CONVERGENCE AND RESILIENCE OF THE FINE-GRAINED PARALLEL INCOMPLETE LU FACTORIZATION FOR NON-SYMMETRIC PROBLEMS

Evan Coleman

Naval Surface Warfare Center
Dahlgren Division
17320 Dahlgren Rd
Dahlgren, VA, USA
ecole028@odu.edu

Masha Sosonkina

Department of Modeling, Simulation
and Visualization Engineering
Old Dominion University
5115 Hampton Blvd, Norfolk, VA, USA
msosonki@odu.edu

ABSTRACT

This paper presents an investigation into the convergence of the fine-grained parallel algorithm for computing an incomplete LU factorization for non-symmetric and indefinite matrices. The fine-grained parallel incomplete LU factorization is a nonlinear fixed point iteration and convergence has not been extensively studied for problems that are not symmetric positive definite. This work investigates the convergence of the algorithm for these more difficult problems and additionally investigates how the occurrence of a computing fault may impact the convergence of the algorithm for the problems being studied. The results obtained suggest that this class of problems presents challenges for the fine-grained parallel incomplete LU factorization (less than 30% of configurations converge naturally), and that while the occurrence of a fault can cause significant negative effects, the simple algorithmic change advocated here can completely ameliorate the effects of a fault.

Keywords: incomplete LU factorizations, preconditioning, fine-grained parallelism, fault tolerance

1 INTRODUCTION

Fine-grained methods are increasing in popularity recently due to their ability to be parallelized naturally on modern co-processors such as GPUs and Intel Xeon Phis. Many examples of recent work using fine-grained parallel methods are available (Chow, Anzt, and Dongarra 2015, Chow and Patel 2015, Anzt, Chow, and Dongarra 2015). A specific area of interest is on techniques that utilize fixed point iteration, i.e., $x = G(x)$ for some vector x and map G. These methods can be computed in either a synchronous or asynchronous manner which helps tolerate latency in high-performance computing (HPC) environments. Looking forward to the future of HPC, it is important to keep in mind the need for developing algorithms that are resilient to faults. On future platforms, the rate at which faults occur is expected to decrease dramatically (Cappello, Geist, Gropp, Kale, Kramer, and Snir 2009, Cappello, Geist, Gropp, Kale, Kramer, and Snir 2014, Asanovic, Bodik, Catanzaro, Gebis, Husbands, Keutzer, Patterson, Plishker, Shalf, Williams, et al. 2006, Geist and Lucas 2009). The fine-grained parallel incomplete LU (FGPILU) factorization (Chow and Patel 2015), which is the focus of this work, is a popular preconditioning method that can be used as a building block for iterative linear-system solvers geared towards novel computing platforms. Typically when working with difficult problems, preconditioning techniques move beyond the fill-in level-based ILU factorizations, of

which FGPILU is representative, to threshold-based ones such as ILUT (Saad 2003), or to even to algebraic multilevel preconditioning such as ARMS (Saad and Suchomel 2002).

The FGPILU algorithm is a nonlinear fixed point iteration, and while convergence of the algorithm is guaranteed for some neighborhood around the solution, questions remain regarding the practical performance of the algorithm with respect to difficult problems. For the purposes of this work, "difficult" problems are defined to be those that satisfy any one of the following three criteria: (i) non-symmetric, (ii) not diagonally dominant, or (iii) ill-conditioned (including indefinite matrices). The majority of the work on the algorithm so far has focused on matrices that are symmetric and positive-definite (SPD) (Chow and Patel 2015, Chow, Anzt, and Dongarra 2015, Coleman, Sosonkina, and Chow 2017), and the performance of the algorithm on non-symmetric and indefinite matrices has not been firmly established. Moreover, if the convergence of the algorithm for these classes of problems is less than desirable, they may be more prone to suffer divergence when faced with a fault.

Developing algorithms that are resilient to faults is of paramount importance and fine-grained parallel (fixed point) methods are no exception. Faults can be divided into two categories: hard faults and soft faults (Bridges, Ferreira, Heroux, and Hoemmen 2012). Hard faults cause immediate program interruption and typically come from negative effects on the physical hardware components of the system or on the operating system itself. Soft faults represent all faults that do not cause the executing program to stop and are the focus of this work. Most often, these faults refer to some form of data corruption that is occurring either directly inside of, or as a result of, the algorithm that is being executed. This paper examines the potential impact of soft faults on the fine-grained parallel incomplete LU factorization, and also investigates the use of fine-grained parallel incomplete LU algorithm generated preconditioners on Krylov subspace solvers. The main contributions of this work are analyzing the ability of the FGPILU fixed point iteration to complete successfully when attempting to solve difficult problems under a myriad of different configurations, investigating how the convergence is affected by the occurrence of a soft fault, and demonstrating that the effects of a fault can be mitigated by the simple checkpointing scheme proposed in Coleman, Sosonkina, and Chow 2017. The structure of this paper is organized as follows: In Section 2, a brief summary of some related studies is provided. In Section 3, an overview of fixed-point iterations specific to their use in high performance computing is given. In Section 4, background information is provided for the fine-grained parallel incomplete LU algorithm. In Section 5, a theoretical underpinning of the fine-grained parallel incomplete LU algorithm with respect to its convergence for nonsymmetric or indefinite problems is explored. In Section 6, a series of numerical results are provided, while Section 7 concludes.

2 RELATED WORK

Other work on using (conventional) incomplete LU factorizations for solving difficult problems from various disciplines has been conducted previously, including the more general studies found in Chow and Saad 1997 and Benzi, Haws, and Tuma 2000. The increase in faults for future HPC systems is detailed in several different places (Asanovic, Bodik, Catanzaro, Gebis, Husbands, Keutzer, Patterson, Plishker, Shalf, Williams, et al. 2006, Cappello, Geist, Gropp, Kale, Kramer, and Snir 2009, Cappello, Geist, Gropp, Kale, Kramer, and Snir 2014, Snir, Wisniewski, Abraham, Adve, Bagchi, Balaji, Belak, Bose, Cappello, Carlson, et al. 2014, Geist and Lucas 2009). An initial look into fault tolerance for the FGPILU algorithm is provided in Coleman, Sosonkina, and Chow 2017; the variants of the FGPILU algorithm discussed therein build on ideas from various methods for fault tolerance (Sao and Vuduc 2013, Bridges, Ferreira, Heroux, and Hoemmen 2012, Hoemmen and Heroux 2011). A more general approach for simulating the occurrence of faults is taken in this study. Several recent studies have adopted similar techniques (Coleman and Sosonkina 2016, Coleman, Jamal, Baboulin, Khabou, and Sosonkina 2017, Elliott, Hoemmen, and Mueller 2015, Elliott, Hoemmen, and Mueller 2014, Stoyanov and Webster 2015). The fault model used in this paper is a combination of a modified version of the one initially developed in Coleman and Sosonkina 2016 that was used

in Coleman, Sosonkina, and Chow 2017. Details on the fault model used here are provided in Section 6.1. Fine-grained parallel methods, specifically parallel fixed point methods, are an area of increased research activity due to the practical use of these methods on HPC resources. An initial exploration of fault tolerance for the FGPILU factorization studied here is provided in Coleman, Sosonkina, and Chow 2017 and **?**, and an exploration of resilience for stationary iterative linear solvers (i.e., Jacobi) is given in Anzt, Dongarra, and Quintana-Ortí 2015. Fault tolerance for fixed point algorithms has been investigated in Stoyanov and Webster 2015, and a more general exploration of fault tolerance for fine-grained methods is provided in Coleman and Sosonkina 2017. The general convergence of parallel fixed point methods has been explored extensively (Addou and Benahmed 2005, Frommer and Szyld 2000, Bertsekas and Tsitsiklis 1989, Ortega and Rheinboldt 2000, Baudet 1978, Benahmed 2007).

3 FIXED POINT ITERATION

Fixed point iterations are concerned with finding solutions to the iteration $x^{k+1} = G(x^k)$ where $G : \mathbb{R}^n \to \mathbb{R}^n$ is composed of component-wise functionals g_i such that

$$x_1 = g_1(\vec{x}) \ , \ x_2 = g_2(\vec{x}) \ , \ \ldots \ x_n = g_n(\vec{x}) \ ,$$

where the subscript represents the *component*, the iteration superscripts have been removed, and the vector notation is added to emphasize that each individual functional used to update a specific component can (potentially) rely on all other components. In a parallel computing environment, the task of finding the update for an individual (or set of) component(s) can be assigned to an individual processing element. In a system that relies on synchronous updates, the functionals all utilize the same components of $\vec{x}$. In particular, $x_i^{k+1} = g_i(\vec{x}^k)$ for all components $i \in \{1, 2, \ldots, n\}$. On the other hand, in the asynchronous case processors will use the latest information available to them. This will lead to different update patterns for each of the individual functionals, each of which will be utilizing components that are updated a different number of times. The convergence of parallel fixed point iterations is discussed in the literature for both the synchronous (Addou and Benahmed 2005) and asynchronous (Frommer and Szyld 2000) cases among many other sources (Bertsekas and Tsitsiklis 1989, Ortega and Rheinboldt 2000, Baudet 1978, Benahmed 2007).

4 FINE-GRAINED PARALLEL INCOMPLETE LU FACTORIZATION

The fine-grained parallel incomplete LU (FGPILU) factorization approximates the true LU factorization and writes a matrix A as the product of two factors L and U where, $A \approx LU$. Normally, the individual components of both L and U are computed in a manner that does not allow easy use of parallelization. The recent FGPILU algorithm proposed in Chow and Patel 2015 allows each element of both the L and U factors to be computed independently. The algorithm progresses towards the incomplete LU factors that would be found by a traditional algorithm in an iterative manner. To do this, the FGPILU algorithm uses the property $(LU)_{ij} = a_{ij}$ for all (i, j) in the sparsity pattern S of the matrix A, where $(LU)_{ij}$ represents the (i, j) entry of the product of the current iterate of the factors L and U. This leads to the observation that the FGPILU algorithm (given in Algorithm 1) is defined by the following two nonlinear equations:

$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right) , \ u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \ . \tag{1}$$

Following the analysis presented in (Chow and Patel 2015), it is possible to collect all of the unknowns l_{ij} and u_{ij} into a single vector x, then express these equations as a fixed-point iteration, $x^{(p+1)} = G\left(x^{(p)}\right)$, where the function G implements the two nonlinear equations described above. The FGPILU algorithm is

given in Algorithm 1. Keeping with the terminology used in Chow and Patel 2015 and Chow, Anzt, and

Algorithm 1: FGPILU algorithm as given in (Chow and Patel 2015).

Input: Initial guesses for $l_{ij} \in L$ and $u_{ij} \in U$
Output: Factors L and U such that $A \approx LU$

```
for sweep = 1, 2, ..., m do
    for (i, j) ∈ S do in parallel
        if i > j then
            l_ij = (a_ij − Σ_{k=1}^{j−1} l_ik u_kj)/u_jj
        else
            u_ij = a_ij − Σ_{k=1}^{i−1} l_ik u_kj
```

Dongarra 2015, each pass the algorithm makes in updating all of the l_{ij} and u_{ij} elements (alternatively: each element of the vector x) is referred to as a "sweep". After each sweep of the algorithm, the L and U factors progress towards convergence. At the beginning of the algorithm, the factors L and U are set with an initial guess. In this study, the initial L factor will be taken to be the lower triangular part of A and the initial U will be taken to be the upper triangular portion of A (as in Chow and Patel 2015, Coleman, Sosonkina, and Chow 2017, Anzt, Chow, Saak, and Dongarra 2016). Adopting the technique used in Chow and Patel 2015, Chow, Anzt, and Dongarra 2015, Coleman, Sosonkina, and Chow 2017, a scaling of the input matrix A is first performed such that the diagonal elements of A are equal to one. As pointed out in Chow and Patel 2015, this diagonal scaling is able to help maintain reasonable convergence rates for the algorithm, and the working assumption in this paper is that all matrices have been scaled in this manner.

5 CONVERGENCE OF THE FGPILU FACTORIZATION

The analysis to show convergence of the FGPILU algorithm relies on properties of the Jacobian associated with the nonlinear mapping defined by $G : \mathbb{R}^m \to \mathbb{R}^m$ where m represents the number of non-zero terms in the matrix A. To define the Jacobian, an order of the elements in both the L and U factors (which together constitute all of the elements in the vector $\vec{x}$ from Section 3) needs to be defined. In particular, an ordering g will map a pair of (i, j) coordinates specifying the location of a non-zero term in A to an index of the vector x, or to the set $\{1, 2, 3, \ldots, m\}$ where $m = nnz(L) + nnz(U)$. That is,

$$x_{g(i,j)} = \begin{cases} l_{ij} & i > j \\ u_{ij} & i \leq j \ . \end{cases}$$

Given this, the two nonlinear equations that define FGPILU, i.e., Eq. (1), can be rewritten such that,

$$G_{g(i,j)} = \begin{cases} \frac{1}{x_{g(j,j)}} \left(a_{ij} - \sum\limits_{1 \leq k \leq j-1} x_{g(i,k)} x_{g(k,j)} \right) & i > j \\ a_{ij} - \sum\limits_{1 \leq k \leq i-1} x_{g(i,k)} x_{g(k,j)} & i \leq j \ , \end{cases} \tag{2}$$

where both sums are taken over all pairs, (i,k) and $(k,j) \in S(A)$. The Jacobian itself can then be written $G'(x) = J(G(x))$ and is defined by the following equations (Chow and Patel 2015):

$$\begin{aligned}
\frac{\partial G_{g(i,j)}}{\partial x_{g(k,j)}} &= -\frac{x_{g(i,k)}}{x_{g(j,j)}}, k < j \\
\frac{\partial G_{g(i,j)}}{\partial x_{g(i,k)}} &= -\frac{x_{g(k,j)}}{x_{g(j,j)}}, k < j \\
\frac{\partial G_{g(i,j)}}{\partial x_{g(j,j)}} &= -\frac{1}{x_{g(j,j)}^2}\left(a_{ij} - \sum_{k=1}^{j-1} x_{g(i,k)} x_{g(k,j)}\right)
\end{aligned} \tag{3}$$

for a row in the Jacobian where $i > j$ (i.e., corresponding to an unknown $l_{ij} \in L$). Conversely, for a row $i \leq j$ (i.e., corresponding to an unknown $u_{ij} \in U$), the partial derivatives are given by:

$$\frac{\partial G_{g(i,j)}}{\partial x_{g(i,k)}} = -x_{g(i,k)}, k < i, \qquad \frac{\partial G_{g(i,j)}}{\partial x_{g(k,j)}} = -x_{g(i,k)}, k < i\ . \tag{4}$$

Under the assumption that there is a single fixed point solution x^* of the nonlinear iteration defined by $G(x)$, the following result given in Theorem 1 provides convergence for the nominal FGPILU algorithm:

Theorem 1 (Frommer and Szyld 2000)**.** *Assume that x^* lies in the interior of the domain of G and that G is F-differentiable at x^*. If $\rho(G'(x^*)) < 1$, then there exists some local neighborhood of x^* such that the asynchronous iteration defined by G converges to x^* given that the initial guess is inside of this neighborhood.*

Details of this analysis are provided in Chow and Patel 2015. As pointed out in that paper, one consequence of Theorem 1 is that the algorithm will be successful when the norm of the Jacobian is small. Examining the equations that define the partial derivatives inside of the Jacobian, this implies that the FGPILU algorithm will be effective when the terms on the diagonal are large and the off diagonal terms are small; indicating that the FGPILU algorithm will perform well for matrices that are diagonally dominant.

5.1 Improving the Convergence of the FGPILU Algorithm

In this work, an investigation is made into the performance of the FGPILU algorithm with respect to more difficult problems. For a given problem, the FGPILU algorithm may fail to converge; i.e., the nonlinear residual norm (see Eq. (2)) fails to decrease below a desired threshold or diverges entirely. Additionally, the structure of the input matrix may preclude unmodified use of the FGPILU algorithm (e.g., due to zeros on the diagonal). If the progression of the algorithm reaches a point where the norm of the Jacobian is greater than one, the fixed point iteration no longer represents a (local) contraction and further sweeps will not help the algorithm make progress towards the desired preconditioning factors. Even if the FGPILU algorithm converges to a set of preconditioning factors, it is possible that, if the system was changed too much to ensure convergence, the preconditioning factors will not aid in the convergence of the associated Krylov solver. In fact, it is possible for the resulting L and U factors to actually slow convergence or prevent convergence entirely (see both Table 2 and (**?**)). In an effort to improve the convergence of the FGPILU algorithm, this study focuses on employing two techniques; both aim to increase the diagonal dominance of the original matrix, which will in turn reduce the norm of the Jacobian and help ensure that the fixed point iteration continues to make progress.

The first technique involves reordering the matrix in order to aid the convergence of the algorithm. Three reorderings are considered here. The first is the `MC64` reordering that attempts to permute the largest entries

of the matrix to the diagonal (**?**), the second is the approximate minimum degree (`AMD`) as implemented in `MATLAB` since it has previously been observed to help convergence of the FGPILU algorithm on non-symmetric problems (Chow and Patel 2015, Benzi, Haws, and Tuma 2000), the other is the Reverse Cuthill-Mckee ordering (`RCM`) which attempts to reduce the bandwidth of the matrix which can potentially aid in the convergence of the FGPILU algorithm and has shown to be effective in the case of symmetric, positive-definite (SPD) matrices (Chow, Anzt, and Dongarra 2015, Chow and Patel 2015, Coleman, Sosonkina, and Chow 2017). After the ordering is applied, the second technique consists of an α-shift that is performed in the manner originally suggested in **?**. Specifically, the original input matrix A can be written, $A = D - B$, where D holds only the diagonal elements of A, and B contains all other elements. Instead of performing the incomplete LU factorization on the original matrix A, the factorization is instead applied to a matrix that is close to A but has an increased level of diagonal dominance. In particular, the incomplete LU factorization can be applied to $\hat{A} = (1+\alpha)D - B$, where $\hat{A} \approx A$ but the size of the diagonal has been increased. This α-shift technique has been used historically for improving the stability of the preconditioning factors generated by conventional incomplete LU factorizations, but given the discussion above in Section 5 concerning the fine-grained incomplete LU factorization that is the subject of this work, it is reasonable to expect this shift to improve the convergence of the FGPILU algorithm. Moreover, since incomplete LU factorizations are by nature, approximate, using the preconditioning factors obtained from applying the FGPILU algorithm to $\hat{A}$ before a Krylov solve of the original matrix A can be expected to accelerate the overall convergence for reasonable values of α. These claims will be explored numerically through the remainder of the paper.

To track the progression of the FGPILU algorithm, a common metric to monitor the progression of the FGPILU algorithm is the so-called nonlinear residual norm (Chow and Patel 2015, Chow, Anzt, and Dongarra 2015, Coleman, Sosonkina, and Chow 2017). This is a value

$$\tau = \sum_{(i,j)\in S} \left| a_{ij} - \sum_{k=1}^{\min(i,j)} l_{ik} u_{kj} \right| , \tag{5}$$

where the set S contains all of the non-zero elements in both L and U, i.e., the non-zero pattern on which the incomplete LU factorization is sought. The nonlinear residual norm decreases as the number of sweeps progresses and the factors produced by the algorithm become closer to the conventional L and U factors that would be computed by a traditional incomplete LU factorization. Because of this it is possible to detect convergence of the algorithm if the nonlinear residual norm is reduced by some pre-specified order of magnitude. However, it should be noted that L and U factors are capable of being used successfully as preconditioning factors (Chow and Patel 2015), and that in practice it may not be necessary to perform very many sweeps of the FGPILU algorithm.

For the purposes of this study, a single fault tolerant variant of the FGPILU algorithm from Coleman, Sosonkina, and Chow 2017 is tested. The variant of the algorithm that was selected monitors the progression of the nonlinear residual norm τ and rejects updates if the update causes the nonlinear residual norm to increase. In particular, the following condition is checked: $\tau^{(sweep)} > \gamma \cdot \tau^{(sweep+r)}$, where the parameter γ controls how monotonic the convergence of the nonlinear residual norm is expected to be, and the parameter r controls how long to delay (in terms of the number of sweeps) in between performing the check.

6 NUMERICAL RESULTS

The test problems that were used in this study are intended to form a representative but not complete set of matrices that are harder to solve than the simpler SPD problems that have been utilized previously. The convergence of the fixed point iteration associated with the FGPILU algorithm displays good convergence with this type of problem (Coleman, Sosonkina, and Chow 2017, Chow, Anzt, and Dongarra 2015, Chow and Patel 2015). However, solving fixed point iterations that feature nonlinear functionals (i.e., in Algorithm 1)

is often difficult, and developing the associated convergence theory is also typically hard to accomplish (see for example: Bertsekas and Tsitsiklis 1989, Ortega and Rheinboldt 2000).

The experimental setup for this study is an NVIDIA Tesla K80 GPU on the Turing high performance cluster at Old Dominion University. The nominal, fault-free iterative incomplete factorization algorithms and iterative solvers were taken from the MAGMA open-source software library (Innovative Computing Lab 2015), and minimal modifications were made to the existing MAGMA source code in order to implement the modifications to the FGPILU algorithm, add the α-shift, and to inject faults into the algorithm. All of the results provided in this study reflect double precision, real arithmetic. The test matrices used here come from a variety of sources. The first comes from the seminal work on the performance of incomplete LU factorization for indefinite matrices (Chow and Saad 1997), `fs_760_3`. The next matrix comes from the domain of circuit simulation, `ecl32`, and has been studied previously (**?**, **?**). The last matrix comes from the set of 8 SPD matrices that were studied in previous works on the FGPILU algorithm (Chow, Anzt, and Dongarra 2015, Coleman, Sosonkina, and Chow 2017, Chow and Patel 2015), and is the matrix among those eight with the largest condition number (as estimated by `MATLAB`'s `CONDEST` function); 'offshore'. Condition numbers for the 8 previously studied SPD problems range from 1.11e+03 to 2.24e+13. A brief summary of all three matrices is provided in Table 1. The matrices that are presented here attempt to give some indication as to the performance of the nonlinear fixed point iteration associated with the FGPILU with respect to matrices that are more challenging computationally than the problems that are featured in the majority of the previous work on the algorithm (i.e. Chow and Patel 2015, Chow, Anzt, and Dongarra 2015, Coleman, Sosonkina, and Chow 2017).

Table 1: Characteristics of the matrices used: Column `Sym?` reflects the symmetry, `PD?` provides positive-definiteness, `Dim`—number of rows, and `Non-zeros`–number of non-zeros in each matrix.

Matrix Name	Abbr.	Sym?	PD?	`CONDEST`	Dim.	Non-zeros	Description
fs_760_3	FS	N	N	9.93E+19	760	5,816	chemical engineering
ecl32	ECL	N	N	9.41E+15	51,993	380,415	circuit simulation
OFFSHORE	OFF	Y	Y	2.24E+13	259,789	4,242,673	electric field diffusion

6.1 Fault Model

In this study, faults are modeled as perturbations similar to several recent studies (Coleman and Sosonkina 2016, Coleman, Sosonkina, and Chow 2017, Stoyanov and Webster 2015); the goal being producing fault tolerant algorithms for future computing platforms that are not too dependent on the precise mechanism of a fault (e.g. bit flip). Modifying the perturbation-based fault model described in Coleman, Sosonkina, and Chow 2017, a single data structure is targeted and a small random perturbation is injected into each component transiently. For example, if the targeted data structure is a vector x and the maximum size of the perturbation-based fault is ϵ, then proceed as follows: generate a random number $r_i \in (-\epsilon, \epsilon)$, and then set $\hat{x}_i = x_i + r_i$ for all values of i. The resultant vector $\hat{x}$ is, thus, perturbed away from the original vector x. In this study, faults are injected into the FGPILU algorithm following the perturbation based methodology described above. Due to the relatively short execution time of the FGPILU algorithm on the given test problems, a fault is induced only once during each run, and the fault is designated to occur at a random sweep number before convergence. Three ranges for faults injected by the perturbation-based model were considered: $r_i \in (-0.01, 0.01), r_i \in (-1, 1)$, and $r_i \in (-100, 100)$. Due to the non-deterministic nature of the fault model, multiple runs are conducted for each value of r_i and the data is averaged in order to obtain representative results.

6.2 Convergence of the FGPILU fixed point iteration and associated Krylov solver

In these fault-free experiments, the convergence of the FGPILU algorithm is examined for three different levels (0,1, and 2) of the incomplete LU factorization (see Benzi 2002 or Saad 2003 for a clear description of levels of incomplete LU factorizations), and three different values of α in the α-shift. Note that regardless of the ordering being utilized, all runs start with a symmetrically scaled matrix such that the entries on the diagonal are less than or equal to 1. As such, appropriate values for α range from 0 to 1 and in this study three discrete values were selected: 0, 0.5, 1.0. More extreme values for α can help improve the convergence of the FGPILU algorithm by increasing the diagonal dominance of the matrix that the FGPILU algorithm is applied to, but this comes at the expense of preparing the preconditioner for a problem increasingly less related to the original problem. As an example, for the `OFFSHORE` problem with `AMD` ordering and symmetric scaling, the FGPILU algorithm converges in a smaller number of sweeps for increasing values of α, but the overall performance of the Krylov subspace solver deteriorates. Details are provided in Table 2.

Table 2: Effects of increasing α for the `OFFSHORE` problem.

α	FGPILU Sweeps	Krylov solver iterations	Krylov solver time
0	24	30	24.8067
1	9	56	46.4995
10	5	144	130.0958

For each of the three matrices that were tested: four orderings were tested (`MC64, AMD, RCM`, and the natural ordering), 3 level of ILU fill-in were tested (levels 0, 1, and 2), and 3 factors for α were used (0, 0.5, and 1.0). This leads to a total of 108 permutations to test. Of these 108 combinations, 84 (77.78%) led to a case were the FGPILU algorithm converged, but only 29 (26.85%) resulted in a successful GMRES solve of the entire linear system using a restart parameter of 50 and a tolerance of 1e-10. Details for those 29 cases are provided below in Table 3.

Table 3: Successful runs with their parameter combinations.

Matrix	**Ordering**	α	**ILU Level**	**Sweeps**	**Krylov Its.**	**Time (s)**
offshore	AMD	0	0	19	30	18
offshore	AMD	0.5	0,1,2	10,11,11	40,34,34	24,55,144
offshore	AMD	1	0,1,2	8,9,9	56,54,54	34,96,229
offshore	RCM	0	0	19	19	35
offshore	RCM	0.5	0,1,2	10,11,11	37,34,34	68,306,771
offshore	RCM	1	0,1,2	9,9,9	54,54,54	101,484,1226
offshore	Natural	0	0	22	22	84
offshore	Natural	0.5	0,1,2	11,12,12	38,34,34	146,312,695
offshore	Natural	1	0,1,2	9,10,10	54,54,54	210,491,1104
ecl32	AMD	0	2	15	127	104
ecl32	RCM	0	2	24	9	39
ecl32	Natural	0	2	18	11	16
fs_760_3	AMD	0	2	55	3	0.4
fs_760_3	RCM	0	1,2	52,63	2,2	0.4,0.4
fs_760_3	MC64	0	1	16	3	0.3
fs_760_3	Natural	0	1	16	3	0.3

In general, higher levels of fill are capable of producing better preconditioning factors (Benzi, Haws, and Tuma 2000, Chow and Saad 1997), but come at the cost of increased storage and computational costs. There is an inherent trade-off in using higher fill levels to produce incomplete factors that are closer to the full L

and U factors that must be evaluated. A few other general observations: the two non-symmetric problems tend to perform better with smaller values of α and higher levels of fill-in allowed, and the level of ILU fill-in tends to not have as much of an impact on whether or not the problem can be solved when compared to the ordering or value for α, but affects the performance. In the results found here, the benefit of having more complete L and U factors from going to a higher fill-in level tends to be outweighed by the increased computational cost of the fixed-point iteration associated with the FGPILU algorithm for a drastically larger number of elements. As an example of the drastic increase in the number of non-zero elements for each of the matrices, consider the data in Table 4.

Table 4: Increase in non-zeros for different levels of ILU fill-in.

Matrix	**nnz(ILU-0)**	**nnz(ILU-1)**	**nnz(ILU-2)**
offshore	4.502E+06	9.974E+06	2.172E+07
ecl32	4.324E+05	9.473E+05	1.954E+06
fs_760_3	6.576E+03	1.757E+04	3.231E+04

6.3 Resilience of the FGPILU fixed point iteration

The experiments conducted in this section reflect the resilience of the FGPILU algorithm with respect to transient soft faults. Resilience is provided by checkpointing and restoring prior data based on the progression of the nonlinear residual norm. To illustrate the resilience of the FGPILU algorithm, only combinations of ordering, ILU-level, and α from Section 6.2 that were successful in the fault-free scenario have been selected for experimentation. A single set of parameters for the checkpointing check, $\tau^{(sweep)} > \gamma \cdot \tau^{(sweep+r)}$, is used. Both γ and r were set to one so that a strict check on the monotonicity of the nonlinear residual norm is performed after every sweep. For SPD problems, this level of check may be unnecessary (Coleman, Sosonkina, and Chow 2017), but this provides the maximum level of protection for the FGPILU algorithm and provides a measure of how effective this check can be for the more difficult problems under investigation in this study. A summary of the data found in these experiments is provided in Table 5, which depicts the

Table 5: Solver performance using FGPILU with no fault tolerance (`NoFT`) and checkpointing (`CP`) .

Scenario	Success Rate (NoFT)	Success Rate (CP)	Timing Ratio	Sweeps Ratio	Its. Ratio
Total	46.65%	100.00%	1.02	0.63	1.01
Small fault	88.59%	100.00%	1.03	0.69	1.03
Medium fault	42.94%	100.00%	1.01	0.48	1.00
Large fault	14.71%	100.00%	1.00	0.73	0.99

percentage of runs that succeeded—resulted in a successful linear system solve—subject to faults (column `Scenario`), when no fault tolerance (column `NoFT`) and the checkpointing FGPILU variant (column `CP`) were employed, respectively. Three ratios of the results with `CP` and `NoFT` are shown in Table 5 as `Timing`, `Sweeps`, and `Its`, defining the timing increase, reduction in the total number of sweeps needed, and the change in the GMRES iterations, respectively. The checkpointing algorithm mitigates well the potential impact of a fault. Note that the largest benefit comes from correcting the impact of a large fault. Smaller faults—which cause effects similar to those produced by bit flips in a less significant bit of the mantissa—tend to be corrected naturally by the iterative nature of the fixed-point iteration. Another important factor in comparing any fault tolerance methods is quantifying how much overhead they introduce. Due to the non-deterministic block-asynchronous nature of the GPU implementation of the FGPILU algorithm in the absence of faults and the inherent randomness involved in the fault model utilized in this study, it is difficult to compare individual cases. However, comparing runs utilizing the same parameters over all cases where both the fault-free variants and the checkpointing variant solved the linear system successfully, there is about a 2% increase in the time required to reach a solution in order to provide fault tolerance to the

FGPILU algorithm using this methodology. There is more of an impact on cases with small faults since it is often possible for the iterative nature of the algorithm to correct the impact of a sufficiently small fault. Note that varying the parameters γ and r that determine the frequency and strictness of the check could change both the efficiency and efficacy of the checkpointing variant of the FGPILU.

7 CONCLUSION & FUTURE WORK

This study has presented some experiments and analysis concerning the convergence of the fine-grained parallel incomplete LU factorization proposed in Chow and Patel 2015 with respect to more difficult problems than have previously been studied. Additionally, initial work concerning the resilience of the FGPILU algorithm with respect to transient soft faults has been explored for this same difficult problem set. Moving forward, further adaptations to the FGPILU algorithm are possible. This series of experiments has worked with various levels of ILU factorization, while a variant of ILUT that takes advantage of fine-grained parallelism called ParILUT (**?**) has been proposed recently, and previous work on the performance of incomplete factorizations for the solution of indefinite and non-symmetric problems (Chow and Saad 1997, Benzi, Haws, and Tuma 2000) has indicated that different styles of incomplete factorization may be more effective for these classes of problems. This suggests that future work on fine-grained preconditioners should include recent developments in fine-grained factorizations when possible. It would be also helpful to compare the performance of various solvers, such as Bi-CGSTAB and TFQMR, with the preconditioning factors that are generated by the FGPILU algorithm. Another avenue for future exploration includes expanding the experimentation conducted with respect to various fault tolerance techniques. In particular, the checkpointing scheme proposed here could be further analyzed in an attempt to optimize the amount of computational overhead induced as opposed to the increase in convergence rate.

ACKNOWLEDGMENTS

This work was supported in part by the Air Force Office of Scientific Research under the AFOSR award FA9550-12-1-0476, by the U.S. Department of Energy (DOE), Office of Advanced Scientific Computing Research Exascale Computing Project, under the grant DE-SC-0016564 through the Ames Laboratory, operated by Iowa State University under contract No. DE-AC00-07CH11358, by the U.S. Department of Defense High Performance Computing Modernization Program, through a HASI grant, and through the ILIR/IAR program at the Naval Surface Warfare Center - Dahlgren Division. This work was also supported by the Turing high performance computing cluster at Old Dominion University.

REFERENCES

Addou, A., and A. Benahmed. 2005. "Parallel synchronous algorithm for nonlinear fixed point problems". *International Journal of Mathematics and Mathematical Sciences* vol. 2005 (19), pp. 3175–3183.

Anzt, H., E. Chow, and J. Dongarra. 2015. "Iterative sparse triangular solves for preconditioning". In *European Conference on Parallel Processing*, pp. 650–661. Springer.

Anzt, H., E. Chow, J. Saak, and J. Dongarra. 2016. "Updating incomplete factorization preconditioners for model order reduction.". *Numerical Algorithms* vol. 73 (3), pp. 611–630.

Anzt, H., J. Dongarra, and E. S. Quintana-Ortí. 2015. "Tuning stationary iterative solvers for fault resilience". In *Proceedings of the 6th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, pp. 1. ACM.

Asanovic, K., R. Bodik, B. Catanzaro, J. Gebis, P. Husbands, K. Keutzer, D. Patterson, W. Plishker, J. Shalf, S. Williams et al. 2006. "The landscape of parallel computing research: A view from Berkeley". Techni-

cal report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley.

Baudet, G. M. 1978. "Asynchronous iterative methods for multiprocessors". *Journal of the ACM (JACM)* vol. 25 (2), pp. 226–244.

Benahmed, A. 2007. "A convergence result for asynchronous algorithms and applications". *Proyecciones (Antofagasta)* vol. 26 (2), pp. 219–236.

Benzi, M. 2002. "Preconditioning techniques for large linear systems: a survey". *Journal of computational Physics* vol. 182 (2), pp. 418–477.

Benzi, M., J. C. Haws, and M. Tuma. 2000. "Preconditioning highly indefinite and nonsymmetric matrices". *SIAM Journal on Scientific Computing* vol. 22 (4), pp. 1333–1353.

Bertsekas, D. P., and J. N. Tsitsiklis. 1989. *Parallel and distributed computation: numerical methods*, Volume 23. Prentice hall Englewood Cliffs, NJ.

Bridges, P., K. Ferreira, M. Heroux, and M. Hoemmen. 2012. "Fault-tolerant linear solvers via selective reliability". *arXiv preprint arXiv:1206.1390*.

Cappello, F., A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir. 2009. "Toward exascale resilience". *The International Journal of High Performance Computing Applications* vol. 23 (4), pp. 374–388.

Cappello, F., A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir. 2014. "Toward exascale resilience: 2014 update". *Supercomputing frontiers and innovations* vol. 1 (1).

Chow, E., H. Anzt, and J. Dongarra. 2015. "Asynchronous iterative algorithm for computing incomplete factorizations on GPUs". In *International Conference on High Performance Computing*, pp. 1–16. Springer.

Chow, E., and A. Patel. 2015. "Fine-grained parallel incomplete LU factorization". *SIAM journal on Scientific Computing* vol. 37 (2), pp. C169–C193.

Chow, E., and Y. Saad. 1997. "Experimental study of ILU preconditioners for indefinite matrices". *Journal of Computational and Applied Mathematics* vol. 86 (2), pp. 387–414.

Coleman, E., A. Jamal, M. Baboulin, A. Khabou, and M. Sosonkina. 2017. "A Comparison and Analysis of Soft-Fault Error Models using FGMRES and ARMS RBT". In *Proceedings of the 12th International Conference on Parallel Processing and Applied Mathematics*. ACM.

Coleman, E., and M. Sosonkina. 2016. "Evaluating a Persistent Soft Fault Model on Preconditioned Iterative Methods". In *Proceedings of the 22nd annual International Conference on Parallel and Distributed Processing Techniques and Applications*.

Coleman, E., and M. Sosonkina. 2017. "Fault Tolerance for Fine-Grained Iterative Methods". In *Proceedings of the 7th annual Virginia Modeling, Simulation, and Analysis Center Capstone Conference*. Virginia Modeling, Simulation, and Analysis Center.

Coleman, E., M. Sosonkina, and E. Chow. 2017. "Fault Tolerant Variants of the Fine-Grained Parallel Incomplete LU Factorization". In *Proceedings of the 2017 Spring Simulation Multiconference*. Society for Computer Simulation International.

Elliott, J., M. Hoemmen, and F. Mueller. 2014. "Evaluating the impact of SDC on the GMRES iterative solver". In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pp. 1193–1202. IEEE.

Elliott, J., M. Hoemmen, and F. Mueller. 2015. "A Numerical Soft Fault Model for Iterative Linear Solvers". In *Proceedings of the 24nd International Symposium on High-Performance Parallel and Distributed Computing*.

Frommer, A., and D. B. Szyld. 2000. "On asynchronous iterations". *Journal of computational and applied mathematics* vol. 123 (1), pp. 201–216.

Geist, A., and R. Lucas. 2009. "Major computer science challenges at exascale". *International Journal of High Performance Computing Applications*.

Hoemmen, M., and M. Heroux. 2011. "Fault-tolerant iterative methods via selective reliability". In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC). IEEE Computer Society*, Volume 3, pp. 9. Citeseer.

Innovative Computing Lab 2015. "Software distribution of MAGMA". http://icl.cs.utk.edu/magma/.

Ortega, J. M., and W. C. Rheinboldt. 2000. *Iterative solution of nonlinear equations in several variables*. SIAM.

Saad, Y. 2003. *Iterative methods for sparse linear systems*. SIAM.

Saad, Y., and B. Suchomel. 2002. "ARMS: An algebraic recursive multilevel solver for general sparse linear systems". *Numerical linear algebra with applications* vol. 9 (5), pp. 359–378.

Sao, P., and R. Vuduc. 2013. "Self-stabilizing iterative solvers". In *Proceedings of the Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, pp. 4. ACM.

Snir, M., R. Wisniewski, J. Abraham, S. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson et al. 2014. "Addressing failures in exascale computing". *International Journal of High Performance Computing Applications*.

Stoyanov, M., and C. Webster. 2015. "Numerical analysis of fixed point algorithms in the presence of hardware faults". *SIAM Journal on Scientific Computing* vol. 37 (5), pp. C532–C553.

AUTHOR BIOGRAPHIES

EVAN COLEMAN is a lead scientist with the Naval Surface Warfare Center Dahlgren Division. He holds an MS in Mathematics from Syracuse University and is working on a PhD in Modeling and Simulation from Old Dominion University. His email address is ecole028@odu.edu.

MASHA SOSONKINA is a Professor of Modeling, Simulation and Visualization Engineering at Old Dominion University. Her research interests include high-performance computing, large-scale simulations, parallel numerical algorithms, and performance analysis. Her email address is msosonki@odu.edu.