EasyChair Preprint

№ 4382

Handling Soft Errors in Krylov Subspace Methods by Exploiting Their Numerical Properties

Muhammed Emin Ozturk, Gagan Agrawal, Yukun Li and Ching-Shan Chou

October 12, 2020

Handling Soft Errors in Krylov Subspace Methods by Exploiting Their Numerical Properties

Muhammed Emin Ozturk, University of Utah
Gagan Agrawal, Augusta University
Yukun Li, University of Central Florida
Ching-Shan Chou, Ohio State University

***Abstract*— Krylov space methods are a popular means for solving sparse systems. In this paper, we consider three such methods: GMRES, Conjugate Gradient (CG) and Conjugate Residual (CR). We focus on the problem of efficiently and accurately detecting soft errors leading to silent data corruption (SDC) for each of these methods. Unlike limited amount of previous work in this area, our work is driven by analysis of mathematical properties of the methods. We identify a term we refer to as *energy norm*, which is decreasing for our target class of methods. We also show other applications of *error norm* and *residual value*, and expand the set of algorithms to which they can be applied. We have extensively evaluated our method considering three distinct dimensions: accuracy, magnitude of undetected errors, and runtime overhead. First, we show that our methods have high detection accuracy rate. We gain over 90% detection rate for GMRES in most of the scenario and matrices. For most case in CG and CR, we gain over 70% detection rate as well. Second, we show that for soft errors that are not detected by our methods, the resulting inaccuracy in the final results are small. Finally, we also show that the run-time overheads of our method are low.**

Keywords – Fault-tolerance, Soft errors, Iterative Solvers, Krylov Subspace, GMRES, Conjugate Gradients, Conjugate Residual

I. Introduction

As we are approaching the Exascale era, there is a growing emphasis on resilience. One of the reasons is that with increasing number of nodes (or cores), the likelihood of a failure increases. Together with factors like the use of smaller feature-size [23], we are seeing greater likelihood of *soft errors*, which involve bit flips leading to silent data corruption (SDC).

There has been considerable recent interest in both detecting soft errors at the software level, and making software resilient against such errors [19], [16], [12]. However, most of the techniques presented in the literature to date do not consider the specific numerical properties of the underlying methods. For example, redundancy has been used as a promising solution [8], [17] where one duplicates the execution and crosschecks that state across different instances. Clearly, such an approach is expensive in terms of the resources required.

In this paper, we focus on a class of numerical methods referred to as *Krylov subspace* methods. These methods are designed to solve large sparse matrix eigenvalue problems or a large linear system of equations, both of which are among the most important problems in scientific computing. *Krylov subspace methods* play a crucial role in how we can handle large, sparse, and non-symmetric matrix problems. Examples of such methods include Conjugate Gradient (CG), Conjugate Residual (CR), and Generalized Minimum Residual (GMRES). So far, there has only been a limited amount of work on addressing the problem of soft error detection for these methods [2], [11], [2], [11]. Most of these solutions have been heuristic in nature, and do not build on top of the analysis of numerical properties of these methods. For example, multiple efforts have used analysis of residual values across iterations as an indicator of soft errors [2], [11] in the case of CG. However, the value of residual is not strictly monotonically decreasing for CG, and this results in an indicator of limited accuracy. In the case of GMRES, Bridges *et al.* [20] developed FT-GMRES (Fault Tolerant GMRES), which involves a different algorithm, and thus the approach is hard to apply to an existing code.

This paper revisits existing literature on the analysis of these methods [25], [4], [24] and identifies properties that lead to efficient and accuracy detectors of soft errors. Particularly, we identify a term we refer to as *energy norm*, which is monotonically decreasing for our target class of methods. We also show other applications of *error norm* and *residual value*, and expand the set of algorithms to which they can be applied. We add additional heuristics to these methods to develop an overall methodology.

We have extensively evaluated our proposed methods using several real matrices. Our evaluation shows high detection accuracy (especially for GMRES and CG). Moreover, we find that average error due to undetected errors is small, indicating that most significant errors get detected. Finally, we see low overheads of applying our method.

II. Related Work

There have been certain efforts on the detection of soft errors in iterative methods for linear systems, as we will describe in this section. However, to the best of our knowledge, detectors covering the space of Krylov

subspace solvers that exploit their numerical properties have not been developed.

Bronevetsky *et al.* proposed that the *residual* that is normally used to check convergence can be used as a soft error detector by observing abnormal value changes across iterations [2]. They performed experiments with several iterative solvers, including CG, which is one of the well known Krylov Subspace Methods. However, they did not apply this approach on GMRES or CR. Moreover, the Residual method applied on CG is just a heuristic, because the residual value is not strictly monotonically decreasing in CG. Similarly, in Liu *et al.* [11], the impact of soft errors was evaluated on five applications, including CG. Recently, Ozturk *et al.* [18] developed a function that has monotonic decreasing property and thus is able to detect soft errors in CG.

Sao *et al.* [22] developed a methodology for iterative solvers such as steepest descent (SD) and conjugate gradient (CG) to make them fault tolerant. They designed self-stabilizing versions of these methods, which are resilient to soft errors. However, their approach requires rewriting an application with a different algorithm, and is not applicable towards making an existing code resilient. There has been limited work proposed regarding for fault tolerance related work in GMRES. Bridges *et al.* [20] proposed FT-GMRES (Fault Tolerant GMRES) – however, the approach also requires rewriting an application with different algorithm to make it resilient towards soft errors. Recently, Pachajoa *et al.* [19] proposed a node failure-tolerant preconditioned conjugate gradient method that is capable of recovering from node failures without additional spare nodes.

Another series of efforts apply machine learning approach in soft error detection. Liu *et al.* combine on-line detection with off-line training [14]. Their proposed method uses machine learning and is effective when the residual (or another term) is not known to be monotonically decreasing under normal execution. However, there is a substantial overhead of off-line training.

III. Background

Large sparse matrix eigenvalue problems or large linear system of equations are a critical part of scientific computing and practice of science and engineering. *Krylov subspace methods* are regarded as one of the ten most important classes of numerical methods for solving large sparse matrix problems [6], as they play a crucial role in how we can handle large, sparse, and non-symmetric matrix problems. This section provides an overview of the Krylov subspace and Krylov subspace methods.

A. Krylov Subspace and Krylov Subspace Methods

Given an $n \times n$ matrix A, and a vector of length n (b), Krylov subspace is is the subspace spanned by the vectors of the Krylov sequence:

$$K_m(A, b) = span\{b, Ab, A^2b, \ldots, A^{m-1}b\}$$

The meaning of $span$ here is that every vector in K_m can be defined by the linear combination of the basis {b, Ab, $A^2b, \ldots, A^{m-1}b$} [10]. A Krylov subspace method for solving a linear system $Ax = b$, or a *Krylov space solver* is an iterative method that starts with an initial approximation x_0 and a *residual* r_0 calculated as $r_0 = b - Ax_0$. At the m-th step, an approximation the solution of $Ax = b$, i.e, x_m, is found in x_0 + K_m. Till reaching a possible exact solution, it iterates x_n such that

$$x_n - x_0 = q_{n-1}(A)r_0 \in K_{n+1}(A, r_0)$$

where q_{n-1} is a polynomial of degree at most $n-1$[10].

To solve a linear system, well-known simple iterative methods such as Jacobi or Gauss elimination might be considered. However, those methods are not efficient when we solve large and highly sparse matrices since they have $O(n^3)$ time complexity. Moreover, Krylov subspace methods are also applicable when those matrices are indefinite and asymmetrical. The well-known Krylov subspace methods are Arnoldi, Conjugate Residual (CR), Conjugate Gradient(CG), Generalized Minimum Residual (GMRES), Biconjugate Gradient (Bi-CG), Quasi Minimal Residual (QMR), and Minimal Residual (MINRES). We focus on CR, CG and GMRES in this paper.

When deciding which Krylov subspace solver to use, we need to consider properties of matrix. If the matrix is Symmetric Positive Definite (SPD), Conjugate Gradient is considered as best candidate for solving large linear systems in terms of workload. However, if the matrix is not symmetric then GMRES should be first choice due to its effectiveness. Fig.1 represents the tree for conditions in which Krylov Subspace methods that we can apply.

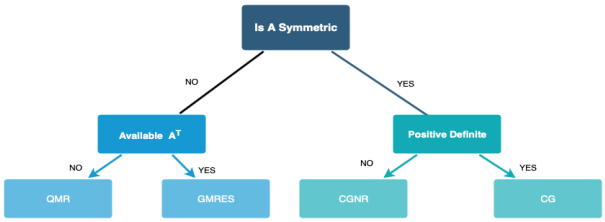

Fig. 1: Decision tree for Choosing Krylov Subspace methods

B. Generalized Mininal Residual (GMRES)

The Generalized Minimal Residual method has been proposed by Saad and Schults [26]. GMRES can be applied to an unsymmetrical system and it proceeds by creating a sequence of orthogonal vectors. GMRES is the most robust method among Krylov Subspace methods although it is expensive in terms of the number of operations. There are many ways to implement the GMRES Algorithm. In the paper, we use a variation of the GMRES Algorithm (Algorithm 1) that utilizes

the Modified Gram-Schmidt orthogonalization in the Arnoldi process [25].

In Algorithm 1, the prim vector of Krylov Subspace is initialized (line 1-3) and the next vector to span Krylov Subspace is generated (line 6). Arnoldi method is applied (line 7-17) – this is an iterative solver also known as orthogonal projection method that is used for building an orthogonal basis of the Krylov subspace K_m.

Algorithm 1 GMRES

```
1: Ax=b linear system and initial vector x_0.
2: r_0 = b − Ax_0 /*Residual Vector*/
3: β = ||r_0||_2
4: k = 0
5: while not converged do
6:     v_{j+1} = Aq_j
7:     /* Orthogonalize (Arnoldi Process) */
8:     while i < j do
9:         h_{i,j} = q_i * v_{j+1}
10:        v_{j+1} = v_{j+1} − h_{i,j} q_i
11:    end while
12:    h_{j+1,j} = ||v_{j+1}||_2
13:    if h_{j+1,j} == 0 then
14:        Solution is x_{j−1}
15:        Return
16:    end if
17:    q_{j+1} = v_{j+1}/h_{j+1,j}
18:    y_j = argmin||H(1 : j + 1, 1 : j)y − βe_1||_2
19:    k = k + 1
20: end while
21: return x_k
```

C. Conjugate Gradient

Conjugate gradient (CG) solves linear system $Ax = b$ where A is symmetric positive definite (SPD). It was initially proposed by Hestenes and Stiefel in 1952 [9]. CG calculates successive approximations $x_k = \alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_k p_k$ to the solution vector. The algorithm continues until the residual $r_k = b - Ax_k$ is sufficiently small.

Algorithm 2 Conjugate Gradient

```
1: Choose initial vector x_0.
2: r_0 = b − Ax_0
3: p_0 = r_0
4: k = 0
5: while not converged do
6:     α_k = (r_k^T r_k) / (p_k^T A p_k)
7:     x_{k+1} = x_k + α_k p_k
8:     r_{k+1} = r_k − α_k A p_k
9:     β_k = (r_{k+1}^T r_{k+1}) / (r_k^T r_k)
10:    p_{k+1} = r_{k+1} + β_k p_k
11:    k = k + 1
12: end while
13: return x_k
```

The pseudo-code given as Algorithm 2 starts out with an initial approximation x_0 (usually a zero vector). In line 2, the residual r_0 represents the direction of the error in the initial result Ax_0. The initial residual is also used as the initial direction p_0 in which to search for a new solution (line 2). In each iteration, the algorithm searches in the direction p_k for a new solution vector x_k (line 10). The distance in which to move in the direction p_k is given by the term α_k (line 6). The residual and search direction vectors are then updated (lines 8-11).

D. Conjugate Residual

The conjugate residual method was also introduced by Hestenes and Stiefel (1952) and Stiefel (1955), even if they did not mention the method's name in their first paper [9]. CR is an iterative, polynomial-based algorithm where x_i minimizes error norm $\|e_k\|_2 = \|f - Ax_i\|_2$ over the i-dimensional Krylov space [25].

$$K_i(A, r_0) = span\{b, Ar_0, A^2 r_0, \ldots, A^{i-1} r_0\}$$

It can also be defined as different derivation of GMRES for specific case where A is a Hermitian matrix [21]. CR is closely related to the CG method with similar construction and convergence properties. Like CG, it computes x at most over n steps. However, it requires more storage and more numerical operations than CG [21]. The required work per loop in CR is $6n$ multiplication operations and one matrix-vector product and required storage for the vectors is $5n$ [21]. The other fundamental difference from Conjugate Gradient is that CR can be used for matrices that are symmetric but not necessarily positive definite [25]. Calculation of α and β is only differences from the implementation of Conjugate Gradient as it seen in Algorithm 5 (line 9-13).

IV. Methodology for Detecting Silent Data Corruption

We now describe our proposed set of approaches, which are all based on theoretical characteristics of Krylov Subspace methods. It should be noted that we focus only on detecting errors and not on recovering from them. After the detection process is completed, recovery can be done by using low-cost application-level checkpoints as Liu *et al.* indicated in their work [13]. Similarly, our presentation and implementation assumes execution on a single machine – the idea can be applied to a larger-scale execution without changes (and SDCs become more likely as the number of cores increases).

A. Mathematical Characteristics of Krylov Subspace Methods

In this section, we explain the mathematical background of our proposed approach and illustrate significant features of Krylov Supspace methods that enable us to detect soft errors. The terms used correspond to the algorithms presented in the last section.

- **Theorem 1.1** (2.4 in [4])
  For Conjugate Residual and MINRES on an SPD system $Ax = b$, the error $\|e_m\|_2$ or $\|x * - x_k\|_2$ is monotonically decreasing

Proof is based on [4].

We use the terminology used in the previous section.

$$x_m = x_{m-1} + \alpha_{m-1}p_{m-1}$$
$$= .....$$
$$= x_k + \alpha_{k+1}p_k + \ldots + \alpha_{m-1}p_{m-1}$$
$$= x_{k-1} + \alpha_k p_{k-1} + \alpha_{k+1}p_k + \ldots + \alpha_{m-1}p_{m-1}$$

From the last two statements above, it can be written that:

$$\|x_m - x_{k-1}\|^2 - \|x_m - x_k\|^2 = ...$$
$$= (x_m - x_{k-1})^T(x_m - x_{k-1}) - (x_m - x_k)^T(x_m - x_k)$$
$$= 2\alpha_k p_{k-1}^T(\alpha_{k+1}p_k + \ldots + \alpha_{m-1}p_{m-1}) + \alpha_k^2 p_{k-1}^T p_{k-1}$$
$$>= 0$$

since Theorem 2.2 in [4] indicates that $\alpha_i >= 0$ and $p_i^T p_j >= 0$, the *energy norm error* $\|x^* - x_k\|_A$ is strictly decreasing in CR and MINRES. On the other hand, for CG, the same term is monotonically decreasing [4].

- **Theorem 1.2** (2.5 in [4])
  For Conjugate Residual and MINRES on an SPD system $Ax = b$ the error $\|e_m\|_A$ or Energy Norm Error $\|x^* - x_k\|_A$ is strictly decreasing.

From the proof above, it can be written that:

$$\|x_m - x_{k-1}\|_A^2 - \|x_m - x_k\|_A^2 = ...$$
$$... = (x_m - x_{k-1})^T A(x_m - x_{k-1}) - (x_m - x_k)^T A(x_m - x_k)$$
$$... = 2\alpha_k p_{k-1}^T A(\alpha_{k+1}p_k + ... + \alpha_{m-1}p_{m-1}) + \alpha_k^2 p_{k-1}^T A p_{k-1}$$
$$... = 2\alpha_k p_{k-1}^T(\alpha_{k+1}p_k + ... + \alpha_{m-1}p_{m-1}) + \alpha_k^2 p_{k-1}^T p_{k-1}$$
$$> 0$$

B. Exploiting Minimization Property

In CG, when we use a SPD A matrix, the *quadratic form* $\phi = \frac{1}{2}x^T Ax - x^T b$ is bounded and $Ax = b$ is calculated by minimizer of that quadratic form. As a Krylov subspace method, CG has the minimization characteristic of any Krylov subspace method, i.e., to minimize the quadratic form within each Krylov subspace [4]. When $b = Ax^*$ and $2\phi = x_k^T Ax_k - 2x_k^T Ax^*$, minimizing quadratic form is equal to minimizing the expression $\|x^* - x_k\|_A = (x^* - x_k)^T A(x^* - x_k)$ [4]. Then the A-norm of the CG error is *monotonically decreasing* [24].

Ozturk *et al.* [18] developed algorithmic method based on this property, that is, monotonically decreasing function $f_k = x_k^T(-r_k - b)$ which we will refer to as *Error Norm*.

1) **Energy Norm Detection Method:** Although error norm detection method gives reasonable performance in soft error detection, it has high time complexity, i.e., $O(n)$. In this paper, we propose new methodology for Conjugate Gradient based on minimization property of Krylov subspace methods. The specific term is $J_k = \alpha_k r_k^T r_k$. Computationally, detection based on this term has significantly lower overhead than the error norm indicator, that is $O(1)$ since $r_k^T r_k$ is already computed as part of implementation of CG (Line 11, Algorithm 3). We call this approach *Energy Norm Indicator* as we use the property that energy norm is strictly decreasing (Theorem 1.1).

$$\|e_k\|_A^2 = e_k^T A e_k$$
$$= r_k^T (A^{-1})^T r_k$$

Note that A is SPD. Therefore it is equal to its transpose. Then,

$$\|e_k\|_A^2 = r_k^T A^{-1} r_k$$

The differences of energy norm $\|e_k\|_A^2 - \|e_{k+1}\|_A^2 = J_k$ is a decreasing function.

$$\|e_k\|_A^2 - \|e_{k+1}\|_A^2 = r_k^T A^{-1} r_k - r_{k+1}^T A^{-1} r_{k+1}$$
$$... = (r_{k+1} + \alpha_k A^T p_k)^T A^{-1}(r_{k+1} + \alpha_k A p_k) - r_{k+1}^T A^{-1} r_{k+1}$$
$$... = r_{k+1}^T A^{-1} r_{k+1} + \alpha_k p_k^T r_{k+1} + \alpha_k r_{k+1} p_k + \alpha_k^2 p_k^T A p_k - r_{k+1}^T A^{-1} r_{k+1}$$

$$\|e_k\|_A^2 - \|e_{k+1}\|_A^2 = \alpha_k r_k^T r_k$$

where the last equality comes from the fact that $p_k A p_k = \frac{r_k^T r_k}{\alpha_k}$ and $r_{k+1}^T p_k = 0$. Notice $r_{k+1} p_k = 0$ can be proved by the spanned Krylov supspace property and the orthogonality of the residues, or by using $p_{k+1} = r_{k+1} + \beta_k p_k$ recursively.

Next, we consider,

$$J_k = \alpha_k r_k^T r_k$$

J_k function is a decreasing but not monotonically decreasing so it is prone to false positives. However, it is better than the error norm indicator in terms of time complexity since its time complexity is $O(1)$, as noted earlier.

2) **Monotonic Residual Detection Method:** One of the fundamental properties of certain algorithms in Krylov Subspace is that the residual norm of the approximate solution at each iteration is monotonically decreasing. Recently certain algorithms in iterative solvers, particularly the Conjugate Gradient, have applied expected (near-) monotonic decrease in the residual as error detection method [11]. However, in the case of CG, the method is only a heuristic since residual is not guaranteed to being strictly monotonic in CG for a SPD system.

The fundamental idea of GMRES is to minimize the residual $\|b - Ax_n\|_2$ at the n-th iteration where x_m is a vector in the Krylov space Unlike Conjugate Gradient, the residual decreases monotonically in GMRES since the corresponding Krylov spaces are nested and algorithm itself minimizes the residual directly [25]. Formally,

$$\|r_{m+1}\|_2 < \|r_m\|_2$$

The implementation of detection method based on residual is given as Algorithm 4. As noted earlier, in Conjugate Gradient, the decreasing of $\|r_m\|_2$ is not

Algorithm 3 CG with Detection by Energy Norm and Error Norm with Sharp Decreases

1: Choose initial vector x_0.
2: $r_0 = b - Ax_0$
3: $p_0 = r_0$
4: $k = 0$
5: Choose targeted variable $(p||r||x)$
6: $m =$ targeted iteration
7: **while** not converged **do**
8: **if** $k = m$ **then**
9: Inject bit error into targeted variable
10: **end if**
11: $\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$
12: $x_{k+1} = x_k + \alpha_k p_k$
13: $r_{k+1} = r_k - \alpha_k A p_k$
14: $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
15: $p_{k+1} = r_{k+1} + \beta_k p_k$
16: */**Error Detection Methods**/*
17: $f_k = x_k^T(-r_k - b)$ *// Error Norm Method*
18: $\kappa_k = f_k - f_{k-1}$
19: **if** $\kappa_k > f_{k-1} * C$ **then**
20: *Error detected by Error Norm Indicator*
21: **else if** $f_k = NAN \parallel INF$ **then**
22: *Error detected by Error Norm Indicator*
23: **end if**
24: $\theta = \kappa_k + \kappa_{k-1} + \kappa_{k-2} + \kappa_{k-3} + \kappa_{k-4}$
25: $\gamma = \frac{\theta}{5}$
26: **if** $\gamma < \kappa_k$ **then**
27: *Error detected by sharp decrease*
28: **end if**
29: */* Energy Norm Method */*
30: $J_k = \alpha_k r_k^T r_k$
31: $\kappa_k = J_k - J_{k-1}$
32: **if** $\kappa_k > J_{k-1} * C$ **then**
33: *Error detected by Energy Norm Indicator*
34: **else if** $J_k = NAN \parallel INF$ **then**
35: *Error detected by Energy Norm Indicator*
36: **end if**
37: $k = k + 1$
38: **end while**
39: **return** x_k

guaranteed. Instead, if x^* denotes the exact solution, $Ax^* = b$, and $e_k = x^* - x_k$ is the kth error in the CG algorithm, e_k^2 is monotonically decreasing. As we described earlier, this property is used by Ozturk *et al.* [18] as leading idea of the error norm indicator which is able to detect soft error in CG effectively.

Similar to GMRES, CR also minimizes the residual $\|b - Ax_n\|_2$ at the nth iteration where x_m is a vector in the Krylov space as pointed out in Theorem 1.1 above. Therefore, the residual decreases monotonically in CR as well. A detection method can be designed which simply observes the behaviour of the residual during the iteration. As it seen in Algorithm 5 (line 23), our detection function just observe behaviour of the residual during iteration.

$$\|r_{m+1}\|_2 < \|r_m\|_2$$

C. Additional Heuristic: Sharp Decrease

Even if error norm $f_k = x_k^T(-r_k - b)$ proposed by Ozturk et al. [18], $\|e_m\|_2$ (energy norm) and residual $\|r_{m+1}\|_2$ (for CR and GMRES) are guaranteed to be monotonically decreasing for certain cases, an approach based on these properties can be prone to false negatives.

Algorithm 4 GMRES with Detection by Monotonic Residual

1: Ax=b linear system and initial vector x_0.
2: $r_0 = b - Ax_0$ /*Residual Vector*/
3: $\beta = ||r_0||_2$
4: $k = 0$
5: **while** not converged **do**
6: v_{j+1} = Aq_j
7: **while** $i < j$ **do**
8: /*Arnoldi Process*/
9: **if** $k = m$ **then**
10: *Choose vector x, y, h, v, s, r, c*
11: *Inject bit error into targeted variable with targeted bit range*
12: **end if**
13: $h_{i,j} = q_i * v_{j+1}$
14: $v_{j+1} = v_{j+1} - h_{i,j} q_i$
15: **end while**
16: /* Perform GMRES outer iteration */
17: .....
18: /* Monotonic Residual */
19: **if** $\|r_{m+1}\|_2 < \|r_m\|_2 * C$ **then**
20: Error detected
21: **else if** $\|r_{m+1}\|_2 = NAN \parallel INF$ **then**
22: Error detected
23: **end if**
24: $k = k + 1$
25: **end while**
26: **return** x_k

This is because soft errors might result in an unexpected sharp decrease in the indicator value. This limits the detection accuracy. To deal with this issue, we add a new heuristic, which is to observe any sharp change of these functions.

V. Experimental Result

We performed all experiments on a cluster where each node has two fourteen-core Xeon E5-2680v4 processors operating at 2.4 GHz, with 512 GB main memory. We used open source implementation of GMRES and CG and improved it with our detection methods. Our presentation and implementation assumes execution on a single machine, however the idea can be applied to a larger-scale execution without any changes – our detection methods are independent of the number of nodes in the system.

A. Experiment Design

We focus on soft faults/errors (bit flips) since they usually do not result in immediate program interruption, instead they cause incorrect output. In our experiments, for each of the detection methods and algorithms, we consider the percentage of detected bit flips in different scenarios (varying iteration number when the flip happened, the array whose element was impacted, and bit range for the bit flip insertion). For each case, we repeat experiments 100 times and report average detection rates. In calculating accuracy rate, a bit flip is considered successfully detected if it is 1) detected within 10 iterations of the insertion, and 2) no false positives have occurred before the bit flip insertion.

Three metrics is used in our evaluations are as follow. First, we measure the efficacy of our method by computing detection accuracy rate, as defined above. Second, for cases where bit flips go undetected, we calculate the

Algorithm 5 Conjugate Residual with Detection by Monotonic Residual and Error Norm

Choose initial vector x_0.
$r_0 = b - Ax_0$
$p_0 = r_0$
$k = 0$
while not converged **do**
  if $k = m$ **then**
    Inject bit error into targeted variable
  end if
  $\alpha_k = \frac{r_k^T A r_k}{(Ap_k)^T Ap_k}$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = r_k - \alpha_k A p_k$
  $\beta_k = \frac{r_{k+1}^T A r_{k+1}}{r_k^T A r_k}$
  $p_{k+1} = r_{k+1} + \beta_k p_k$
  $k = k + 1$
  /***** *Detection Methods* *****/
  $f_k = x_k^T(-r_k - b)$ /* *Error Norm Method* */
  $\kappa_k = f_k - f_{k-1}$
  if $\kappa_k > f_{k-1} * C$ **then**
    Error detected by Error Norm
  else if $f_k = NAN \parallel INF$ **then**
    Error detected by by Error Norm
  end if
  if $\|r_{m+1}\|_2 < \|r_m\|_2 * C$ **then** /* *Monotonic Residual Method*/
    Error detected by Monotonic Residual
  else if $\|r_{m+1}\|_2 = NAN \parallel INF$ **then**
    Error detected by Monotonic Residual
  end if
end while
return x_k

log of the relative error based on the formula, which is the relative difference between computed solution and the true solution x_e of $Ax = b$.

$$\text{Relative Error} = \log\left(\frac{\|x - x_e\|_2}{\|x_e\|_2}\right).$$

We calculate the average of the relative error values among the cases that cannot be detected. Finally, we also consider the runtime overheads of different methods.

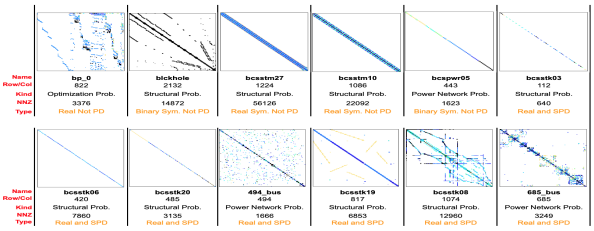

Fig. 2: Matrices from University of Florida Sparse Matrix Collection [5]. NNZ stands for number of non-zero value, N represent number of Row and Column

University of Florida Sparse Matrix Collection is an actively growing sparse matrix set that is widely used by the numerical linear algebra community [5]. We evaluated these Krylov space methods with chosen square linear systems from the University of Florida Sparse Matrix Collection that had the following properties: matrix size [100, 3000], symmetric, positive definite, non-symmetric, non-positive definite and real. Feature and structure of matrices in our experiments are given in Fig.2.

B. *Efficacy of Our Methods*

1) **GMRES:** Our first experiment involved application of the simple residual method on GMRES. As can be seen in Algorithm 4, GMRES contains nested loop. Iterations of the inner loop are Arnoldi Process that orthogonalizes the matrix. Due to the fact that most of the time is spent inside of this loop during the execution, we inject fault into beginning of the inner loop. We defined 7 different vectors as target for fault injection since they require significant memory. These are x, y, h, v, s, r, and c. The x represents solution vector and SDC occurred in the x vector has significant effect on the final result. Our method is capable to detect all occurred SDC in x vector for these 5 different matrices. Fig.6 represent the size of fault injected vectors in our experiment – the possibility of SDC is proportional with vector size. Therefore, catching soft error in larger vectors is more important than a smaller one. As is seen in Fig.3 and Fig.4, our method has a high accuracy rate (except bp_0 matrix) in detecting soft error occurring in y and h vectors, which are the larger vector.

Our main observation is that the effectiveness and performance of our method is based on the matrix type. As can be seen form Figures 3 and 4, the experiment with bp_0 is less successful than other matrices – our method has lower detection rate as well as higher segmentation error case. The reason is that the sparsity structure of bp_0 is different from other matrices, i.e, the distribution of non-zero value in the matrix is non-homogeneous as it seen in Fig.2.

In certain cases, segmentation error occurred due to the soft error after fault injection. We ignore those cases and focus on the SDC that leading us incorrect result, since restarting with last checkpoint is a simple option in these cases. We have also calculated the average of the relative error of our method in GMRES among the cases that cannot be detected with these 5 matrices. While a large deviation in final result is possible, these are arising from matrices that are small in size, and thus, bit-flips at these locations are relatively unlikely.

2) **Conjugate Gradient:** For CG, we evaluate the performance of Energy Norm method and compare with Error Norm method proposed in earlier work. There are three major vectors in design of CG (Algorithm 3) – x, r, and p. 5 different real matrices with SPD property obtained from the University of Florida Matrix Collection [5] were used – 494_bus, bcsstk06, bcstk20, bcsstk19, and bcsstk08. Detail information of those matrices is given in Fig.2.

Fig.8 demonstrate the performance of Energy Norm method on CG. We can see that Energy Norm can reasonably overcome soft error in CG for r and p cases. However, it is not so effective for x cases. Therefore, we have combined Energy norm with Error Norm, resulting in improved detection accuracy. For example, our methods has almost 60% accuracy rate in detection of soft error occurred in x vectors with 494_bus and

Accuracy Rate of Monotonic Residual Detection in GMRES

Injection Time	20%															40%													
Matrices	bcspwr05			bcsstm10			bcsstm27			blckhole			bp_0			bcspwr05			bcsstm10			bcsstm27			blckhole			bp_0		
Vectors	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r
60-63	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	77%	100%	*	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	82%	100%	*
56-59	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	38%	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	26%	100%	*
52-55	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	3%	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	100%	*
48-51	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	0%	100%		100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	100%	
Injection Time	60%															80%													
Matrices	bcspwr05			bcsstm10			bcsstm27			blckhole			bp_0			bcspwr05			bcsstm10			bcsstm27			blckhole			bp_0		
Vectors	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r	h	v	r
60-63	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	83%	100%	*	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	82%	100%	*
56-59	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	20%	100%	*	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	19%	100%	*
52-55	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	100%	0%
48-51	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	100%	0%

Fig. 3: Accuracy rate for Residual method in GMRES – fault injection into h, v, r vectors (we exclude x case since it has 100% detection rate for all cases). * represents program termination case due to injected fault

Accuracy Rate of Monotonic Residual Detection in GMRES

Injection Time	20%															40%													
Matrices	bcspwr05			bcsstm10			bcsstm27			blckhole			bp_0			bcspwr05			bcsstm10			bcsstm27			blckhole			bp_0		
Vectors	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y
60-63	78%	*	100%	87%	87%	100%	*	93%	100%	98%	100%	100%	3%	59%	0%	41%	*	100%	83%	78%	100%	*	72%	100%	78%	100%	100%	18%	40%	0%
56-59	43%	*	100%	66%	80%	100%	*	85%	100%	100%	100%	100%	*	*	*	*	*	100%	65%	83%	100%	*	71%	100%	100%	100%	100%	*	100%	*
52-55	*	41%	100%	*	*	100%	*	90%	100%	100%	100%	100%	*	99%	*	43%	52%	*	83%	*	100	*	72%	100%	99%	100%	100%	50%	99%	*
48-51	24%	58%	*	90%	85%	100%	*	90%	*	100%	100%	100%	*	99%	*	43%	40%	*	83%	90%	100	*	76%	*	100%	100%%	100%	*	99%	*
Injection Time	60%															80%													
Matrices	bcspwr05			bcsstm10			bcsstm27			blckhole			bp_0			bcspwr05			bcsstm10			bcsstm27			blckhole			bp_0		
Vectors	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y	c	s	y
60-63	15%	*	100%	67%	78%	*	*	66%	*	90%	100%	100%	46%	60%	0%	13%	*	100%	63%	69%	*	*	62%	100%	92%	100%	100%	46%	80%	0%
56-59	*	*	100%	62%	*	100%	*	*	100%	100%	*	100%	90%	99%	*	28%	*	100%	67%	*	100%	*	*	100%	100%	100%	100%	100%	99%	*
52-55	39%	31%	*	66%	*	100%	*	62%	100%	99%	*	100%	*	100%	*	100%	7%	100%	76%	65%	100%	*	73%	100%	100%	100%	100%	*	100%	*
48-51	39%	24%	*	90%%	85%	100%	*	73%	100%	100%	100%	100%	*	100%	*	86%	43%	100%	81%%	93%	100%	*	71%	100%	100%	100%	100%	*	*	*

Fig. 4: Accuracy rate for Residual method in GMRES – fault injection into c, s, y vectors * represent program termination case due to injected faults

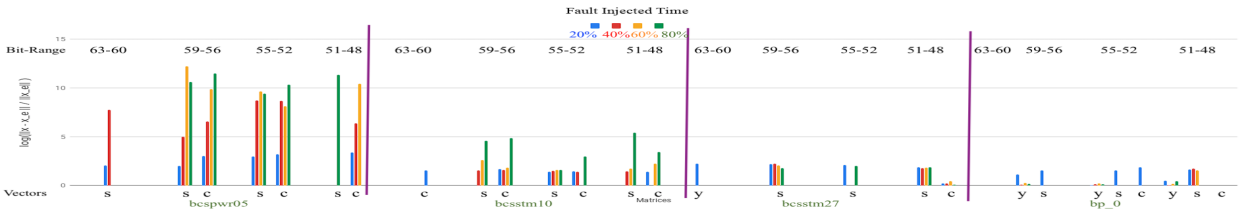

Fig. 5: Inaccuracy in Results Due to Undetected Errors (The Relative Error) – Residual detection method on GMRES (Missing values are not applicable)

Vectors Size in GMRES							
Vectors	x	y	h	v	s	r	c
Size	n	mr+1	(mr+1)*mr	n*(mr+1)	mr	n	mr

Fig. 6: The fault injected vectors size in GMRES. The mr stands for number of inner iteration limit (Iteration Number of Arnoldi). The n is size of solution vector.

bcsstk06 matrices. The average of all matrices in CG for the cases where we inject fault into r and p within bit-range 63-60 is 75%. It even has better performance for bit-range 59-56 which is almost 90%.

We also measured the average relative error of our method on CG. Fig.9 demonstrate the change of solution in log10 (Relative Error). As it seen in the Fig.9, all of the results are negative, showing low errors. For instance, the log (base 10) of the overall relative error for the x case in bit range 63-60 is -1.01815 (Time frame 60%, bcsstk06). This means that most of the soft errors that result in large error in the solution vector are detected by our detection methods. On the other hand, Fig.10 stands for the relative error of Error Norm itself (without any improvement or combination). Error Norm itself has a large average of relative error for the case in which fault injected in x vectors in the bit-range 63-60.

The other observation is that unlike GMRES, the matrix type does not have a large impact on the performance of our methods as it seen in Fig.8. Ultimately, it can be concluded that Energy Norm, based on minimization property of Krylov Subspace method, is effective in either detecting errors or limiting the impact of soft errors on final results.

3) **Conjugate Residual:** In the case of CR, since the residual vectors should be A-orthogonal or conjugate, a detector based on monotonic property of the residual can be used. Like CG, three vectors of interest are x, r, and p. First, we only apply monotonic residual detection method for catching error in CR. Then, to increase detection rate for the x vector, we combine residual method with the error norm method. We have used 1000*1000 SPD artificial matrix as well as several real matrices attained from Universify of Florida Sparse Matrix Collection (bcspwr5, bcsstk03, bcsstk06, bcsstk20).

Fig.11 shows the accuracy rate of the method combining residual with error norm. Results are taken from case in which fault injected into r vector for 1000*1000 artificial SPD matrix. It really has high detection rate particularly for scenario that faults are injected in bit range 59-56. Fig.12 stands for detection rate of Monotonic Residual Method with Error Norm in CR for cases

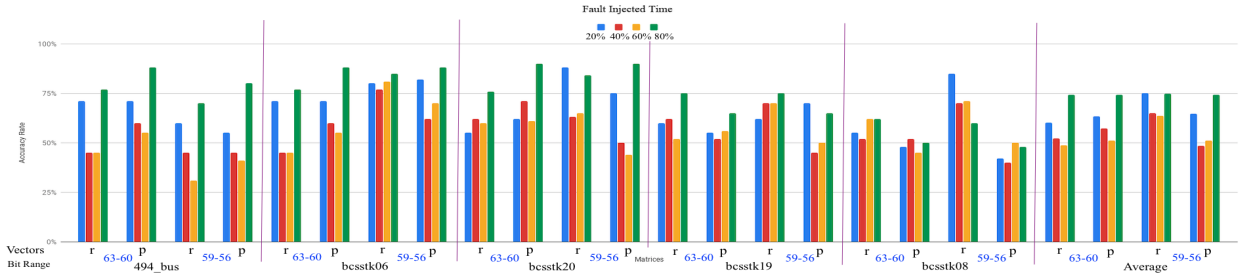

Fig. 7: Accuracy rate of Energy Norm method on Conjugate Gradient – Results reported for 5 matrices and the overall averages

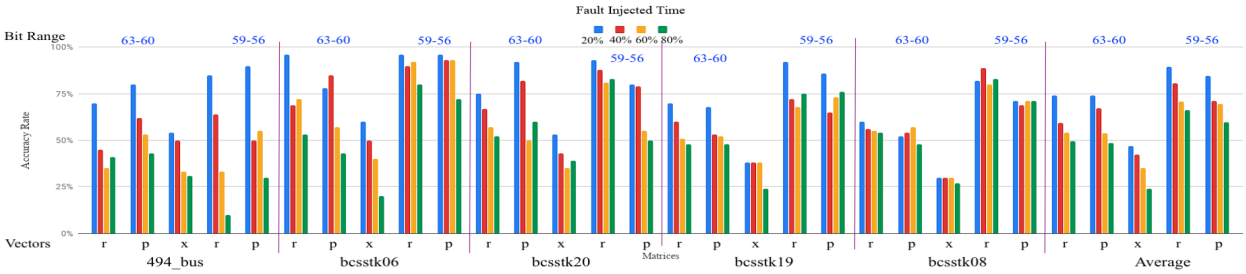

Fig. 8: Accuracy rate of Energy Norm combined with Error Norm on Conjugate Gradient (we exclude x case in bit-range 59-56 since it has low performance) – Results reported for 5 matrices and the overall averages

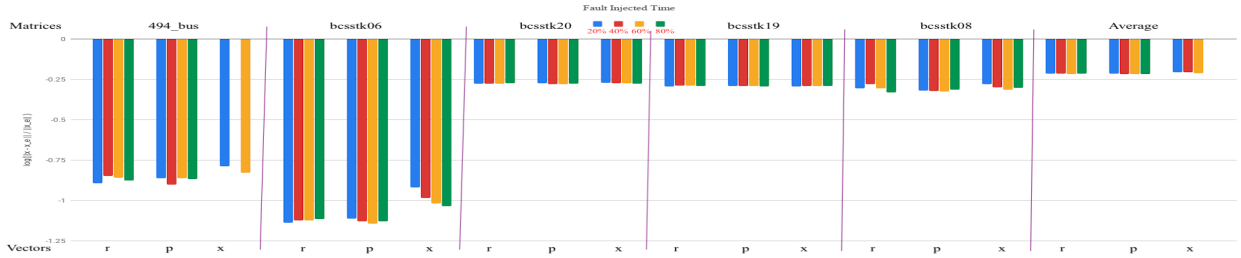

Fig. 9: Inaccuracy in Results Due to Undetected Errors (The Relative Error) in bit-range 63-60 – Energy Norm with Error Norm on Conjugate Gradient

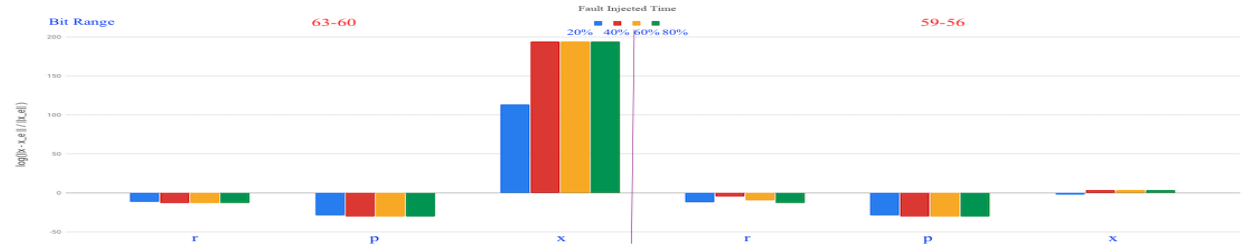

Fig. 10: Inaccuracy in Results Due to Undetected Errors (The Relative Error)

where fault injected into X. In Fig.13 we have shown all result case by case taken from 4 different real matrices. According to these results, it can be concluded that the combined method is quite effective. We have also calculated the average Relative Error values among the cases where soft error can not be detected. The result is given in Fig.14. It simply indicate that for all scenarios in the experiment most of the soft errors that cause large error in the solution vector are detected by our detection methods.

Monotonic Residual Detection Method with Error Norm in Conjugate Residual

Injection Time	20%	40%	60%	80%
Matrices	Simple 1000*1000 SPD			
60-63	76%	70%	63%	52%
56-59	93%	84%	82%	80%
52-55	53%	58%	64%	66%

Fig. 11: Detection Accuracy of Residual detection method with Error Norm on Conjugate Residual (r vector - artificial SPD matrix)

C. Overhead of Our Methods

In this section, we focus on the overall time overhead of our proposed methods for soft error detection in CR, GMRES and CG. As explained and proved above, the time complexity of Energy Norm is $O(1)$. The time complexity of Monotonic Residual detection method depends on calculation of residual vector. We compare overhead of Monotonic Residual method, Energy Norm and Error Norm in CG, CR and GMRES. One of the reason behind this small overhead is that Residual is calculated already in the implementation of these application to decide termination of iteration. As it seen in the Fig.15, overhead caused by our methods are modest, which further establishes the effectiveness of our methods.

D. Discussion: Other Methods in the Literature

As explained in the related work section, there has only been limited work proposed regarding to fault tolerance in GMRES. Chen [3] proposed a methodology for iterative solvers such as GMRES, CG, and BiCG. His methodology applies invariants to check if SDC occured or not. However, his approach has additional computation and parallel communication to check invariant of algorithm (as also indicated by Elliott as well *et al.* [7]). Bridges *et al.* [20] proposed FT-GMRES (Fault Tolerant GMRES) – this approach requires rewriting an application with a different algorithm to make it resilient towards soft errors. Elliott *et al.* [7] evaluated effect of soft error on the GMRES Krylov Subspace methods as well. They basically defined theoretical upper limits for some value in the application and by using this bound they proposed SDC detector. Their detection method is only able to detect error that exceeds the bound, which depends on the input matrix.[7].

Unlike GMRES, there have been certain efforts on the detection of soft error in Conjugate Gradient. Till now, many researcher have used the residual as a soft error detector for CG [18], [11]. However, as we explained above the residual value is not guaranteed to decreasing for all iterations, and thus the approach is only a heuristic. It is not strictly monotonically decreasing and then this approach is only heuristic. Therefore, it leads false positive cases.

Recently, Ozturk *et al.* [18] proposed monotonically decreasing function which we refer to as the Error Norm,

Monotonic Residual Detection Method with Error Norm in Conjugate Residual

Injection Time	20%	40%	60%	80%
Matrices	Fault Injected into X vector			
Simple SPD	52%	46%	32%	28%
bcspwr05	45%	32%	26%	30%
bcsstk03	65%	70%	100%	60%
bcsstk06	60%	44%	20%	31%
bcsstk20	44%	35%	30%	36%

Fig. 12: Detection Accuracy of Residual detection method with Error Norm on Conjugate Residual

to observe abnormal routine in CG. They also combined their function with Invariant checking algorithm (proposed by Chen [3] and improved by Loh *et al.* [15]). This method checks the invariant ϕ_k which is the conjugate property of the direction vector p_k [18].

$$\phi_k = \frac{p_k^T A p_{k-1}}{\|p_k\|_2 \|A p_{k-1}\|_2}$$

However, calculating ϕ_k is expensive and results in substantial overheads[18]. Therefore, we combined Error Norm approach with the function called Energy Norm where the time complexity of Energy Norm calculation is O(1).

Scholl *et al.* [1] proposed following equation as a soft error detection method that they called λ detection:

$$b^T p_i = x_k^T w_i$$

in which w_i represent Ap_i for $i < k$. The fundamental idea behind this approach is observing inner product on the left-hand side in the equation since it is independent over continuous iteration. Scholl *et al.* [1] assigned this left-hand side in a scalar variable $\lambda_i = b^T p_i$ so that they can check whether the equality fails to hold within some tolerance or not. As demonstrated by Ozturk *et al.* [18], λ detection has high overheads and works only for higher bits.

VI. Conclusion

This paper has focused on the problem of detecting soft errors in the case of Krylov subspace methods, particularly, CG, GMRES, and CR. By identifying key numerical properties, we develop efficient and accurate detectors of soft errors. Particularly, we identify a term we refer to as *energy norm*, which is decreasing for our target class of methods. Next, though detectors based on error norm and residual value had been used in the past, we have expanded the set of algorithms on which they can be applied.

We have extensively evaluated our proposed methods using several real matrices. Our evaluation shows high detection accuracy (especially for GMRES and CG). Moreover, we find that average error due to undetected errors is small, indicating that most significant errors get detected. Finally, we see low overheads of applying our method.

References

[1] Michael A Kffochte Alexander Scholl, Claus Braun and Hans-Joachim Wunderlich. Efficient algorithm-based fault tolerance for sparse matrix operations. *46th Annual IEEE/IFIP International Conference on. IEEE*, 2016.

[2] Greg Bronevetsky and Bronis de Supinski. Soft error vulnerability of iterative linear algebra methods. *Proceedings of the 22nd annual international conference on Supercomputing, ACM*, 2008.

[3] Zizhong Chen. Online-abft: An online algorithm based fault tolerance scheme for soft error detection in iterative methods. *ACM SIGPLAN*, 48(8).

[4] Michael Saunders David Chin-Lung Fong. Cg versus minres: An empirical comparison. pages 17(1):44–62, 2012.

[5] Yifan Hu Davis Timothy. The university of florida sparse matrix collection. 2011.

Monotonic Residual Detection Method with Error Norm (Proposed Indicator) In Conjugate Residual

Injection Time	20%					40%				
Matrices	Average	bcspwr5	bcsstk03	bcsstk06	bcsstk20	Average	bcspwr5	bcsstk03	bcsstk06	bcsstk20
60-63	75%	65%	81%	77%	75%	65%	59%	66%	71%	65%
56-59	83%	73%	80%	92%	86%	83%	81%	72%	94%	83%
52-55	34%	25%	65%	35%	12%	27%	22%	50%	30%	6%
Injection Time	**60%**					**80%**				
Matrices	**Average**	**bcspwr5**	**bcsstk03**	**bcsstk06**	**bcsstk20**	**Average**	**bcspwr5**	**bcsstk03**	**bcsstk06**	**bcsstk20**
60-63	56%	50%	64%	53%	55%	53%	37%	78%	45%	52%
56-59	80%	91%	78%	74%	76%	75%	71%	71%	82%	74%
52-55	29%	25%	40%	42%	9%	23%	13%	41%	31%	6%

Fig. 13: Accuracy rate of Residual method with Error Norm on Conjugate Residual (r vectors)

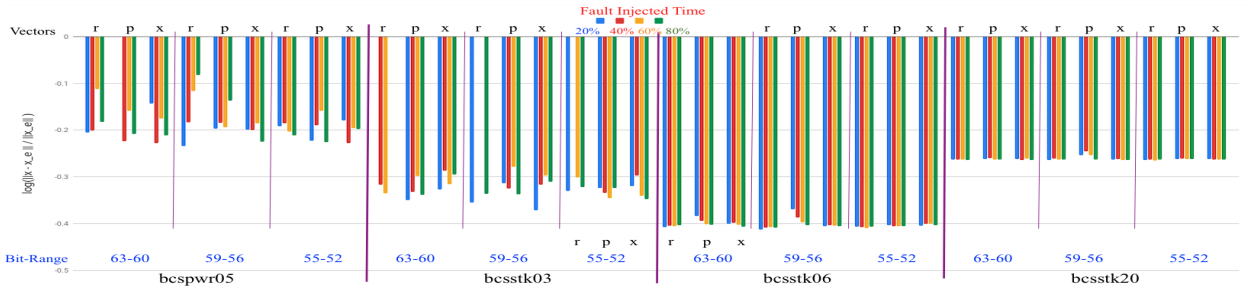

Fig. 14: Relative Error with Residual method with Error Norm on Conjugate Residual

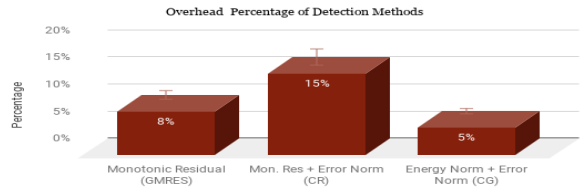

Fig. 15: Overhead of detection methods; Monotonic Residual (GMRES), Mon. Res. with ERROR Norm (CR) and Energy Norm with Error Norm (CG)

[6] Jack Dongarra, Thomas Herault, and Yves Robert. *Fault tolerance techniques for high-performance computing*, chapter 1, pages 3–85. Springer International Publishing, 2015.

[7] James Elliott, Mark Hoemmen, and Frank Mueller. Evaluating the impact of SDC on the GMRES iterative solver. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, May 19-23, 2014*, pages 1193–1202, 2014.

[8] David Fiala, Frank Mueller, Christian Engelmann, Rolf Riesen, Kurt B. Ferreira, and Ron Brightwell. Detection and correction of silent data corruption for large-scale high-performance computing. In *SC Conference on High Performance Computing Networking, Storage and Analysis, SC '12, Salt Lake City, UT, USA - November 11 - 15, 2012*, page 78, 2012.

[9] Magnus R Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[10] Martin H.Gutknecht. A brief introduction to krlov space methods for solving linear systems. *Frontiers of Computational Science. Springer,Berlin,Heidelberg*, pages 53–62, 2007.

[11] Mehmet Can Kurt Jiaqi Liu and Gagan Agrawal. A practical approach for handling soft errors in iterative applications. *IEEE International Conference on Cluster Computing*, 2015.

[12] Jieyang Chen Dingwen Tao Sihuan Li Panruo Wu Hongbo Li Kaiming Ouyang Yuanlai Liu Fengguang Song Liang, Xin and Zizhong Chen. Correcting soft errors online in fast fourier transform. 2017.

[13] Jiaqi Liu and Gagan Agrawal. A methodology for application-level asynchronous checkpointing for mpi applications. In *In Euro-Par 2014 Resilience Workshop(Euro-Par2014WS), November,2014*, 2014.

[14] Jiaqi Liu and Gagan Agrawal. Soft error detection for iterative applications using offline training. *IEEE 23rd International Conference on High Performance Computing (HiPC)*, 2016.

[15] Kewal K. Saluja Loh, Felix and Parameswaran Ramanathan. Fault tolerance through invariant checking for iterative solvers. *VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), 2016 29th International Conference on. IEEE*, 2016.

[16] Gokcen Kestor Joseph Manzano Osman Unsal Samrat Chatterjee Mutlu, Burcu Ozcelik and Sriram Krishnamoorthy. Characterization of the impact of soft errors on iterative methods. 2018.

[17] Xiang Ni, Esteban Meneses, Nikhil Jain, and Laxmikant V. Kalé. Acr: automatic checkpoint/restart for soft and hard error protection. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC'13, Denver, CO, USA - November 17 - 21, 2013*, 2013.

[18] Muhammed Emin Ozturk, Marissa Renardy, Yukun Li, Gagan Agrawal, and Ching-Shan Chou. A novel approach for handling soft error in conjugate gradients. *In 2018 IEEE 25th International Conference on High Performance Computing (HiPC), pp. 193-202. IEEE, 2018)*, 2018.

[19] Christina Pacher Pachajoa, Carlos and Wilfried N. Gansterer. Node-failure-resistant preconditioned conjugate gradient method without replacement nodes. 2019.

[20] Kurt B Ferreira Michael A.Heroux Patrick G, Bridges and Mark Hoemme. Fault-tolerant iterative methods via selective reliability. 2012.

[21] Yousef Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[22] Piyush Sao and Richard Vuduc. Self-stabilizing iterative solvers. In *The Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems.ACM,2013*, 2013.

[23] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. Dram errors in the wild: a large-scale field study. *Commun. ACM*, 54(2):100–107, 2011.

[24] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. 1994.

[25] Daniel B.Szyld Valeria Simoncini. Recent computational development in krylov subspace methods for linear systems. pages 14(1):1–59, 2007.

[26] Martin H.Schults Yuocef Saad. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, pages 7(3):856–869, 1986.