Optimal Real Number Codes for Fault Tolerant Matrix Operations

Zizhong Chen
Department of Mathematical and Computer Sciences
Colorado School of Mines, Golden, CO 80401.
zchen@mines.edu

ABSTRACT

Today s long running high performance computing applications typically tolerate fail-stop failures by checkpointing. However, applications such as dense linear algebra computations often modify a large mount of memory between checkpoints and checkpointing usually introduces considerable overhead when the number of processors used for computation is large. It has been demonstrated in [13] that single fail-stop failure in ScaLAPACK matrix multiplication can be tolerated without checkpointing at a decreasing overhead rate of $1/\sqrt{p}$, where p is the number of processors used for computation. Multiple simultaneous processor failures can be tolerated without checkpointing by encoding matrices using a real-number erasure correction code. However, the floating-point representation of a real number in today's high performance computers introduces round off errors which can be enlarged and cause the loss of precision of possibly all digits during recovery when the number of processors in the system is large. In this paper, we present a class of Reed-Solomon style real-number erasure correcting codes which is numerically optimal during recovery. We analytically construct the numerically best erasure correcting codes for 2 erasures and develop an approximation method to computationally construct numerically good codes for 3 or more erasures. We prove that it is impossible even for the numerically best minimum redundancy erasure correcting codes to correct all erasure patterns when the total number of processors is large. We give the conditions that guarantee to correct all two erasures. Experimental results demonstrate that the proposed codes are numerically much more stable than existing codes.

1. INTRODUCTION

While the peak performance of the contemporary high performance computing (HPC) systems continues to grow exponentially, it is getting more and more difficult for scientific applications to achieve high performance due to both the complex architecture of and the increasing failures in these systems. Schroeder and Gibson from Carnegie Mellon University (CMU) recently studied the system logs of 22 HPC systems in Los Alamos National Laboratory (LANL) and found that the mean-time-to-interrupt (MTTI) for these HPC systems varies from about half a month to less than half a day [44, 45, 46]. In order to use these systems efficiently and avoid restarting computations from beginning after failures, applications have to be able to tolerate failures. Today's long running scientific applications typically tolerate failures by checkpointing [6, 14, 37, 38, 41, 47]. Checkpointing can usually be used in different type of systems and to a wide range of applications. However, when applications modify a large mount of memory between two consecutive checkpoints, checkpointing often introduces a considerable overhead [30, 36].

Matrix operations (such as matrix multiplication, solving system of linear equations, and finding eigenvalues and eigenvectors, etc) are fundamental operations in science and engineering. Some important linear algebra operations such as Gaussian elimination have been proved to be able to scale to more than 100,000 processors and achieve more than one petaflops on today's HPC systems [1]. However, today's widely used dense linear algebra software such as ScaLAPACK [7] and PLAPACK [26] usually modifies a large mount of memory between checkpoints, therefore, checkpointing techniques often introduce a considerable overhead into the computation. The high frequency of failures and the large number of processors in the next generation HPC systems will further exacerbate the problem.

In [13], a highly scalable checkpoint-free techniques was proposed to tolerate single fail-stop failure in high performance matrix operations on large scale HPC systems. It was also demonstrated that the overhead rate of this scheme decreases with a speed of $1/\sqrt{p}$ when the number of processors p increases. However, in order to tolerate multiple simultaneous process failures with minimum redundancy, a real number version Reed-Solomon style erasure correcting codes have to be used to encode the input matrices.

In existing Reed-Solomon style real number erasure correcting codes, the generator matrices mainly include: Vandermonde matrix (Vander) [28], Vandermonde-like matrix for the Chebyshev polynomials (Chebvand) [8], Cauchy matrix (Cauchy), Discrete Cosine Transform matrix (DCT), Discrete Fourier Transform matrix (DFT) [22, 23], Gaussian random matrix [11, 12], and Grassmannian frame matrix [43]. If there is no round-off errors in the representation of a real number, these generator matrices can all be used as the encoding matrices of the proposed checkpoint-free techniques in [13].

However, in today's computer arithmetic where no computation is exact due to round-off errors, it is well known [27] that, in solving a linear system of equations, a condition number of 10^k for the coefficient matrix leads to a loss of accuracy of about k decimal digits in the solution. The coefficient matrix of the system of equations to be solved during recovery may be any square sub-matrix (including minor) of the generator matrix. Therefore, in order to get a numerically good recovery for any erasure patterns, *any square sub-matrix (including minor) of the generator matrix has to be well-conditioned.*

But the generator matrices from existing Reed-Solomon style real number erasure correcting codes mentioned above all contain many ill-conditioned sub-matrices when the sizes of these generator matrices are large. Therefore, in these real number codes, when certain erasure patterns occur, an ill-conditioned linear system has to be solved to reconstruct an approximation of the original information, which can cause the loss of precision of possibly all digits in the recovered numbers. To the best of our knowledge, it is still open whether there exists any arbitrarily large generator matrix that can correct all erasures or not. It is also an open problem how to find the codes with optimal numerically stability.

In this paper, we present a class of numerically optimal Reed-Solomon style real-number erasure correcting codes. We construct the numerically optimal erasure correcting codes for two erasures analytically and develop an approximation method to approximate the numerically optimal codes for three or more erasures computationally. We explore the property of generator matrices that are able to correct all erasure patterns. We prove no minimum redundancy codes can correct all erasure patterns when the size of processors is large and the number of erasures is more than one. We give an upper bound on the number of processors so that all two erasure patterns can be corrected. Experimental results demonstrate that our codes are numerically much more stable than existing codes.

Although we only focus on the correcting of erasures in this paper, it is also possible to use our codes (generator matrices) to correct errors through the l_1 minimization techniques proposed in [9, 19]. While this paper develops the codes for fault tolerant matrix operations, the codes can also be used in many other fields such as compressive sensing [20] and fault tolerant combinatorial and dynamic systems [28].

The rest of the paper is organized as following. Section 2 introduces techniques for fault tolerant matrix operations. In Section 3, we explore the numerical properties of existing real number codes and present a class of real number codes that have optimal numerical stability. In Section 4, we analytically construct the numerically best erasure correcting codes for two erasures. Section 5 develops an approximation method to approximate the numerically optimal codes for three or more erasures computationally. In Section 6, we compare various real number codes experimentally. Section 7 concludes the paper and discusses the future work.

2. FAULT TOLERANT MATRIX OPERATIONS

Matrix operations are fundamental for science and engineering. Incorporating fault tolerance into matrix operations has been extensively studied for many years by many researchers [3, 4, 5, 8, 11, 12, 13, 14, 25, 29, 30, 32, 33, 36, 39, 40, 48].

In [29], the algorithm based fault tolerance (ABFT) is proposed to detect, locate, and correct miscalculations. The idea is to encode the original matrices using real number codes and then re-design algorithms to operate on the encoded matrices. In [29], Huang and Abraham proved that the encoding relationship in the input encoded matrices is preserved **at the end of the computation no matter which algorithm is used** to perform the computation. Therefore, processor miscalculations can be detected, located, and corrected at the end of the computation. ABFT researches have mostly focused on detecting, locating, and correcting miscalculations or data corruption where failed processors are often assumed to be able to continue their work but produce incorrect calculations or corrupted data. The error detection are often performed at the end of the computation by checking whether the final computation results satisfy the encoding relationship or not.

However, in a distributed environment, if a failed processor stops working, then we need to be able to detect, locate, and recover the data **in the middle of the computation**. In order to be able to recover in the middle of the computation, a global consistent state of the application is often required. Checkpointing and message logging are typical approaches to maintain or construct such a global consistent state. In [14, 15, 16, 17, 30], real number erasure correcting codes are used to encode the checkpoint data to maintain a global consistent state with redundancy periodically.

Recently, in [13], it has been demonstrated that fault tolerance (for fail-stop failures) for large scale parallel matrix operations on today's large HPC systems can be achieved without any checkpointing (or message logging) by encoding the original matrices into larger weighted checksum matrices using real-number erasure correcting codes. The scheme is highly scalable with low overhead. The overhead rate decreases with a speed of $1/\sqrt{p}$ when the number of processors p increases.

Traditional erasure correcting codes based on finite fields do **NOT** work [11] for the techniques in [29, 13]. Real number codes have to be used to encode the input matrices. In order to be able to recover from multiple simultaneous failures of any patterns, the encoding matrix have to be chosen very carefully. This encoding matrix is often called the generator matrix of a linear code in coding theory. The goal of this paper is to find an appropriate generator matrix to encode the input matrices so that multiple simultaneous failures in large scale parallel matrix operations can be recovered without any checkpointing (or message logging).

3. REAL-NUMBER CODES FOR FAULT TOLERANT MATRIX OPERATIONS

The research on real number codes can be dated back to [34]. Recently, codes based on random matrices [10, 11, 12] and Grassmannian frames [24, 43] have been proposed to improve the numerically stability of the recovery. However, it is still an open problem what is the numerically best real number codes. In this section, we discuss some popular real number codes and propose a class of new real number codes which have optimal numerical stability.

Let $x = (x_1, x_2, ..., x_n)^T \in \mathcal{R}^n$ denote the original information, and $G_{m\times n}$ denote a m by n real number matrix.

The redundant information $c = (c_1, c_2, ..., c_m)^T \in \mathcal{R}^m$ is calculated by

$$\begin{cases} g_{11}x_1 + \ldots + g_{1n}x_n & = c_1 \\ & \vdots \\ g_{m1}x_1 + \ldots + g_{mn}x_n & = c_m. \end{cases} \quad (1)$$

$G_{m\times n}$ is often called the generator matrix of the linear code. We also call $G_{m\times n}$ the encoding matrix for fault tolerant matrix operations. In a fault tolerant matrix operations, the original information x_i is the local matrix in the local memory of a processor. Without loss of generality and for the simplicity of the discussion, in this paper, we assume x_i is just a real number.

The relationship in (1) actually establishes m equalities between the original data x and the redundant information c. If k (where $k \leq m$) elements of x is erased, then the m equalities become a system of linear equations with k unknowns. When the generator $G_{m\times n}$ is appropriately chosen, the lost k elements in x can be able to be reconstructed through solving this system of linear equations with k unknowns.

The real number coding theory problem we want to solve is: *how to choose the generator matrix $G_{m\times n}$ in (1), such that, after any no more than m erasures in x, a good approximation of all erased elements in x can still be reconstructed by solving the system of linear equations derived from (1)?*

3.1 Real-Number Codes Derived from Finite Field Codes

In [35], Nair and Abraham proved that, for any finite field code, there is a corresponding code in real number field. In the existing codes derived from finite fields, the generator matrices mainly include: Vandermonde matrix (Vander) [28], Vandermonde-like matrix for the Chebyshev polynomials (Chebvand) [8] Cauchy matrix (Cauchy), Discrete Cosine Transform matrix (DCT), and Discrete Fourier Transform matrix (DFT) [22, 23]. These generator matrices all contain many ill-conditioned sub-matrices when the size the generator matrices become large. Therefore, in these codes, when certain erasure patterns occur, an ill-conditioned linear system of equations has to be solved to reconstruct an approximation of the original information, which can cause the loss of precision of possibly all digits in the recovered numbers.

3.2 Real-Number Codes Based on Random Matrices

In [10, 11, 12], Gaussian and uniform random matrices have been proposed as the encoding (generator) matrices. It is well know that Gaussian random matrices are well conditioned [21]. Note that any sub-matrix of a Gaussian random matrix is still a Gaussian random matrix, therefore, Gaussian random matrix can guarantee the recovery of the lost data with high probability.

3.3 Real-Number Codes Based on Grassmannian Frames

While Gaussian random codes is good with high probability, it is nondeterministic. It has been shown in [31] that Gaussian random distribution in $\mathcal{R}^n$ is equivalent to uniform random distribution in $\mathcal{S}^{n-1}$. Uniformly distributed points on hyper spheres tend to maximize the minimum sphere distance between points. If the sphere distance of two points is zero, the corresponding two columns of the matrix are the same. The sub-matrices containing these two columns are singular. When two vectors are the same, the correlation of the two vectors is 1. The Grassmannian frame idea minimize the maximum correlations between columns of the generator matrices [24, 43].

A sequence of vectors $\{g_k\}_{k=1}^n \in \mathcal{R}^m$ is called a Grassmannian frame if it is the solution to

$$\min_{\{f_k\}_{k=1}^n \in \mathcal{R}^m, <f_k, f_k>=1} \left\{ \max_{i\neq j} \{< f_i, f_j >\} \right\} \quad (2)$$

The Grassmannian frame code is defined as the code whose generator matrix is $G_{m\times n} = (g_1, g_2, \ldots, g_n)$.

Minimizing the maximum correlations is equivalent to maximizing the minimum angle between columns of the generator matrices. The problem of maximizing the minimum angle between vectors on a hyper sphere is called the Grassmannian (line) packing problem [18]. It is hard to find optimal arrangements of points even on a 2-sphere (i.e. $m = 3$). Steve Smale has listed the problem of "distribution of points on the 2-sphere" as the problem #7 of a total of 18 unsolved mathematics problems in twenty-first century [42]. There are no general analytical solutions for this problem except for some special combinations of m and n [2].

3.4 Real-Number Codes with Optimal Numerical Stability

The Grassmannian frame code minimizes the maximum correlations between columns of the generator matrices, however, the accuracy during recovery is directly related only to the condition number of the equations to be solved and condition number is a property that associated with more than two columns of a matrix. Even if the maximum correlations are minimized, it is still possible that the generator matrix contains a singular sub-matrix. Therefore, in order to get better codes, we decide to work on minimizing the maximum condition numbers of sub-matrices of a generator matrix directly.

The recovery process involves solving a system of linear equations with one of the sub-matrices from the generator matrix G as the coefficient matrix. The coefficient matrix can be any sub-matrix including minor of G. It is well know that in order to get a numerically good solution, the coefficient matrix have to be well conditioned. Therefore, in order to be able recovery from all erasure patterns, the generator matrix G have to satisfy any square sub matrix including minor of G have to be well conditioned.

If the worst conditioned sub-matrix of G is well conditioned, then all the sub-matrix of G will be well conditioned. Therefore, we look for generator matrices G for which the condition number of the worst conditioned sub-matrix is minimized.

There are finite number of sub-matrices in G, therefore, we can rank these sub-matrices. Let G_i denote the i^{th} sub-matrix of G, then the matrix G^* that minimize the condition number of the worst conditioned sub-matrix of G is the solution of the following minimax problem.

$$f(m,n) = \min_{G_{m\times n} \in \mathcal{R}^{m\times n}} \left\{ \max_i \{\kappa(G_i)\} \right\} \quad (3)$$

The code G^* obtained from the solution of the above minimax problem (3) is numerically best in the sense that the generator matrix obtained has the condition number of the worst conditioned sub-matrix minimized. $f(m,n)$ is the condition number of the worst conditioned sub-matrix of the optimal G^* obtained. it is well known [27] that, in solving a linear system of equations, a condition number of 10^k for the coefficient matrix leads to a loss of accuracy of about k decimal digits in the solution. Therefore, $f(m,n)$ an be used to estimate the worst case recovery accuracy. For example in IEEE standard 754 floating point numbers, there are 16 digits of accuracy. Then the worst case recovery can guarantee an accuracy of $16 - \log_{10} f(m,n)$ digits.

The minimax problem specified in (3) is also difficult even if G is restricted on matrices with unit norm columns. Actually, when G is restricted on matrices with unit norm columns, the problem also becomes finding optimal arrangements of points on hyper-sphere. As we discussed before, it is hard to find optimal arrangements of points even on a 2-sphere. Steve Smale has listed this problem of "distribution of points on the 2-sphere" as the problem #7 of a total of 18 unsolved mathematics problems in twenty-first century [42].

4. OPTIMAL REAL-NUMBER CODES FOR TWO ERASURES

In what follows we will solve problem (3) for the special case when $m = 2$. The generator matrices we obtain is the generator matrices for numerically best real-number codes for two erasures. $f(2,n)$ we obtain is the condition number of the worst conditioned sub-matrix of the numerically best real-number codes for two erasures.

If there are elements with value zero in the generator matrix, there will be singular 1×1 sub-matrices in the generator matrix. Therefore, when solving (3), we just need to consider generator matrices with none of their elements being 0. Without loss of generality, we assume the elements of G is non-zero. When $m = 2$, it is enough to just consider all the 2×2 sub-matrices.

For any $2 \times n$ matrix $G_{2\times n} \in \mathcal{R}^{2\times n}$, let g_j denote the j^{th} column of $G_{2\times n}$. Let G_{ij} denote the sub-matrix of $G_{2\times n}$ consisting of the column i and j of $G_{2\times n}$.

THEOREM 1. *Let*

$$f(2,n) = \min_{G_{2\times n}\in\mathcal{R}^{2\times n}} \left\{ \max_{i,j} \{\kappa(G_{ij})\} \right\} \tag{4}$$

Then,

$$f(2,n) = \sqrt{\frac{1+\cos\frac{\pi}{n}}{1-\cos\frac{\pi}{n}}} \tag{5}$$

The following generator matrix is one of the solutions for (4)

$$G = \begin{pmatrix} \cos\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \dots & \frac{(2n-1)\pi}{2n} \\ \sin\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \dots & \frac{(2n-1)\pi}{2n} \end{pmatrix}$$

PROOF. In a polar coordinate system (Figure 1), the i^{th} column of $G_{2\times n}$ can be represented as

$$g_j = \begin{pmatrix} r_i\cos\theta_i \\ r_i\sin\theta_i \end{pmatrix}$$

$G_{2\times n}$ can be represented as

$$G_{2\times n} = \begin{pmatrix} r_1\cos\theta_1 & \dots & r_n\cos\theta_n \\ r_1\sin\theta_1 & \dots & r_n\sin\theta_n \end{pmatrix}$$

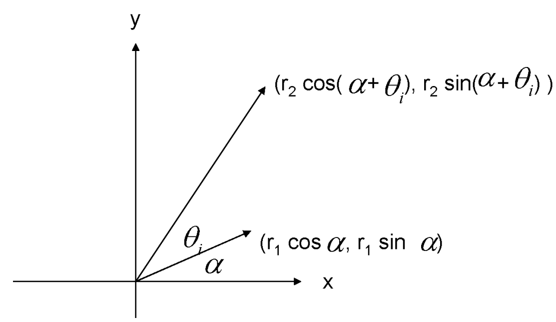

Figure 1: Polar coordinate representation for a sub-matrix

$G_{i,j}$ can be represented as

$$G_{ij} = \begin{pmatrix} r_i\cos\theta_1 & r_j\cos\theta_j \\ r_i\sin\theta_1 & r_j\sin\theta_j \end{pmatrix}$$

Therefore,

$$G_{ij}^T G_{ij} = \begin{pmatrix} r_i^2 & r_i r_j\cos(\theta_j-\theta_i) \\ r_i r_j\cos(\theta_j-\theta_i) & r_j^2 \end{pmatrix}$$

Note that $G_{ij}^T G_{ij}$ is symmetric and positive definite, therefore, all its eigenvalues are positive real numbers. Let $\lambda_{max}(G_{ij}^T G_{ij})$ denote the maximum eigenvalue of $G_{ij}^T G_{ij}$ and $\lambda_{min}(G_{ij}^T G_{ij})$ denote the minimum eigenvalue of $G_{ij}^T G_{ij}$, then

$$\begin{aligned} \lambda_{max}(G_{ij}^T G_{ij}) &= \frac{r_i^2+r_j^2}{2} + \\ & \sqrt{\frac{(r_i^2+r_j^2)^2}{2} + r_i^2 r_j^2(\cos(\theta_j-\theta_i)-1)} \end{aligned}$$

$$\begin{aligned} \lambda_{min}(G_{ij}^T G_{ij}) &= \frac{r_i^2+r_j^2}{2} - \\ & \sqrt{\frac{(r_i^2+r_j^2)^2}{2} + r_i^2 r_j^2(\cos(\theta_j-\theta_i)-1)} \end{aligned}$$

Therefore,

$$\begin{aligned} \kappa(G_{ij}) &= \sqrt{\frac{\lambda_{max}(G_{ij}^T G_{ij})}{\lambda_{min}(G_{ij}^T G_{ij})}} \\ &= \sqrt{\frac{1+\sqrt{1+\frac{4(\cos^2(\theta_j-\theta_i)-1)}{\frac{r_i^2}{r_j^2}+\frac{r_i^2}{r_j^2}}}}{1-\sqrt{1+\frac{4(\cos^2(\theta_j-\theta_i)-1)}{\frac{r_i^2}{r_j^2}+\frac{r_i^2}{r_j^2}}}}} \\ &\geq \sqrt{\frac{1+|\cos(\theta_j-\theta_i)|}{1-|\cos(\theta_j-\theta_i)|}} \end{aligned} \tag{6}$$

The equality in (6) **is achieved** when $r_i = r_j$.

Note that the relationship (6) is for any 2×2 sub-matrix of $G_{2\times n}$, therefore, the $G_{2\times n}$ that solve problem (4) has to satisfy $r_1 = r_2 = \ldots = r_n = r$. Note that for any matrix M, $\kappa(M) = \kappa(rM)$, therefore, during the computation of $f(2,n)$, it is enough to just consider $G_{2\times n}$ whose $r_1 = r_2 = \ldots = r_n = 1$.

When $r_1 = r_2 = \ldots = r_n = 1$, columns of $G_{2\times n}$ can be treated as vectors on a unit circle centered at (0,0). If there

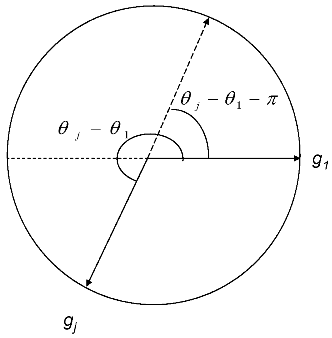

Figure 2: If $(\theta_j - \theta_1)$ is larger than π, then there is a vector $g = -g_j$ for which the angle $(\theta - \theta_1)$ is less than π and $|\cos(\theta_j - \theta_1)| = |\cos(\theta - \theta_1)|$.

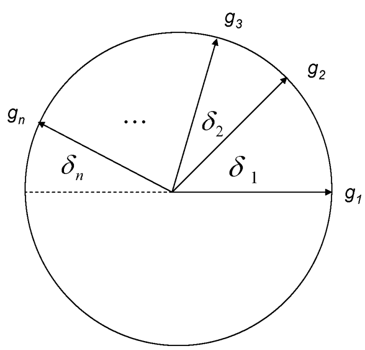

Figure 3: Adjust columns of G such that $\delta_1 + \delta_2 + \ldots + \delta_n = \frac{\pi}{2}$.

is a vector g_j for which the angle $(\theta_j - \theta_1)$ is larger than π (see g_j in Figure 2), then there is a vector $g = -g_j$ for which the angle $(\theta - \theta_1)$ is less than π and $|\cos(\theta_j - \theta_1)| = |\cos(\theta - \theta_1)|$. Therefore, it is enough to just consider $G_{2\times n}$ for which the angle between g_1 and any other g_j is less than π during the calculation of $f(2, m)$. Without affecting the calculation of $f(2, m)$, the columns of $G_{2\times n}$ can be exchanged so that the angle between g_1 and g_j increases as j increases (see Figure 3 for such an arrangement). Let δ_i denote the angle between the newly re-arranged g_i and g_{i+1} for $i = 1, 2, \ldots, n-1$ and δ_n denote the angle between g_n and $-g_1$, then

$$\delta_1 + \delta_2 + \ldots + \delta_n = \frac{\pi}{2} \tag{7}$$

The angle between g_i and g_j is

$$\theta_j - \theta_i = \delta_i + \ldots + \delta_{j-1} \tag{8}$$

Therefore,

$$\begin{aligned}
f(2,n) &= \min_{G_{2\times n}} \{\max_{ij}\{\kappa(G_{ij})\}\} \\
&= \min_{r_1=\ldots=r_n=1} \{\max_{ij}\{\kappa(G_{ij})\}\} \\
&= \min_{\delta_1,\ldots,\delta_n} \left\{ \max_i \left\{ \sqrt{\frac{1+|\cos(\theta_j-\theta_i)|}{1-|\cos(\theta_j-\theta_i)|}} \right\} \right\} \\
&= \min_{\delta_1,\ldots,\delta_n} \left\{ \sqrt{\frac{1+|\cos(\min_i\{\delta_i\})|}{1-|\cos(\min_i\{\delta_i\})|}} \right\} \\
&\geq \min_{\delta_1,\ldots,\delta_n} \left\{ \sqrt{\frac{1+|\cos(\frac{\sum_i^n \delta_i}{n})|}{1-|\cos(\frac{\sum_i^n \delta_i}{n})|}} \right\} && (9) \\
&= \sqrt{\frac{1+\cos(\frac{\pi}{n})}{1-\cos(\frac{\pi}{n})}} && (10)
\end{aligned}$$

The equality in (9) **is achieved** when

$$\delta_1 = \delta_2 = \ldots = \delta_n = \frac{\pi}{n}. \tag{11}$$

There is infinite number of optimal $2 \times n$ matrices (codes) with non-zero elements that satisfy (11).

The following is a sample optimal $2 \times n$ matrix (i.e. a sample numerically best real number erasure correcting code for 2 erasures)

$$G = \begin{pmatrix} \cos\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \ldots & \frac{(2n-1)\pi}{2n} \\ \sin\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \ldots & \frac{(2n-1)\pi}{2n} \end{pmatrix}$$

$\square$

Note that when n is large,

$$\begin{aligned}
f(2,n) &= \sqrt{\frac{1+\cos\frac{\pi}{n}}{1-\cos\frac{\pi}{n}}} \\
&\approx \frac{2n}{\pi}
\end{aligned}$$

Therefore, the condition number of the worst conditioned sub-matrix of even the numerically best real-number codes increase to infinite approximately linearly when the number of original data items (processors) n increases. It is **impossible** for even the numerically best 2-erasure code to correct **all** possible 2-erasures when the number of data items (processors) is large. The introduced numerical errors can be arbitrarily large during recovery when n is arbitrily large.

In order to guarantee to correct **ALL** possible 2-erasures in IEEE standard 754 floating point numbers (16 digits of accuracy) with k digits of accuracy The total number of data items (processors) n has to satisfy

$$n \leq 10^{16-k} \times \frac{\pi}{2} \tag{12}$$

If $n \geq 10^{16} \times \frac{\pi}{2}$, all 16 digits in the IEEE standard 754 floating point numbers will be lost. However, this can be avoided by divide n processors into sub-groups of the size s and encode the input matrices within each sub-group. In order to guarantee to correct all possible 2-erasures with k digits of accuracy in each sub-group, the number of processors s in each sub-group has to satisfy

$$s \leq 10^{16-k} \times \frac{\pi}{2}$$

Therefore, **with the increase of the redundancy information**, we can guarantee to correct all possible 2-erasures with k digits of accuracy.

5. CONSTRUCT NUMERICALLY GOOD CODES BY UNCONSTRAINT OPTIMIZATION

As discussed in Section 3, it is one of Smale's 18 unsolved mathematic problems [42] in the twenty-first century to obtain analytical solution for the minimax problem specified in (3) even if $m = 3$ and G is restricted on matrices with unit norm columns. Instead of solving (3) directly, in this section, we propose to compute approximate solutions of (3) by solving another unconstrained optimization problem. We prove that, for $m = 2$, the solution obtained by solving the new unconstraint optimization problem is the same as the solution obtained by solving (3) directly.

Inspired by the $m = 2$ case, in what follows, we restrict the choice of the generator matrix G within matrices whose column g_j satisfy $||g_j||_2 = g_j^T g_j = 1$, where $j = 1, 2, \ldots, n$. We restrict the choice of sub-matrix within $m \times m$ matrices.

Let G_j denotes the j^{th} $m \times m$ sub-matrix of $G_{m\times n}$ (the order here can be any order one likes). Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$ denotes the m eigenvalues of $G_j^T G_j$, then

$$\det(G_j^T G_j) = \prod_{i=1}^{m} \lambda_i$$

$$\sum_{i=1}^{m} \lambda_i = tr(G_j^T G_j) = m$$

$$1 \leq \lambda_1 < m$$

Therefore,

$$\begin{aligned}
\det(G_j) &= \sqrt{\det(G_j^T G_j)} \\
&= \sqrt{\prod_{i=1}^{m} \lambda_i} \\
&= \sqrt{\frac{\lambda_1 . \prod_{i=1}^{m-1} \lambda_i}{\kappa(G_j^T G_j)}} \\
&\leq \sqrt{\frac{\left(\frac{\lambda_1 + \sum_{i=1}^{m-1} \lambda_i}{m}\right)^m}{\kappa(G_j^T G_j)}} \\
&\leq \frac{2^{\frac{m}{2}}}{\kappa(G_j)}
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\det(G_j) &= \sqrt{\det(G_j^T G_j)} \\
&= \sqrt{\prod_{i=1}^{m} \lambda_i} \\
&\geq \sqrt{\lambda_m^m} \\
&\geq \sqrt{\frac{1}{\left(\frac{\lambda_1}{\lambda_m}\right)^m}} \\
&= \frac{1}{\kappa(G_j)^m}
\end{aligned}$$

Therefore, for a fixed m, if $\det(G_j)$ is small, then $\kappa(G_j)$ will be large. if $\kappa(G_j)$ is large, then $\det(G_j)$ will be small.

Note that,

$$\begin{aligned}
\det G_j &= \sqrt{\det(G_j^T G_j)} \\
&= \sqrt{\prod_{i=1}^{m} \lambda_i} \\
&\leq \sqrt{\left(\frac{\sum_{i=1}^{m} \lambda_i}{m}\right)^m} \\
&= 1
\end{aligned}$$

Therefore, in order to make the sub-matrices of G well-conditioned, we need to maximize the determinants of the sub-matrices. If any sub-matrix of G has a large condition number, then $\prod_{i=1}^{m} \det(G_i)$ will be small. Therefore, we propose to approximate the numerically best codes by solving the following optimization problem.

$$h(m, n) = \max_{G_{m\times n} \in \mathcal{R}^{m\times n},\ ||g_j||_2 = 1} \left\{ \prod_i \det(G_i) \right\} \tag{13}$$

If m dimensional polar coordinate systems are used to represent the elements of $G_{m\times n}$, then the constrained optimization problem (13) becomes an unconstrained optimization problem. Standard unconstrained optimization techniques can then be used to solve this maximization problem.

The solution (matrix G) obtained by solving (13) usually produce numerically very good real-number codes (see Section 5 for experimental comparisons to currently known best code).

To our surprise, when $m = 2$, the optimal matrix G obtained by solving (13) is exactly the same as the numerically best code obtained by solving (4) directly.

Theorem 2. *Let*

$$h(2, n) = \max_{G_{2\times n} \in \mathcal{R}^{m\times n},\ ||g_j||_2 = 1} \left\{ \prod_i \det(G_i) \right\} \tag{14}$$

Then,

$$h(2, n) = \prod_{j=1}^{n-1} \left\{ \sin^{n-j} \frac{j\pi}{n} \right\} \tag{15}$$

The following generator matrix is one of the solutions for (5):

$$G = \begin{pmatrix} \cos\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \ldots & \frac{(2n-1)\pi}{2n} \\ \sin\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \ldots & \frac{(2n-1)\pi}{2n} \end{pmatrix}$$

Proof. When $m = 2$, if we use the same polar coordinate system as in Section 3.4 and exchange columns of $G_{2\times n}$ similarly to get the arrangement g_j in Figure 4, then

$$G_{2\times n} = \begin{pmatrix} \cos\theta_1 & \ldots & \cos\theta_n \\ \sin\theta_1 & \ldots & \sin\theta_n \end{pmatrix}$$

$$\theta_j = \theta_1 + \sum_{k=1}^{j-1} \delta_k$$

The sub-matrix G_{ij} (consisting of the column i and j of $G_{2\times n}$) can be represented as

$$G_{ij} = \begin{pmatrix} \cos\theta_1 & \cos\theta_n \\ \sin\theta_1 & \sin\theta_n \end{pmatrix}$$

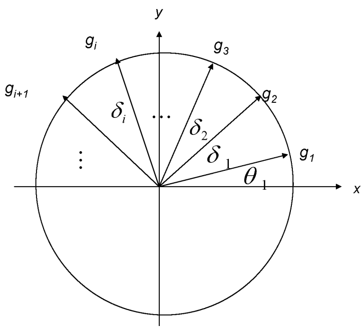

Figure 4: Adjust columns of G such that $\theta_j = \theta_1 + \sum_{k=1}^{j-1} \delta_k$.

Therefore,

$$\det(G_{ij}) = \sin(\theta_j - \theta_i)$$

$$\begin{aligned}
\prod_{j>i} \det(G_{ij}) &= \prod_{j>i} \sin(\theta_j - \theta_i) \\
&= \prod_{j>i} \sin(\sum_{k=i}^{j-1} \delta_k) \\
&= (\sin\delta_1 \ldots \sin\delta_{n-1}) \\
&\quad (\sin(\delta_1+\delta_2) \ldots \times \sin(\delta_{n-2}+\delta_{n-1})) \\
&\quad \vdots \\
&\quad \sin(\delta_1 + \ldots + \delta_{n-1})
\end{aligned}$$

Note that, when $x_i > 0$

$$\prod_{i=1}^{n} x_i \leq \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^n.$$

When $x_1 = x_2 = \ldots = x_n = x$, the equality **is achieved** and

$$\prod_{i=1}^{n} x_i = x^n.$$

Therefore, when $\delta_1 = \delta_2 = \ldots = \delta_{n-1} = \delta$, for all $j = 0, 1, \ldots, n-2$, $\prod_{k=0}^{n-1-j} \sin(\sum_{t=0}^{j} \delta_{k+t})$ achieve their maximum $\sin^{n-1-j}((j+1)\delta)$ **at the same time**. Therefore, $\prod_{j>i} \det(G_{ij})$ achieves its maximum

$$\prod_{j>i} \det(G_{ij}) = \sin^{n-1}\delta \sin^{n-2} 2\delta \ldots \sin^1((n-1)\delta)$$

Rearrange the right hand side of the above formula, we have

$$\begin{aligned}
\prod_{j>i} \det(G_{ij}) &= (\sin^{n-1}\delta \times \sin^1((n-1)\delta) \\
&\quad (\sin^{n-2} 2\delta \times \sin^2((n-1)\delta) \\
&\quad (\sin^{n-3} 3\delta \times \sin^3((n-3)\delta) \\
&\quad \vdots
\end{aligned}$$

When $\sin j\delta = \sin((n-j)\delta)$, **all** $\sin^{n-j} j\delta \times \sin^j((n-j)\delta)$ achieve, **at the same time**, their maximum

$$\left(\frac{(n-j)\sin j\delta + j\sin((n-j)\delta)}{n}\right)^n = (\sin j\delta)^n.$$

Note that, when $\delta_1 = \delta_2 = \ldots = \delta_{n-1} = \delta$,

$$\pi = \delta_1 + \ldots + \delta_{n-1} + \delta_n = (n-1)\delta + \delta_n$$

Therefore,

$$\delta < \frac{\pi}{n-1}$$

Therefore, $\sin j\delta = \sin((n-j)\delta)$ implies

$$j\delta = \pi - (n-j)\delta$$

Therefore, implies

$$\delta = \frac{\pi}{n}$$

When $\delta_1 = \delta_2 = \ldots = \delta_{n-1} = \delta$,

$$\delta_n = \pi - (\delta_1 + \ldots + \delta_{n-1}) = \frac{\pi}{n}.$$

Therefore, when $\delta_1 = \delta_2 = \ldots = \delta_n = \frac{\pi}{n}$, $\prod_i \det(G_i)$ achieve its maximum.

□

6. EXPERIMENTAL RESULTS

The numerical properties of real number codes from Vandermonde matrices, Cauchy matrices, DCT matrices, DFT matrices, and Gaussian random matrices has been fully analyzed and compared in [11]. Experimental results indicate that real number codes from Gaussian random matrices are much more stable than the other codes.

In this section, we will compare the numerically best codes with real number codes from Gaussian random matrices and Grassmannian frame matrices. When the number of erasures $m = 2$, there are exact analytical expressions for the generator matrices of both the Grassmannian code and the numerically best code. The numerically best code are the same as the Grassmannian code when $m = 2$. They are both numerically optimal. However, when the number of erasures $m \geq 3$, the Grassmannian code is not optimal anymore. Therefore, we focus on the comparison the numerically stability of these codes for more than two erasures.

When the number of erasures $m \geq 3$, most of time, there are no exact analytical expressions for the generator matrices of all three codes except for very few combinations of m and n. Therefore, most of time, we have to use approximation codes in practice.

However, for $m = 3$ and $n = 10$, the mathematically optimal (without any computational approximation) Grassmanian (packing) codes are given in [18]. It is a hexakis bi-antiprism. The columns of the corresponding generator matrix (i.e. the coordinates of the 10 points in three dimensional space) are given in [2]. Therefore, in this paper, we choose to compare the numerical stability of the three codes to tolerate three failures in ten processors. Table 2 gives the corresponding generator matrix of the Grassmannian code.

It is mathematically difficult to obtain analytical expressions for the numerically best codes for three or more erasures. Therefore, for numerically best codes, we use the approximation codes computed by solving the unconstrained

Table 1: A generator matrix from Gaussian random matrices with mean $\mu = 0$ and standard deviation $\sigma = 1$.

g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1
0.0582	-0.2290	0.1256	-1.1022	-2.6053	-2.0564	-0.0062	-1.0216	-0.9579	-2.0886
-1.6885	1.0350	-1.2976	0.7591	-0.8609	-0.7067	-1.3709	-1.9139	-0.7915	0.5943
-1.2755	-1.5523	-0.8135	0.3585	0.0536	-0.9256	-0.4202	-0.8843	-0.8012	0.8242

Table 2: A generator matrix from Grassmannian frames matrices

g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1
1.0000	0.6101	0.6101	0.6101	0.6101	0.6101	0.6101	0	0	0
0	0.7923	0.3961	-0.3961	-0.7923	-0.3961	0.3961	0.8660	-0.8660	0
0	0	0.6861	0.6861	0	-0.6861	-0.6861	0.5000	0.5000	-1.0000

optimization problem in Section 4 to participate the comparison. Table 3 gives the corresponding generator matrix of the numerically best code.

Gaussian random codes are simple to generate using a pseudo Gaussian random number generator. Table 1 gives the corresponding generator matrix of the Gaussian random code. The matrix is generated using MATLAB. Actually, this code is only a statistical approximation of the Gaussian random codes. However, this is how we generate and use Gaussian random codes in practice.

It is well known [27] that, in solving a linear system of equations, a condition number of 10^k for the coefficient matrix leads to a loss of accuracy of about k decimal digits in the solution. The coefficient matrix of the system of equations to be solved during recovery can be any square sub-matrix (including minor) of the generator matrix. Therefore, in what follows, we focus on comparing the condition numbers of the sub-matrices of all three generator matrices.

The size of the generator matrices is 3×10, therefore, the total number of 3×3 sub-matrices in each generator matrix is 120.

Table 4 gives the condition numbers of the 10 worst conditioned 3×3 sub-matrices in all three generator matrices. Table 4 demonstrates that the condition numbers of all 10 worst-conditioned 3×3 sub-matrices of the numerically best code are much more smaller than that of the other two codes. Therefore, in the worst case scenarios, the numerically best code is numerically much more stable than both the Gaussian random code and the Grassmannian code. Condition number is a property that associated with more than two columns of a matrix. The Grassmannian codes maximizes only the minimum angle between any two columns of the generator matrix. When the minimum angle between any two columns of the generator matrix achieves its global maximum, it is still possible that three columns of a generator matrix are in the same plan, therefore, the generator matrix contains a singular sub-matrix. This is exactly the reason why we get one singular sub-matrix in the Grassmannian codes. Therefore, Grassmannian codes are generally **NOT** optimal unless $m = 2$ where a sub-matrix only contains 2 columns. The numerically best code minimizes the maximum condition numbers of all sub-matrices, therefore, has a much better numerically stability in the worst case scenarios.

Figure 5 gives the distribution of all 120 condition numbers of all 120 3×3 sub-matrices for both the Grassmannian

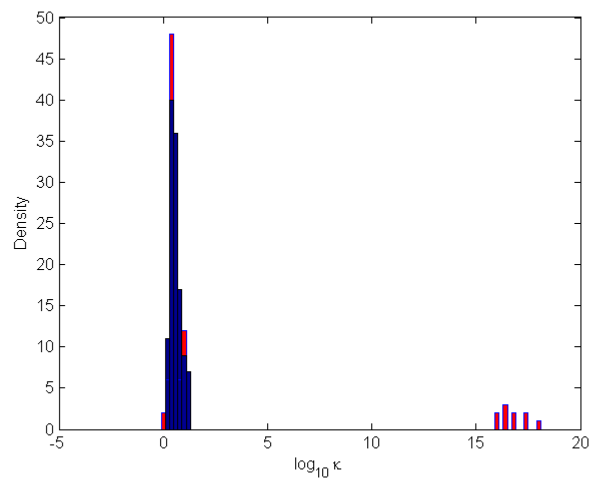

Figure 5: Condition number distribution for Grassmannian frame codes and optimal codes.

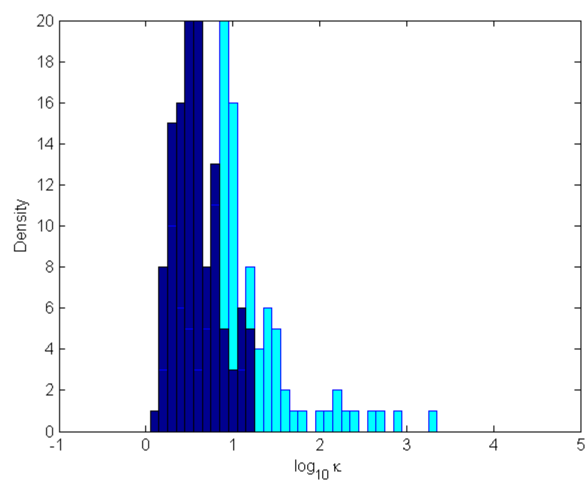

Figure 6: Condition number distribution for Gaussian random codes and optimal codes.

Table 3: A generator matrix from numerically best real number codes

g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1
-0.5566	0.1467	0.7247	0.9919	0.4631	-0.6691	0.5614	-0.2353	0.0686	-0.6749
0.8095	0.7985	0.4905	0.1217	-0.1332	0.2351	-0.6914	-0.0325	0.9466	-0.5804
0.1871	0.5839	0.4839	0.0365	0.8763	0.7050	0.4547	0.9714	-0.3149	0.4556

Table 4: The condition numbers of the 10 worst-conditioned 3×3 sub-matrices in different generator matrices

Grass	$0.1 * 10^{17}$	$0.1 * 10^{17}$	$0.3 * 10^{17}$	$0.3 * 10^{17}$	$0.3 * 10^{17}$	$0.5 * 10^{17}$	$0.7 * 10^{17}$	$2.3 * 10^{17}$	$2.3 * 10^{17}$	Inf
Rand	111	131.3	145.1	168.6	199	250.4	366.7	457.4	786.7	1891.1
Best	12.1652	12.4318	12.4371	13.2190	13.5483	14.7503	15.6580	16.1104	16.1609	16.5355

code (red) and the numerically best code (blue). Figure 5 demonstrates that the numerically best code are at least as good as the Grassmannian code in average cases.

Figure 6 shows the distribution of all 120 condition numbers of all 120 3×3 sub-matrices for both Gaussian random code (cyan) and the numerically best code (blue). Figure 6 indicates that the numerically best code are much more stable than Gaussian random codes in average cases.

7. CONCLUSION

In this paper, we present a class of numerically best real-number codes for fault tolerant matrix operations on large HPC systems. We give an analytical expressions for the numerically best erasure correcting codes for two erasures and develop an approximation method to computationally approximate the numerically best codes for more than two erasures. Experiment results demonstrate that our codes are numerically much more stable than existing codes.

In the near future, we would like to explore better approximation methods to computationally approximate the numerically best codes for three or more erasures.

8. REFERENCES

[1] Top 500 supercomputer sites. http://www.top500.org.

[2] http://www.research.att.com/ njas/grass/dim3

[3] J. Anfinson and F. T. Luk A Linear Algebraic Model of Algorithm-Based Fault Tolerance. *IEEE Transactions on Computers*, v.37 n.12, p.1599-1604, December 1988.

[4] P. Banerjee, J. T. Rahmeh, C. B. Stunkel, V. S. S. Nair, K. Roy, V. Balasubramanian, and J. A. Abraham Algorithm-based fault tolerance on a hypercube multiprocessor. *IEEE Transactions on Computers*, vol. C-39:1132–1145, 1990.

[5] V. Balasubramanian and P. Banerjee Compiler-Assisted Synthesis of Algorithm-Based Checking in Multiprocessors. *IEEE Transactions on Computers*, vol. C-39:436-446, 1990.

[6] A. Bouteiller, P. Lemarinier, G. Krawezik, and F. Cappello. Coordinated checkpoint versus message log for fault tolerant MPI. *Procceedings of International Conference on Cluster Computing (Cluster 2003)*, Honk Hong, December, 2003.

[7] L. S. Blackford, J. Choi, A. Cleary, A. Petitet, R. C. Whaley, J. Demmel, I. Dhillon, K. Stanley, J. Dongarra, S. Hammarling, G. Henry, and D. Walker. ScaLAPACK: a portable linear algebra library for distributed memory computers - design issues and performance. In *Supercomputing '96: Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM)*, page 5, 1996.

[8] D. L. Boley, R. P. Brent, G. H. Golub, and F. T. Luk. Algorithmic fault tolerance using the lanczos method. *SIAM Journal on Matrix Analysis and Applications*, 13:312–332, 1992.

[9] E. Candes, M. Rudelson, L. Tao, R. Vershynin Error Correction via Linear Programming. *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS05)*, IEEE, 2005. pp. 295-308

[10] Z. Chen and J. Dongarra. Numerically Stable Real-Number Codes Based on Random Matrices. In *ut-cs-04-526* June 9, 2004.

[11] Z. Chen and J. Dongarra. Numerically stable real number codes based on random matrices. In *Proceeding of the 5th International Conference on Computational Science (ICCS2005)*, Atlanta, Georgia, USA, May 22-25, 2005. LNCS 3514, Springer-Verlag.

[12] Z. Chen and J. Dongarra. Condition Numbers of Gaussian Random Matrices. *SIAM Journal on Matrix Analysis and Applications*, Volume 27, Number 3, Page 603-620, 2005.

[13] Z. Chen, and J. Dongarra. Algorithm-Based Fault Tolerance for Fail-Stop Failures. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 19, No. 12, December, 2008.

[14] Z. Chen, and J. Dongarra. Highly Scalable Self-Healing Algorithms for High Performance Scientific Computing. *IEEE Transactions on Computers*, July, 2009.

[15] Z. Chen, G. E. Fagg, E. Gabriel, J. Langou, T. Angskun, G. Bosilca, and J. Dongarra. Fault tolerant high performance computing by a coding approach. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP 2005, June 14-17, 2005, Chicago, IL, USA*. ACM, 2005.

[16] Z. Chen, G. E. Fagg, E. Gabriel, J. Langou, T. Angskun, G. Bosilca, and J. Dongarra. Building Fault Survivable MPI Programs with FT-MPI Using Diskless Checkpointing. In *University of Tennessee*

Computer Science Department Technical Report. Technical Report UT-CS-04-540, 2004.

[17] Z. Chen. *Scalable techniques for fault tolerant high performance computing.* Ph.D. thesis, University of Tennessee, Knoxville, TN, USA, 2006.

[18] J. H. Conway, R. H. Hardin and N. J. A. Sloane Packing Lines, Planes, etc.: Packings in Grassmannian Spaces. *Experimental Mathematics*, Vol. 5, No. 2, 1996

[19] D. L. Donoho For most large undetermined systems of linear equations the minimal ‘1-norm near-solution is also the sparsest near-solution. *Communications on Pure and Applied Mathematics*, Volume 59 Issue 6, Pages 797 - 829

[20] D. L. Donoho Compressed sensing. *IEEE Trans. on Information Theory*, 52(4), pp. 1289 - 1306, April 2006

[21] A. Edelman. Eigenvalues and condition numbers of random matrices. *SIAM J. Matrix Anal. Appl.*, 9(4):543–560, 1988.

[22] Ferreira, P. Stability issues in error control coding in complex field, interpolation, and frame bounds. *IEEE Signal Processing Letters*, vol.7 No.3,(2000) pp.57-59.

[23] Ferreira, P., Vieira, J. Stable DFT codes and frames, *IEEE Signal Processing Letters*, vol.10 No.2,(2003) pp.50-53.

[24] V. K. Goyal and J. Kovacevic Quantized Frame Expansions with Erasures *Applied and Computational Harmonic Analysis* vol.10, 203 233 (2001)

[25] J. Gunnels, R. van de Geijn, D. Katz, E. Quintana-Ort Fault-Tolerant High-Performance Matrix Multiplication: Theory and Practice *Proceedings of the 2001 International Conference on Dependable Systems and Networks (DSN'01)* , Washington, DC, USA, 2001.

[26] P. Alpatov, G. Baker, C. Edwards, J. Gunnels, G. Morrow J. Overfelt, R. van de Geijn, and J. Wu. PLAPACK: Parallel Linear Algebra Libraries Design Overview *Proc. of the SC97 Conference*, ACM, San Diego, CA, 1997.

[27] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The John Hopkins University Press, , 1989.

[28] C. N. Hadjicostis. *Coding Approaches to Fault Tolerance in Combinational and Dynamic Systems*, Kluwer Academic Publishers, 2002.

[29] K.-H. Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Transactions on Computers*, vol. C-33:518–528, 1984.

[30] Y. Kim. *Fault Tolerant Matrix Operations for Parallel and Distributed Systems.* Ph.D. dissertation, University of Tennessee, Knoxville, June

[31] D. E. Knuth. *The Art of Computer Programming*, Addison-Wesley Professional, 2 edition, October 15, 1998.

[32] J. Langou, Z. Chen, G. Bosilca, and J. Dongarra. Recovery Patterns for Iterative Methods in a Parallel Unstable Environment *SIAM Journal on Scientific Computing*, 30(1):102-116, 2007.

[33] F. T. Luk and H. Park An analysis of algorithm-based fault tolerance techniques. *SPIE Adv. Alg. and Arch. for Signal Proc.*, vol. 696, 1986, pp. 222-228.

[34] T. Marshall Coding of Real-Number Sequences for Error Correction: A Digital Signal Processing Problem. *IEEE Journal on Selected Areas in Communications*, Volume 2, Issue 2, Mar 1984 Page(s): 381 - 392

[35] Nair, S. S. and Abraham, J. A.: Real-number codes for fault-tolerant matrix operations on processor arrays, IEEE Transactions on Computers, vol. C-39,(1990) pp.300-304.

[36] J. S. Plank, Y. Kim, and J. Dongarra. Fault Tolerant Matrix Operations for Networks of Workstations Using Diskless Checkpointing. *IEEE Journal of Parallel and Distributed Computing*, 43, 125-138 (1997).

[37] J. S. Plank, K. Li, and M. A. Puening. Diskless checkpointing. *IEEE Trans. Parallel Distrib. Syst.*, 9(10):972–986, 1998.

[38] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software – Practice & Experience*, 27(9):995–1012, September 1997.

[39] J. L. Sung and G. R. Redinbo. Algorithm-Based Fault Tolerant Synthesis for Linear Operations. *IEEE Transactions on Computers*, Volume 45 , Issue 4, April 1996.

[40] G. Robert Redinbo. Generalized Algorithm-Based Fault Tolerance: Error Correction via Kalman Estimation. *IEEE Transactions on Computers*, Volume 47, Issue 6,June, 1998.

[41] G. Stellner. CoCheck: Checkpointing and process migration for MPI. *Proceedings of the 10th International Parallel Processing Symposium (IPPS'96)*, Honolulu, Hawaii, April, 1996.

[42] S. Smale. Mathematical Problems for the Next Century. *Mathematics: Frontiers and Perspectives*, Ed. V. Arnold, M. Atiyah, P. Lax, and B. Mazur, Providence, RI: Amer. Math. Soc., 2000.

[43] T. Strohmer and R. W. Heath Grassmannian frames with applications to coding and communication *Applied and Computational Harmonic Analysis*, Volume 14, Issue 3, Pages 257-275, May 2003.

[44] B. Schroeder and G. A. Gibson. A large-scale study of failures in high-performance computing systems. *Proceedings of the International Conference on Dependable Systems and Networks (DSN2006)*, Philadelphia, PA, USA, June 25-28, 2006.

[45] B. Schroeder and G. A. Gibson. Understanding Failures in Petascale Computers. *Journal of Physics: Conference Series*, 78, 2007.

[46] G. A. Gibson, B. Schroeder, and J. Digney. Failure Tolerance in Petascale Computers. *CTWatchQuarterly*, Volume 3, Number 4, November 2007.

[47] C. Wang, F. Mueller, C. Engelmann, and S. Scot. Job Pause Service under LAM/MPI+BLCR for Transparent Fault Tolerance. In *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium, March, 2007, Long Beach, CA, USA.*

[48] S. J. Wang and N.K. Jha. Algorithm-Based Fault Tolerance for FFT Networks. *IEEE Transactions on Computers*, Volume 43, Issue 7,July, 1994.