

# INTRODUCING THE PARTIAL STRASSEN ALGORITHM FOR ENTANGLEMENT RENORMALIZATION

*Aaron Dayton (Speaker) – Kiana Gallagher – Dr. Thomas E. Baker*  
*Department of Physics & Astronomy, University of Victoria, Victoria, Canada*



Chaires de recherche  
du Canada

Canada Research  
Chairs

QUANTUM BC  
research · training · innovation

Canada



**NSERC**  
**CRSNG**



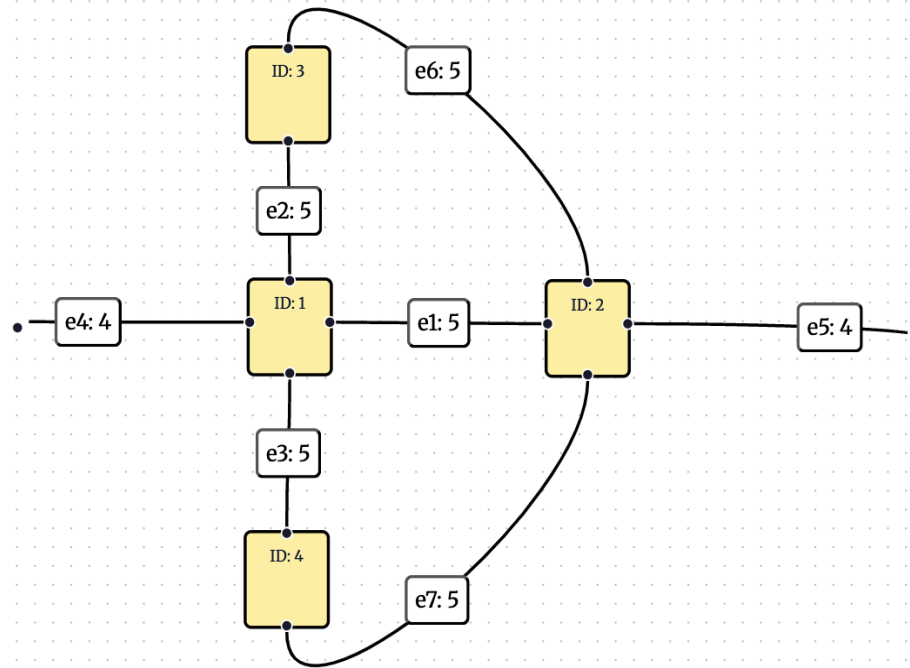
Digital Research  
Alliance of Canada

UVIC

# MOTIVATION

- We require efficient classical methods to compare against quantum computations
  - *Lattice density functional theory*
  - *Variational quantum eigensolvers*
  - *Entanglement renormalization*
- Tensor contractions reduce to matrix multiplications

URL: [dmrgenie.rs.uvic.ca](http://dmrgenie.rs.uvic.ca)  
GitHub: [bakerte/DMRGenie.jl](https://github.com/bakerte/DMRGenie.jl)



Drag and Drop

Tensor Rank:

Upload

Color gradient slider

Tensor Operations

Contract

Decompositions

SVD

QR

LQ

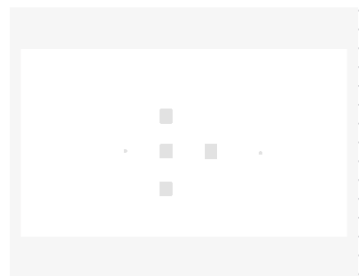
Eigenvalue

Edit Functions

Split Edge

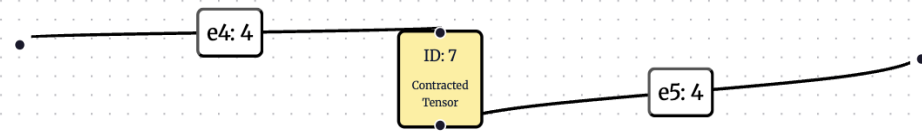
Connect Edge

Copy



Vertical toolbar with icons: +, -, a square, and a lock.

URL: [dmrgenie.rs.uvic.ca](http://dmrgenie.rs.uvic.ca)  
GitHub: [bakerte/DMRGenie.jl](https://github.com/bakerte/DMRGenie.jl)



Drag and Drop

Tensor Rank:

Upload



Tensor Operations

Contract

Tensor One ID:

Tensor Two ID:

Submit

Decompositions

SVD

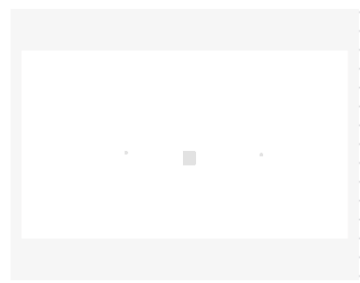
QR

LQ

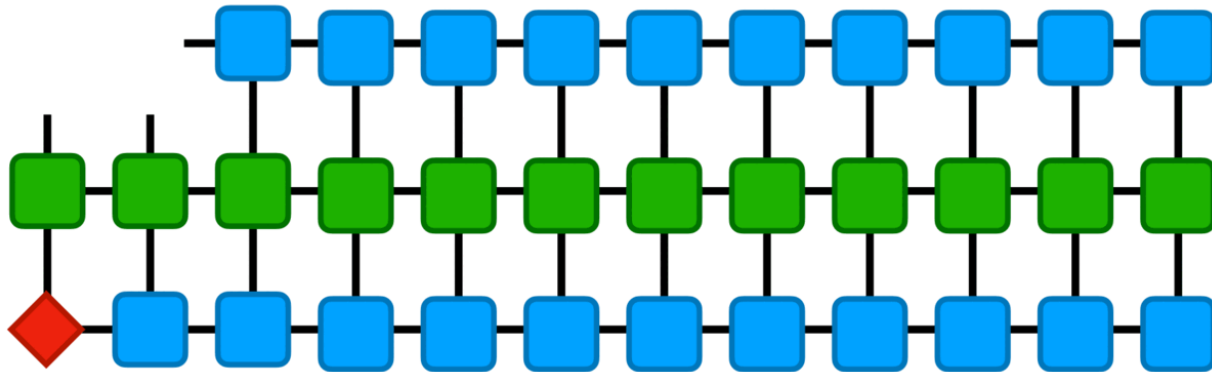
Eigenvalue

Edit Functions

Split Edge



URL: [dmrgenie.rs.uvic.ca](https://dmrgenie.rs.uvic.ca)  
GitHub: [bakerte/DMRGenie.jl](https://github.com/bakerte/DMRGenie.jl)



Model

Hamiltonian  ?

Compute Correlations  ?

Symmetry  ?

Number of Sites:

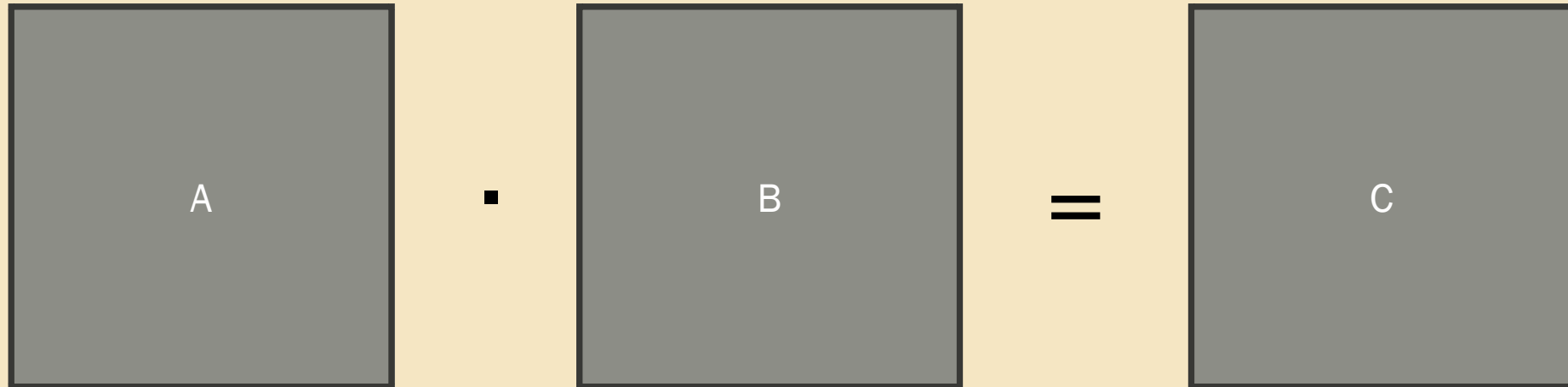
Run Algorithm



# THE STRASSEN ALGORITHM

*FOR MATRIX MULTIPLICATION*

# THE STRASSEN ALGORITHM



# THE STRASSEN ALGORITHM

The diagram illustrates the Strassen algorithm for matrix multiplication. It shows three 2x2 matrices arranged horizontally, separated by a multiplication dot and an equals sign. Each matrix is represented as a 2x2 grid of cells.

- The first matrix, labeled  $A$ , has cells containing  $A_{11}$ ,  $A_{12}$ ,  $A_{21}$ , and  $A_{22}$ .
- The second matrix, labeled  $B$ , has cells containing  $B_{11}$ ,  $B_{12}$ ,  $B_{21}$ , and  $B_{22}$ .
- The third matrix, labeled  $C$ , has cells containing  $C_{11}$ ,  $C_{12}$ ,  $C_{21}$ , and  $C_{22}$ .

The equation is represented as: 
$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

# THE STRASSEN ALGORITHM

## Traditional Multiplication

**8** Multiplications

**4** Additions

## Strassen Multiplication

**7** Multiplications

**18** Additions

# THE STRASSEN ALGORITHM

## Traditional Multiplication

**8** Multiplications

**4** Additions

## Strassen Multiplication

**7** Multiplications

**18** Additions

*(Single-level vs. Multi-level)*

# THE STRASSEN ALGORITHM

Traditional Multiplication

8 Multiplications

4 Additions

$$O(n^3)$$

Strassen Multiplication

7 Multiplications

18 Additions

*(Single-level vs. Multi-level)*

$$O(n^{\log_2 7}) = O(n^{2.81})$$

# THE STRASSEN ALGORITHM

*Traditional*

$$M_1 = A_{11}B_{11}$$

$$M_2 = A_{12}B_{21}$$

$$M_3 = A_{11}B_{12}$$

$$M_4 = A_{12}B_{22}$$

$$M_5 = A_{21}B_{11}$$

$$M_6 = A_{22}B_{21}$$

$$M_7 = A_{21}B_{12}$$

$$M_8 = A_{22}B_{22}$$

*Strassen*

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_2 = (A_{21} + A_{22})B_{11}$$

$$M_3 = A_{11}(B_{12} - B_{22})$$

$$M_4 = A_{22}(B_{21} - B_{11})$$

$$M_5 = (A_{11} + A_{12})B_{22}$$

$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$

$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

# THE STRASSEN ALGORITHM

*Traditional*

$$C_{11} = M_1 + M_2$$

$$C_{12} = M_3 + M_4$$

$$C_{21} = M_5 + M_6$$

$$C_{22} = M_7 + M_8$$

*Strassen*

$$C_{11} = M_1 + M_4 - M_5 + M_7$$

$$C_{12} = M_3 + M_5$$

$$C_{21} = M_2 + M_4$$

$$C_{11} = M_1 - M_2 + M_3 + M_6$$

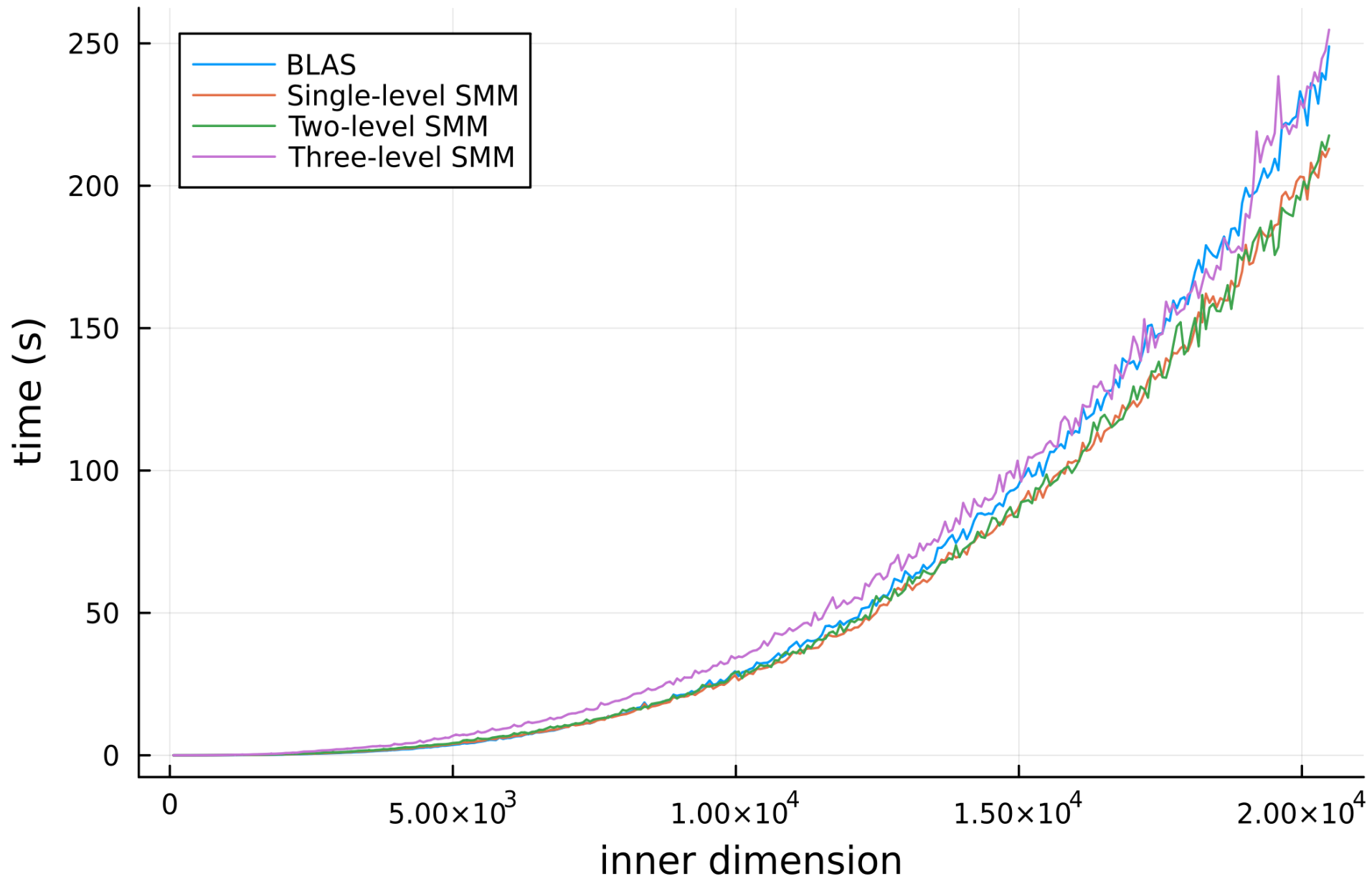
# OUR RESULTS

We implemented a general Strassen algorithm in **Julia**

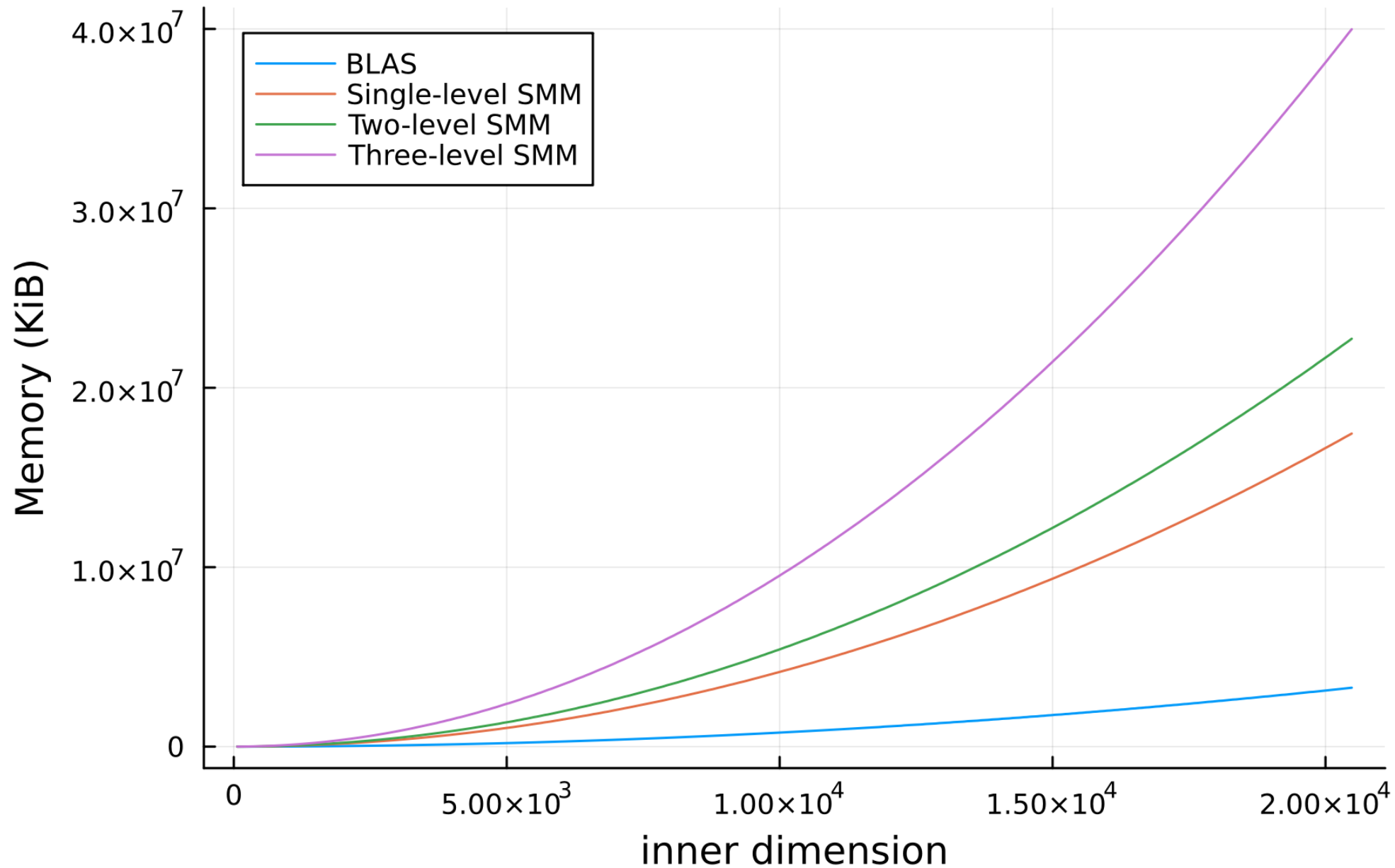


*[github.com/bakerte/StrassOPen.jl](https://github.com/bakerte/StrassOPen.jl)*

# Runtime comparison of BLAS and multi-level Strassen

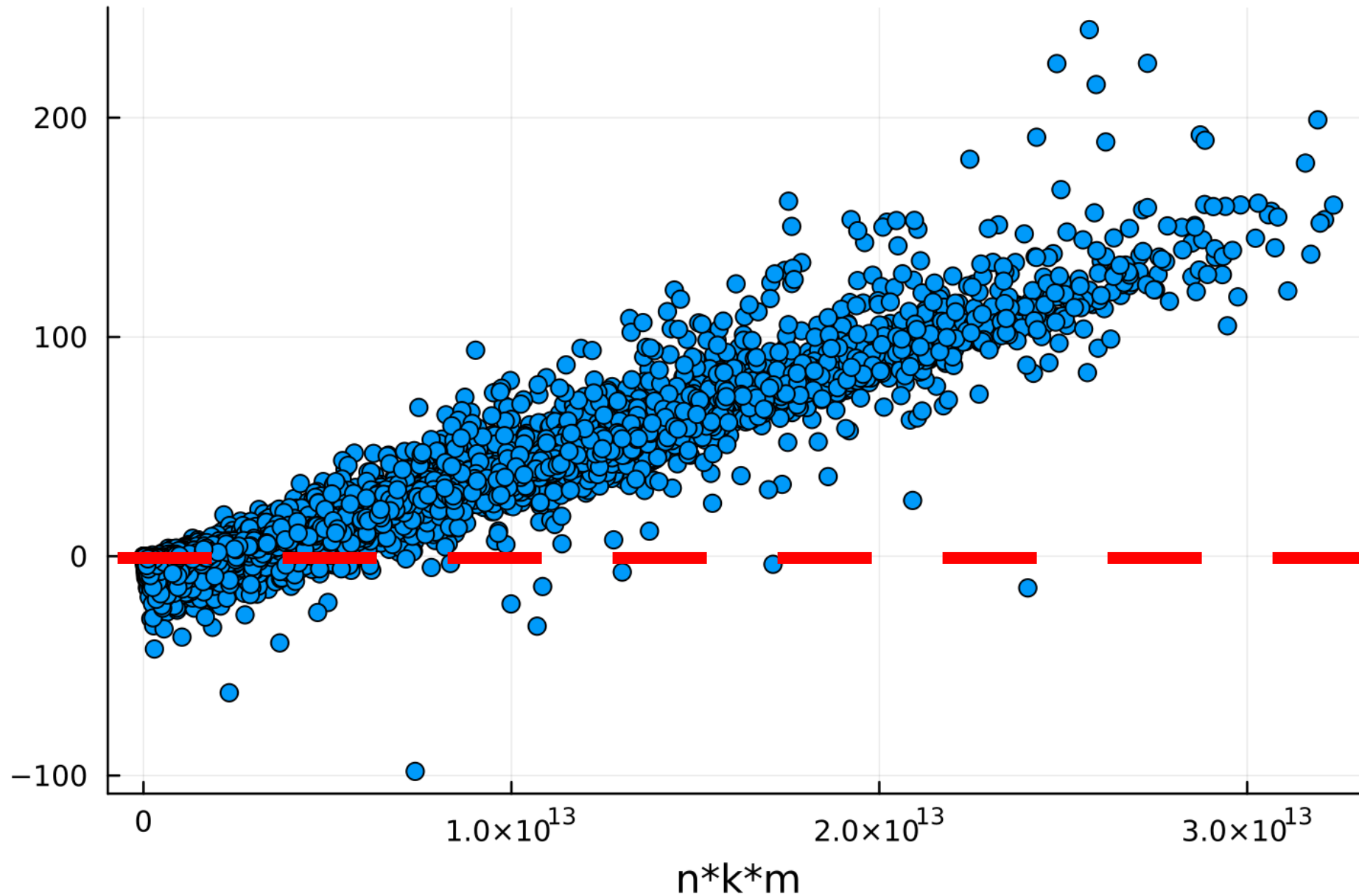


# Memory comparison of BLAS and multi-level Strassen



BLAS times minus two-level Strassen times (s)

Time difference versus  $n*k*m$



# INITIAL CONCLUSIONS

- The single and two-level Strassen algorithms are applicable
- There is a Goldilocks zone of applicability:
  1. Matrix size is large enough for speedup
  2. Matrix size is not too large so that there is sufficient memory (memory ceiling)



# THE PARTIAL-STRASSEN ALGORITHM

*A NEW VARIANT*

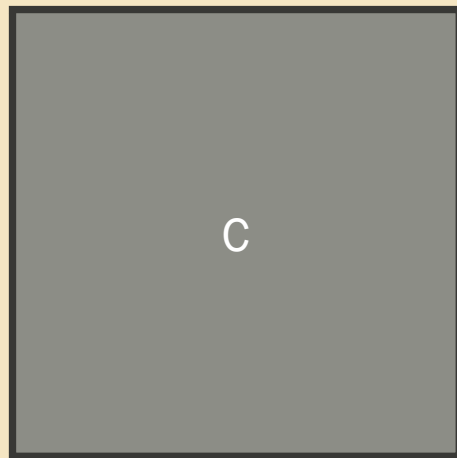
# THE PARTIAL-STRASSEN ALGORITHM

Modifications:

1. Store some  $M_i$  matrices in  $C$  and some in pre-allocated buffers
2. Subdivide only the  $M_i$  matrices stored in  $C$

# THE PARTIAL-STRASSEN ALGORITHM

Store some  $M_i$  matrices in C



$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_2 = (A_{21} + A_{22})B_{11}$$

$$M_3 = A_{11}(B_{12} - B_{22})$$

$$M_4 = A_{22}(B_{21} - B_{11})$$

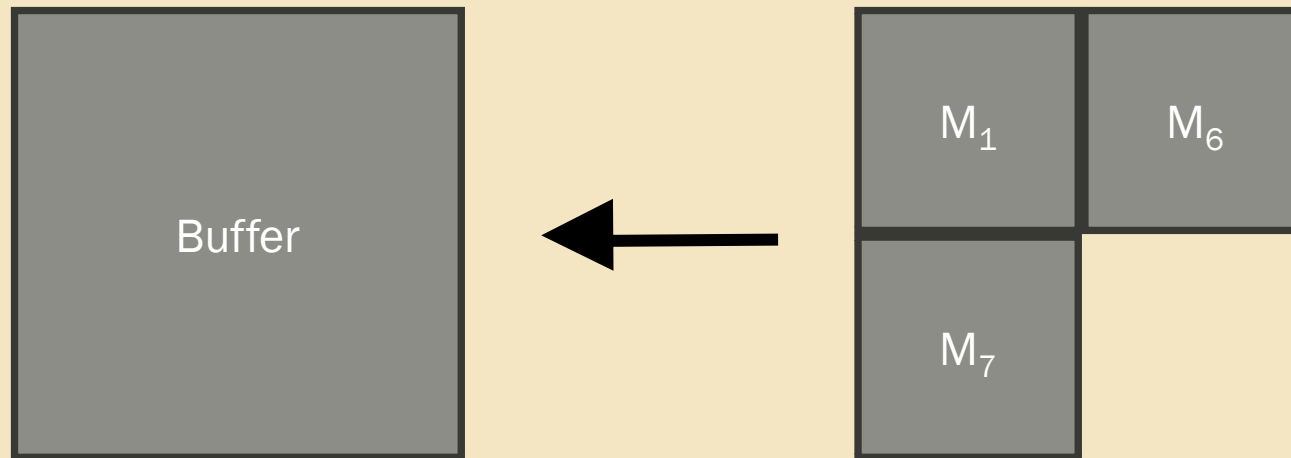
$$M_5 = (A_{11} + A_{12})B_{22}$$

$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$

$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

# THE PARTIAL-STRASSEN ALGORITHM

Store remaining  $M_i$  matrices in a buffer



$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_2 = (A_{21} + A_{22})B_{11}$$

$$M_3 = A_{11}(B_{12} - B_{22})$$

$$M_4 = A_{22}(B_{21} - B_{11})$$

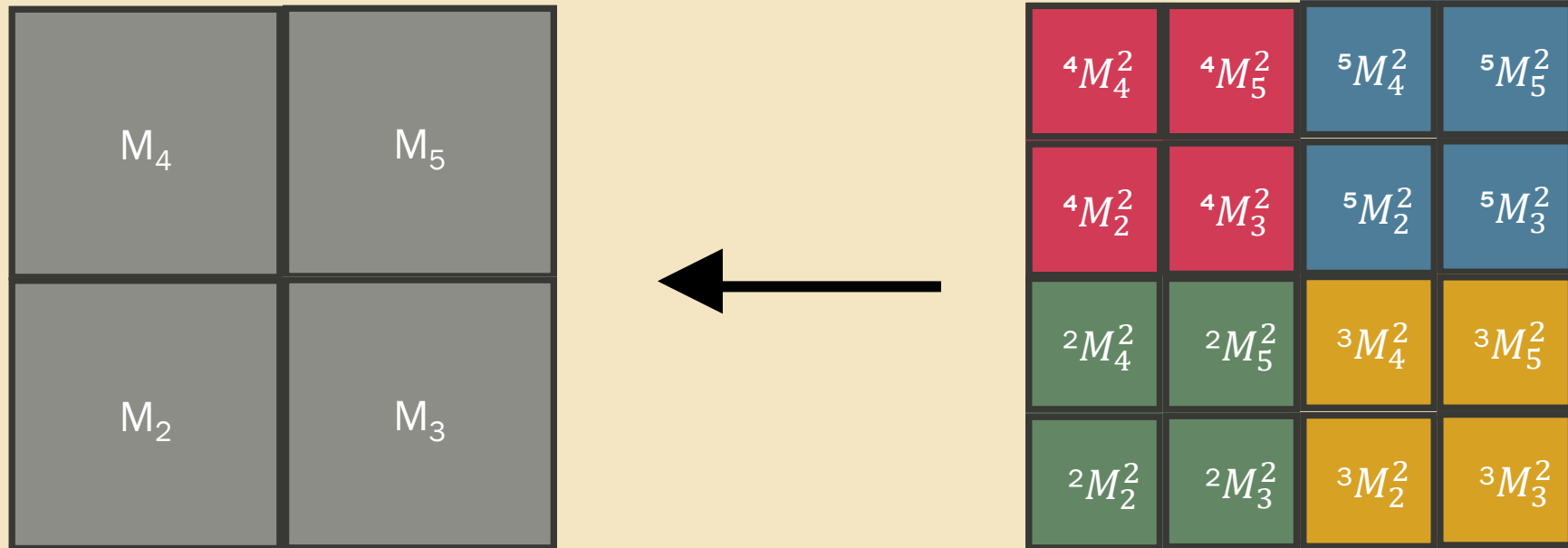
$$M_5 = (A_{11} + A_{12})B_{22}$$

$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$

$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

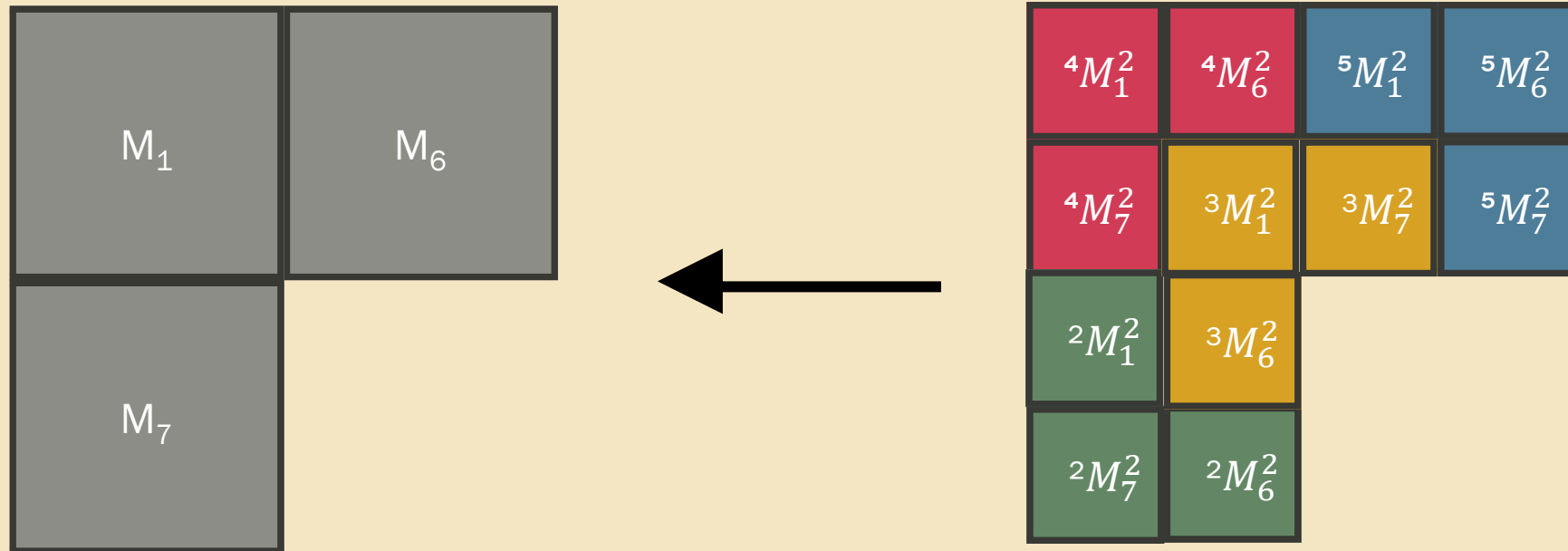
# THE PARTIAL-STRASSEN ALGORITHM

Further subdivide matrices which are stored in C



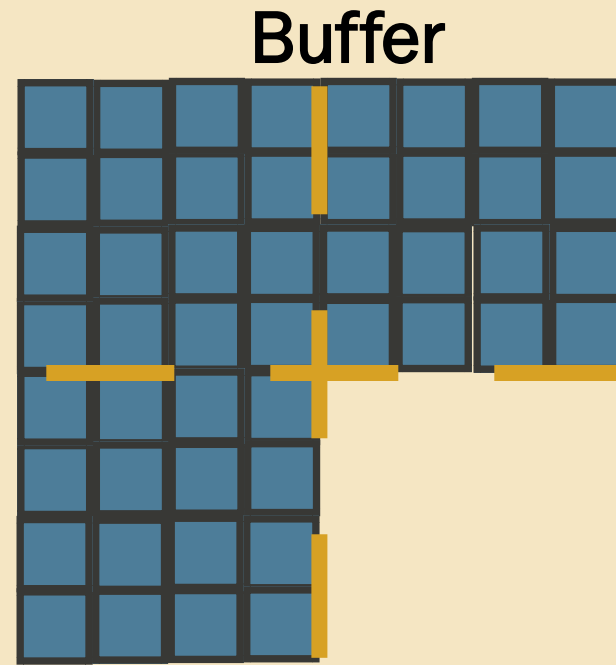
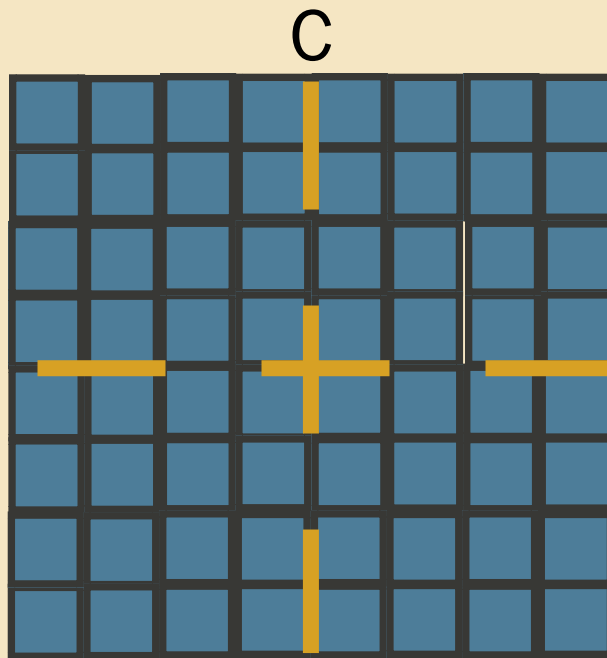
# THE PARTIAL-STRASSEN ALGORITHM

Store the  ${}^jM_i^2$  which don't fit into  $\mathbf{C}$  in our buffer (single-thread representation)



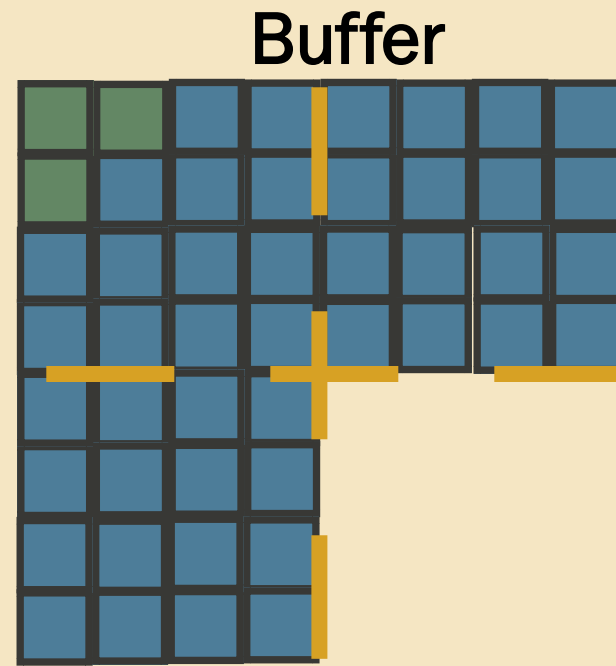
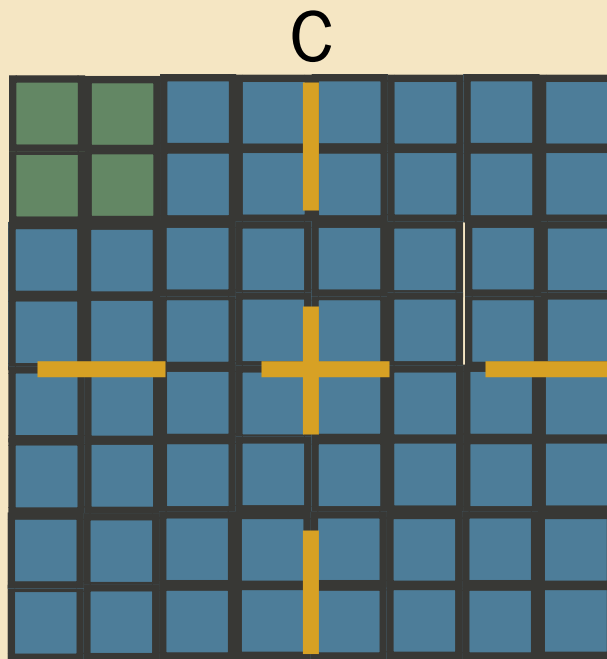
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



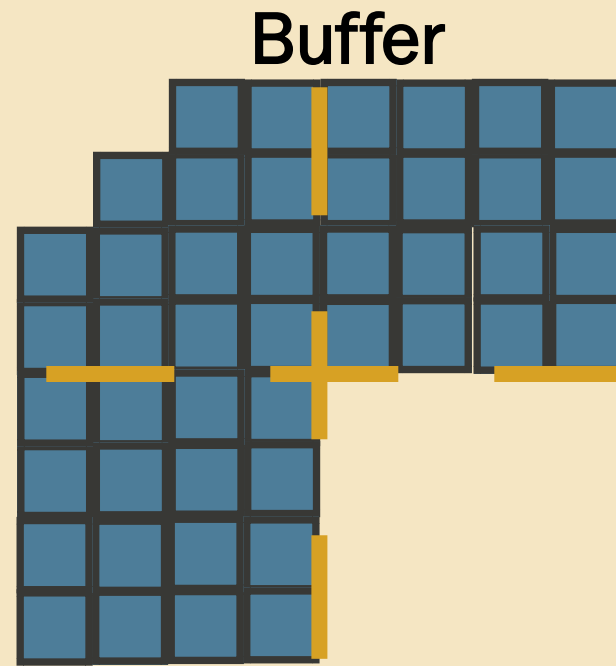
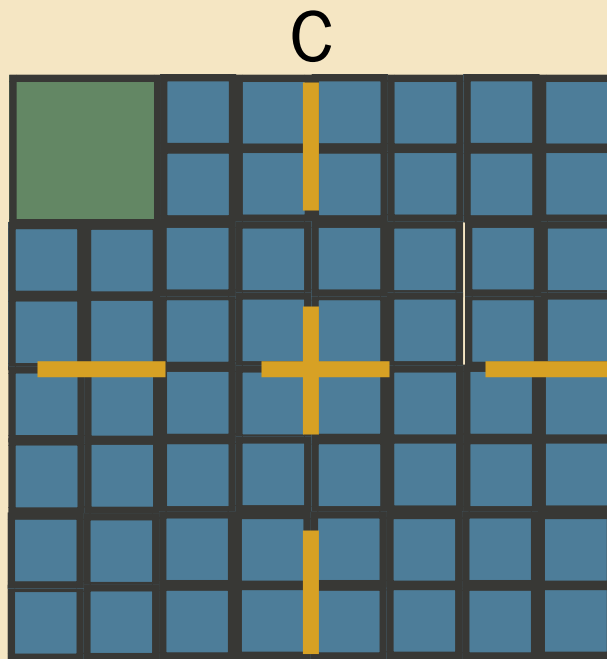
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



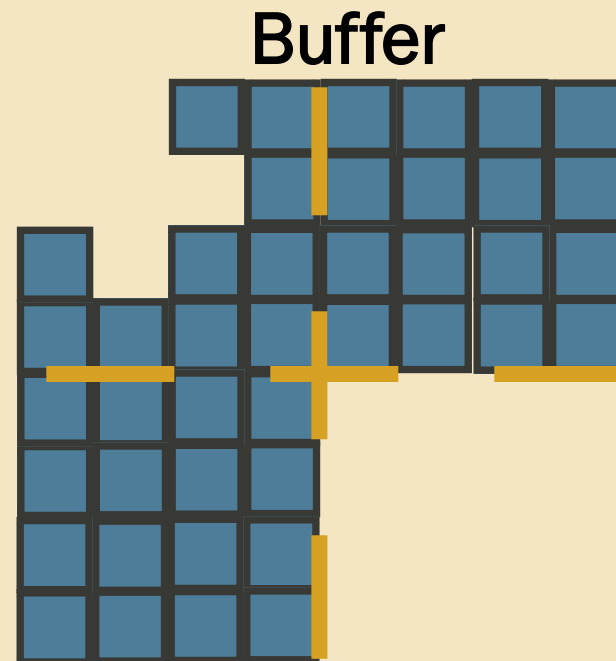
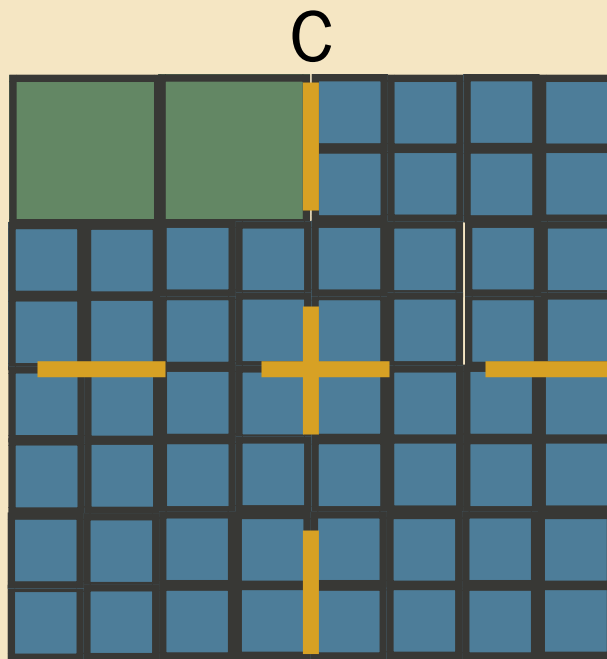
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



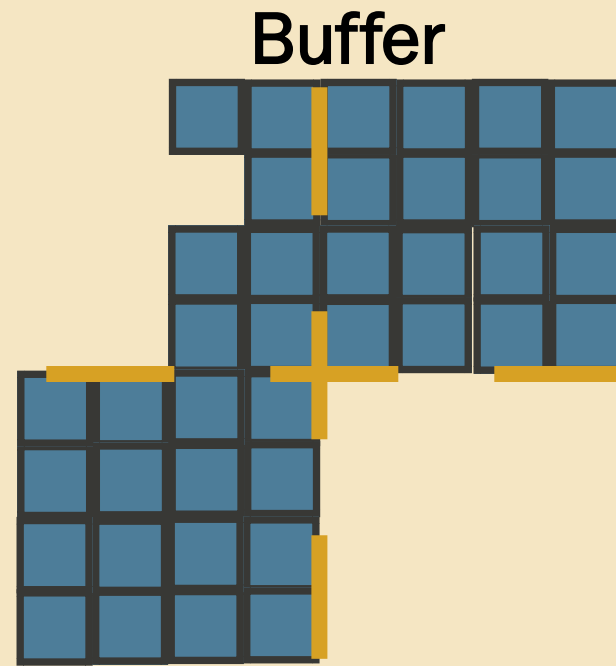
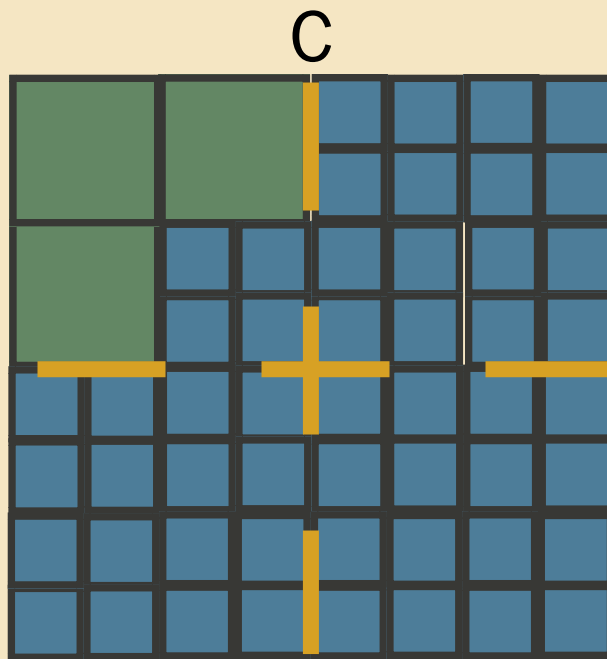
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



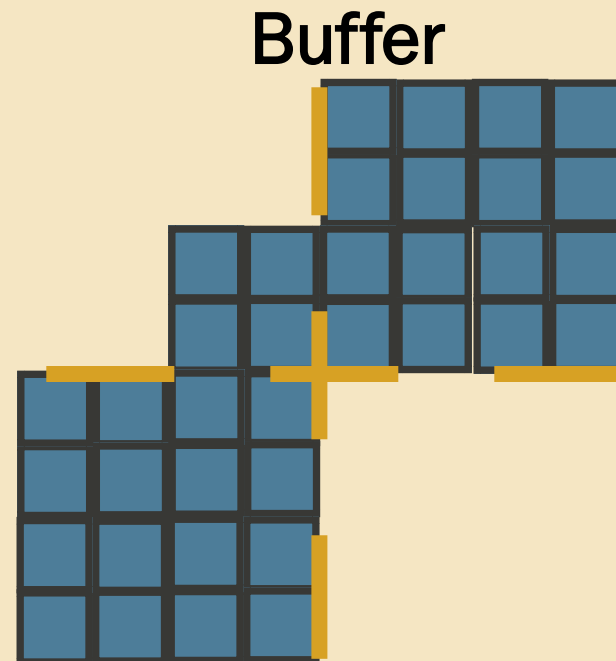
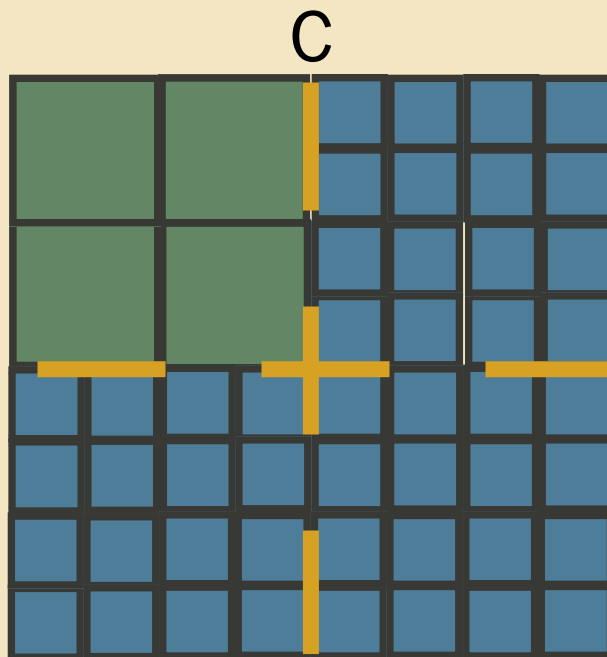
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



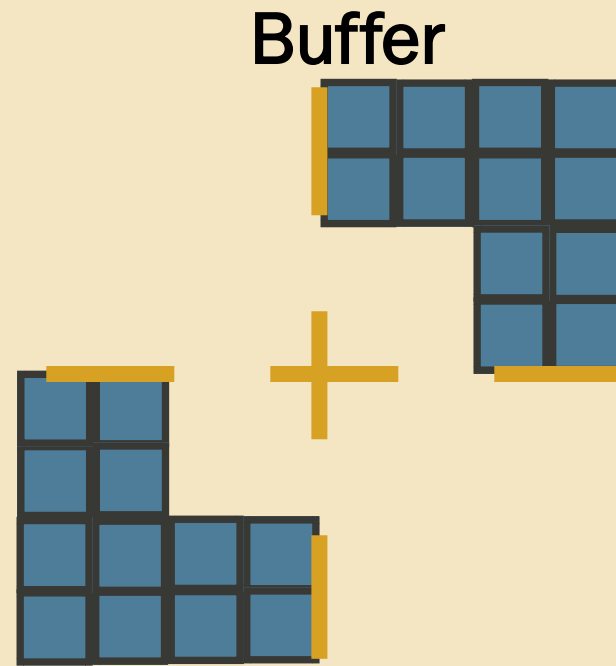
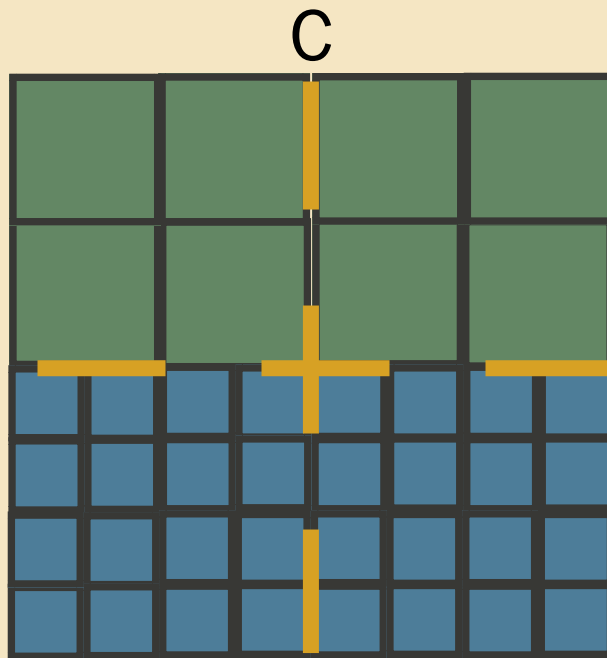
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



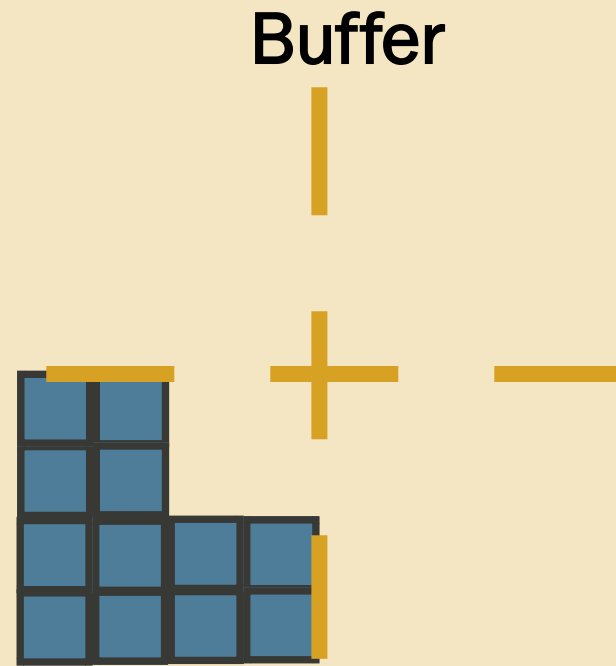
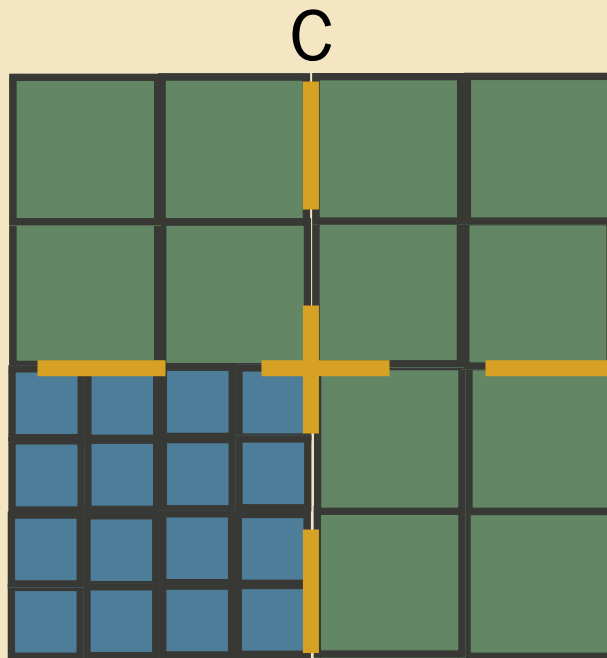
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



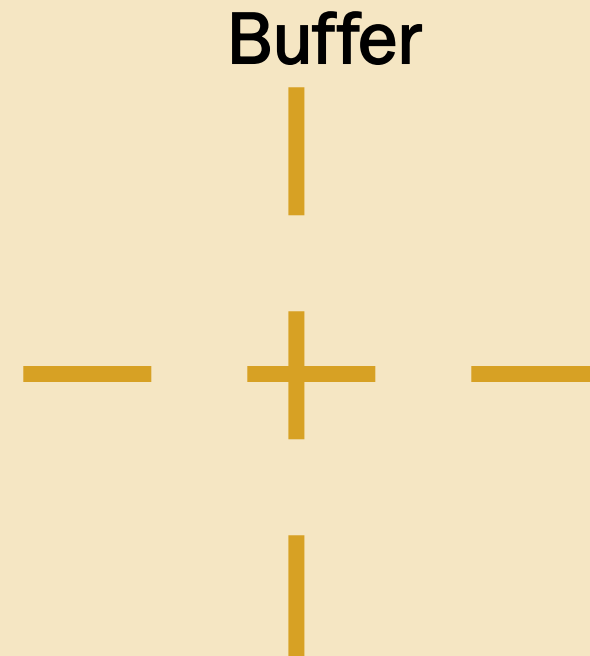
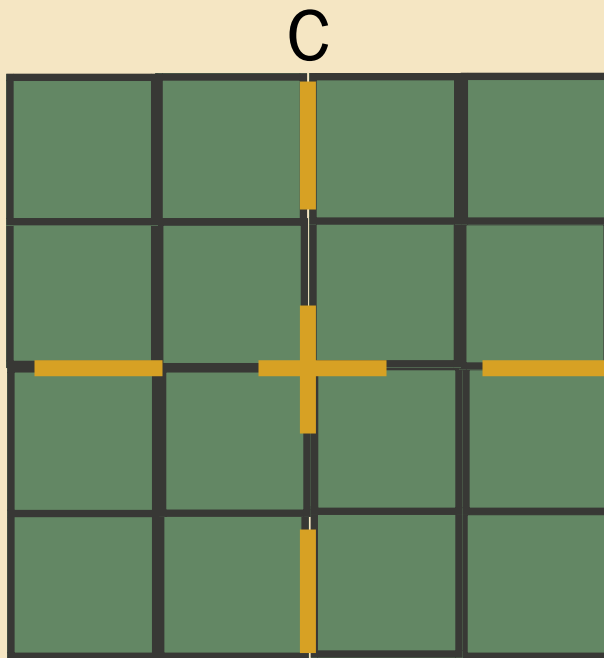
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



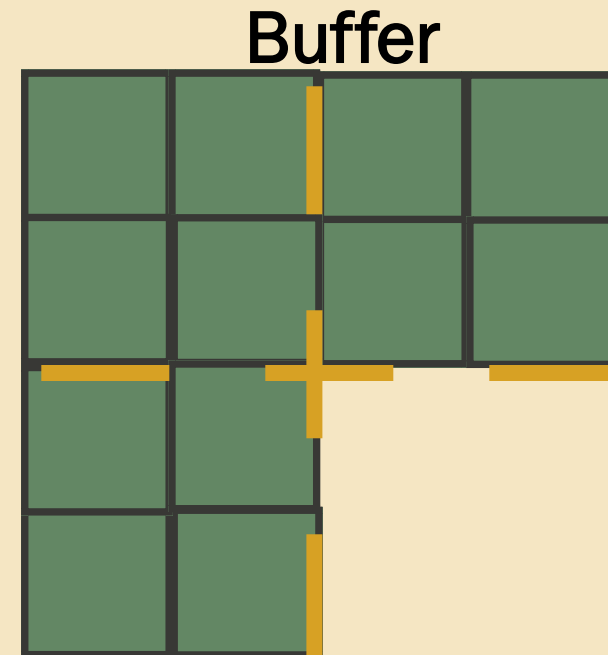
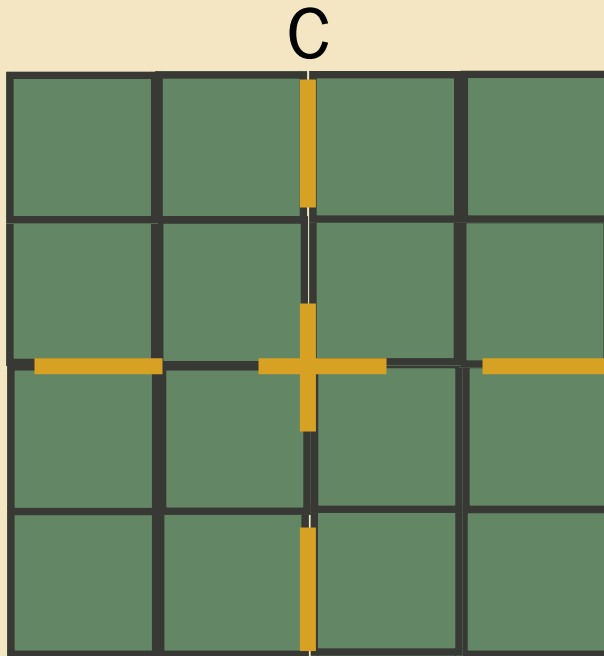
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



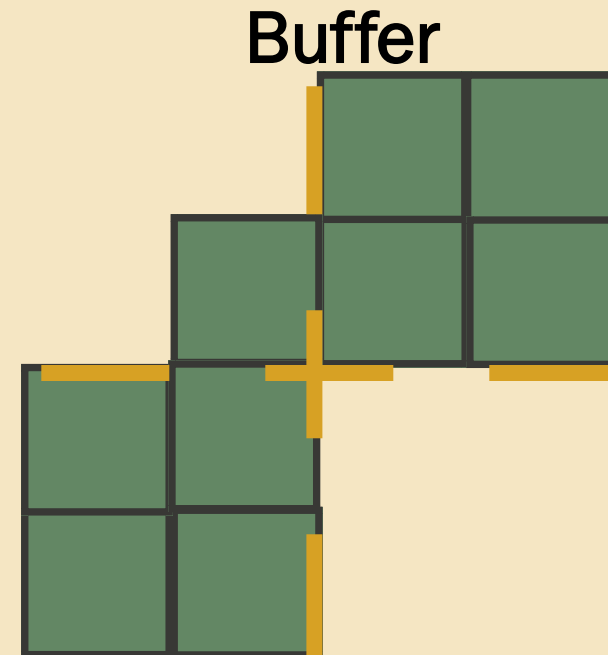
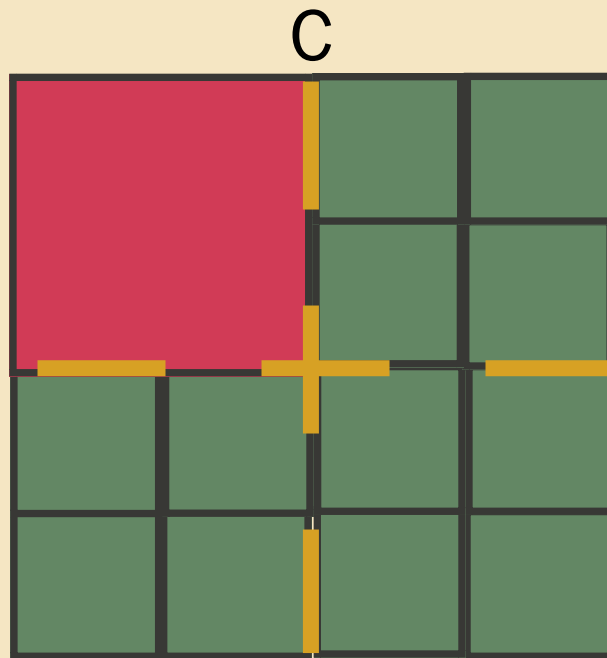
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



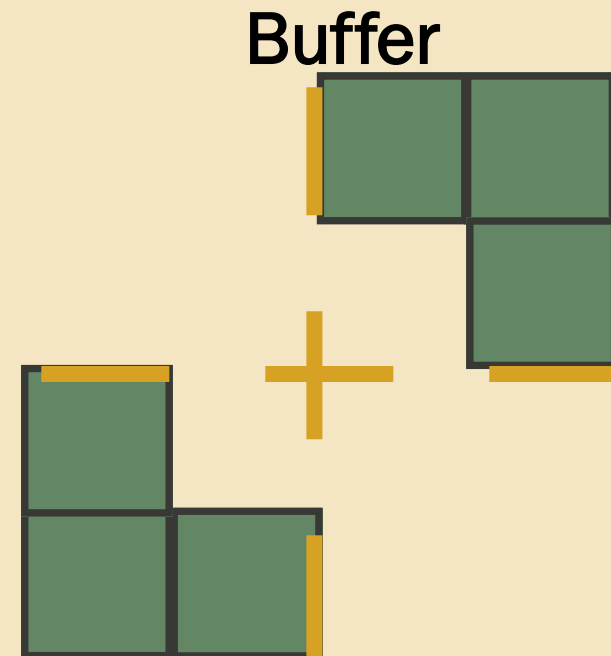
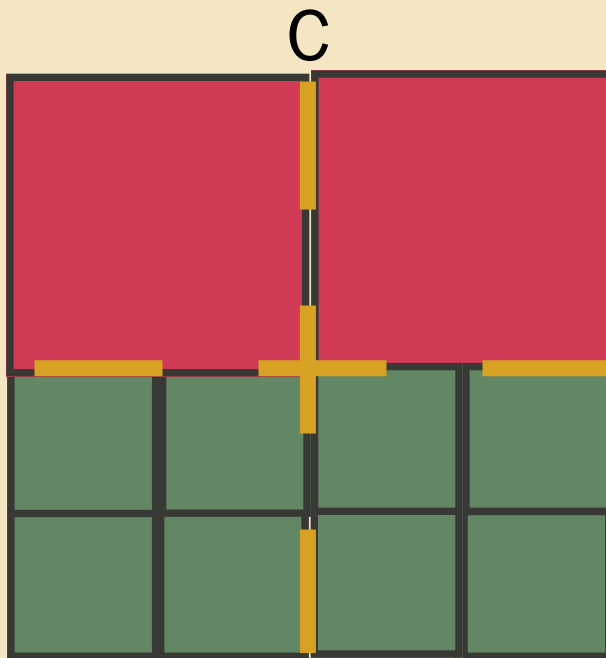
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



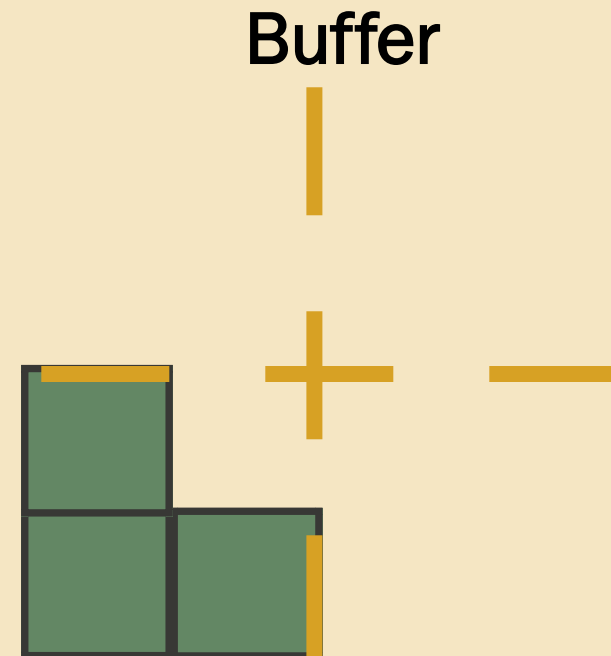
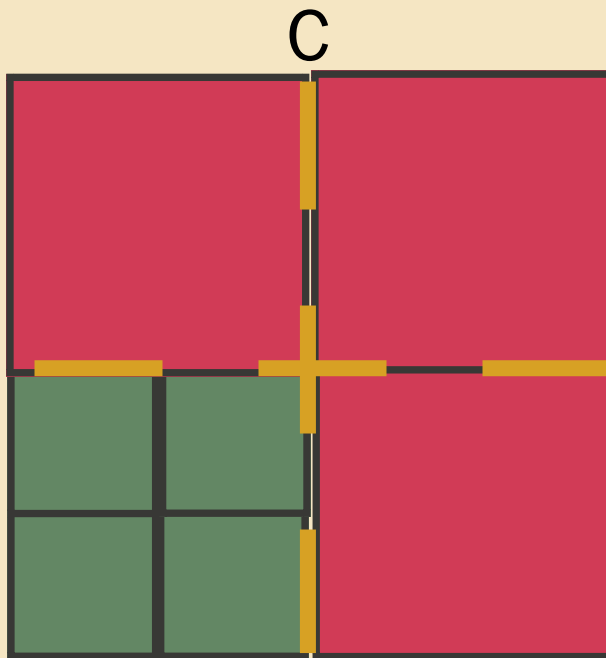
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



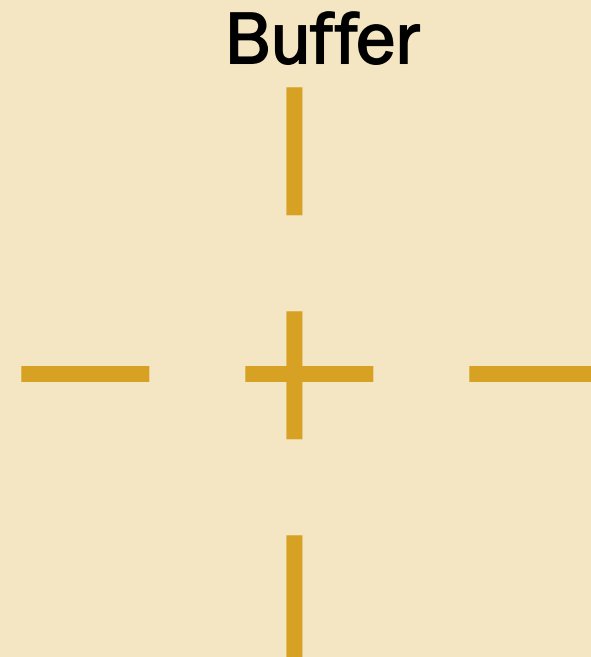
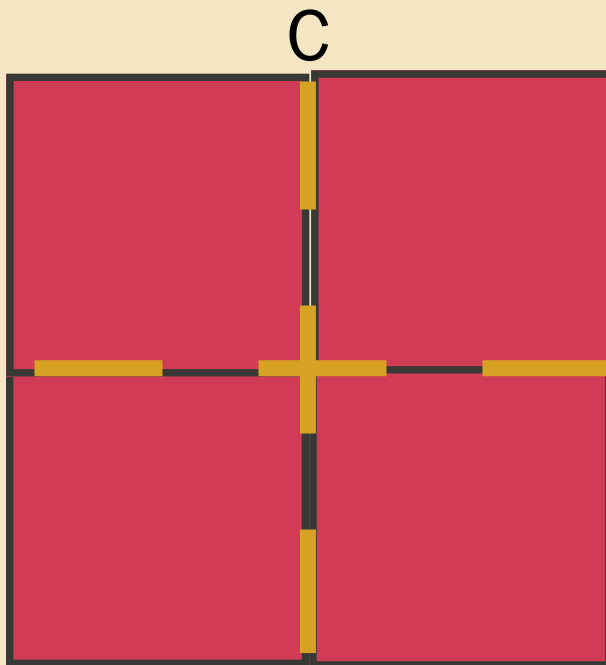
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



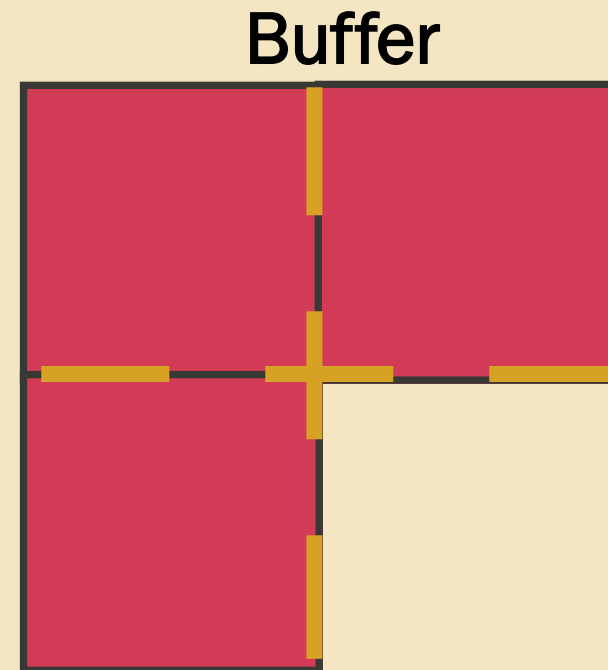
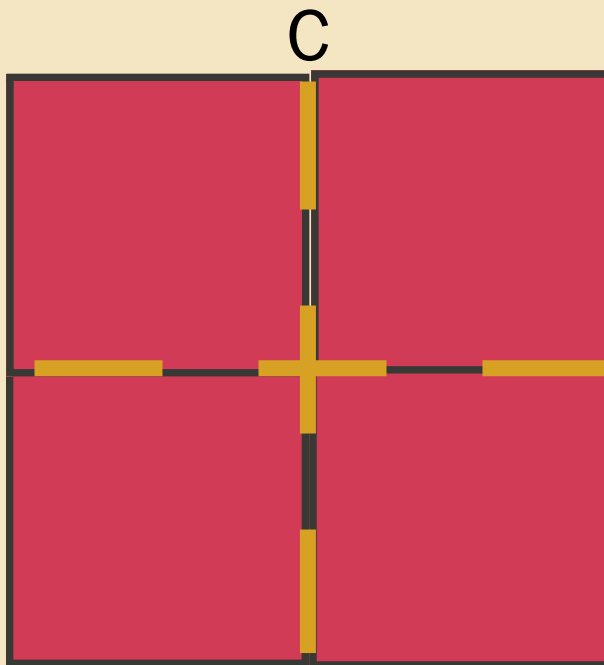
# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up



# THE PARTIAL-STRASSEN ALGORITHM

Repeat all the way down, then build back up

C



# BENEFITS

- Limited memory usage  $cn^2$
- Practical running time

$T =$  Total operations

$k =$  Number of levels

$$T = 4^k \left[ \left(\frac{n}{2^k}\right)^2 + \left(\frac{n}{2^k}\right)^3 \right] + \sum_{i=1}^k \left[ 3 * 4^{i-1} \left[ 2 \left(\frac{n}{2^i}\right)^2 + \left(\frac{n}{2^i}\right)^3 \right] + 8 \left(\frac{n}{2^i}\right)^2 \right] + O(n)$$

# BENEFITS

- Limited memory usage  $cn^2$
- Practical running time

*T = Total operations*

*k = Number of levels*

For sufficiently large  $n$  and  $k$

$$T \approx \frac{3}{4}n^3 + \left(\frac{3}{2}k + 4\right)n^2 + O(n)$$

***Matrix Multiplication in 75% time***  
*(for large matrices)*

# IMPLEMENTATION

- We implemented a single-threaded version of the Partial Strassen Algorithm which can achieve 80% the runtime of BLAS
- We have a multi-threaded version in early development which beats BLAS
- There is a lot of room for improvement

# TAKEAWAY

*We can speed up large matrix multiplications and tensor contractions with limited memory consumption*

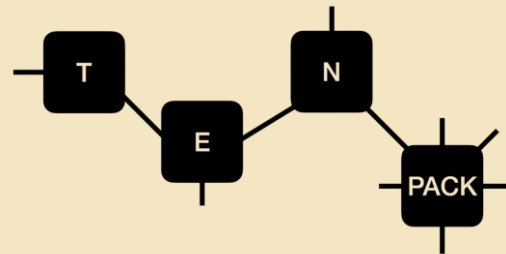


 **DMRGenie**

*[bakerte/DMRGenie.jl](#)*

**DMRjulia**

*[bakerte/DMRJtensor.jl](#)*



*[bakerte/TensorPACK.jl](#)*



*[bakerte/StrassOPen.jl](#)*

# THANK YOU

*Aaron Dayton (Speaker) – Kiana Gallagher – Dr. Thomas E. Baker*

*Department of Physics & Astronomy, University of Victoria, Victoria, Canada*



Chaires de recherche  
du Canada

Canada Research  
Chairs

QUANTUM BC  
research · training · innovation

Canada



**NSERC**  
**CRSNG**



Digital Research  
Alliance of Canada

**UVIC**