"Real" and "Complex" Network Codes

Saurabh Shintre¹, Sachin Katti², Sidharth Jaggi³,
Bikash Kumar Dey¹, Dina Katabi², Muriel Médard²

¹Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India
²CSAIL/LIDS, Massachusetts Institute of Technology, Cambridge, MA
³Department of Information Engineering, Chinese University of Hong Kong, Hong Kong
{saurabh,bikash}@ee.iitb.ac.in, {skatti,dk,medard}@mit.edu, jaggi@ie.cuhk.edu.hk

Abstract. Algebraic approaches to Network Coding focuses on considering data as element of a finite field (FFNC) and use finite field algebra at the interior nodes. As an alternative, we propose Arithmetic Network Codes (ANC) in which the nodes perform finite precision arithmetic over real and complex fields. We demonstrate the use of such codes in Multi-Resolution Transmission and natural advantages gained in Wireless Networks. We show, through simulations, that although real field can be considered as an asymptotically large finite field, the performance of similar coding strategies give different results in both scenarios. Our results aim at finding the parameters that govern performance of ANC.

1. Introduction

The paradigm of network coding [1], i.e., coding at all nodes of a network rather than just at the edges, has been intensively studied recently. It has been shown that for a variety of problems “Min-Cut” of the network can be achieved by performing linear operations at the intermediate nodes [2]. A typical construction of such network codes considers data as element of a finite field \( \mathbb{F}_q \) and performs algebra at the intermediate node. It has been shown that sufficiently large field size \( q \), such construction achieves min-cut with probability close to 1 [3].

However, for sources or networks better described by real (or complex) arithmetic, as in some real world applications like images and sounds, it might be advantageous to consider network codes that are linear over real (or complex) fields. We motivate ANCs via two “real world” communication problems – multi-resolution multicast, and wireless multicast.

1.1. Multi-resolution multicast

A natural generalization to the multicast problem is to consider multi-resolution multicast, i.e., allowing different receivers to have different download capacities. Suppose each sink wishes to reproduce with minimum distortion, a sequence of real numbers from the source, which for a real case can be pixel values. For the usual implementation of FFNCs, even if a receiver’s capacity is only marginally less than the source’s rate, the minimum

*Submitted as “Real” and “Complex” Network Codes: Promises and Challenges to IEEE Workshop on Network Coding and Applications, NetCod, 2008.
mean-square error of the receiver’s estimate can still be high. The problem arises since the $\mathbb{R}$-vector space in which the source’s data is naturally embedded is quite different from the $\mathbb{F}_q$-vector space over which arithmetic is performed by the network’s nodes. Hence, the resulting distortion can be much higher. Instead, building on powerful results from the field of compressed sensing [4], ANCs that can be designed and implemented in a low complexity and distributed manner are shown [5] to obtain a graceful trade-off between throughput and distortion.

1.2. Wireless network coding

Wireless networks are significantly different from wired networks for several reasons – their broadcast nature, and the presence of interference, noise, and fading. Current network codes usually only exploit the broadcast property [6] and try to avoid interference. Most code designs for wireless networks assume noiseless links, and either no interference due to careful scheduling or that interfering packets are dropped. These assumptions essentially transform the wireless network coding problem into something like a wired network coding problem.

Consider the example illustrated in Fig. 1. The two sources $S_1$ and $S_2$ want to communicate two real numbers $r_1$ and $r_2$ respectively to each other over a wireless channel. However, they are not in each other’s transmission range and so they have to communicate via a common repeater $R$. With only routing, this requires four time slots as shown in Fig. 1(a). With FFNCs, the repeater can broadcast $r_1 + r_2$ to both $S_1$ and $S_2$ in a single time slot and thus require only three time slots (Fig. 1(b)). However, with appropriate power control and synchronization, if $S_1$ and $S_2$ simultaneously transmit $r_1$ and $r_2$ respectively, then $R$ receives $r_1 + r_2$. $R$ can then broadcast $r_1 + r_2$ to both $S_1$ and $S_2$ in just two time slots (Fig. 1(c)).
2. Challenges
The above scenarios, and the work outlined in [5] indicate the promising possibilities of ANCs. As FFNCs have been studied very well and it has been shown that for sufficiently large field size the probability of transform being invertible is close to 1. As real field can be considered as a finite field of infinite size, similar results should hold. Intuitively, the performance of FFNC over \( \mathbb{F}_{2^m} \) should be comparable to \( m \)-bit finite precision ANC. We show that this is not true. For FFNCs two transforms of the same rank can transmit the same precision but over finite precision arithmetic two transforms of the same rank can differ in terms of the precision provided. This in turn results from the fact that for ANC operations are over a normed vector space and hence precision depends not only on the rank but also on its numerical stability. We show that a well studied quantity “Condition Number” of the transform (explained in Section 4) is a measure of the numerical stability and plays a big role in the construction of ANC. It also remains a big challenge to find transform with good (meaning small) condition number because random choice of matrices fail to give good performance.

3. Example
We illustrate our point with an example. Consider the line network shown in Figure 2 with one source, one sink, and \( k \) serial relays. There are \( C \) links between each consecutive pair of nodes. We consider a simple code where each node performs the same linear transform \( M \). This is for ease of exposition – similar results can be obtained even if different operations are performed at each node. We consider one FFNC, and two kinds of ANCs.

First, we examine the performance of an FFNC. The first node chooses a random \( C \times C \) matrix \( M \) with coefficients chosen uniformly i.i.d. from \( \mathbb{F}_q \). The probability that \( M \) is singular can be shown to be at most \( 1 - \frac{C}{q} \) [3]. Let \( q \) equal \( 2^m \). Then the probability that after \( k \) hops the linear transform is not invertible is at most \( C 2^{-m} \), which is independent of \( k \).

Second, suppose the coefficients of \( M \) are chosen from \( \mathbb{R} \). Each node applies the transform \( M \) with \( m \)-bit precision. Let the eigenvalues of largest and smallest magnitude of \( M \) be \( \lambda_l \) and \( \lambda_s \) respectively. Then with probability 1, \( |\lambda_l| \neq |\lambda_s| \). After \( k \) hops, the ratio of the magnitudes of the largest and the smallest eigenvalues of the overall transform matrix is \( |\lambda_l/\lambda_s|^k \). If \( |\lambda_l/\lambda_s|^k > 2^m \), then the overall transform matrix is computationally singular. Note that this happens even though the rounding off is done only at the receiver. This seems to imply that quantization is the source of the problem, and that the overall performance must necessarily degrade exponentially with network size. However, the last part of this example shows that this degradation is significantly less for well-conditioned matrices.

As a third case, suppose \( M \) is a random unitary matrix. Therefore, all its eigenvalues are of unit magnitude. If quantization is done only at the receiver then, unlike the
second case, there is no loss of rank in the overall transform. However, let us now consider the effect of rounding off after transformation at each repeater. Every addition or multiplication of \( m \)-bit precision incurs a normalized (w.r.t. the magnitude of the result) quantization noise of at most \( 2^{-m} \). Every component of the output vector is obtained after \( C \) multiplications and \((C - 1)\) additions. So, after the transformation and rounding off at the first repeater, every component of the message vector is corrupted by a quantization error of at most \( 2C2^{-m} \). This error accumulates at each stage, and after \( k \) stages, the error in each component is at most \( 2kC2^{-m} \). So, the mean square error (MSE) after \( k \) stages is at most \( 2kC^{3/2}2^{-m} \). The MSE grows only linearly with the size of the network, i.e., the loss of precision (in bits) grows logarithmically with the size of the network.

There are several interesting issues raised by the example above. First, at a high level, in large networks the behavior of ANCs is intriguingly different from that of FFNCs. Second, the effect of quantization depends on how “well-behaved” the underlying linear transforms are – unitary transforms show much better performance than general transforms do. This leads to the third point, which is the difficulty of actually using unitary transforms as ANCs. For one, a node may have more outgoing links than incoming links. The local linear transform matrix must have more rows than columns, and therefore cannot be unitary. Further, what matters is the linear transform across cutsets of the network, and this is a global property of the code – using unitary matrices at individual nodes cannot guarantee global unitarity, especially if one wishes to design the ANC in a distributed manner. Even if one knew the entire network, it may not even be possible to design such a code.

**4. The Condition Number of a Matrix**

Consider the transform \( \tilde{y} = [A]\tilde{x} \), where \( [A] \) is a square matrix. If \([A]\) is invertible and the computation \([A]\tilde{x}\) is carried out exactly, it is possible to recover \( \tilde{x} \) as \([A]^{-1}\tilde{y}\) with no distortion. However, from the rounded-off value of \( \tilde{y} \) it is possible only to get an estimate \( \hat{x} \) of \( \tilde{x} \). If the rounding-off error in \( \tilde{y} \) is denoted by \( \Delta \tilde{y} \) and the corresponding error \( \hat{x} - \tilde{x} \) in the estimate \( \hat{x} \) is denoted by \( \Delta x \), then one would like to minimize the normalized error \( \Delta \hat{x}/\tilde{x} \). (In particular, \( \log_2(\Delta \hat{x}/\tilde{x}) \) is the loss in floating point precision for fixed normalized error \( \Delta y/\tilde{y} \) in the output.) In other words, the “goodness” of the transform \([A]\) is indicated by the maximum value of

\[
\frac{||\Delta \hat{x}/||\tilde{x}||}{||\Delta \tilde{y}/||\tilde{y}||} \leq \frac{||[A]^{-1}\Delta y/||[A]^{-1}\tilde{y}||}{||\Delta \tilde{y}/||\tilde{y}||}.
\]

This quantity can be readily seen to be equal to the product of the matrix norms of \([A]\) and its inverse. Any matrix norm that is \( \text{sub-modular} \) (i.e., satisfies \( ||[A][B]|| \leq ||[A]||||[B]|| \)) may be used. Thus the condition number of \([A]\), denoted by \( \kappa([A]) \), is defined as \( ||[A]||||[A]^{-1}|| \). Let \( a_{ij} \) and \( \hat{a}_{ij} \) respectively be the entries of \([A]\) and \([A]^{-1}\). The following properties of condition numbers will be useful.

- For the Frobenius norm on matrices, the corresponding condition number, denoted \( \kappa_F([A]) \), equals \( \sqrt{\sum a_{ij}^2} / \sqrt{\sum \hat{a}_{ij}^2} / \sqrt{\dim([A])} \).
- For the row-sum operator norm on matrices, the corresponding condition number, denoted \( \kappa_o([A]) \), equals \( \max_i | \sum a_{ij} | (\max_i \sum |a_{ij}|) \).
• $\kappa([A]) \geq 1$ and $\kappa([A]) = 1$ for unitary matrices.
• $\kappa([A][B]) \leq \kappa([A])\kappa([B])$ for any two matrices $[A], [B]$.
• $\log_2(\kappa([A]))$ is clearly the loss in precision in the estimate $\hat{x}$ due to rounding-off of $\tilde{y}$. In particular, there exist $\tilde{x} \neq \hat{x}$ such that $[A]\tilde{x} = [A]\hat{x}$.

5. Simulation Results

In this section, we present some MATLAB simulation results to illustrate that random arithmetic coding produces overall transform matrices with very large condition number. We consider the line network in Fig. 2 with $C = 2$ links from any node to the next, and the network obtained by stacking $k$ butterfly networks as shown in Fig. 3. In either case, each edge carries a vector of length $n$. Then for the line network, the transformation at each node is taken to be a $nC \times nC$ matrix with components chosen i.i.d. uniformly distributed in $[0, 1]$. The matrices at different nodes are also chosen i.i.d. The overall transform matrix from the source to the destination is computed by multiplying all the matrices. Fig. 4 shows the condition number, averaged over 100 different experiments, of the overall transform matrix as a function of $k$. The figure shows a $\log_2$ plot for each $n = 1, 2, 3, 4$.

![Figure 3. Stacked butterfly network](image)

![Figure 4. Condition number of the overall transform for the line network](image)

The results suggest that with random arithmetic coding, the loss of precision of the source data increases linearly with the network size.

6. Discussion

We have presented a theoretical framework for Real and Complex Nodes, using an analogy with algebraic network coding. Our examples state clear advantages of using real
network codes (easy extension to complex field) to achieve higher throughput. Most important of these advantages is that real network codes achieve mincut of the subgraph for each individual source-sink pair, where as algebraic coding achieves mincut of the entire graph. We fail to develop an equivalent random coding scheme for ANCs and show that loss in quantization becomes very high as the size of network increases. The quantization loss is characterized by the condition number of the overall transform which can not be contained through random choice of matrices. In a later work, we have given the upper and lower bounds to the rate achieved in terms of the condition numbers.

Future work involves the development of deterministic algorithms to minimize the quantization loss as well as analysis of complexity and throughput for generalized networks. Simulations and implementation in the above mentioned example is also required.

References