Successive Approximation for Coded Matrix Multiplication

Shahrzad Kiani and Stark C. Draper

shahrzad.kianidehkordi@mail.utoronto.ca , stark.draper@utoronto.ca

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

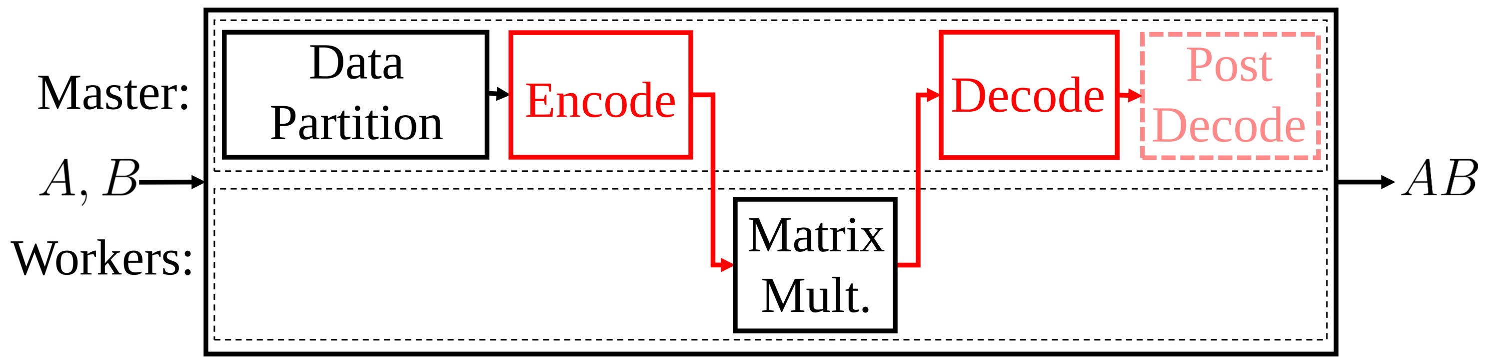
NSERC CRSNG   DIDI SCHOLARSHIP

Abstract

In this project, we combine ideas of approximate and coded computing to further accelerate computation. We develop two successive approximated coding (SAC) methods, each produces a sequence of increasingly accurate approximations of the desired computation as more and more workers report in. SAC guarantees exact recovery upon completion of a sufficient number of workers. We theoretically provide design guidelines for our SAC methods, and numerically show that SAC achieves a better accuracy-speed tradeoff in comparison with previous methods.

Coded Computing Using Polynomial Bases

Goal: Compute $C = AB$, $A \in \mathbb{R}^{(N_x \times N_z)}$, $B \in \mathbb{R}^{(N_z \times N_y)}$ across N workers.
Method: Distributed coded matrix multiplication to mitigate stragglers.



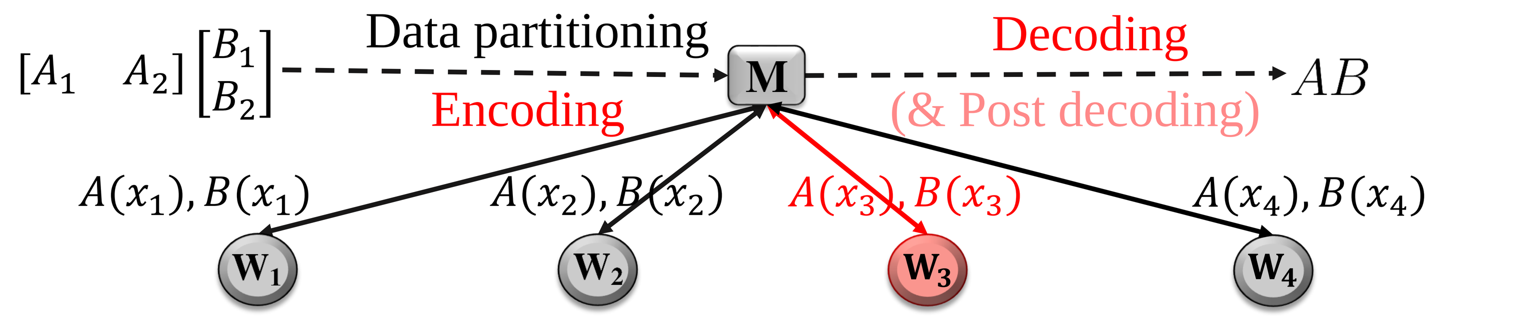
- **Encoding via polynomial basis :** $\{T_k(x)\}_{k=1:K}$, $K = 2$.

Exp.	MatDot (MD)	OrthoMD (OMD)	Lagrange (Lag)
Basis	Monomial	Orthonormal	Lagrange
$A(x)$	$A_1T_1(x) + A_2T_2(x)$		
$B(x)$	$B_1T_2(x) + B_2T_1(x)$	$B_1T_1(x) + B_2T_2(x)$	

- **Decoding via polynomial basis** $\{Q_r(x)\}_{r=1:R}$ to recover $\{C_r\}_{r=1:R}$, $R = 3$.

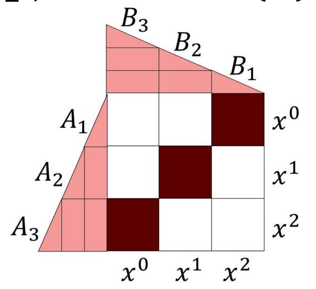
$$\begin{bmatrix} Q_1(x_{i_1}) & Q_2(x_{i_1}) & Q_3(x_{i_1}) \\ Q_1(x_{i_2}) & Q_2(x_{i_2}) & Q_3(x_{i_2}) \\ Q_1(x_{i_3}) & Q_2(x_{i_3}) & Q_3(x_{i_3}) \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} A(x_{i_1})B(x_{i_1}) \\ A(x_{i_2})B(x_{i_2}) \\ A(x_{i_3})B(x_{i_3}) \end{bmatrix}$$

- **Post decoding (not always needed):** $C = \sum_{k=1}^{K} \alpha_k A(y_k)B(y_k)$. $\mathcal{Y} = \{y_k\}_k$



Accuracy-Speed Tradeoff via Approximation

Benchmark: ε−approximate MatDot (εAMD) [Jeong et al].
- **Idea:** Evaluate $A(x)B(x)$ at sufficiently small points $x \in \{x_n\}_{n=1:N}$.
- **Exp)** $K = 3$, $A(x)B(x) = (A_1 + A_2x + A_3x^2)(B_1x^2 + B_2x + B_3)$



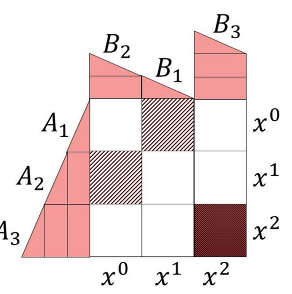
$$= A_1B_1 + (A_1B_2 + A_2B_3)x$$
$$+ (\mathbf{A_1B_1 + A_2B_2 + A_3B_3})x^2 \quad \} \ P(x) \cong A(x)B(x) \text{ Approx. Rec, } R_1 = 3$$
$$+ (A_2B_1 + A_3B_2)x^3 + A_3B_1x^4 \quad \text{Exact. Rec, } R = 5$$

Successive Approximated Coding (SAC)

Goal: Extend single-layer approximation of εAMD to SAC.

Proposed Method 1) Group-wise SAC (G-SAC)
- $\sim R$ approximation layers, D groups of Approx. layers.
- **Exp 1)** Two-group SAC, $D = 2, K_d \in \{2, 1\}$
- **Idea:** Carefully re-order the coefficients of the encoding polynomials.

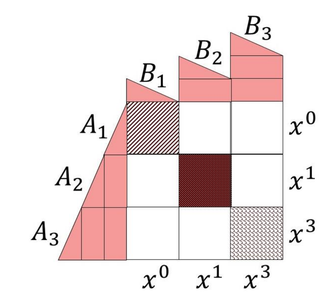
$$A(x)B(x) = (A_1 + A_2x + A_3x^2)(B_3x^2 + B_1x + B_2)$$



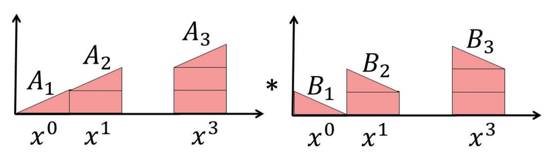
$$= A_1B_2 + (\mathbf{A_1B_1 + A_2B_2})x \quad \} \ P_1(x),\ R_1 = 2$$
$$+ (A_1B_3 + A_2B_1 + A_3B_2)x^2 \quad P_2(x),\ R_2 = 3$$
$$+ (A_2B_3 + A_3B_1)x^3 \quad P_3(x),\ R_3 = 4$$
$$+ (\mathbf{A_3B_3})x^4 \quad P_4(x),\ R = 5$$

- **Exp 2)** Multi-group SAC, $D = 3, K_d \in \{1,1,1\}$.
- **Idea:** In analogy with discrete Conv., inject delays amongst coefficients of encoding polynomials to avoid interference b/w them.

$$A(x)B(x) = (A_1 + A_2x + A_3x^3)(B_3x^3 + B_2x + B_1)$$



$$= \mathbf{A_1B_1} \quad \} \ P_1(x),\ R_1 = 1$$
$$+ (A_1B_2 + A_2B_1)x \quad P_2(x),\ R_2 = 2$$
$$+ \mathbf{A_2B_2}x^2 \quad \ddots$$
$$+ (A_3B_1 + A_1B_3)x^3$$
$$+ (A_2B_3 + A_3B_2)x^4$$
$$+ \mathbf{A_3B_3}x^6 \quad P_6(x),\ R = 7$$

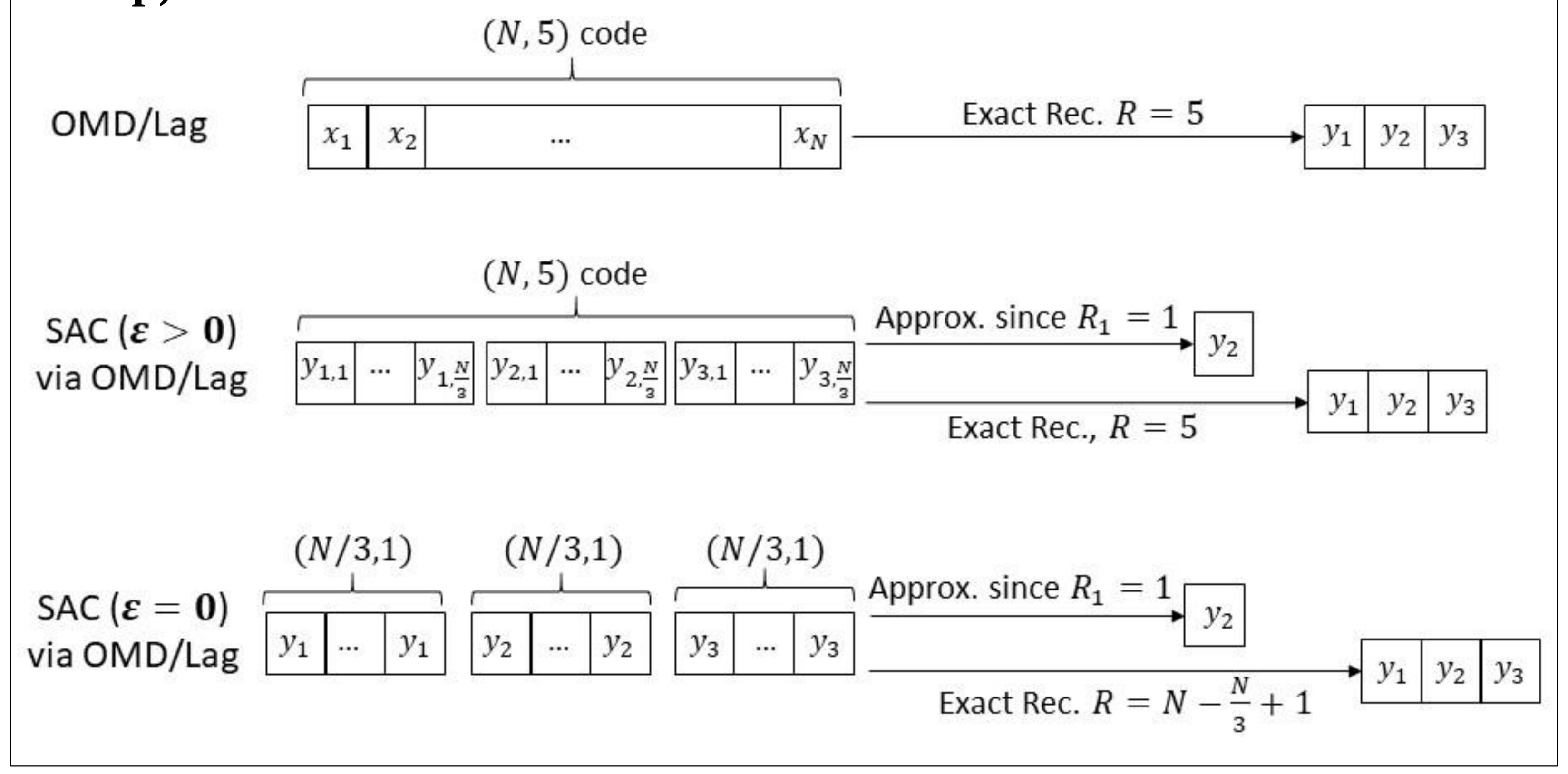


Proposed Method 2) Layer-wise SAC (L-SAC)
- Apply SAC to the codes that require post-decoding, (e.g., OMD, Lag).
- Evaluate $A(x)B(x)$ at points $x \in \mathcal{X}_{SAC}$, distinct from that of OMD and Lag.
- **Idea:** (1) Divide $\mathcal{X}_{SAC}$ to K disjoint splits, e.g., $\{y_{k,i}\}_{k\in[K], i\in[N/K]}$.

  (2) Set $y_{k,i}$ in split k to be "ε−close" to $y_k \in \mathcal{Y}$.

$\Rightarrow$ successively recover the terms that make up $C = \sum_{k=1}^{K} \alpha_k A(y_k)B(y_k)$.

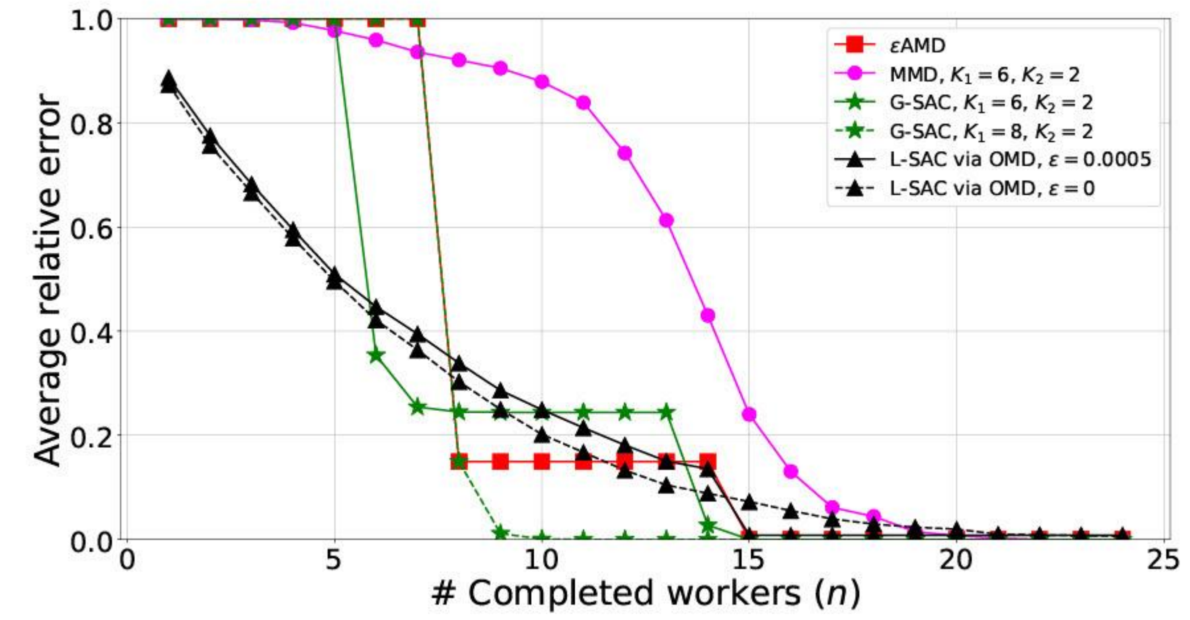
- **Exp)** $K = 3$



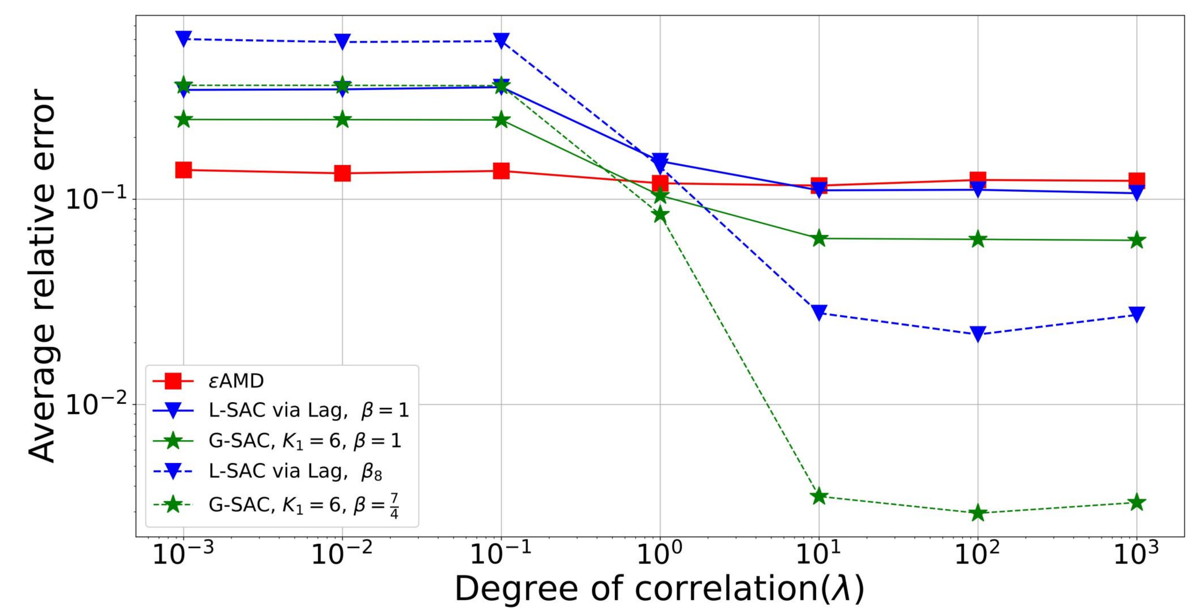
Simulation Results

Source of errors:
$$\underbrace{\sqrt{\|C - \tilde{C}_n\|_F^2}}_{\text{Error}} \le \underbrace{\sqrt{\|C - C_n\|_F^2}}_{\text{Approx. Err.}} + \underbrace{\sqrt{\|C_n - \tilde{C}_n\|_F^2}}_{\text{Computation Err.}}$$

Setting:
- $N = 40$, $K = 8$, $A_k \in \mathbb{R}^{100\times1000}$ and $B_k \in \mathbb{R}^{1000\times100} \sim \mathcal{N}(0,1)$.
- MMD: (30,6) MD code for $\sum_{k=1}^{6} A_kB_k$ and (10,2) MD code for $\sum_{k=7}^{8} A_kB_k$
- G-SAC: D=2, $K_d \in \{6,2\} + K_d \in \{8,0\}$, $\mathcal{X}_{complex} = \{0.15e^{\frac{i2\pi n}{N}}\}_{n=1}^{N}$.
- L-SAC via OMD: $y_{k,i}$ ε-close to $\mu_k^{(8),cheby}$. $\varepsilon \in \{5 \times 10^{-4}, 0\}$.



- Fig 2: L-SAC via Lag: $y_{k,i}$ ε-close to k. $\varepsilon = 3.33 \times 10^{-2}$.
- Fig 2: $A_k = \lambda A^{(0)} + A_k^{(1)}$ and $B_k = \lambda B^{(0)} + B_k^{(1)}$. $A^{(0)}, \ldots, B_k^{(1)} \sim \mathcal{N}(0,1)$.



Takeaways:
- G-SAC ($K_1 = 8$) similar to εAMD upto $n = 8$, improve in estimate as $n \uparrow$.
- G-SAC ($K_1 = 6$) earlier estimates since $n = 6$, better estimates $n \ge 14$.
- L-SAC via OMD continuous improvement since $n = 1$.
- When $\varepsilon = 0$, slightly better estimation, but wait longer for exact recovery.
- Fig 2: even for $n = 8$, G-SAC & L-SAC better estimates than εAMD if highly correlated (λ large) and parameters set optimally.

Relevant Publications

S. Kiani, and S. C. Draper, "Successive Approximation for Coded Matrix Multiplication", to be published at IEEE ISIT, June, 2022.

S. Kiani and S. C. Draper, "Successive Approximation for Coded Matrix Multiplication", submitted to IEEE JSAIT, (in review).