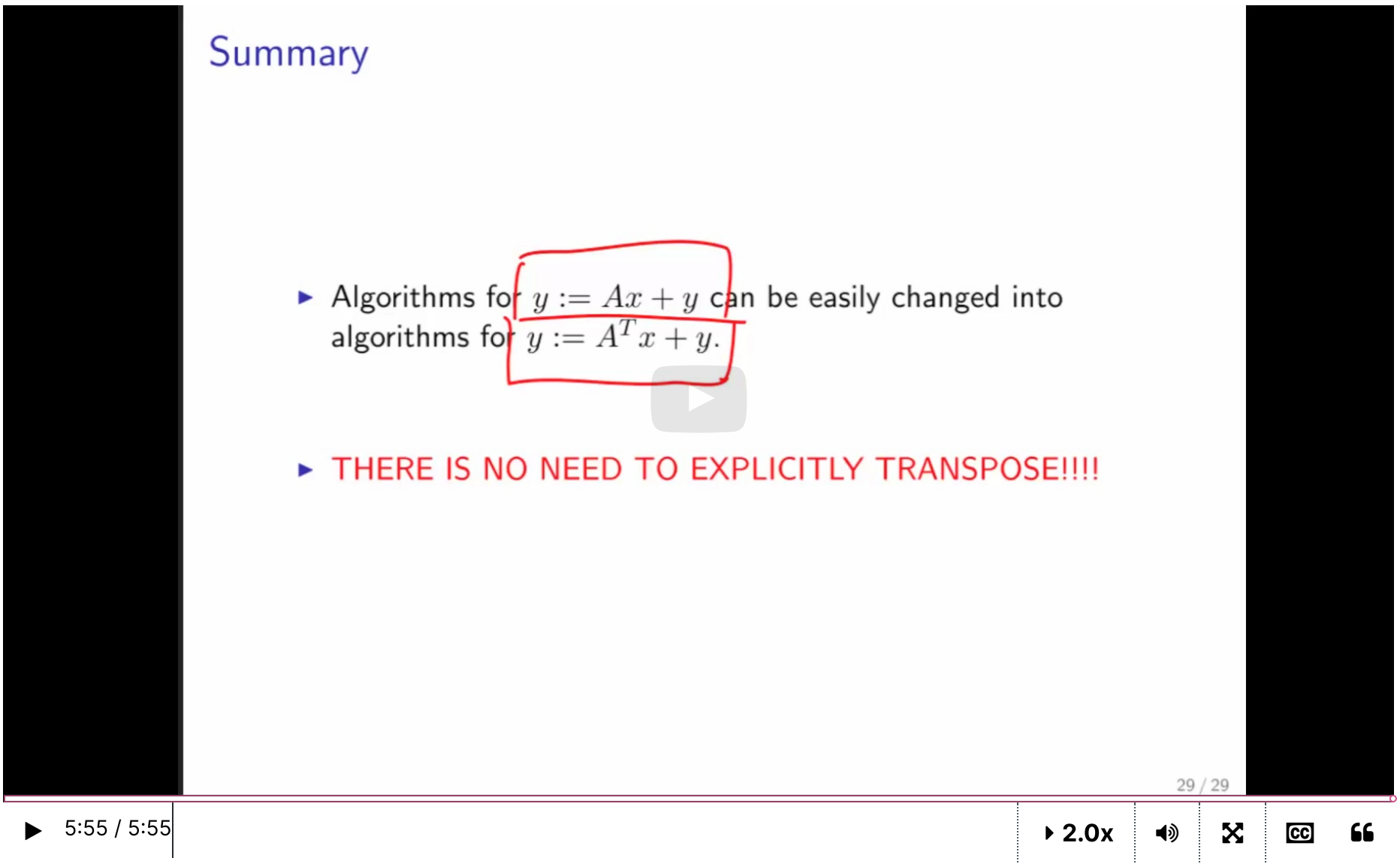
4.3.1 Transpose Matrix-Vector Multiplication

Bookmark this page

Week 4 due Oct 24, 2023 19:42 IST

4.3.1 Transpose Matrix-Vector Multiplication



Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Reading Assignment

0 points possible (ungraded)

Read Unit 4.3.1 of the notes. [LINK]

Done

Submit

Correct

Discussion

Topic: Week 4 / 4.3.1

Hide Discussion

Add a Post

Show all posts by recen

Calculator

- Does not a matrix-vector multiplication require memops? — 2
  In the reading (p. 133) and the video is said: "A matrix-vector multiplication requires 2mn flops. Transposing a matrix requires 2mn memops (mn ...
- what's the point of transposing if you could do the dot product? — 3
  why did we ever transpose if we could just take the dot product and be more efficient in our memory processing? i'm don't understand what it m...

Homework 4.3.1.1

1/1 point (graded)

Algorithm: $y := \text{MVMULT\_T\_UNB\_VAR1}(A, x, y)$

Partition $A \rightarrow \left(A_L \middle| A_R \right)$, $y \rightarrow \left(\frac{y_T}{y_B} \right)$

where A_L is $m \times 0$ and y_T is 0×1

while $m(y_T) < m(y)$ **do**

Repartition

$$\left(A_L \middle| A_R \right) \rightarrow \left(A_0 \middle| a_1 \middle| A_2 \right), \left(\frac{y_T}{y_B} \right) \rightarrow \left(\frac{y_0}{\psi_1} \over y_2 \right)$$

$\psi_1 := a_1^T x + \psi_1$

Continue with

$$\left(A_L \middle| A_R \right) \leftarrow \left(A_0 \middle| a_1 \middle| A_2 \right), \left(\frac{y_T}{y_B} \right) \leftarrow \left(\frac{y_0}{\psi_1} \over y_2 \right)$$

endwhile

Algorithm: $y := \text{MVMULT\_T\_UNB\_VAR2}(A, x, y)$

Partition $A \rightarrow \left(\frac{A_T}{A_B} \right)$, $x \rightarrow \left(\frac{x_T}{x_B} \right)$

where A_T is $0 \times n$ and x_T is 0×1

while $m(A_T) < m(A)$ **do**

Repartition

$$\left(\frac{A_T}{A_B} \right) \rightarrow \left(\frac{A_0}{a_1^T} \over A_2 \right), \left(\frac{x_T}{x_B} \right) \rightarrow \left(\frac{x_0}{\chi_1} \over x_2 \right)$$

$y := \chi_1 a_1 + y$

Continue with

$$\left(\frac{A_T}{A_B} \right) \leftarrow \left(\frac{A_0}{a_1^T} \over A_2 \right), \left(\frac{x_T}{x_B} \right) \leftarrow \left(\frac{x_0}{\chi_1} \over x_2 \right)$$

endwhile

Write routines

- `[ y_out ] = Mvmult_t_unb_var1( A, x, y )`
- `[ y_out ] = Mvmult_t_unb_var2( A, x, y )`

that compute $y := A^T x + y$ using the algorithms in this unit.

Some links that will come in handy:

- Spark (alternatively, open the file `LAFF-2.0xM/Spark/index.html`)
- PictureFLAME (alternatively, open the file `LAFF-2.0xM/PictureFLAME/PictureFLAME.html`)

You may want to use the following scripts to test your implementations (these should be in your directory `LAFF-2.0xM/Programming/Week04/`):

- test_Mvmult_t_unb_var1.m
- test_Mvmult_t_unb_var2.m

☑ Done/Skip

✔

View document with most algorithms and implementations for this week.

Submit

Calculator

Answers are displayed within the problem

Homework 4.3.1.2

2/2 points (graded)

Implementations achieve better performance (finish faster) if one accesses data consecutively in memory. Now, most scientific computing codes store matrices in "column-major order" which means that the first column of a matrix is stored consecutively in memory, then the second column and so forth. Now, this means that an algorithm that accesses a matrix by columns tends to be faster than an algorithm that accesses a matrix by rows. That, in turn, means that when one is presented with more than one algorithm, one should pick the algorithm that accesses the matrix by columns.

Our FLAME notation makes it easy to recognize algorithms that access the matrix by columns.

• For the matrix-vector multiplication of $Ax + y$, would you recommend the algorithm that uses dot products or the algorithm that uses axpy operations?

axpy ✔ Answer: axpy

• For the matrix-vector multiplication of $A^T x + y$, would you recommend the algorithm that uses dot products or the algorithm that uses axpy operations?

dot ✔ Answer: dot

Explanation

Answer: When computing $Ax + y$, it is when you view Ax as taking linear combinations of the *columns* of A that you end up accessing the matrix by columns. Hence, the `axpy`-based algorithm will access the matrix by columns.
When computing $A^T x + y$, it is when you view $A^T x$ as taking dot products of *columns* of A with vector x that you end up accessing the matrix by columns. Hence, the `dot`-based algorithm will access the matrix by columns.

‹ Previous Next ›

Your Privacy Choices

Connect

Idea Hub
Contact Us
Help Center
Security
Media Kit

Calculator