の乗算を行っている．また，性能を評価するために新しいベンチマークを開発した．このベンチマークは従来のPHOLDベンチマークと類似しているが，シミュレーションエラーの検出を支援する変数と計算が含まれている．このベンチマークでは，行列の乗算がイベントとなる．逆計算を適用できるようにするために，行列の乗算には要素が整数環上の非特異行列が使用される．

5. 結果

実行結果の評価では，Jaunsコードは行列のサイズが小さいと最も性能が高くなり，Backstrokeコードは行列のサイズが大きくなるにつれて性能が向上した．2つのコードの性能は行列のサイズが20の場合で交差し，それよりも小さいとJaunsコードの方が，大きいとBackstrokeコードの方が性能が高くなる．行列のサイズが640の場合では，Backstrokeコードの性能は手書きのコードとほぼ同じだった．これは，Backstrokeコードが中間の計算結果を保存しないように最適化されているため，コミットコードが呼び出されるまでメモリトレースが特定のサイズまでしか成長しないからだ．

また，行列のサイズが十分に大きい場合，JanusコードはLDU分解の後に乗算を実行しているため，操作の総数が標準の手順の約5/3倍になっているので，実行時間もほぼ2倍になると予測される．また，同じく行列のサイズが十分に大きい場合，Backstrokeコードは中間の計算結果を保存しないように最適化されているため，サイズnの行列に対し $2n^2$ のメモリ操作回数がある．行列の乗算の実行時間はO(n^3)であるため，このオーバヘッドは無視できると推測される．

また，Janusコードは追加の情報の保存を必要としないため，メモリの節約を行うことができる．ただし，Janusコードは可逆性の保証のためにアサーションの記述が必要であるため，非可逆で最適な操作よりも多くの操作が必要になる可能性がある．

6. 有用性

行列のサイズに応じて，Janusが生成したコードかBackstrokeが生成したコードのどちらが最適かを決めることができる．したがって，両方のコードを含み，行列のサイズに応じて最適なコードを呼び出すような実装が可能である．

7. 限界・短所

今回のベンチマークで実行した計算は行列の乗算であるため，他の計算に対する性能の指標にはならない可能性がある．また，今回のベンチマークは要素が整数環上の非特異

行列を使用しているため，環とならないデータ型の要素をもつ行列や特異行列が出てくる行列の乗算では性能が低くなる可能性がある.

8. 次に何を読めばいいか？

PHOLDとはHOLDベンチマークの離散（Parallel）版である．HOLDは一連のイベントをランダム（2つのイベント間の時間は指数分布から決定）に発生させるベンチマークである．PHOLDはPDES-graph（たぶん，ノードがLP，エッジがLP間の繋がりのグラフ）を取得し，すべてのLPに対して均一かつランダムにイベントをスケジュールする.

Benchmarking PDES Algorithms Vincent Bonnet 3539733 v2.1 p18-19より