基于编码分布式快速哈达玛变换的多元 LDPC 码译码算法研究

刘锐，黎勇
（重庆大学计算机学院，重庆 400044）

摘 要：尽管多元 LDPC 码纠错性能优异且能抗突发错误，但高译码复杂度仍制约了其实际应用。在其经典的 FHT-QSPA 译码中，快速哈达玛变换（FHT）及其逆变换（IFHT）是校验节点更新的主要瓶颈。基于此，提出了基于系统型 MDS 码的编码分布式 FHT 方案。该方案在主节点上将信道概率建模为矩阵并对其进行切分，再编码生成冗余子矩阵；其后，将所有子矩阵卸载到从节点并行执行 FHT 和 IFHT，然后将计算结果传回主节点并完成最终译码。编码冗余的嵌入克服了节点掉队问题，稳定地提升了变换效率，从而加速了整个译码过程。与先前的编码矩阵乘法方案相比，所提方案编码复杂度更低、译码恢复数值精度更高，并保持了高效蝶形运算结构，降低了从节点计算复杂度。耗时对比和译码性能分析表明，所提方案相比传统单节点 FHT 方案快了约 3.8 倍，大幅提升了 FHT-QSPA 译码效率，且没有译码性能损失。

关键词：多元 LDPC 码译码；编码分布式计算；快速哈达玛变换
中图分类号：TN911
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2023199

Study on coded distributed fast Hadamard transform based non-binary LDPC code decoding algorithm

LIU Rui, LI Yong
School of Computer Science, Chongqing University, Chongqing 400044, China

Abstract: Despite the excellent error correction performance and burst error resistance capability of non-binary LDPC codes, the complexity of the decoding algorithms hinders their broader application. In the classic FHT-QSPA decoding, the fast Hadamard transform (FHT) and its inverse transform (IFHT) have become the main bottleneck for updating the check nodes. Therefore, a coded distributed FHT scheme based on systematic MDS codes was proposed. In the scheme, the channel probability was modeled by the master node as a matrix and it was segmented, and those sub-matrices were encoded into redundant ones. Then, all the sub-matrices were offloaded to worker nodes to perform parallel FHT and IFHT, and the results were sent back to the master node for the final decoding. By embedding redundant information, the proposed scheme resolved the straggling problem, improving the efficiency, and accelerating the entire decoding process. Comparisons with other coded matrix multiplication schemes, the proposed scheme provides lower encoding complexity, higher numerical accuracy in decoding recovery, and maintains an efficient butterfly operation, effectively reducing the computational complexity of worker nodes. The time comparison and decoding performance analysis reveal that the proposed scheme achieves up to approximately 3.8 times acceleration compared to the traditional single-node FHT scheme, significantly enhances the FHT-QSPA decoding efficiency without affecting the decoding performance.

Keywords: non-binary LDPC code decoding, coded distributed computing, fast Hadamard transform

收稿日期：2023−07−26；**修回日期：**2023−09−27
通信作者：黎勇，yongli@cqu.edu.cn
基金项目：国家自然科学基金资助项目（No.61771081）
Foundation Item: The National Natural Science Foundation of China (No.61771081)

0 引言

非二元/多元 LDPC（NB-LDPC, non-binary LDPC）码是由二元 LDPC 码在有限域（GF, galois field）上扩展而来的[1]。相比二元 LDPC 码，多元 LDPC 码更好地消除了 Tanner 图中的短环，提高了纠错性能[2]。多元 LDPC 码用多个比特表示一个多元符号，获得了较二元 LDPC 码更好的抗突发错误能力[3]。另外，多元符号与高阶调制更匹配，提升了频谱利用率[4]。

尽管多元 LDPC 码有多重优势，但其更高的译码复杂度极大地阻碍了多元 LDPC 码的实际应用。多元 LDPC 码的译码算法以多元和积算法（QSPA, q-ary sum-product algorithm）为代表[1]，后续针对 QSPA 计算复杂度问题，研究者提出了 FHT-QSPA[5]、FFT-QSPA[3]等改进算法。此外，Min-Max[6]、Min-Sum[7]、EMS[8]等译码算法简化了 QSPA 中的迭代步骤，如替换和积操作、降低信息传递数量等，以不同程度的纠错性能损失换取了译码复杂度的降低。现有多元 LDPC 码的硬件加速多基于简化译码算法展开，并在 FPGA 和 GPU 等专用芯片的支持下实现更高的译码吞吐量[9]。

因此，若要在单点计算资源有限且难以搭载昂贵专用芯片的设备上应用低误码率、高复杂度的多元 LDPC 码 QSPA 译码，则可借鉴分布式计算思路，利用多个节点的算力，共同完成计算密集型的译码任务，从而在纠错译码性能和算法复杂度两方面做出更好的折中。

但是，多节点分布式系统中某些节点可能出现通信繁忙、资源抢占、节点失效等情况，从而影响整体任务。这些节点被称为掉队节点。为了克服节点掉队的问题，传统的副本策略使用多个节点负责相同的任务，只要任意节点完成当前任务即可继续后续计算。然而，副本策略中成倍增加的资源消耗使其在资源利用率方面表现欠佳。

近年来，编码分布式计算（CDC, coded distributed computing）（简称编码计算）借鉴编码理论，巧妙地在计算任务中嵌入了冗余信息，克服了节点掉队问题、保护了数据隐私、优化了通信负载，成为分布式计算领域的研究热点[10-13]。以主从架构的编码计算方案为例，主节点能够通过部分从节点的计算结果直接恢复出掉队节点的计算结果，完成整体计算，克服掉队节点的拖累[10]。

在诸多编码计算研究中，大规模矩阵乘法（矩阵−矩阵/矩阵−向量乘法）作为机器学习算法的核心模块受到了较多关注。多项式码编码计算方案在矩阵乘法中嵌入了多项式结构[11]，使主节点可以通过部分从节点结果左乘编码系数矩阵的逆矩阵，求出掉队节点的计算结果。然而多项式码的编码矩阵作为一个范德蒙矩阵，其条件数随节点数量增加呈指数级增加，极大地影响了编码矩阵求逆，导致该方案在实数矩阵的译码恢复上数值稳定性很差。针对此问题，文献[12]通过切比雪夫多项式进一步改进了多项式嵌入结构，大幅提升了译码恢复过程的数值精度，使实数域上的编码分布式矩阵乘法成为可能。此外，文献[13]设计了一套基于系统型 MDS（max distance separable）码的矩阵−矩阵乘法编码计算方案，其编码系数矩阵为随机系数矩阵，不具备范德蒙矩阵结构，在保证可译码性的同时，不会产生译码恢复过程的数值稳定性问题。但是，该方案仅进行了理论推导，未进行实验验证。

除了关注矩阵乘法，编码计算还研究了如何利用掉队节点的部分计算结果[14]计算稀疏性保持[15]等问题。此外，编码分布式计算也嵌入了边缘计算架构[16]、多无人机集群[17]等应用场景，为多种单点资源有限的分布式集群性能优化提供了新方案。

本文受系统型 MDS 码编码计算方案[13]启发，针对多元 LDPC 码的 FHT-QSPA 译码，设计了 FHT 的编码计算加速方案，提升了多元 LDPC 码 FHT-QSPA 译码的效率。具体来看，本文的贡献有以下三点。

1) 提出了基于系统型 MDS 码的编码分布式 FHT 方案。该方案在主节点上将信道概率建模为矩阵并对其进行切分，再编码生成冗余子矩阵；其后，将所有子矩阵卸载到从节点并行执行 FHT 和 IFHT，然后将计算结果传回主节点并完成最终译码。

2) 在子矩阵编码中，通过随机系数编码生成了冗余子矩阵，嵌入了冗余信息，克服了节点掉队问题；同时，本文证明了随机系数编码矩阵能够以概率 1 的可能完成译码恢复，且恢复误差很小。此外，本文方案还保持了 FHT 的高效蝶形运算结构。

3) 多个不同参数多元 LDPC 码的耗时对比和译码性能分析表明，编码分布式 FHT 方案相比传统单节点 FHT 方案最高加速了约 3.8 倍，显著提升了 FHT-QSPA 译码效率，且没有译码性能损失。

1　相关工作

定义在有限域 GF(q) 上的多元 LDPC 码可称为 q-元 LDPC 码，其中 $q=p^r$，p 为素数，且 $r>1$。令 α 表示有限域 GF(q) 的本原元，则有限域 GF(q) 中的所有元素均可用本原元的幂次来表示，即 $\alpha^i, i\in\{0,1,\cdots,q-2,-\infty\}$，其中 $\alpha^{-\infty}\triangleq 0$。当 $p=2$ 时，GF(2^r) 表示二元域的扩展域，其中每一个元素都可以唯一地映射为一个长为 r 的二元序列。因此，GF(2^r) 上的多元 LDPC 码的码字符号可以由 r 个二进制比特来表示。与二元 LDPC 码不同，多元 LDPC 码的校验矩阵中的非零元不是 1，而是有限域 GF(q) 中的 $q-1$ 个非零元 $\alpha^i, i\in\{0,1,\cdots,q-2\}$，即一个 (n,k) q-元 LDPC 码 $\mathcal{C}$ 可由有限域 GF(q) 上的校验矩阵 $\boldsymbol{H}_{m\times n}=[h_{i,j}]$ 的零空间来定义，其中，$h_{i,j}$ 是有限域 GF(q) 中的元素。合法码字 $\boldsymbol{c}$ 和校验矩阵 $\boldsymbol{H}$ 的乘积为 $\boldsymbol{0}$，即 $\boldsymbol{H}^{\mathrm{T}}\boldsymbol{c}=\boldsymbol{0}$。

多元和积算法（QSPA）（又称 BP 算法）是多元 LDPC 码的经典译码算法。在 QSPA 译码中，一次迭代译码可大致分为校验节点更新，变量节点更新和尝试译码三步[9]。针对 QSPA 译码校验节点更新复杂度高的问题，文献[13]提出了等效变换，改进了每个有限域元素相应概率的计算方式，通过快速哈达玛变换代替了校验节点更新中的乘积操作，加速了译码算法。令 q_{mn}^a 表示变量节点 n 的值为有限域元素 a 时校验节点 m 被满足的概率；r_{mn}^a 表示变量节点 n 的值固定为 a 时校验节点 m 被满足的概率。集合 $\mathcal{N}(m)$ 表示与校验节点 m 相连的变量节点的集合。FHT-QSPA 译码的校验节点更新具体步骤如下。

1) 等效变换

$$q'^a_{mn}=q'^{a\div h_{mn}}_{mn} \tag{1}$$

其中，$\div$ 操作表示有限域上的乘法逆操作。

2) 快速哈达玛变换

$$Q'^a_{mn}=\mathrm{FHT}\left[q'^0_{mn},\cdots,q'^{q-1}_{mn}\right] \tag{2}$$

3) 逆快速哈达玛变换

$$r'^a_{mn}=\mathrm{IFHT}\left\{\left(\prod\nolimits_{j\in\mathcal{N}(m)\backslash n}Q'^0_{mj}\right),\cdots,\left(\prod\nolimits_{j\in\mathcal{N}(m)\backslash n}Q'^{q-1}_{mj}\right)\right\} \tag{3}$$

其中，$\backslash$ 表示排除元素操作，$\prod$ 表示连乘操作。

4) 逆等效变换

$$r^a_{mn}=r'^{(a\otimes h_{mn})}_{mn} \tag{4}$$

其中，$\otimes$ 表示有限域上的乘法操作。

式(1)～式(4)表示 FHT-QSPA 校验节点更新过程，需要根据校验矩阵中的非零元素对信道概率向量进行两组变换。若将校验矩阵所有非零元素对应的各符号信道概率表示为一个矩阵 $\boldsymbol{B}\in\mathbb{R}^{2^r\times N}$，则 FHT-QSPA 在校验节点更新过程的各计算步骤复杂度如表 1 所示。复杂度最高的是 FHT 及 IFHT，为 $\mathcal{O}(r\times 2^r\times N)$，其中，$r$ 表示扩展域阶数，2^r 表示有限域元素数量，N 表示校验矩阵中非零元素数量。

表 1　校验节点更新过程的各计算步骤复杂度

公式	GF(⊗)	GF(÷)	$\mathcal{R}$(×)	$\mathcal{R}$(+)
式(1)	—	$\mathcal{O}(2^r\times N)$	—	—
式(2)	—	—	—	$\mathcal{O}(r\times 2^r\times N)$
式(3)	—	—	$\mathcal{O}(2^r\times N)$	$\mathcal{O}(r\times 2^r\times N)$
式(4)	$\mathcal{O}(2^r\times N)$	—	—	—

在上述四步变换中，主要包括有限域上乘法 GF(⊗) 和有限域上乘法逆运算 GF(÷)，以及实数域上的乘法 $\mathcal{R}$(×) 和加法 $\mathcal{R}$(+)。式(1)和式(4)对应的是等效变换和逆等效变换，这两步在有限域上的乘法和乘法逆运算可以通过 LUT（lookup table）来加速。而式(2)和式(3)所代表的 FHT 和 IFHT 是对浮点数类型的信道概率值进行更新。当码字有限域增大或码长增加时，FHT 和 IFHT 的计算复杂度也随之快速增大。

表 2 统计了定义在不同有限域以及不同码长的多元 LDPC 码在单次校验节点更新中 FHT 和 IFHT 两步耗时之和占单次校验节点更新总耗时的比例。由表 2 可知，不同码长的 FHT 耗时和 IFHT 耗时之和超过了单次校验节点更新耗时的 82%；而随着码字有限域增大，其耗时占比超过 90%，成为校验节点更新的主要性能瓶颈。若通过分布式并行加速 FHT 和 IFHT，则可加速多元 LDPC 码校验节点更新，从而加速整体译码过程。

表 2 FHT 和 IFHT 耗时占比

码长	GF(16)	GF(32)	GF(64)	GF(128)	GF(256)
200	82.91%	88.46%	91.89%	93.25%	93.91%
400	83.25%	88.59%	91.77%	93.48%	93.90%
800	83.99%	88.59%	91.79%	93.49%	93.90%
1 600	83.17%	88.43%	91.78%	93.19%	93.62%
2 000	83.98%	88.37%	91.66%	93.27%	93.92%

2 系统型 MDS 码编码分布式 FHT 方案

正如前文分析，当校验矩阵中非零元素增加时，式(2)的规模快速增大。例如，当有限域阶数为64，校验矩阵中非零元素数量为 600 时，需要变换的 $\boldsymbol{q}_{mn}$ 矩阵大小为 64×600。此时，单节点执行 FHT 需要进行 $6\times64\times600=230\,400$ 次加减法操作。此时，若使用分布式计算，则可以将 $\boldsymbol{q}_{mn}$ 矩阵划分为多个子矩阵，并将其卸载到从节点中并行执行，提高 FHT 效率。

然而，在分布式计算的实际场景中，各计算节点受计算优先级、网络通畅程度等因素的影响，可能呈现出不同的计算速度。这样的节点掉队现象使分布式并行的加速效果受到部分掉队节点的拖累，降低整体加速效果。因此，针对 FHT 设计一套抗节点掉队的分布式并行加速算法成为加速多元 LDPC 码译码的关键所在。

2.1 总体流程

设有一个 $n+1$ 个节点组成的分布式计算系统，其中，主节点 W_0 负责 FHT-QSPA 的总体执行、子矩阵编码和掉队结果译码恢复。首先，主节点按列划分概率矩阵 $[\boldsymbol{q}_{mn}]_{2^r\times N}$ 得到子矩阵序列 $\boldsymbol{B}=\{\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_k\}$。其后，主节点对子矩阵序列进行随机线性组合，得到编码子矩阵 $\widetilde{\boldsymbol{B}}_{k+1},\widetilde{\boldsymbol{B}}_{k+2},\cdots,\widetilde{\boldsymbol{B}}_n$。之后，主节点将计算复杂度最高的 FHT 和 IFHT 卸载到从节点 $W_i,i\in\{1,2,\cdots,n\}$ 中并行执行。其中，k 个从节点收到的是未编码子矩阵，而另外 $n-k$ 个从节点收到的是编码后的子矩阵。从节点 W_i 分别对子矩阵进行 FHT，并将计算结果返回给主节点。主节点只需收到最快完成的 k 个子矩阵变换结果便能恢复其他掉队子矩阵变换结果，从而克服了掉队节点对总体计算速度的拖累。方案整体流程如算法 1 所示。

算法 1 基于系统 MDS 码的编码分布式 FHT 方案

输入 信道概率矩阵 $[\boldsymbol{q}_{mn}]_{2^r\times N}$，从节点数量 n，恢复阈值 k

输出 $[\boldsymbol{Q}_{mn}]_{2^r\times N}$

if 节点索引=从节点 #从节点计算任务

$\boldsymbol{B}_i\leftarrow W_0$ #接收子矩阵

$\boldsymbol{C}_i=\mathrm{FHT}(\boldsymbol{B}_i)$ #执行 FHT

$\boldsymbol{C}_i\rightarrow W_0$ # 发送变换结果

else #主节点计算任务

$\boldsymbol{B}=[\boldsymbol{q}_{mn}]$ #赋初值

$\boldsymbol{B}=\{\boldsymbol{B}_1\ \ \boldsymbol{B}_2\cdots\ \ \boldsymbol{B}_k\}$ #划分子矩阵

$\mathbb{B}=\boldsymbol{B}\boldsymbol{G}_{\boldsymbol{B}}^{\mathrm{T}}=\{\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_k,\widetilde{\boldsymbol{B}}_{k+1},\widetilde{\boldsymbol{B}}_{k+2},\cdots,\widetilde{\boldsymbol{B}}_n\}$ # 编码子矩阵

end if

for i=1:n do #分发子矩阵

$W_i\leftarrow\boldsymbol{B}_i,\boldsymbol{B}_i\in\mathbb{B}$

end for

for WaitAny(k) do #译码恢复

$$\mathrm{FHT}(\boldsymbol{B}_s)=\mathrm{Decoding}\{\mathrm{FHT}(\boldsymbol{B}_u),\mathrm{FHT}(\boldsymbol{B}_c)\},\quad s\in I_u\setminus S_u,u\in S_u,c\in S_c$$

end for

$$[\boldsymbol{Q}_{mn}]=[\mathrm{FHT}(\boldsymbol{B}_u),\mathrm{FHT}(\boldsymbol{B}_c),\mathrm{FHT}(\boldsymbol{B}_s)],\quad s\in I_u\setminus S_u,u\in S_u,c\in S_c$$ #输出

图 1 展示了由 5 个从节点组成的编码分布式 FHT 方案，其中前 3 个节点计算未编码子矩阵，后 2 个节点计算编码子矩阵。主节点将 FHT 卸载到各个从节点上，主节点上仅需要进行编码和译码操作。多个从节点收到了尺寸更小的子矩阵，并行执行 FHT，实现分布式并行加速。图 1 中×所示节点为掉队节点，而主节点可基于其他节点结果恢复掉队节点的计算结果。

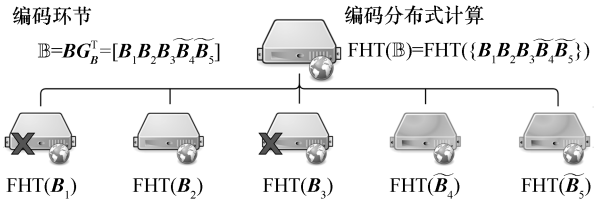

图 1 编码分布式 FHT 方案架构

下面将围绕主节点编码环节、可译码性证明、译码环节和从节点 FHT 计算来展开分析。

2.2 编码环节

首先，本节详细描述了系统型 MDS 码编码分布式 FHT 方案的子矩阵编码思路。主节点按列切分概率矩阵 $[\boldsymbol{q}_{mn}]_{2^r\times N}$ 得到子矩阵序列 $\boldsymbol{B}=\{\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_k\}$。

其后，主节点对子矩阵序列进行随机线性组合，得到编码子矩阵 $\widetilde{\boldsymbol{B}}_{k+1},\widetilde{\boldsymbol{B}}_{k+2},\cdots,\widetilde{\boldsymbol{B}}_n$。设序列 $\mathbb{B}$ 表示所有子矩阵组成的序列，即 $\{\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_k,\widetilde{\boldsymbol{B}}_{k+1},\widetilde{\boldsymbol{B}}_{k+2},\cdots,\widetilde{\boldsymbol{B}}_n\}$。则生成矩阵 $\boldsymbol{G}_{\boldsymbol{B}}$ 定义为

$$\boldsymbol{G}_{\boldsymbol{B}}=\left[\begin{array}{c}\boldsymbol{I}_{k\times k}\\ \hline \boldsymbol{P}_{\boldsymbol{B}}\end{array}\right]=\left[\begin{array}{c}\boldsymbol{I}_{k\times k}\\ \hline \begin{array}{ccc} b_{k+1,1} & \cdots & b_{k+1,k}\\ \vdots & \ddots & \vdots\\ b_{n,1} & \cdots & b_{n,k}\end{array}\end{array}\right] \tag{5}$$

此时，其上半部分 $\boldsymbol{I}$ 为单位矩阵，而下半部分为标准高斯分布 $\mathcal{N}(0,1)$ 上采样的随机数子矩阵 $\boldsymbol{P}_{\boldsymbol{B}}$。因此，通过生成矩阵和未编码子矩阵序列的乘法，即可求出分配给所有从节点的子矩阵序列 $\mathbb{B}$

$$\begin{aligned}\mathbb{B}&=[\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_k]\boldsymbol{G}_{\boldsymbol{B}}^{\mathrm{T}}=\\ &[\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_k,\widetilde{\boldsymbol{B}}_{k+1},\widetilde{\boldsymbol{B}}_{k+2},\cdots,\widetilde{\boldsymbol{B}}_n]\end{aligned} \tag{6}$$

$\boldsymbol{P}_{\boldsymbol{B}}$ 中的每行确定了一组随机系数，对未编码子矩阵进行线性组合，得到对应的编码子矩阵。即 $\widetilde{\boldsymbol{B}}_i=b_{i,1}\boldsymbol{B}_1+\cdots+b_{i,k}\boldsymbol{B}_k,i\in\{k+1,k+2,\cdots,n\}$。

此外，特别值得指出的是，在式(6)中，子矩阵 $\boldsymbol{B}_i$ 被视为标量，生成矩阵 $\boldsymbol{G}_{\boldsymbol{B}}$ 中对应的单位矩阵 $\boldsymbol{I}$ 维度也设定为矩阵划分后的子矩阵数量，即 $k\times k$。然而，实际情况中，每个子矩阵 $\boldsymbol{B}_i$ 的维度是 $2^r\times\dfrac{N}{k}$。因此，编码环节仅需要针对未编码子矩阵进行随机系数标量和子矩阵的乘法，之后再进行线性组合，即可求出对应的编码子矩阵。

2.3　可译码性证明

本节分析系统型 MDS 码编码分布式 FHT 方案的可译码性。通过对其编码矩阵行列式的分析，证明其子方阵行列式以概率 1 的可能是一个非零多项式，从而证明该方案能够以概率 1 的可能完成译码。

引理 1[18]　若矩阵 $\boldsymbol{G}$ 的每一个子方阵都是非奇异矩阵时，该矩阵可以成为一个系统型 MDS 码的编码矩阵。

引理 2　若子矩阵 $\boldsymbol{P}_{\boldsymbol{B}}$ 中的元素都是在标准高斯分布中 $\mathcal{N}(0,1)$ 独立同分布地采样，子矩阵 $\boldsymbol{P}_{\boldsymbol{B}}$ 的每一个子方阵都以概率 1 的可能是非奇异矩阵。

证明　首先通过数学归纳法证明任意 $r\times r$ 阶子矩阵 $(r\leqslant(n-k))$ 的行列式都是非零多项式。当 $r=1$ 时，假设显然成立。接着，假设每一个 $(r-1)\times(r-1)$ 阶子矩阵的行列式都是非零多项式。此时，任意一个 $r\times r$ 阶子矩阵可写为

$$\boldsymbol{S}=\begin{bmatrix} b_{i_1,j_1} & b_{i_1,j_2} & \cdots & b_{i_1,j_r}\\ b_{i_2,j_1} & b_{i_2,j_2} & \cdots & b_{i_2,j_r}\\ \vdots & \vdots & \ddots & \vdots\\ b_{i_r,j_1} & b_{i_r,j_2} & \cdots & b_{i_r,j_r}\end{bmatrix} \tag{7}$$

此矩阵的行列式可以写为

$$\det(\boldsymbol{S})=b_{i_1,j_1}\boldsymbol{D}_1+b_{i_1,j_2}\boldsymbol{D}_2+\cdots+b_{i_1,j_r}\boldsymbol{D}_r \tag{8}$$

其中，$\boldsymbol{D}_i$ 为去掉矩阵 $\boldsymbol{S}$ 的第一行和第 i 列形成的 $(r-1)\times(r-1)$ 阶子矩阵。由于归纳假设任意 $(r-1)\times(r-1)$ 阶子矩阵的行列式都是非零多项式。此外，假设矩阵 $\boldsymbol{S}$ 中的系数集合 $\{b_{i,j}\}$ 使 $\det(\boldsymbol{S})=0$，本文称这个系数集合为整个空间的 0-测度子集。对于给定子矩阵阶数 r，则有 $\binom{k}{r}\binom{n-k}{r}$ 个不同的子矩阵。称使 $\boldsymbol{P}_{\boldsymbol{B}}$ 的任意子方阵行列式为 0 的系数集合 $\{b_{i,j}\}$ 为一个 0-测度系数集合，则该系数集合是由 $\sum_{r=1}^{n-k}\binom{k}{r}\binom{n-k}{r}$ 个不同阶数子矩阵的 0-测度系数子集的并集。当 $\{b_{i,j}\}$ 为高斯分布中采样得到的随机值时，概率 P(系数集合为“0−测度系数集合”)=0。故任意 $r\times r$ 阶子矩阵 $(r\leqslant(n-k))$ 的行列式都是非零多项式，即任意 $(r\leqslant(n-k))$ 阶子矩阵都是非奇异矩阵。证毕。

在引理 1 和引理 2 的基础上，本文给出下列定理。

定理 1　若编码计算方案由生成矩阵 $\boldsymbol{G}$ 确定，$\boldsymbol{P}_{\boldsymbol{B}}$ 为矩阵 $\boldsymbol{G}$ 的子矩阵，其大小为 $(n-k)\times k$，且 $\boldsymbol{P}_{\boldsymbol{B}}$ 中的元素均为标准高斯分布 $\mathcal{N}(0,1)$ 中的独立同分布采样得到。此时，该方案给定了一个系统型的 MDS 码编码计算方案，当主节点收到 n 个子矩阵变换结果中的任意 k 个时，即可以概率 1 的可能译码恢复掉队节点对应的子矩阵变换结果。

在定理 1 所确定的系统型 MDS 码编码计算方案中，从节点数量为 n 个，其中有 k 个节点执行未编码子任务，有 $n-k$ 个节点执行编码子任务。此时，该方案的恢复阈值为 $n-k$，即该方案最多可以容忍 $n-k$ 个节点掉队或失效。

2.4　译码环节

在 n 个从节点和一个主节点组成的分布式计算系统中，简便起见，可令前 k 个节点为未编码子任

务节点，后 $n-k$ 个节点为编码子任务节点。设从节点索引集合为 $i\in\{1,2,\cdots,k,k+1,\cdots,n\}$，则 2 种节点对应索引集合可规定为 $I_u=\{i_1,i_2,\cdots,i_k\}$ 和 $I_c=\{i_{k+1},i_{k+2},\cdots,i_n\}$。

由定理 1 可知，当主节点收到最快完成的 k 个子任务结果时即可完成计算。此时，主节点根据接收到的未编码子任务计算结果数量 n_u 来判断是否需要进行译码。最好情况下，前 k 个未编码子任务都很快完成了计算任务，并将结果返回给主节点，即 $n_u=k$。此时，由于主节点收到了所有想要的未编码子任务计算结果，并不需要进行译码恢复，可直接进行后续计算任务。

而当 $n_u<k$ 时，即未编码节点出现了掉队情况，则主节点需要收到至少 $k-n_u$ 个编码子任务计算结果来恢复未编码子任务的计算结果。设主节点收到 n_c 个编码子任务节点结算结果，2 种节点对应的索引集合分为 $S_u=\{u_1,u_2,\cdots,u_{n_u}\},S_u\subset I_u$；$S_c=\{c_1,c_2,\cdots,c_{n_c}\},S_c\subset I_c$，且 $n_c+n_u=k$。此时，主节点已经收到了足够数量的子任务结果，来恢复掉队节点子任务结果 $S_s=I_u\setminus S_u$，完成译码恢复。

于是，可以得到下列由已知的编码子任务计算结果、未编码子任务结果、掉队未编码子任务结果确定的方程组

$$\overbrace{\mathrm{FHT}(\widetilde{\boldsymbol{B}}_{c_i})}^{\text{已知编码子任务}}=\overbrace{\sum_{u\in S_u}b_{c_i,u}\mathrm{FHT}(\boldsymbol{B}_u)}^{\text{已知未编码子任务}}+\overbrace{\sum_{s\in S_s}b_{c_i,s}\mathrm{FHT}(\boldsymbol{B}_s)}^{\text{掉队未编码子任务}},\forall c_i\in S_c \tag{9}$$

因此，掉队的未编码子任务结果可以通过求解上述方程组来获得。此时方程组(9)中的未知数为 $\mathrm{FHT}(\boldsymbol{B}_s),s\in S_s$，未知数的数量 $n_s\leqslant n_c$，即掉队的未编码子任务的数量小于或等于主节点收到的编码子任务数量。此外，由定理 1 可知，若编码矩阵中的系数为高斯分布中独立同分布采样得到，则满足任意 r 阶子矩阵均是非奇异矩阵，即任意 r 阶子矩阵均为可逆矩阵。根据上述两点，对方程组(9)进行求解即可得到掉队的未编码子任务结果 $\mathrm{FHT}(\boldsymbol{B}_s),s\in S_s$。

例如图 1 所示的编码分布式 FHT 方案，由 5 个从节点组成。其中未编码子任务节点为 $I_u=\{1,2,3\},I_c=\{4,5\}$。若假设主节点收到最快完成计算任务的从节点集合为 $S_u=\{2\},S_c=\{4,5\}$，则 1 号节点和 3 号节点所负责的未编码子任务结果出现了掉队情况，需要通过收到的编码子任务来求解，即求解下列齐次线性方程组，便可求出对应掉队任务结果。

$$\begin{aligned}&\overbrace{\mathrm{FHT}(\widetilde{\boldsymbol{B}_4})}^{\text{已知}}-\overbrace{b_{4,2}\mathrm{FHT}(\boldsymbol{B}_2)}^{\text{已知}}=\overbrace{b_{4,1}\mathrm{FHT}(\boldsymbol{B}_1)}^{\text{未知}}+\overbrace{b_{4,3}\mathrm{FHT}(\boldsymbol{B}_3)}^{\text{未知}}\\&\overbrace{\mathrm{FHT}(\widetilde{\boldsymbol{B}_5})}^{\text{已知}}-\overbrace{b_{5,2}\mathrm{FHT}(\boldsymbol{B}_2)}^{\text{已知}}=\overbrace{b_{5,1}\mathrm{FHT}(\boldsymbol{B}_1)}^{\text{未知}}+\overbrace{b_{5,3}\mathrm{FHT}(\boldsymbol{B}_3)}^{\text{未知}}\end{aligned} \tag{10}$$

2.5　从节点 FHT 计算

前文主要阐述了 FHT 的编码分布式并行加速方案，本节围绕 FHT 的计算方式，阐述了编码分布式 FHT 方案在从节点上的计算任务。

FHT 本身可以通过蝶形运算的形式来表示，图 2 展示了一个 4 维向量通过蝶形运算完成 FHT 的过程。总体来看，FHT 的复杂度可以表示为 $\mathcal{O}(r\times 2^r\times N)$，其中 $2^r\times N$ 表示了待变换矩阵的规模。而经过前文所述的编码环节，每个子矩阵的大小都变为了 $2^r\times\frac{N}{k}$。因此，从节点上的计算复杂度降为 $\mathcal{O}\left(r\times 2^r\times\frac{N}{k}\right)$。

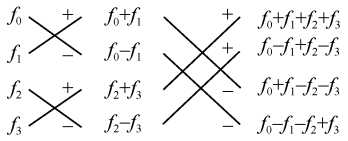

图 2　FHT 蝶形运算计算顺序

3　实验验证

本节针对前述系统型 MDS 码编码分布式 FHT 方案进行了实验验证，分析了编码环节、译码环节以及从节点 FHT 的复杂度，并在 AWS EC2 计算集群上验证了所提方案的有效性。首先，简要介绍了实验环境设置和实验参数设置；其次，针对系统型 MDS 码编码分布式 FHT 方案的编码环节、译码环节以及从节点 FHT 进行了复杂度分析；之后，以目前北斗导航系统中采用的 64 元 200 码长的多元 LDPC 码为基础[19-20]，分析了类似参数的其他多元 LDPC 码字的译码加速效果。此外，还与 FHT-QSPA 进行了译码性能对比，证明了本文方案可以在保证相同的译码精度的情况下，通过分布式并行计算，显著加速 FHT-QSPA 译码。

3.1　实验环境设置

为验证本文所提编码分布式 FHT 方案的译码加速性能，本节基于 AWS EC2 环境搭建了分布式计算集群，并基于 MPI 消息传递库开发了编码分布式 FHT 方案。测试码字上以北斗系统中使用的 64 元(200, 100)LDPC 码为基础，并拓展了不同的有限域阶数和不同的码长。有限域阶数包括 GF(16)、GF(32)、GF(64)、GF(128)和 GF(256)。码长则包括 200、400、800、1 600 和 2 000。在 EC2 实例的选择上，本文选择了 t2.micro 类型的实例，其具体配置为 1 vCPU 和 1 GB 内存。在此基础上，测试了上述码字的 FHT 环节和 IFHT 环节经系统型 MDS 码编码分布式 FHT 方案后取得的加速效果。所有的耗时数据均为 100 个码字完成译码或达到最大迭代次数后单次校验节点更新中各个计算步骤耗时数据的均值。

3.2　复杂度分析

本文所提编码分布式 FHT 方案的主要计算步骤包括编码环节和译码环节，以及从节点 FHT。编码分布式 FHT 方案计算复杂度如表 3 所示。

表 3　编码分布式 FHT 方案计算复杂度

计算环节	$\mathcal{R}(\times)$	$\mathcal{R}(+)$
单节点 FHT	—	$\mathcal{O}(r\times 2^r\times N)$
编码环节	$\mathcal{O}((n-k)\times 2^r\times N)$	$\mathcal{O}\left((n-k)\times(k-1)\times 2^r\times\frac{N}{k}\right)$
译码环节	$\mathcal{O}\left(n_s\times(k-n_u)\times 2^r\times\frac{N}{k}\right)$	$\mathcal{O}\left(n_s\times(k-1)\times 2^r\times\frac{N}{k}\right)$
从节点 FHT	—	$\mathcal{O}\left(r\times 2^r\times\frac{N}{k}\right)$

3.2.1　编码环节复杂度对比

在编码分布式 FHT 方案的编码环节中，主节点需要对划分好的子矩阵进行随机线性组合，即每个子矩阵先乘以一个随机系数再进行相加。因此，编码环节的复杂度由冗余节点数量和子矩阵大小构成。k 表示从节点中执行未编码子任务的节点数量，反映了加速倍数。k 越大，则加速倍数越大，对应 $n-k$ 越小，掉队节点抑制能力也越小，此时编码复杂度越低。因此，本文方案的编码环节复杂度和掉队节点抑制能力存在一个折中。

在相同的 n 个从节点组成的分布式计算环境中，文献[11]所提多项式码编码矩阵乘法方案需要为每一个从节点单独编码生成子矩阵，除该方法中的幂次计算环节外，还需进行 $\mathcal{O}(n\times 2^r\times N)$ 次乘法和 $\mathcal{O}\left(n\times(k-1)\times 2^r\times\frac{N}{k}\right)$ 次加法，2 种运算的复杂度都高于本文方案。

正交多项式码编码矩阵乘法方案[12]的编码复杂度为 $\mathcal{O}(2^r\times N\times k+2^{2r}\times k)$，也高于本文方案编码环节复杂度。

3.2.2　译码环节复杂度对比

此外，若未编码从节点不掉队，本文方案不需要执行译码；若未编码从节点出现掉队，则译码环节需要求解由 n_s 个方程组成的齐次线性方程组。相应的复杂度主要由未编码子任务中的掉队节点数量 n_s 和子矩阵大小构成。

多项式码编码矩阵乘法方案[11]由于其特殊的幂次多项式编码嵌入结构，使多项式码方案在浮点型数据的译码恢复精度很差。而在 FHT 中，多项式码方案随着节点数量增加已不能正确译码。文献[12]和后文恢复误差分析也说明了这一点。但文献[12]所提的正交多项式码编码矩阵乘法的译码复杂度为 $\mathcal{O}(N\times 2^r\times k^2)$，仍然高于本文方案的 $\mathcal{O}\left(n_s\times(k-n_u)\times 2^r\times\frac{N}{k}\right)$。

3.2.3　从节点计算环节复杂度对比

另外，从节点上执行的是规模更小的 FHT，其复杂度是单节点执行的 $\frac{1}{k}$，并且多个从节点并行执行可以加速 FHT。然而，与文献[11]和文献[12]所提出的编码矩阵乘法方案相比，2 种方案将待变换矩阵和 FHT 系数矩阵均进行了编码，破坏了 FHT 系数矩阵的蝶形运算结构，使从节点上 FHT 由蝶形运算退化为矩阵乘法，其计算复杂度则由 $\mathcal{O}\left(r\times 2^r\times\frac{N}{k}\right)$ 大幅增加为 $\mathcal{O}\left(2^{3r}\times\frac{N}{k}\right)$。而本文方案优化了编码冗余嵌入方式，在从节点上维持了 FHT 的高效蝶形运算结构，降低了从节点上的计算复杂度。

3.3　数值稳定性

文献[11]中的多项式码编码矩阵乘法方案，由于其编码生成矩阵内生的范德蒙矩阵结构，其矩阵条件数随着从节点数量增加而快速增加，使编码系数矩阵求逆的数值稳定性很差，极大地影响了浮点型数据的译码恢复，文献[12]的实验同样反映了这一点。

针对 FHT-QSPA 的 FHT 和 IFHT，其计算的是多元符号所对应的各个有限域元素的概率值。而这意味着 FHT-QSPA 对编码分布式算法造成的译码恢复误差更加敏感，一旦译码恢复数值精度较差，必然会影响后续的译码性能。所以，本文采取的恢复误差计算式为

$$E([Q_{mn}],[Q'_{mn}]_{\text{coded}}) = \max([Q_{mn}]-[Q'_{mn}]_{\text{coded}}) \quad (11)$$

其中，$[Q_{mn}]$ 表示单节点自行完成 FHT 的结果，$[Q'_{mn}]_{\text{coded}}$ 表示采用编码分布式计算得到的 FHT 结果。本文对比了所提系统 MDS 码的编码分布式 FHT 方案和文献[11]中提出的多项式码编码矩阵乘法方案以及文献[12]中提出的正交多项式编码矩阵乘法方案的最大恢复误差，统计结果如表 4 所示。

表 4　　3 种方案最大恢复误差

方案	n=5	n=10	n=20
多项式码方案	4.77×10^{-2}	6.06	1.52×10^{5}
正交多项式码方案	1.99×10^{-15}	5.44×10^{-15}	6.42×10^{-13}
本文方案	3.93×10^{-16}	2.84×10^{-16}	7.95×10^{-16}

从表 4 可以看出，3 种编码分布式方案的数值精度随着从节点数量 n 的增加都呈现增加的趋势。然而正如前文分析，多项式码方案的数值稳定性随着节点数量增加而急剧恶化，当从节点超过 5 个时，其最大恢复误差已经大于 0.01，而这样的误差无疑会对 FHT-QSPA 的概率更新产生极大影响；而当节点数继续增大时，多项式码方案已不能正确完成译码恢复。对比来看，正交多项式码方案着重改进了多项式码方案的数值稳定性，在 20 个从节点的分布式设置中仍旧保持了 1×10^{-13} 级别的恢复误差。而本文所提的基于系统型 MDS 码的编码分布式 FHT 方案的最大恢复误差是三者中最小的。

3.4 加速效果

在本文方案中，主要的设计参数有从节点的总数 n、未编码从节点的数量 k、节点的掉队程度 λ 和掉队节点数量 s。其中，从节点总数 n 比较容易理解，n 越大，分布式并行加速效果越好。而参数 k 则表示主节点在 n 个从节点中选择前 k 个节点作为未编码节点，分配未编码子矩阵；剩下的 $n-k$ 个节点则作为冗余节点，分配编码子矩阵。因此本文所提编码分布式 FHT 方案所能容忍的掉队节点数量等于 $n-k$，而对 FHT 和 IFHT 的理论加速倍数为 k。换言之，本文使用冗余节点换取了分布式方案中对掉队节点的容忍。

此时，节点的掉队程度或计算速度可以由主节点进行监控，当节点计算速度正常或掉队程度较低时，k 可以设置得较大，换取较好的加速效果；而当节点繁忙程度增加，节点计算速度下降时，k 则可设得较小，使 $n-k$ 较大，换取较好的掉队抑制效果。

3.4.1 从节点数 n 对加速效果的影响

首先，分析系统型 MDS 码编码分布式并行加速的 FHT 和 IFHT 在执行耗时上的效果。在码字选择上，统计了码长为 200 的 64 元 LDPC 码在校验节点更新中 FHT 和 IFHT 的耗时，并在分布式环境下进行了对应的耗时分析，具体结果如图 3 所示。

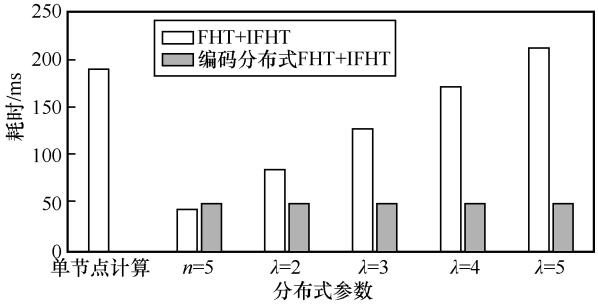

图 3　200 码长的 64 元 LDPC 码编码分布式 FHT 和 IFHT 耗时

图 3 中共有 6 组实验结果。第一组数据表示由单节点自行计算 FHT 和 IFHT 的总耗时，200 码长的 64 元 LDPC 码在单次校验节点更新过程中，FHT 和 IFHT 总耗时约为 191 ms。第二组数据 ($n=5$) 中的白色部分表示使用 5 个从节点对 FHT 和 IFHT 进行分布式计算，其耗时总计约 42 ms，加速倍数约为 4.5 倍；而灰色柱则表示使用编码分布式 FHT 方案对 FHT 和 IFHT 进行加速，其耗时约为 49 ms，加速倍数约为 3.9 倍。因此，编码分布式 FHT 方案通过设置冗余节点，虽造成分布式并行加速倍数稍有降低，但克服了掉队节点的拖累，使分布式 FHT 获得了稳定的加速。

3.4.2 掉队程度 λ 对加速效果的影响

图 3 的第二组数据的白色部分表示分布式并行实验，即 5 个从节点中未设置掉队节点，此时分布式加速倍数接近于理想加速倍数 5 倍。但是当节点出现不同程度的掉队时，其加速性能无疑会劣化。后 4 组数据中的白色部分反映了当 5 个节点中有 1 个节点出现掉队情况时，分布式执行的 FHT 和 IFHT 总耗时。其中，λ 表示节点的掉队程度，即掉队节点的计算速度退化为未掉队节点的 $\frac{1}{\lambda}$。4 组数据中的白色部分反映了当节点掉队程度加深，分布式计算

的实际加速效果不断退化，甚至超过未采用分布式计算的耗时 ($\lambda = 5$)。

加入系统型 MDS 编码分布式计算后的实验数据绘制如图 3 后 4 组数据中的灰色部分所示。在编码分布式 FHT 方案中冗余节点数量设置为 1，因此该方案可以对抗 5 个节点中任意 1 个节点的掉队，而该方案的理论加速性能为 4 倍。与 5 个节点纯分布式计算相比，1 个冗余节点的编码分布式 FHT 方案相比纯分布式方案理论加速性能略弱，总耗时约为 50 ms。但是随着节点掉队程度的加深，编码分布式 FHT 方案的加速性能不受掉队节点的影响，稳定在 50 ms 左右。因此，编码分布式 FHT 方案能够通过冗余节点设计抵抗掉队节点的影响，且节点掉队程度加深对整体加速性能没有劣化。

3.4.3　掉队节点数量 s 对加速效果的影响

由于编码分布式 FHT 方案中通过设置冗余节点来克服节点掉队，而图 3 反映了节点掉队程度的加深对并行算法的加速效果有很大的影响。因此，图 4 对掉队节点数量进行了实验，其中从节点数量为 $n = 10$，冗余节点数量为 $n - k = 5$，节点掉队程度为 $\lambda = 2$，码字参数与图 3 实验一致。编码分布式 FHT 方案可以在冗余节点数量范围内容忍不同数量的掉队节点，而不增加 FHT 和 IFHT 耗时，即基于系统型 (n,k) MDS 码的编码分布式 FHT 方案能够抵抗任意小于或等于 $n-k$ 个节点掉队。当节点掉队数量超过了冗余节点数量时，此时整个分布式系统中的掉队情况更加严重。例如，当 10 个从节点中有 6 个节点掉队时，主节点在收到 4 个从节点结果的基础上至多还需要等待任意 1 个从节点的计算结果便可完成译码恢复。因此，当掉队节点数量超过冗余节点数量时，主节点可以在下一次迭代更新中进一步增大冗余节点数量，以实现更好的抗节点掉队效果；反之，当节点掉队数量小于冗余节点数量时，主节点也可在下一次迭代更新中降低冗余节点数量，达到更好的并行加速效果。

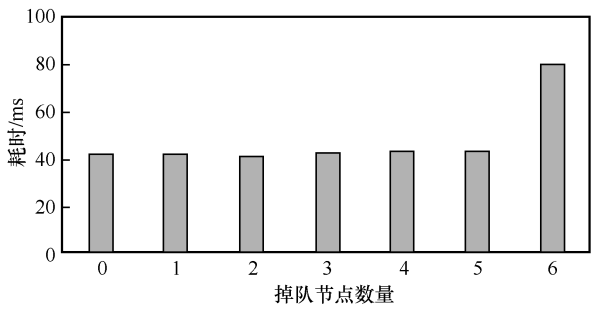

图 4　不同掉队节点数量 s 对编码 FHT 和 IFHT 的耗时影响

3.5　不同参数 NB-LDPC 码加速效果

根据前文分析，编码分布式 FHT 加速方案不受节点掉队程度影响，实现了稳定加速。因此，本文继续使用 5 个从节点，其中有 1 个冗余节点的编码分布式参数设定，分析不同码长和不同有限域阶数上的加速效果。选择掉队节点数量为 1 个，节点掉队程度 λ 为 2。在不同码长的 64 元 LDPC 码上，编码分布式方案对 FHT 和 IFHT 加速效果如图 5 所示。当码长从 200 增长到 2 000，FHT+IFHT 的总耗时从 191 ms 快速增加到了 2 105 ms。而编码分布式 FHT+IFHT 方案能够稳定地并行加速 FHT 和 IFHT，提升两者计算效率。

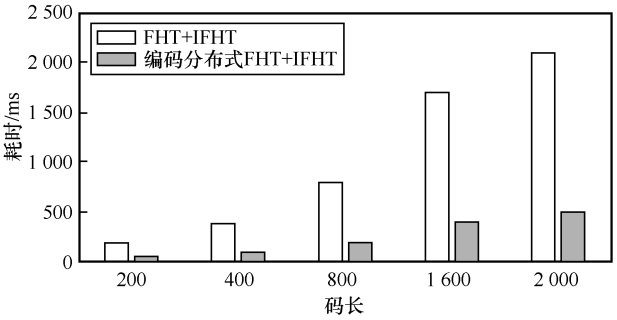

图 5　不同码长的 64 元 LDPC 码 FHT 和 IFHT 耗时

接下来，固定码长为 200，将有限域阶数从 16 增加到 256 时，对应的编码分布式 FHT 加速方案效果如图 6 所示。随着有限域阶数的增加，FHT+IFHT 的总耗时从 49 ms 增长到 866 ms。因此，如图 5 和图 6 中灰色数据所示，特别是针对码长变长和有限域变大的情况，编码分布式 FHT 方案大幅节约了 FHT 和 IFHT 计算时间，加速了校验节点更新，从而加速了整体 FHT-QSPA 译码过程。

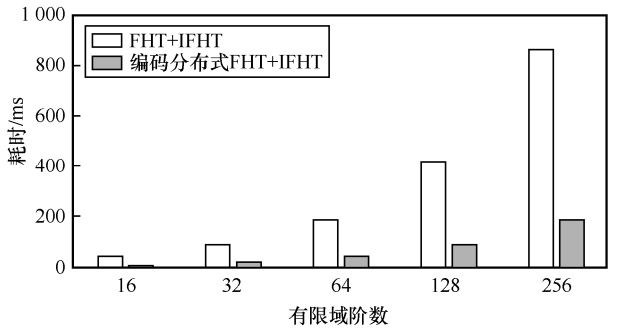

图 6　200 码长的不同有限域阶数 LDPC 码 FHT 和 IFHT 耗时

3.6　译码性能对比

本节选用了码长为 800 的 16 元 LDPC 码作为测试码字。首先通过随机方法得到了码字生成矩

阵，之后对随机生成的信息进行编码，并进行 BPSK 调制，其后经过 AWGN 信道加噪，最后分别使用 FHT-QSPA 和基于编码分布式方案的 FHT-QSPA 进行译码，分析其译码性能。图 7 给出了 2 种算法的译码性能曲线。从图 7 可以看出，基于编码分布式 FHT 方案的译码算法的译码性能相较于原始算法没有损失。

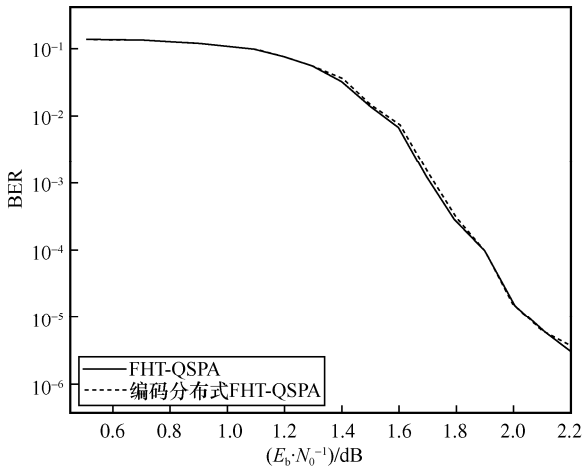

图 7 FHT-QSPA 和编码 FHT-QSPA 的译码性能曲线

除此之外，本文还进一步分析了 FHT-QSPA 和编码分布式 FHT-QSPA 在单次校验节点迭代中 FHT 和 IFHT 的耗时情况，如表 5 所示。其中编码分布式 FHT 采用了 5 个从节点，其中有 1 个冗余节点的分布式设置，而掉队节点数量设置为 1 个。从表 5 可以看出，通过编码分布式的改造，使 FHT 和 IFHT 这 2 个耗时最长环节得到了稳定加速。

表 5 2 种译码算法单次迭代耗时

环节	FHT-QSPA/ms	编码分布式 FHT-QSPA/ms
FHT	97.1	26.0
IFHT	97.5	26.1

4 结束语

本文提出了一种基于系统型 MDS 码的编码分布式 FHT 方案。该方案在主节点上将信道概率建模为矩阵并对其进行切分，再编码生成冗余子矩阵；其后，主节点将所有子矩阵卸载到从节点并行执行 FHT 和 IFHT，然后从节点将计算结果传回主节点完成最终译码。通过设置不同的冗余节点数量，该方案可以在加速性能和抗节点掉队性能之间取得折中。此外，本文证明了所提系统型 MDS 码编码计算方案能够以概率 1 的可能完成掉队任务结果的译码恢复，而译码过程仅需对齐次线性方程组进行求解。

总体来看，本文方案的编译码环节的复杂度均较低，这使编码分布式 FHT 加速方案针对 FHT 和 IFHT 均获得了接近于 k 倍的加速。将本文方案的译码恢复误差与其他编码矩阵乘法方案进行对比，验证了本文方案优秀的数值稳定性。而不同有限域和码长的多元 LDPC 码译码耗时分析，确认了本文方案的加速效果。译码性能曲线对比表明了本文方案可以在不损失译码性能的情况下，稳定地加速译码过程。

另外，本文方案还适于其他变换，例如 FFT 等。通过对待变换矩阵进行编码分布式加速，可以同时实现较高的抗节点掉队性能和保持高效变换计算结构。后续本文将继续研究该方案在其他变换上的应用，拓展系统型 MDS 码编码分布式加速方案的应用场景，提高计算效率。

参考文献：

[1] DAVEY M C, MACKAY D. Low-density parity check codes over GF(q)[C]//Proceedings of IEEE Communications Letters. Piscataway: IEEE Press, 2002: 165-167.

[2] PENG R H, CHEN R R. Design of nonbinary quasi-cyclic LDPC cycle codes[C]//Proceedings of 2007 IEEE Information Theory Workshop. Piscataway: IEEE Press, 2007: 13-18.

[3] SONG H X, CRUZ J R. Reduced-complexity decoding of q-ary LDPC codes for magnetic recording[J]. IEEE Transactions on Magnetics, 2003, 39(2): 1081-1087.

[4] RONG B, JIANG T, LI X M, et al. Combine LDPC codes over GF(q) with q-ary modulations for bandwidth efficient transmission[J]. IEEE Transactions on Broadcasting, 2008, 54(1): 78-84.

[5] LI Y, WANG L, CHEN J B. The design and simulation of q-ary LDPC codes based on the PEG algorithm[C]//Proceedings of 14th IST Mobile and Wireless Communications Summit. Piscataway: IEEE Press, 2005: 1-5.

[6] SAVIN V. Min-Max decoding for non binary LDPC codes[C]//Proceedings of 2008 IEEE International Symposium on Information Theory. Piscataway: IEEE Press, 2008: 960-964.

[7] WANG C L, CHEN X H, LI Z W, et al. A simplified min-sum decoding algorithm for non-binary LDPC codes[J]. IEEE Transactions on Communications, 2013, 61(1): 24-32.

[8] DECLERCQ D, FOSSORIER M. Decoding algorithms for nonbinary LDPC codes over GF(q)[J]. IEEE Transactions on Communications, 2007, 55(4): 633-643.

[9] FERRAZ O, SUBRAMANIYAN S, CHINTHALA R, et al. A survey on high-throughput non-binary LDPC decoders: ASIC, FPGA, and GPU architectures[J]. IEEE Communications Surveys & Tutorials, 2021, 24(1): 524-556.

[10] LEE K, LAM M, PEDARSANI R, et al. Speeding up distributed machine learning using codes[J]. IEEE Transactions on Information Theory, 2018, 64(3): 1514-1529.

[11] YU Q, MADDAH-ALI M, AVESTIMEHR S. Polynomial codes: an optimal design for high-dimensional coded matrix multiplication[C]// Proceedings of the 31st International Conference on Neural Information Processing Systems. New York: ACM Press, 2017: 4406-4416.

[12] FAHIM M, CADAMBE V R. Numerically stable polynomially coded computing[J]. IEEE Transactions on Information Theory, 2021, 67(5): 2758-2785.

[13] JEONG H, YANG Y Q, GROVER P. Systematic matrix multiplication codes[C]//Proceedings of 2019 IEEE International Symposium on Information Theory (ISIT). Piscataway: IEEE Press, 2019: 1-5.

[14] DAS A B, RAMAMOORTHY A. Coded sparse matrix computation schemes that leverage partial stragglers[J]. IEEE Transactions on Information Theory, 2022, 68(6): 4156-4181.

[15] WANG S, LIU J, SHROFF N. Coded sparse matrix multiplication[J]. arXiv Preprint, arXiv: 1802.03430, 2018.

[16] PRAKASH S, DHAKAL S, AKDENIZ M R, et al. Coded computing for low-latency federated learning over wireless edge networks[J]. IEEE Journal on Selected Areas in Communications, 2021, 39(1): 233-250.

[17] ZHOU B N, XIE J F, WANG B Q. Dynamic coded distributed convolution for UAV-based networked airborne computing[C]// Proceedings of 2022 International Conference on Unmanned Aircraft Systems (ICUAS). Piscataway: IEEE Press, 2022: 955-961.

[18] MACWILLIAMS F J, SLOANE N J A. The theory of error-correcting codes[M]. North Holland: North Holland Publishing Co., 1977.

[19] 中国卫星导航办公室. 北斗系统空间信号接口控制文件 B1C（1.0 版）[R]. 2017.
China Satallite Navigation Office. BeiDou navigation satellite system signal in space interface control document B1C (version 1.0)[R]. 2017.

[20] 中国卫星导航办公室. 北斗系统空间信号接口控制文件 B2a（1.0 版）[R]. 2017.
China Satallite Navigation Office. BeiDou navigation satellite system signal in space interface control document B2a (version 1.0)[R]. 2017.

[作者简介]

刘锐（1995− ），男，重庆人，重庆大学博士生，主要研究方向为编码分布式计算、信道编码等。

黎勇（1982− ），男，重庆人，博士，重庆大学教授、博士生导师，主要研究方向为信息编码理论及应用、计算机视觉、医学图像处理等。