Generation of Linear Algebra Algorithms for Automatic Differentiation

Diego Fabregat Traver and Paolo Bientinesi

AICES, RWTH Aachen
fabregat@aices.rwth-aachen.de

International Workshop on Parallel Matrix Algorithms and Applications
June 29 - July 2, 2010
Basel, Switzerland

Automatic Differentiation?

- AD: numerical evaluation of the derivative of a function (computer program)
- $y = sin(x) \xrightarrow{\text{AD Tool}} y' = cos(x)$
- Applications: perturbation analysis, optimization, ...

Motivation - Simple Example

$$f(\alpha, A, x, \beta, y) = \alpha Ax + \beta y \quad \text{(gemv)}$$

Motivation - Simple Example

$$f(\alpha, A, x, \beta, y) = \alpha A x + \beta y \quad \text{(gemv)}$$

$$\frac{df}{dv} = ?$$

Motivation - Simple Example

$$f(\alpha, A, x, \beta, y) = \alpha Ax + \beta y \quad \text{(gemv)}$$

$$\frac{df}{dv} = ?$$

1) $\alpha' Ax + \alpha A'x + \alpha Ax' + \beta' y + \beta y'$

Motivation - Simple Example

$$f(\alpha, A, x, \beta, y) = \alpha Ax + \beta y \quad \text{(gemv)}$$

$$\frac{df}{dv} = ?$$

1) $\alpha' Ax + \alpha A'x + \alpha Ax' + \beta' y + \beta y'$

   ...

   $\alpha A'x + \alpha Ax' + \beta y'$

   ...

32) $\alpha A'x + \beta' y$

$$f(\alpha, A, x, \beta, y) = \alpha Ax + \beta y \quad \text{(gemv)}$$

$$\frac{df}{dv} = ?$$

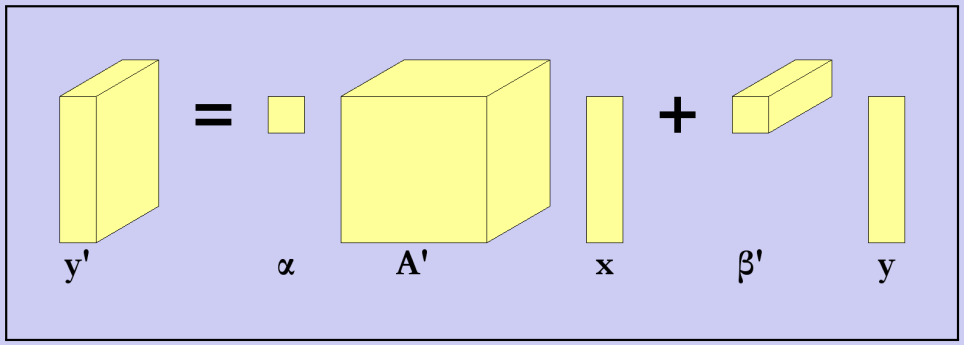
1) $\alpha' Ax + \alpha A'x + \alpha Ax' + \beta' y + \beta y'$

...

$\alpha A'x + \alpha Ax' + \beta y'$

...

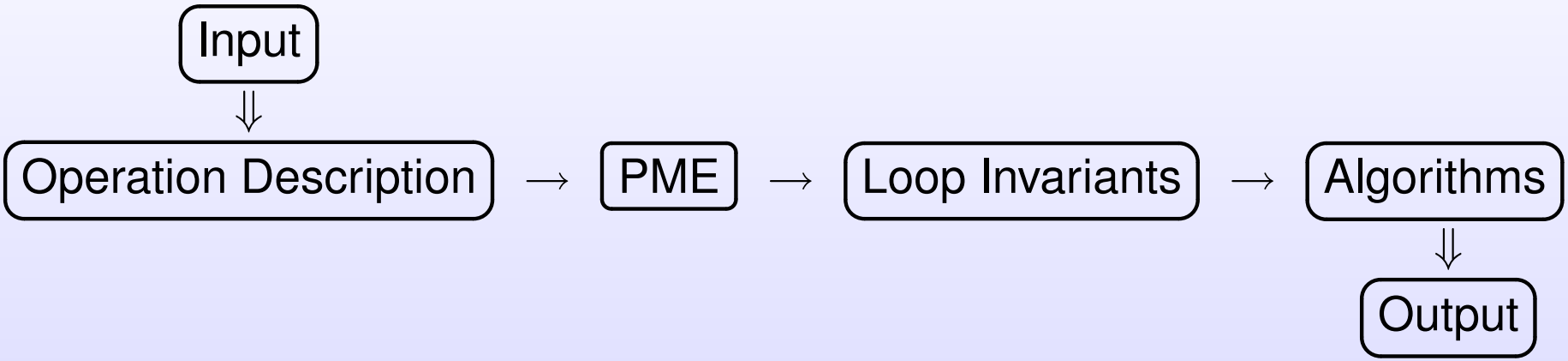
32) $\alpha A'x + \beta' y \quad \rightarrow$

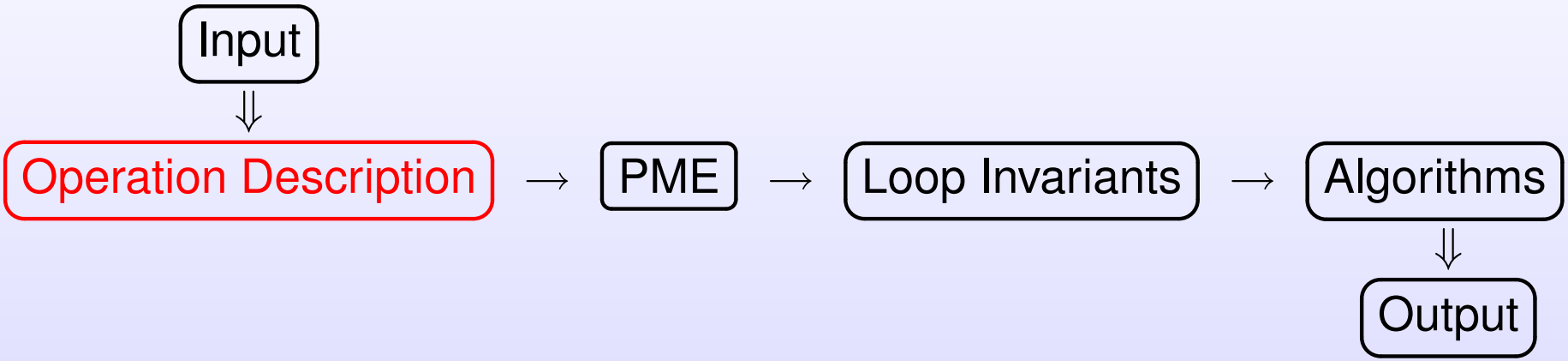


Extra complication: dimensionality

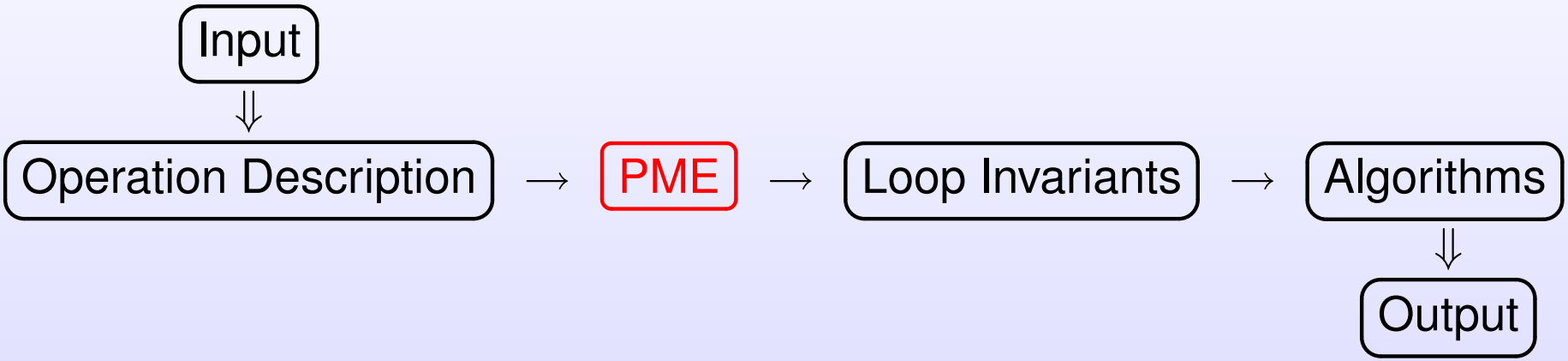
How to map high-dimensional objects onto BLAS?

Outline


Input
Operation Description
PME
Loop Invariants
Algorithms
Output

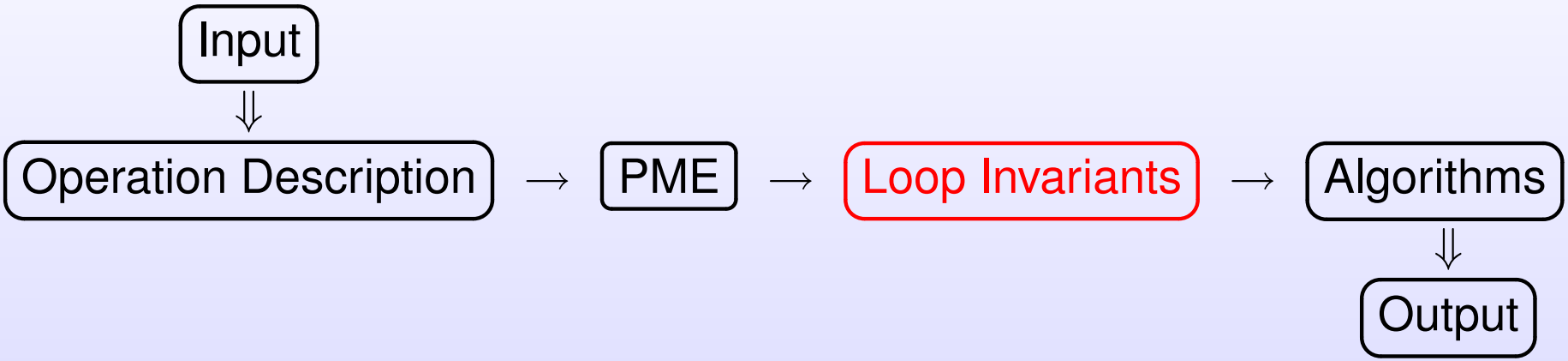


$$f: \quad L := \Gamma(A) \quad \equiv \quad \begin{array}{ll} f_{\text{Pre}}: & \{Input(A) \ \wedge SPD(A) \wedge \\ & \ Output(L) \ \wedge \text{LowTri}(L)\} \\ f_{\text{Post}}: & \{LL^T = A\} \end{array}$$



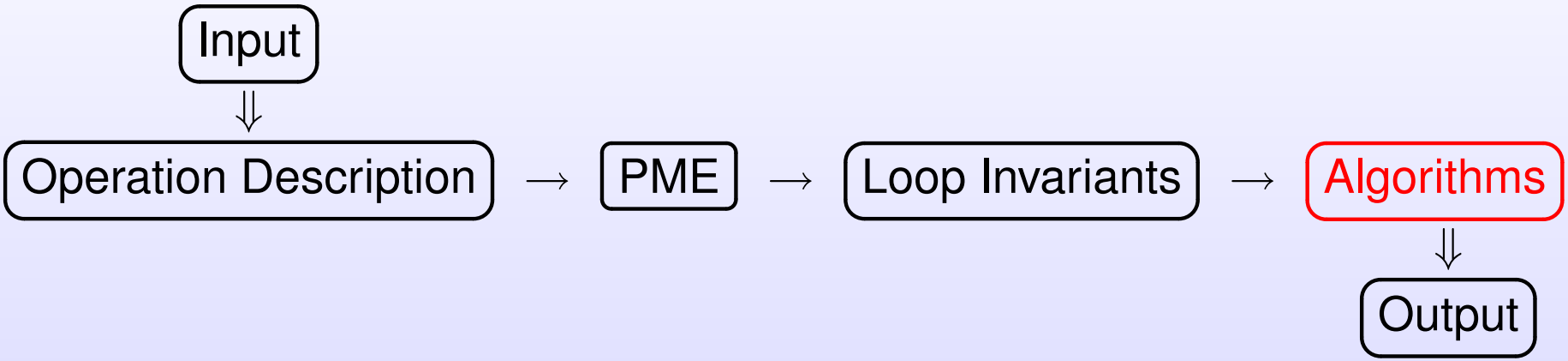
Partitioned Matrix Expression (PME):

$$\left(\begin{array}{c|c} L_{TL} = \Gamma(A_{TL}) & \\ \hline L_{BL} = A_{BL}L_{TL}^{-T} & L_{BR} = \Gamma\left(A_{BR} - L_{BL}L_{BL}^{T}\right) \end{array}\right)$$



Loop Invariants:

- $\left(\begin{array}{c|c} L_{TL} = \Gamma(A_{TL}) & \\ \hline L_{BL} = A_{BL} & L_{BR} = A_{BR} \end{array}\right)$
- $\left(\begin{array}{c|c} L_{TL} = \Gamma(A_{TL}) & \\ \hline L_{BL} = A_{BL}L_{TL}^{-T} & L_{BR} = A_{BR} \end{array}\right)$
- $\left(\begin{array}{c|c} L_{TL} = \Gamma(A_{TL}) & \\ \hline L_{BL} = A_{BL}L_{TL}^{-T} & L_{BR} = A_{BR} - L_{BL}L_{BL}^{T} \end{array}\right)$



Updates:

Variant 1

$$L_{11} = \Gamma(A_{11})$$
$$L_{21} = A_{21} L_{11}^{-T}$$
$$L_{22} = A_{22} - L_{21} L_{21}^{T}$$

Variant 2

$$L_{10} = A_{10} L_{00}^{-T}$$
$$L_{11} = A_{11} - L_{10} L_{10}^{T}$$
$$L_{11} = \Gamma(L_{11})$$

Variant 3

$$L_{11} = A_{11} - L_{10} L_{10}^{T}$$
$$L_{11} = \Gamma(L_{11})$$
$$L_{21} = A_{21} - L_{20} L_{10}^{T}$$
$$L_{21} = L_{21} A_{11}^{T}$$

Outline

$$f(\alpha, x, y) = \alpha x + y \quad (\text{axpy})$$

$$f(\alpha, x, y) = \alpha x + y \quad \text{(axpy)}$$

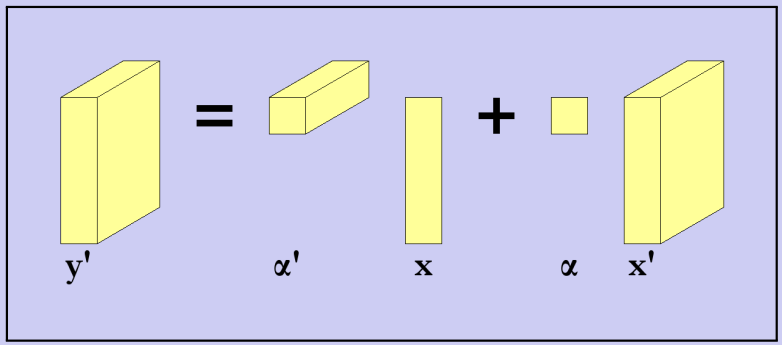
$$\frac{df}{dv} = ?$$

- $\alpha' x + \alpha x' + y'$
- $\alpha x' + y'$
- $\alpha' x + y'$
- $\alpha' x + \alpha x'$
- $\alpha x'$
- $\alpha' x$

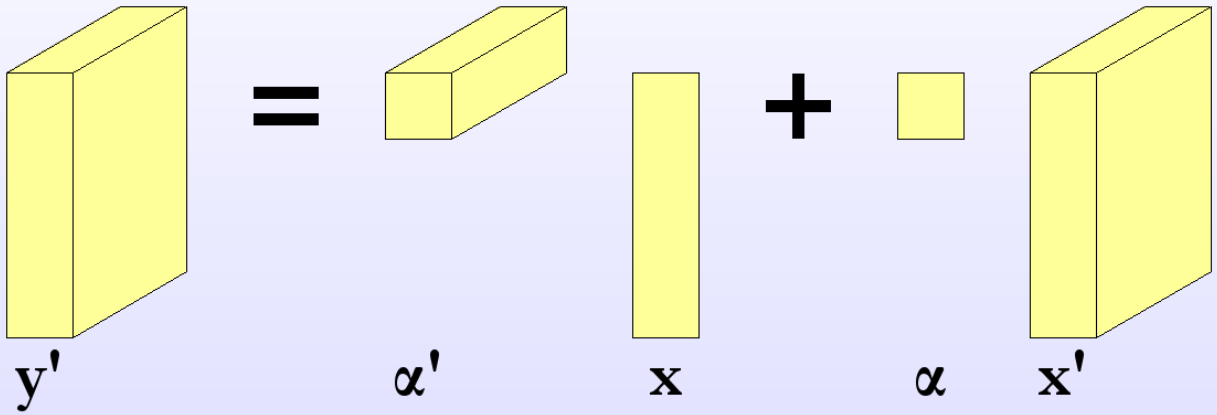
$$f(\alpha, x, y) = \alpha x + y \quad \text{(axpy)}$$

$$\frac{df}{dv} = ?$$

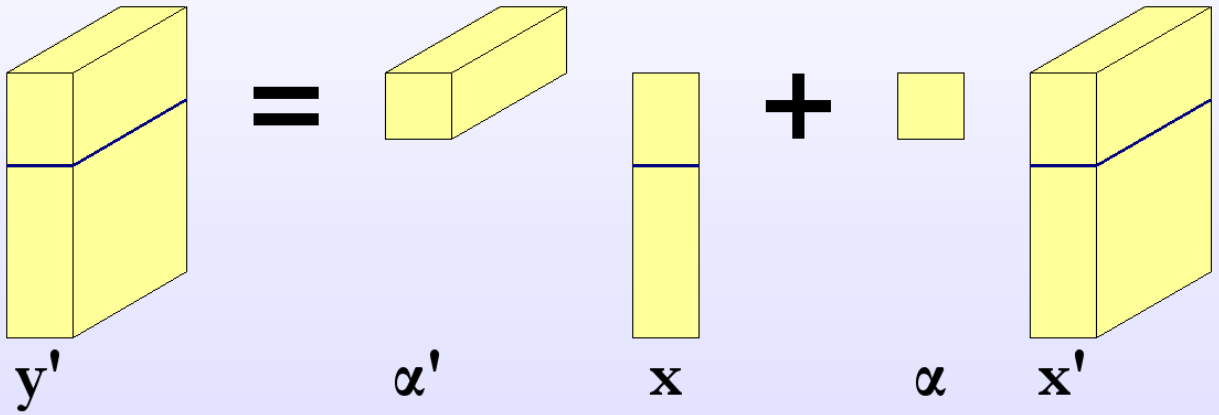
- $\alpha' x + \alpha x' + y'$
- $\alpha x' + y'$
- $\alpha' x + y'$
- $\alpha' x + \alpha x' \quad \rightarrow$
- $\alpha x'$
- $\alpha' x$

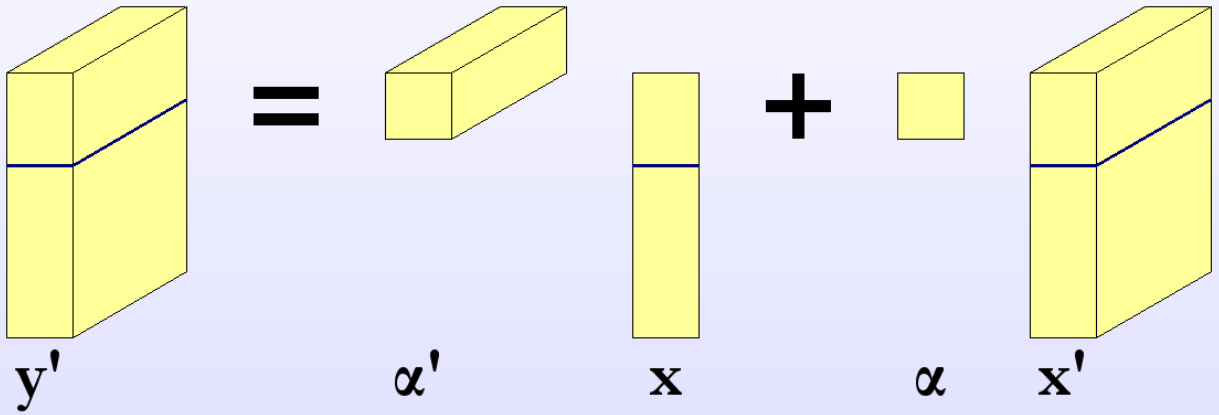


1) D(axpy): $y' = \alpha' x + \alpha x'$



1) D(axpy): Top-Bottom Partitioning

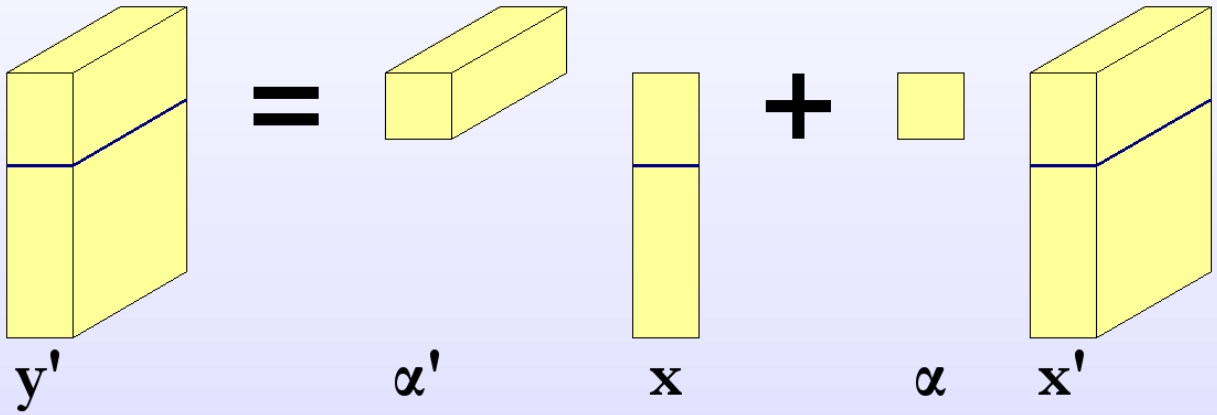




$\Downarrow$

Partitioned postcondition:

$$\begin{pmatrix} y'_T \\ y'_B \end{pmatrix} = \alpha' \begin{pmatrix} x_T \\ x_B \end{pmatrix} + \alpha \begin{pmatrix} x'_T \\ x'_B \end{pmatrix}$$

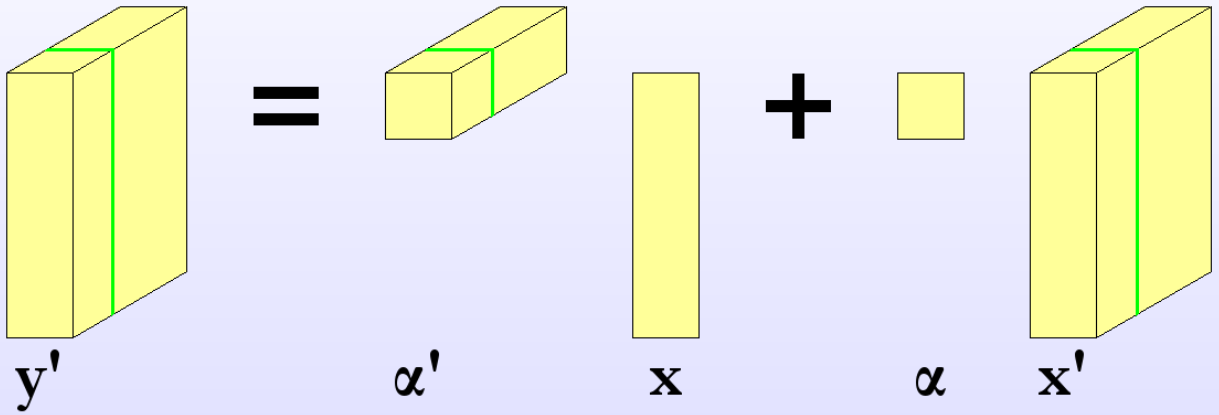


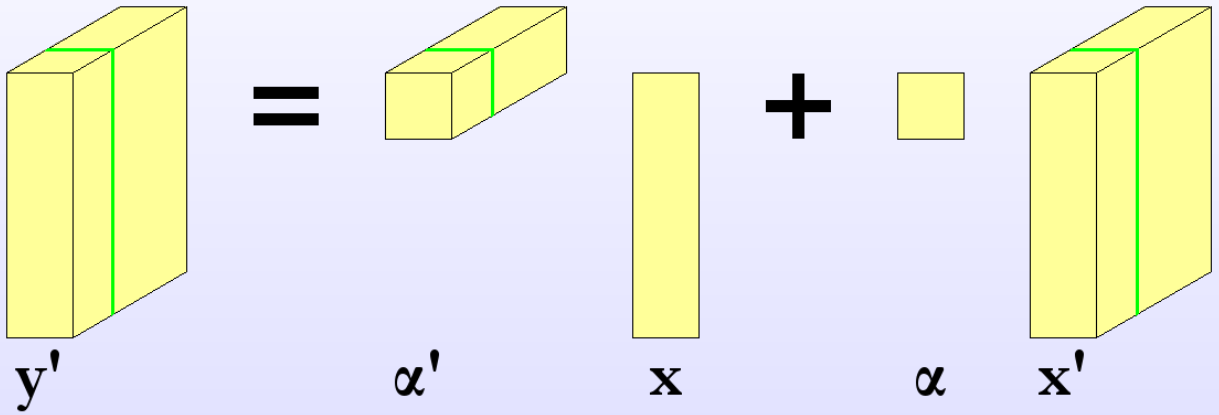
$\Downarrow$

Partitioned Matrix Expression (PME):

$$\left(\begin{array}{l} y'_T = \alpha' x_T + \alpha x'_T \\ \hline y'_B = \alpha' x_B + \alpha x'_B \end{array} \right)$$

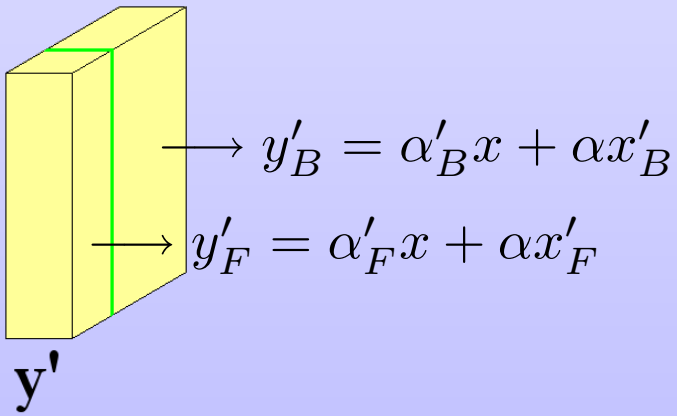
2) D(axpy): Front-Back Partitioning





$\Downarrow$

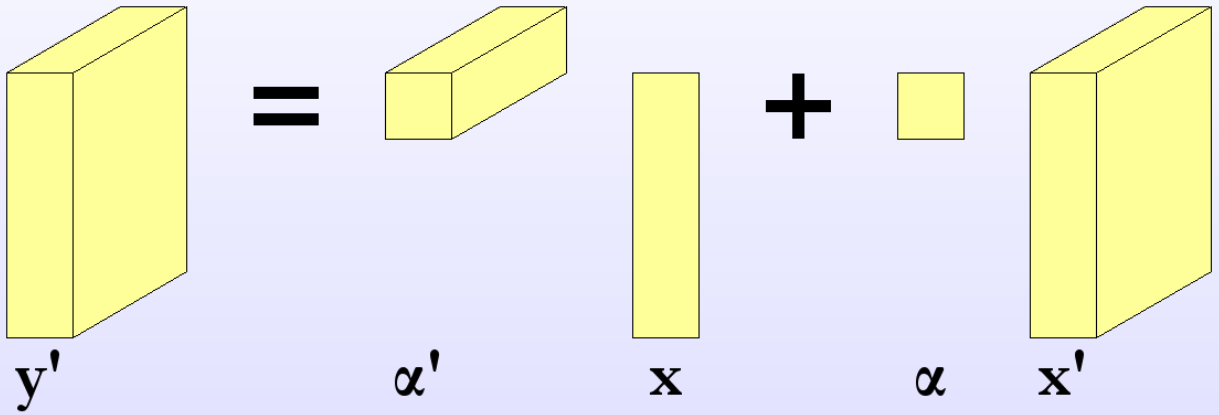
Partitioned Matrix Expression (PME):



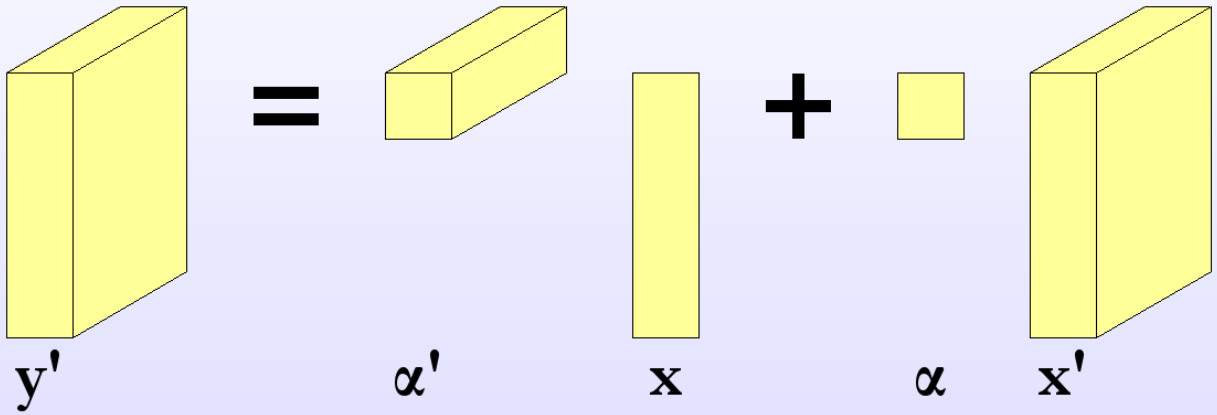
$$\longrightarrow y'_B = \alpha'_B x + \alpha x'_B$$

$$\longrightarrow y'_F = \alpha'_F x + \alpha x'_F$$

3) D(axpy): No Partitioning



3) D(axpy): Ger + Scal



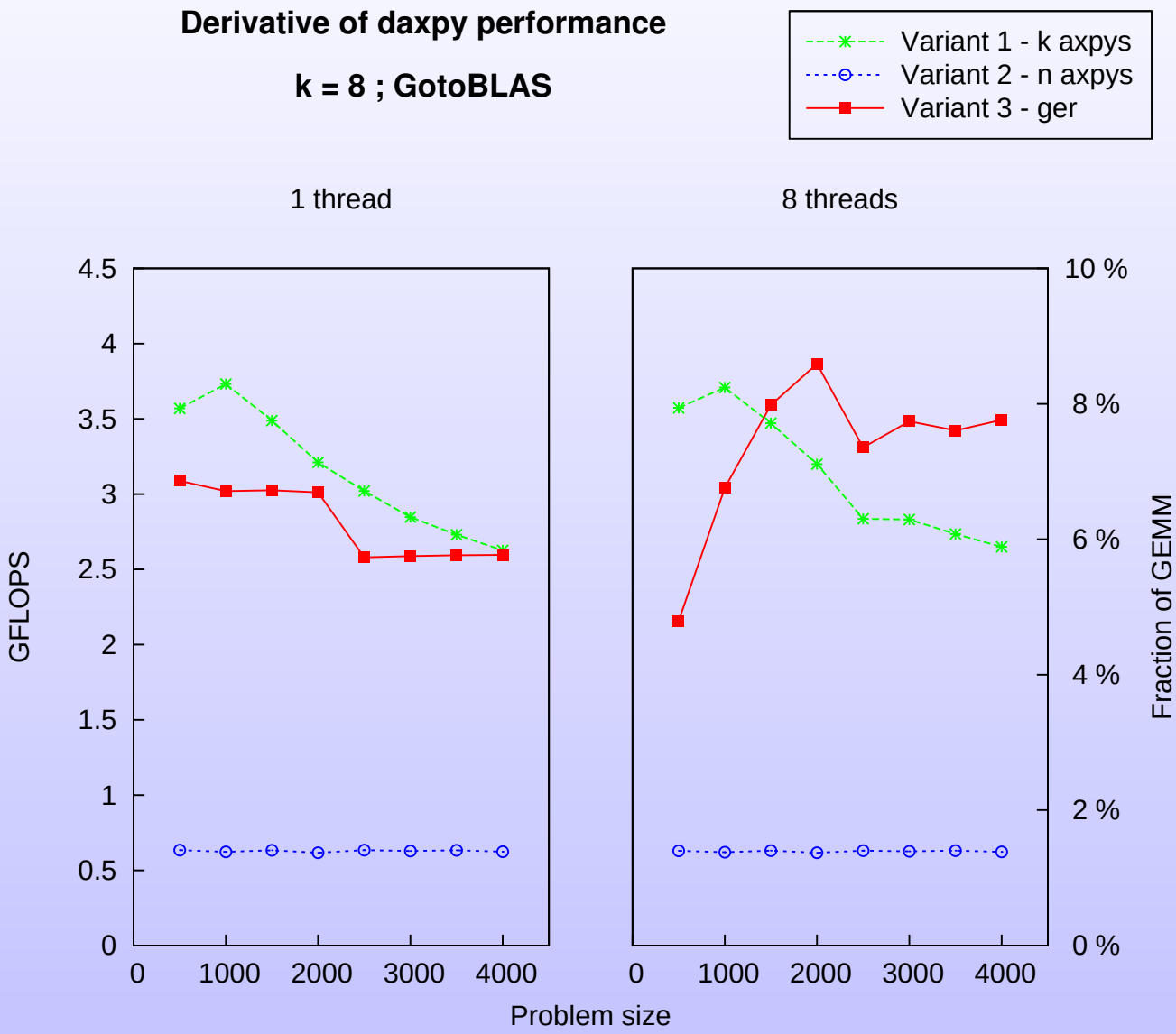
$$\Downarrow$$

$$y' = \alpha x'$$

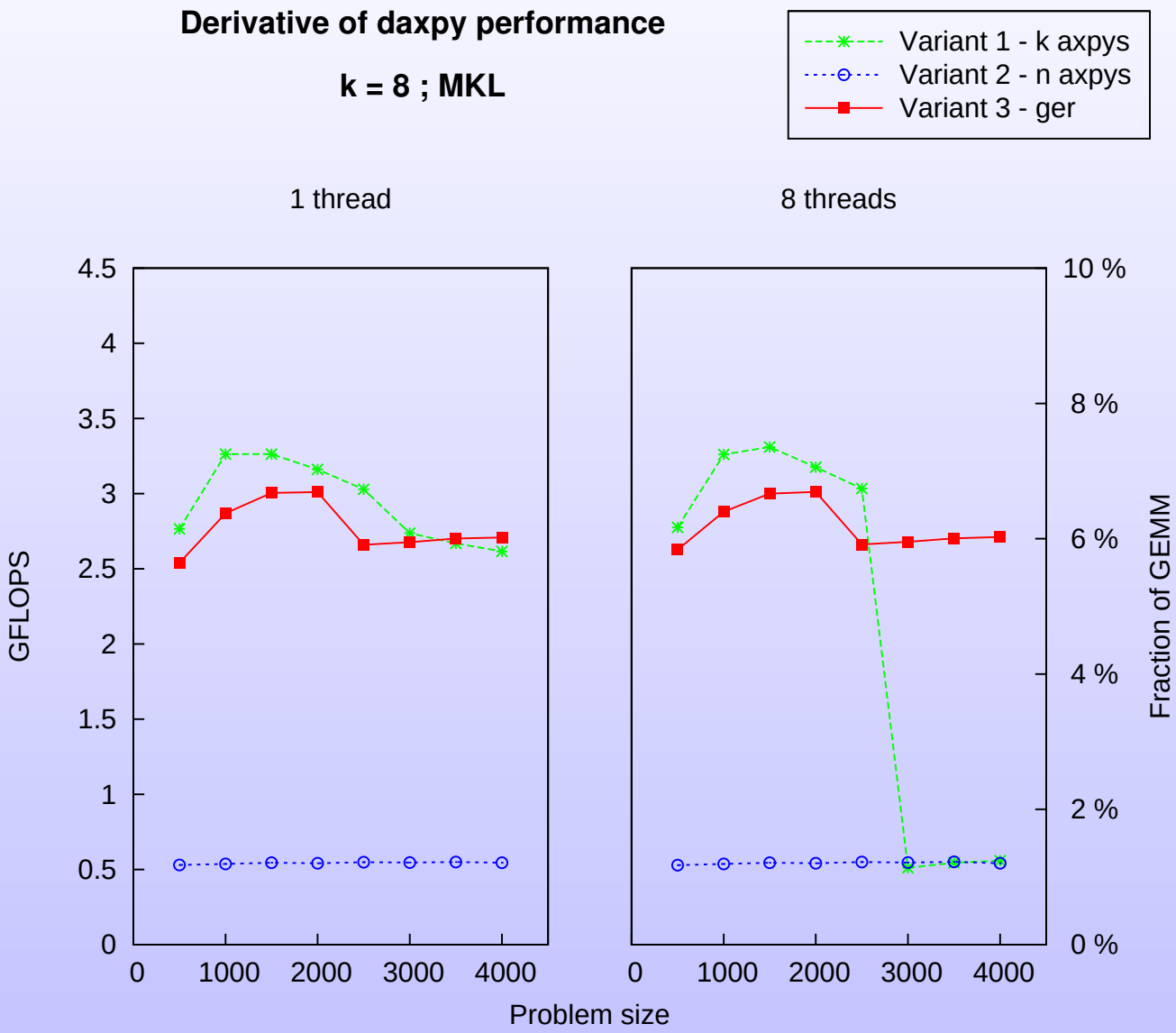
$$y' = \alpha' x + y'$$

Outline

D(axpy) – Performance



D(axpy) – Performance



Conclusions

We have shown:

- Derivative of a linear algebra operation $\longrightarrow$ several kernels
- For each kernel $\longrightarrow$ family of algorithmic variants
- Automation + High Performance

Conclusions

We have shown:

- Derivative of a linear algebra operation $\rightarrow$ several kernels
- For each kernel $\rightarrow$ family of algorithmic variants
- Automation + High Performance

Future Research

- Can we apply this methodology for even higher-dimensional objects?

Thanks to:

- Dr. Edoardo Di Napoli
- Matthias Petschow
- Roman Iakymchuk

Funding from DFG is gratefully acknowledged

Deutsche Forschungsgemeinschaft DFG