Look ahead technique for reduction to Hessenberg form: design of the algorithm and applicability on current hardware

Julien Langou, Matthew Nabity
Department of Mathematical & Statistical Sciences
University of Colorado Denver

May 14, 2009

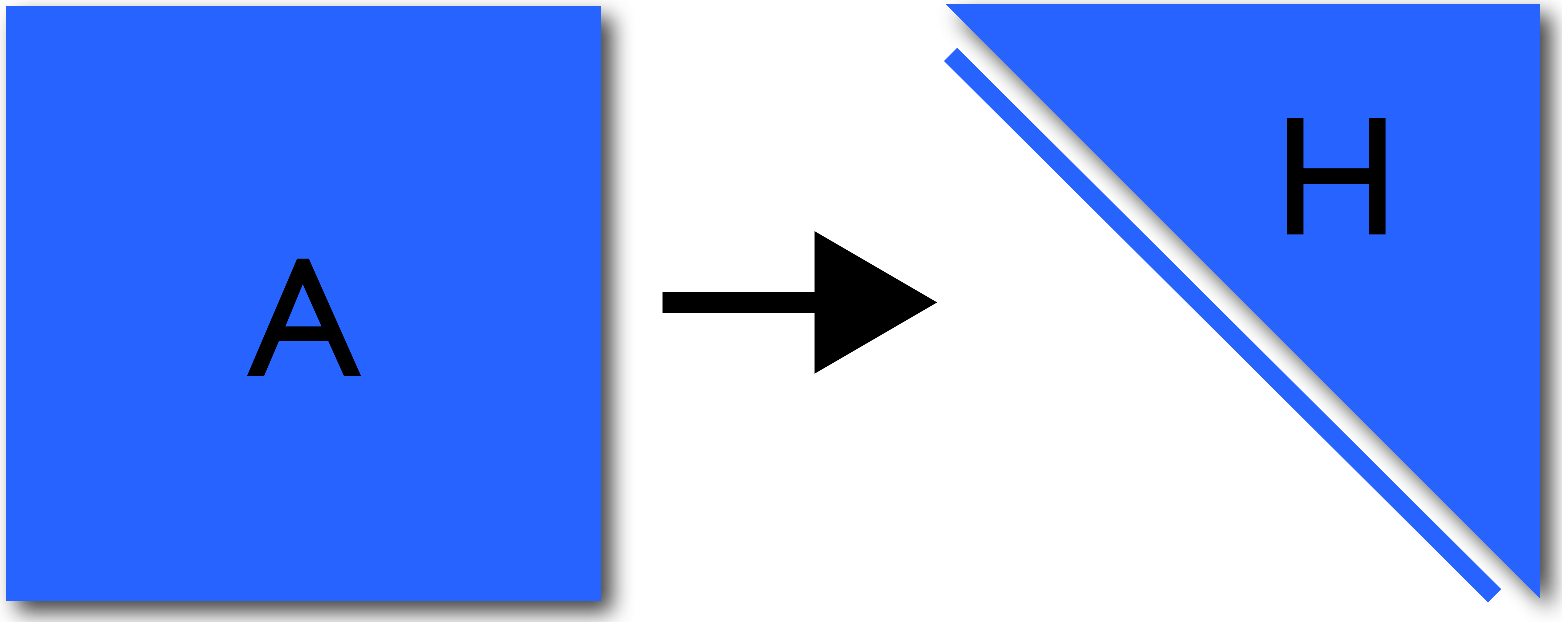
Center for Computational Mathematics

Outline

- Background and motivation
- LAPACK routines (unblocked and blocked)
- Look ahead algorithm
- To do list

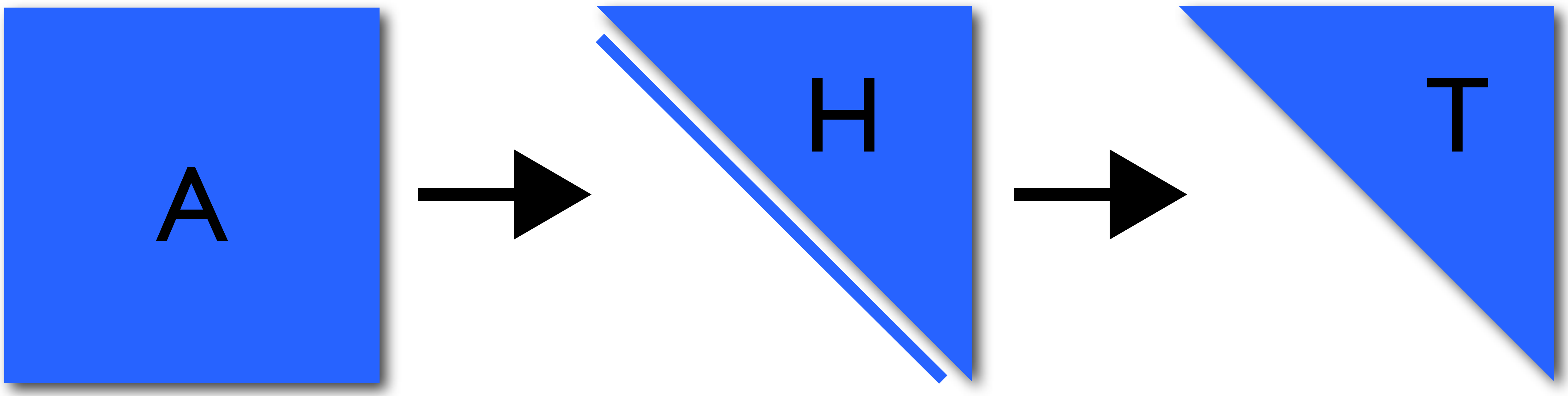
Motivation

- Reduction to Hessenberg form is the first phase of solving the nonsymmetric eigenvalue problem
- $H = Q^T A Q$ with $Q^T Q = I$
- Cost ~ $(10/3)n^3$

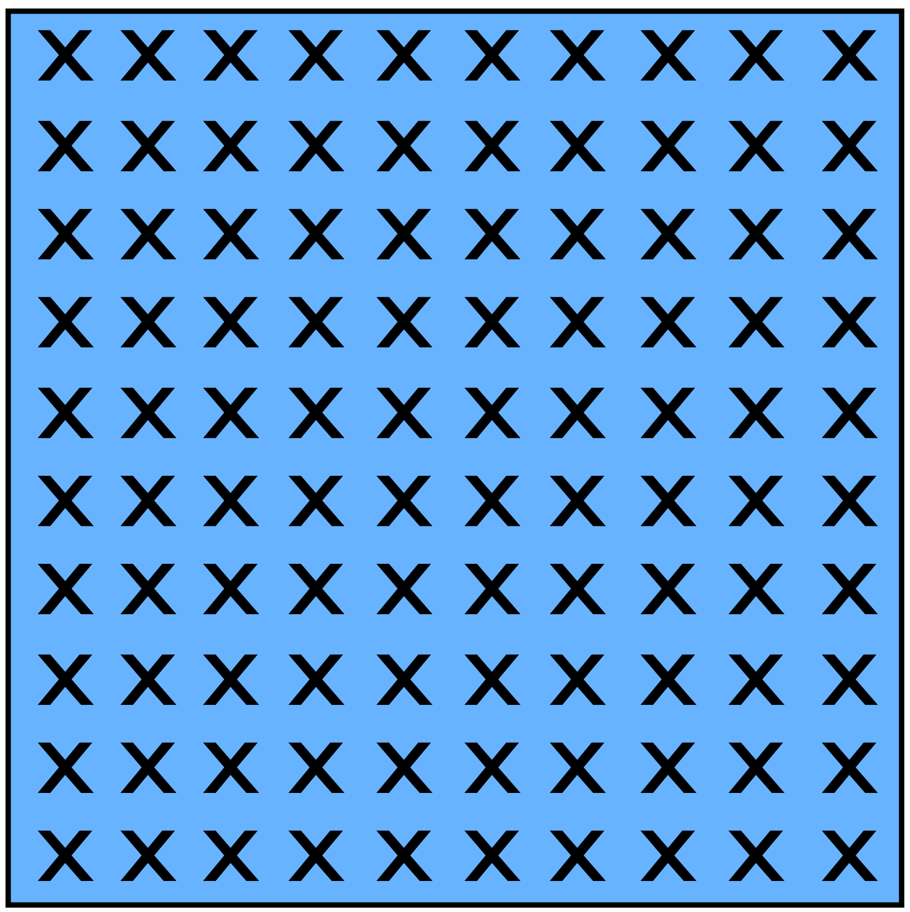


Motivation

- Can continue to Schur Form and to Diagonal form for an eigenvalue revealing factorization
- $Q^T Q = I$, unitary transformations = stable computation

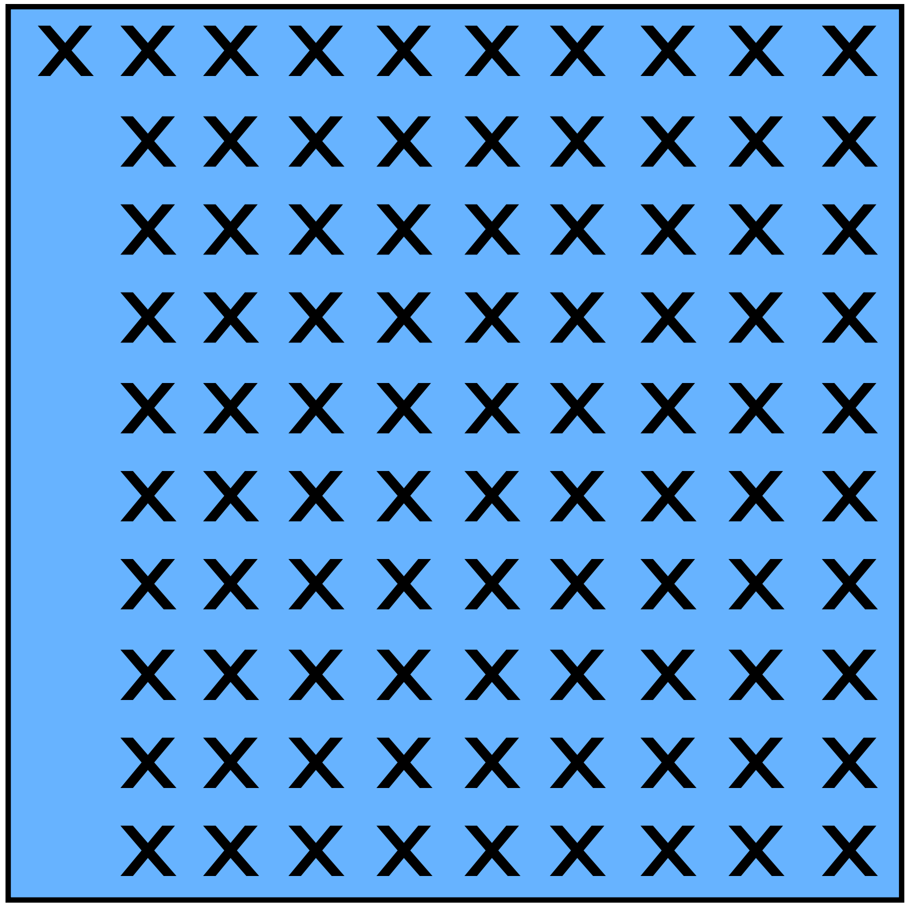


Why Hessenberg Form?



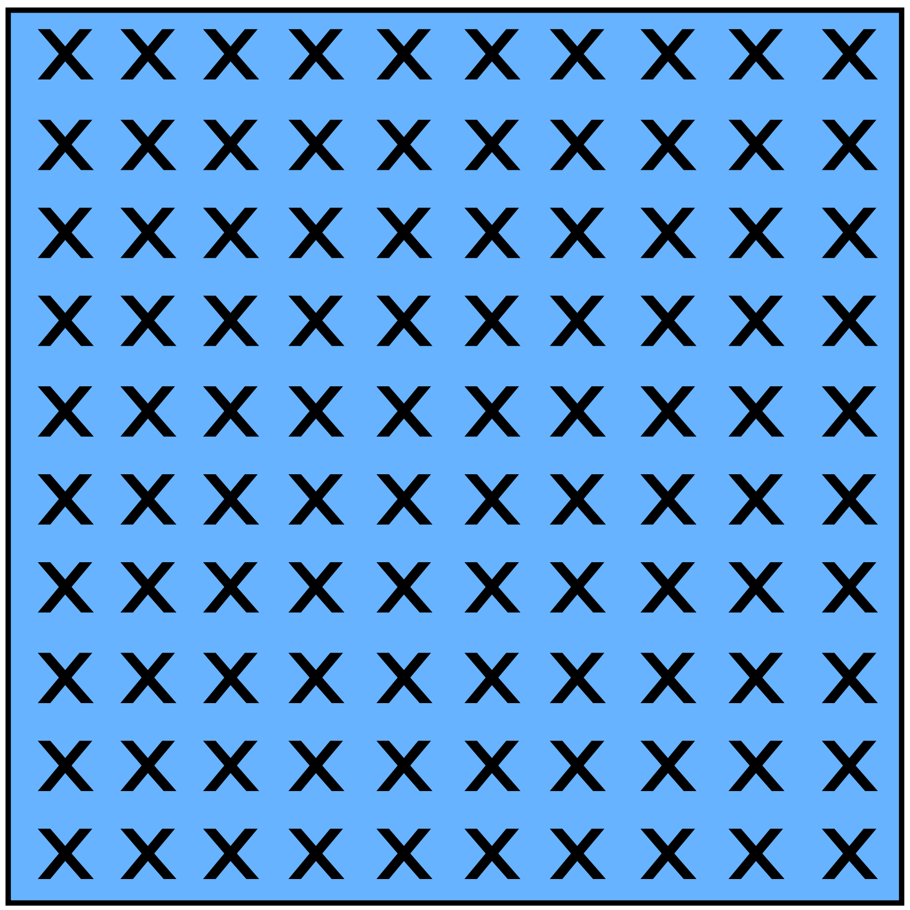
- *A* nXn, nonsymmetric
- Compute *v* and *t* to zero out the first column

Why Hessenberg Form?



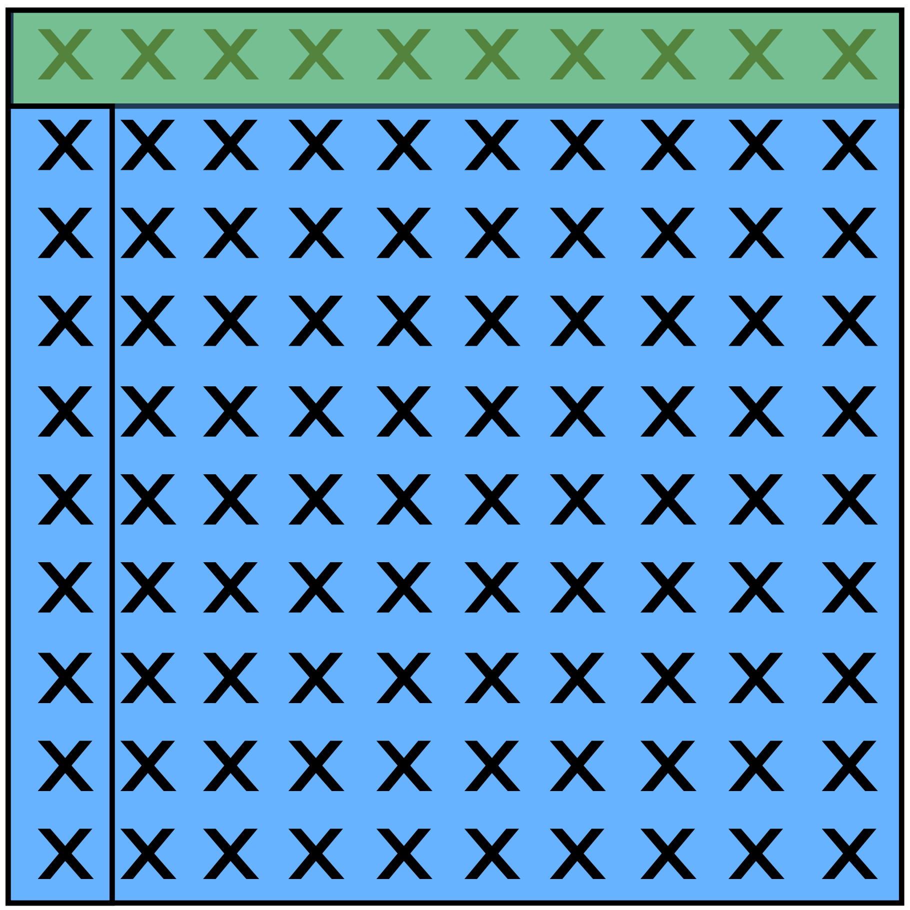
- *A* nXn, nonsymmetric
- Compute *v* and *t* to zero out the first column
- Apply on left: $(I - v_1 t_1 v_1^T)A$

Why Hessenberg Form?



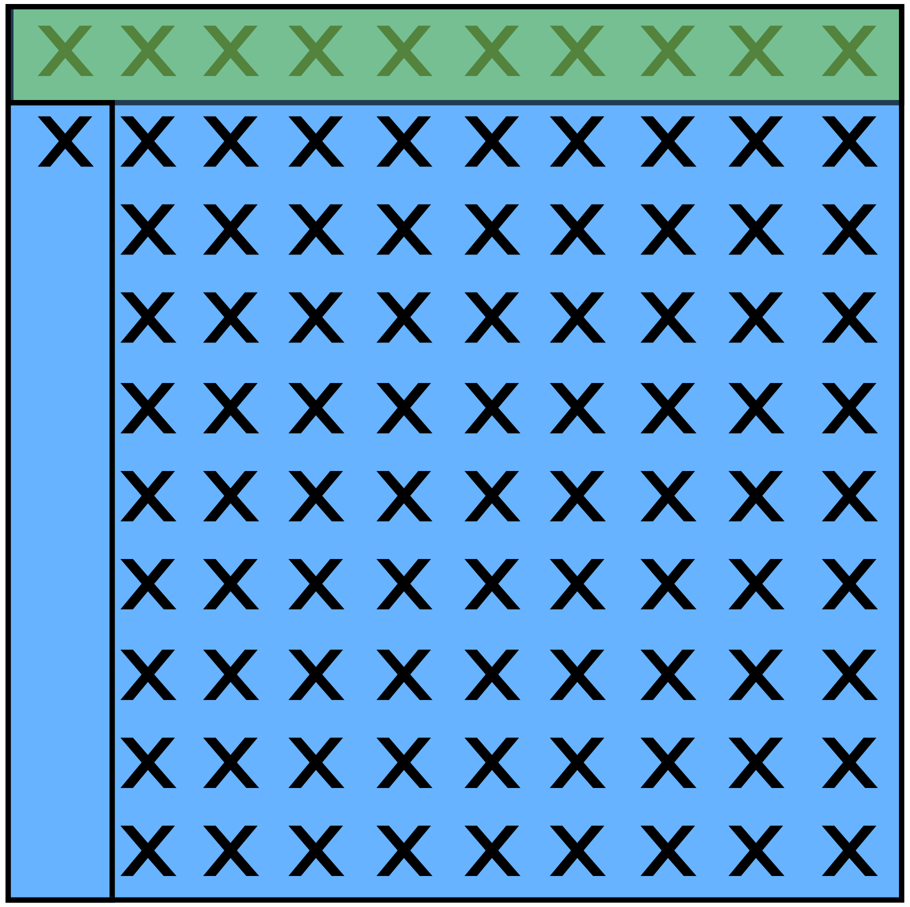
- A nXn, nonsymmetric
- Compute v and t to zero out the first column
- Apply on left:
  $(I - v_I t_I v_I^T)A$
- Apply on right:
  $(I - v_I t_I v_I^T)A(I - v_I t_I v_I^T)$
- Work destroyed...

Hessenberg Reduction
xGEHD2



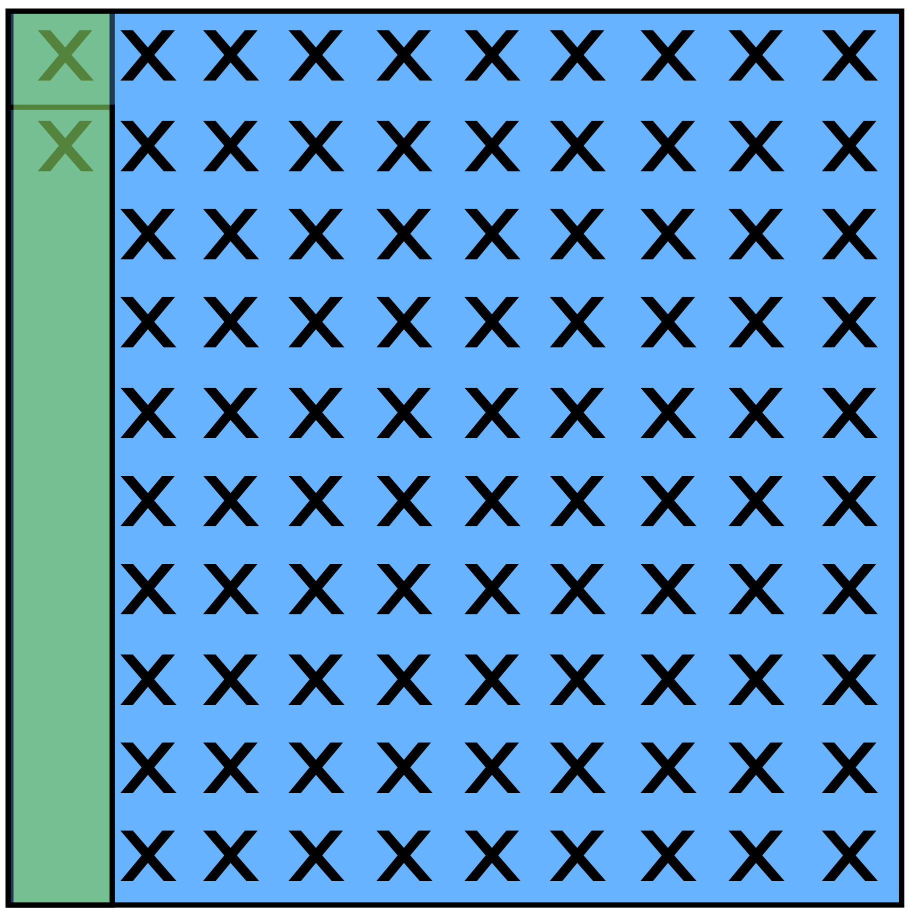
- Compute v_i and t_i

Hessenberg Reduction
xGEHD2



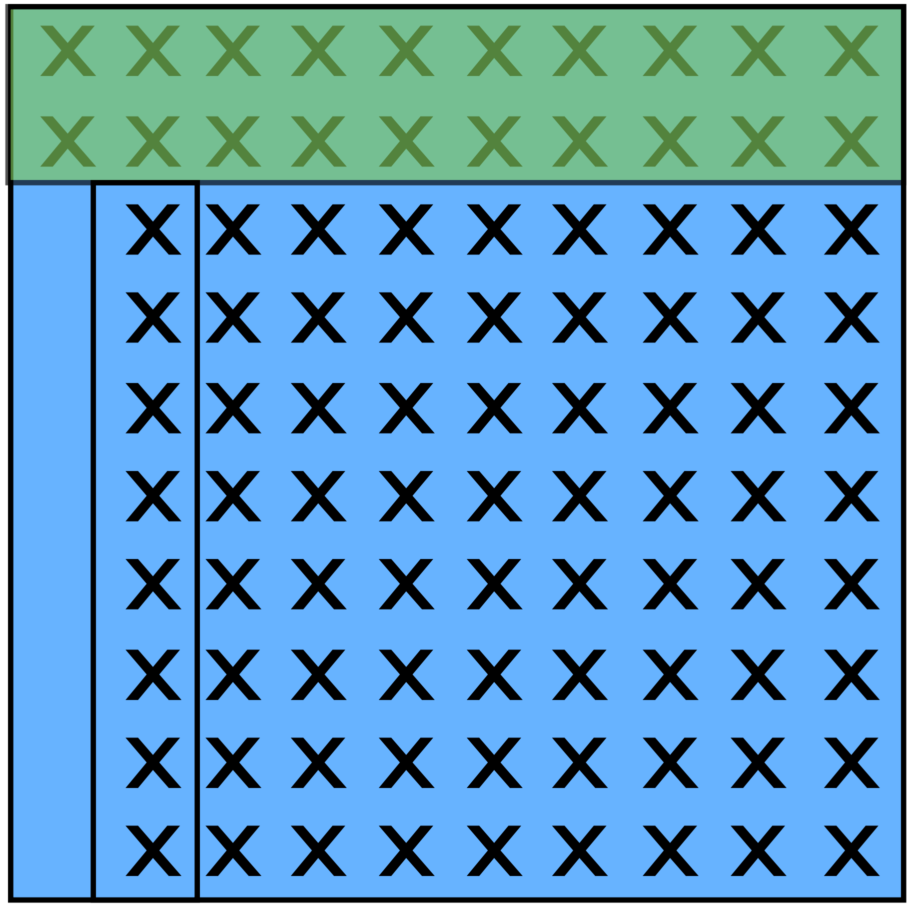
- Compute v_I and t_I
- Apply on left: $(I - v_I t_I v_I^T)A$

Hessenberg Reduction
xGEHD2



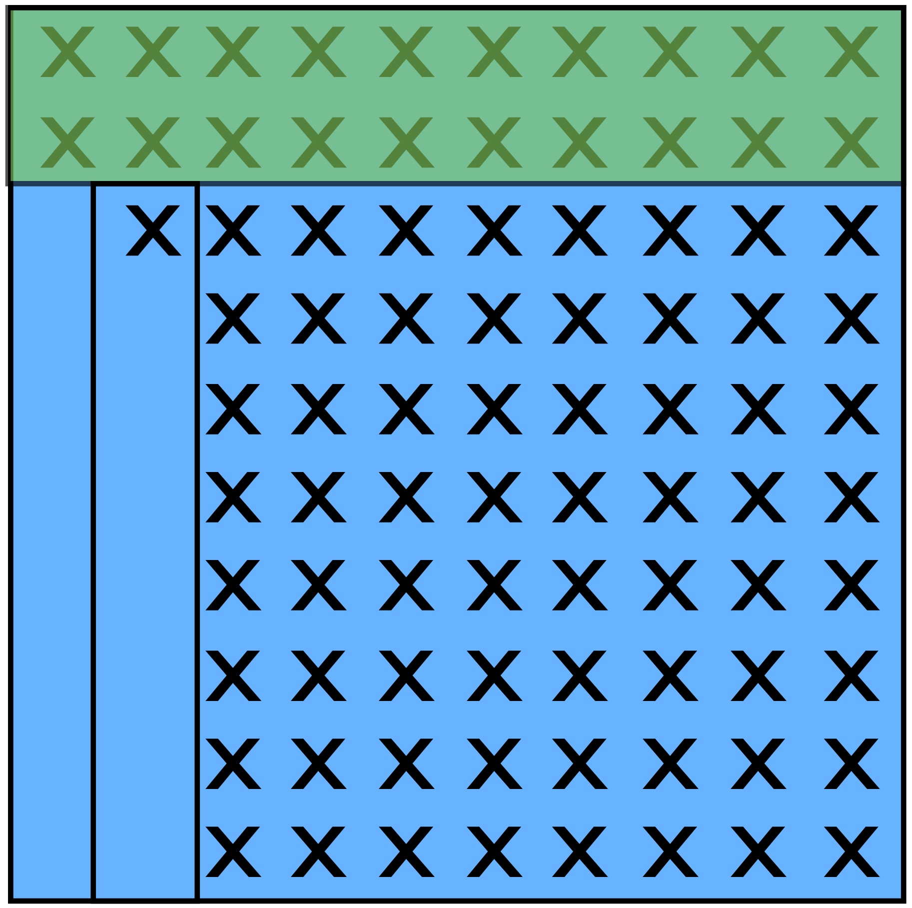
- Compute v_1 and t_1
- Apply on left: $(I - v_1 t_1 v_1^T)A$
- Apply on right: $(I - v_1 t_1 v_1^T)A(I - v_1 t_1 v_1^T)$
- Call updated matrix A_1

Hessenberg Reduction
xGEHD2



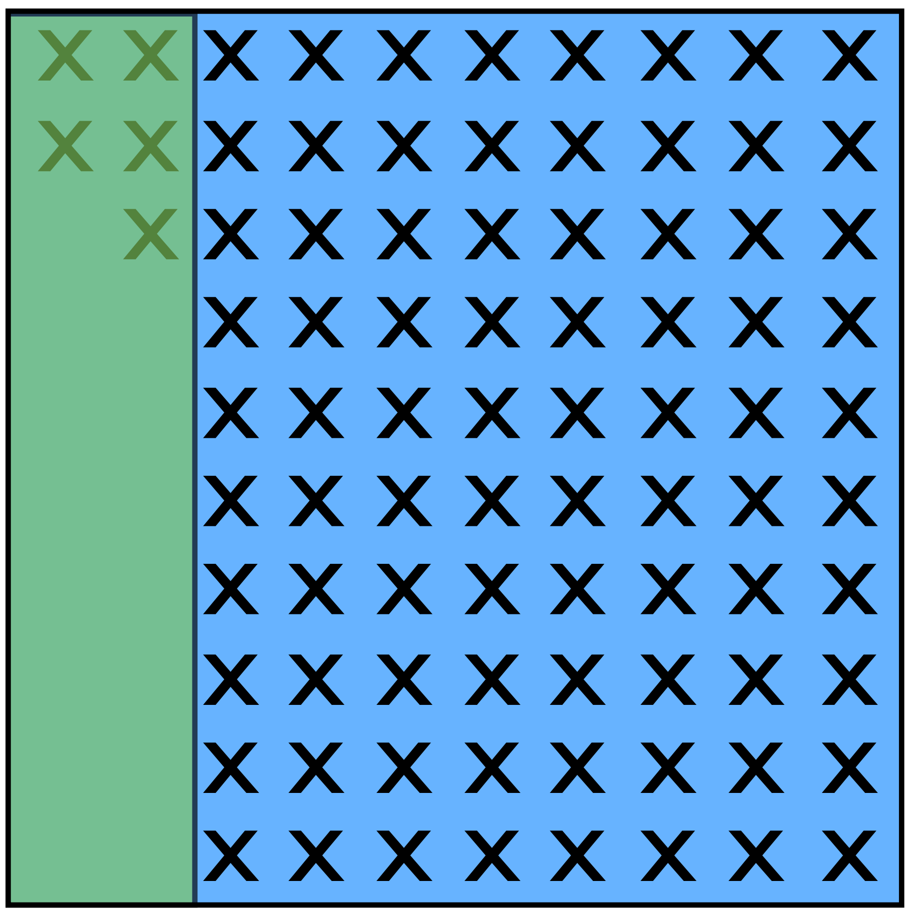
- Compute v_2 and t_2

Hessenberg Reduction
xGEHD2



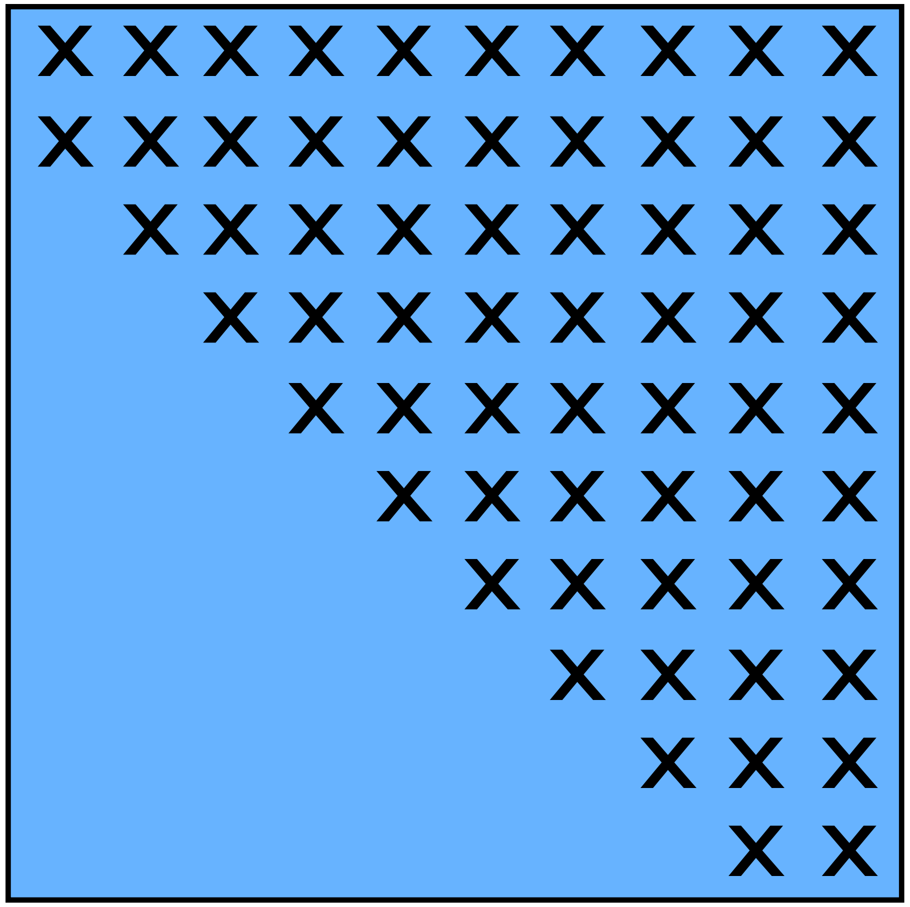
- Compute v_2 and t_2
- Apply on left: $(I - v_2 t_2 v_2^T) A_I$

Hessenberg Reduction xGEHD2



- Compute v_2 and t_2
- Apply on left: $(I - v_2 t_2 v_2^T) A_1$
- Apply on right: $(I - v_2 t_2 v_2^T) A_1 (I - v_2 t_2 v_2^T)$
- Call updated matrix A_2, etc ...

Hessenberg Reduction
xGEHD2



- *A* nXn nonsymmetric
- ~ $(10/3)n^3$
- ~ n^3 data transfers
- Level - 2 BLAS

xGEHD2

- Compute t_1, v_1 from 1st col. of A
- Compute A_1: $(I - v_1 t_1 v_1^T) A (I - v_1 t_1 v_1^T)$
- Compute t_2, v_2 from 2nd col. of A_1
- Compute A_2: $(I - v_2 t_2 v_2^T) A_1 (I - v_2 t_2 v_2^T)$
- Compute t_3, v_3 from 3rd col. of A_2
- Continue. . .

Introducing Blocking in xGEHD2

- Compute t_1, v_1 from 1st col. of A
- Compute A_1: $(I - v_1 t_1 v_1^T) A (I - v_1 t_1 v_1^T)$
- Compute t_2, v_2 from 2nd col. of A_1

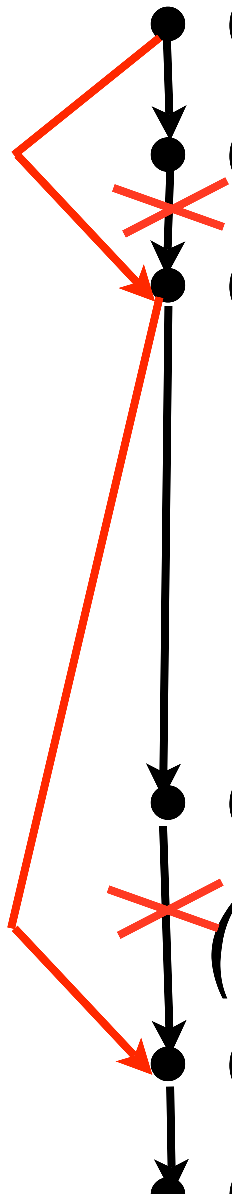
  $= (I - v_1 t_1 v_1^T) A (I - v_1 t_1 v_1^T) e_2$

  $= (I - v_1 t_1 v_1^T) A (e_2 - v_1 t_1 (v_1^T e_2))$

  (we need all of A for this)
- Compute A_2:

  $(I - v_2 t_2 v_2^T) A_1 (I - v_2 t_2 v_2^T)$
- Compute t_3, v_3 from 3rd col. of A_2
- Continue…

Introducing Blocking in xGEHD2



- Compute t_1, v_1 from 1st col. of A
- Compute A_1: $(I - v_1 t_1 v_1^T) A (I - v_1 t_1 v_1^T)$
- Compute t_2, v_2 from 2nd col. of A_1

  $= (I - v_1 t_1 v_1^T) A (I - v_1 t_1 v_1^T) e_2$

  $= (I - v_1 t_1 v_1^T) A (e_2 - v_1 t_1 (v_1^T e_2))$

  (we need all of A for this)
- Compute A_2:

  $(I - v_2 t_2 v_2^T) A_1 (I - v_2 t_2 v_2^T)$
- Compute t_3, v_3 from 3rd col. of A_2
- Continue. . .

Introducing Blocking in xGEHD2

- Compute t_1, v_1 from 1st col. of A
- Compute A_1: $(I - v_1 t_1 v_1^T) A (I - v_1 t_1 v_1^T)$
- Compute t_2, v_2 from 2nd col. of A_1

  $= (I - v_1 t_1 v_1^T) A (I - v_1 t_1 v_1^T) e_2$

  $= (I - v_1 t_1 v_1^T) A (e_2 - v_1 t_1 (v_1^T e_2))$

  (we need all of A for this)
- Compute A_2:

$(I - v_2 t_2 v_2^T) A_1 (I - v_2 t_2 v_2^T)$

- Compute t_3, v_3 from 3rd col. of A_2
- Continue…

Introducing Blocking in xGEHD2

- Compute t_1, v_1 from 1st col. of A
- Compute A_1: $(I - v_1 t_1 v_1^T) A (I - v_1 t_1 v_1^T)$
- Compute t_2, v_2 from 2nd col. of A_1
  $= (I - v_1 t_1 v_1^T) A (I - v_1 t_1 v_1^T) e_2$
  $= (I - v_1 t_1 v_1^T) A (e_2 - v_1 t_1 (v_1^T e_2))$
  (we need all of A for this)
- Compute A_2:
  $(I - v_2 t_2 v_2^T) A_1 (I - v_2 t_2 v_2^T)$
- Compute t_3, v_3 from 3rd col. of A_2
- Continue. . .

Blocking and xGEHRD

Panel

- Compute t_1, v_1 from 1st col. of A
- Compute t_2, v_2 from $A_1 e_2$
- Compute t_k, v_k from $A_{k-1} e_k$

(~ n^2k FLOPs and data transfers)

Update

- Combine into single update

  $A_k = (I - VTV^T)A(I - VTV^T)^T$

(~ $4n^2k$ FLOPS and n^2 data)

Blocking and xGEHRD

Panel

- Compute t_1, v_1 from 1st col. of A
- Compute t_2, v_2 from $A_1 e_2$
- Compute t_k, v_k from $A_{k-1} e_k$

(~ n^2k FLOPs and data transfers)

Update

- Combine into single update
  $A_k = (I - VTV^T)A(I - VTV^T)^T$

(~ $4n^2k$ FLOPS and n^2 data)

Panel

- Compute t_{k+1}, v_{k+1} from 1st col. of A_k
- Compute t_2, v_2 from $A_1 e_2$
- Compute t_{2k}, v_{2k} from $A_{2k-1} e_k$

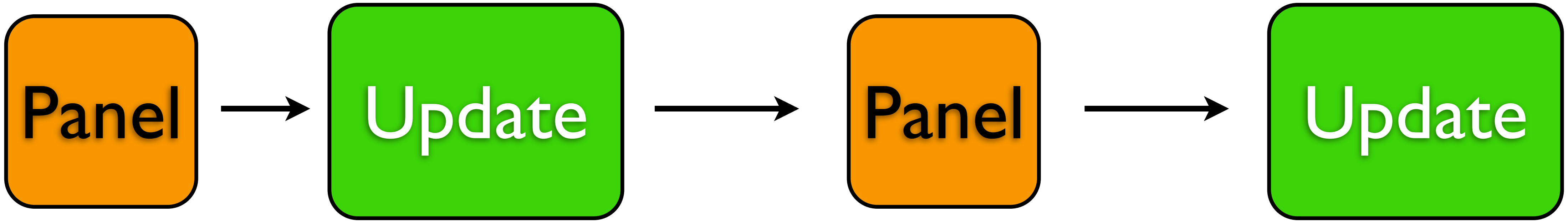
(~ n^2k FLOPs and data transfers)

Update

- Combine into single update
  $A_{2k} = (I - VTV^T)A_k(I - VTV^T)^T$

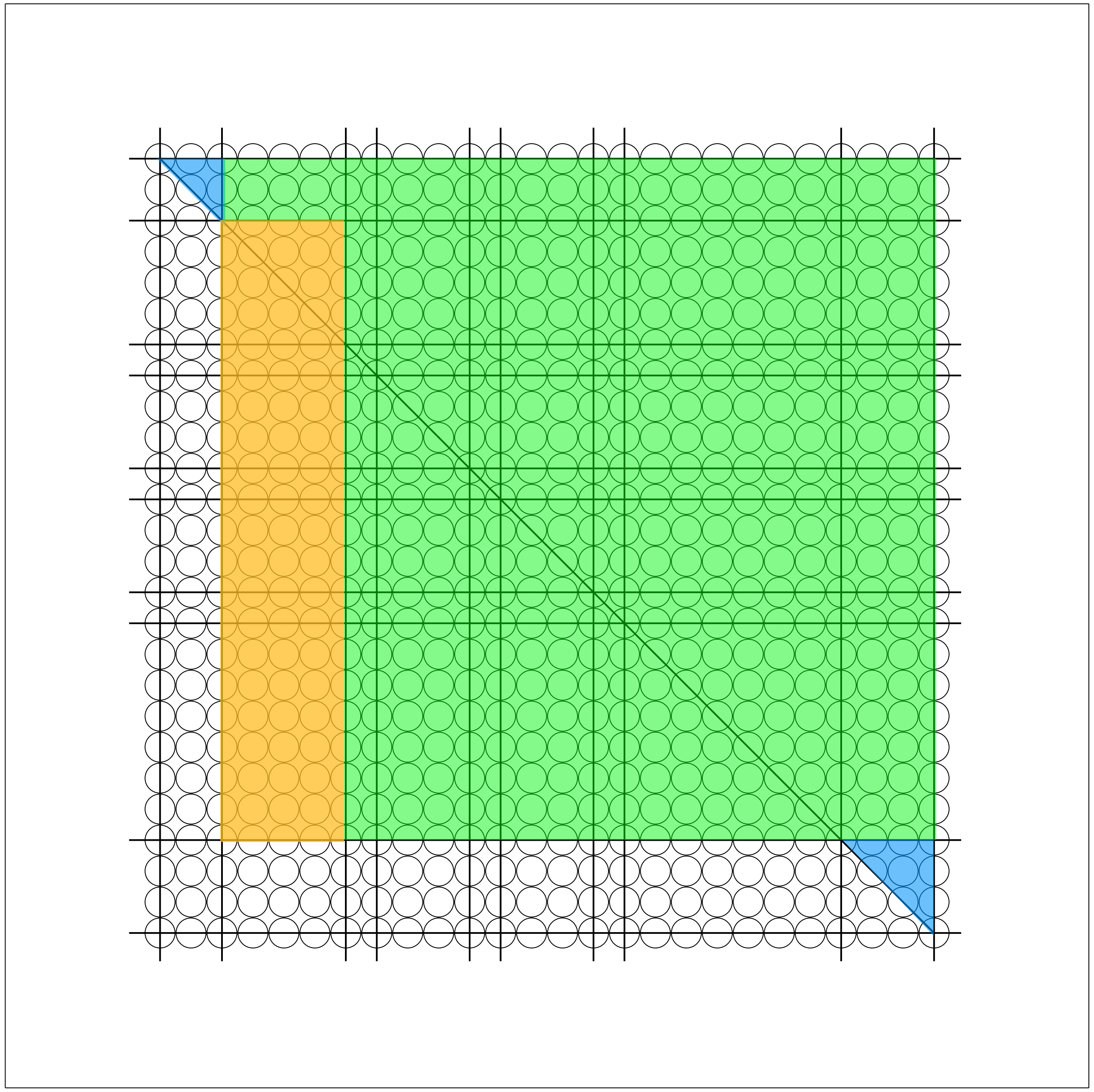
(~ $4n^2k$ FLOPS and n^2 data)

Blocking



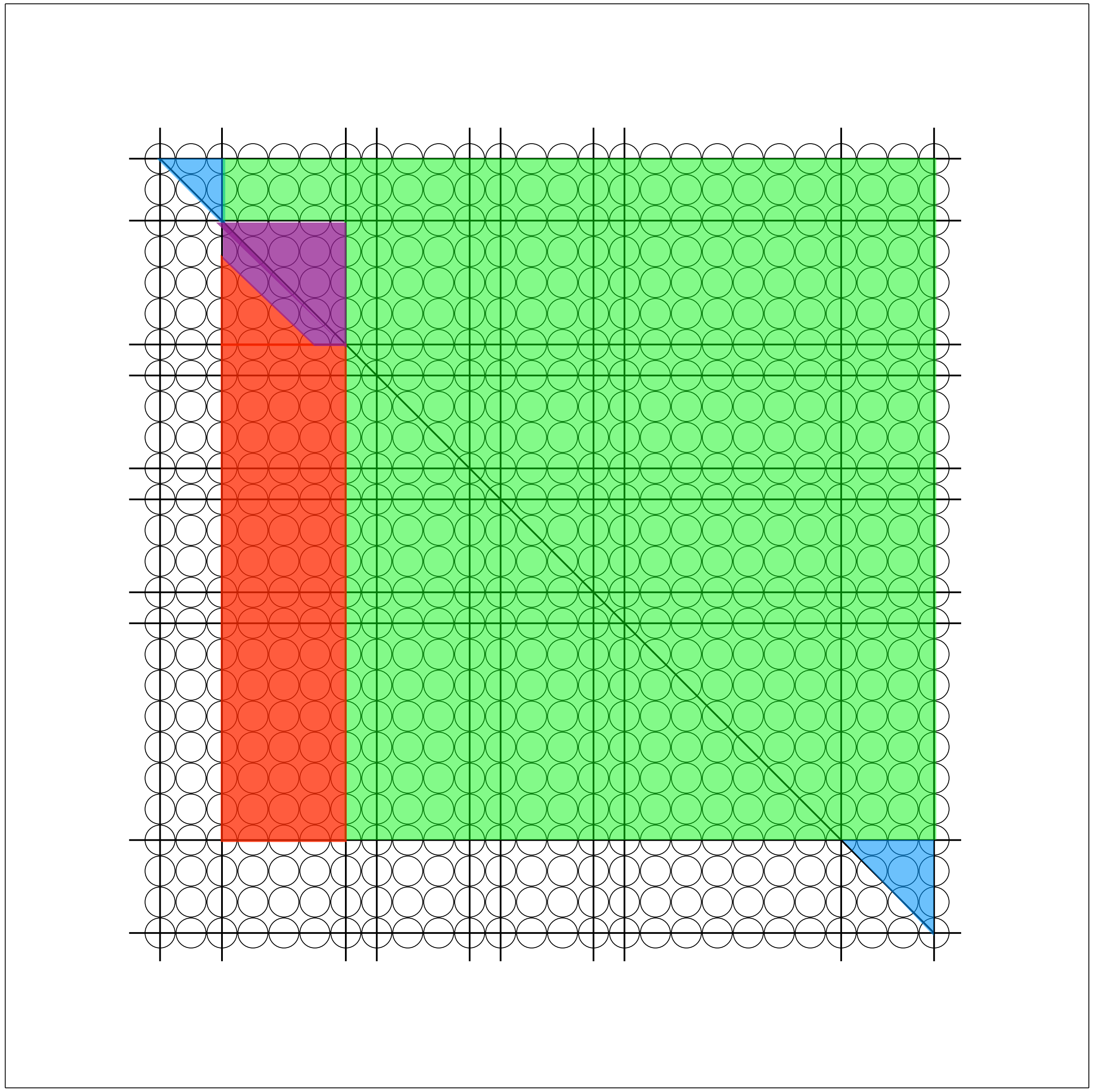
Look ahead

- Can we overlap panel factorization and update???



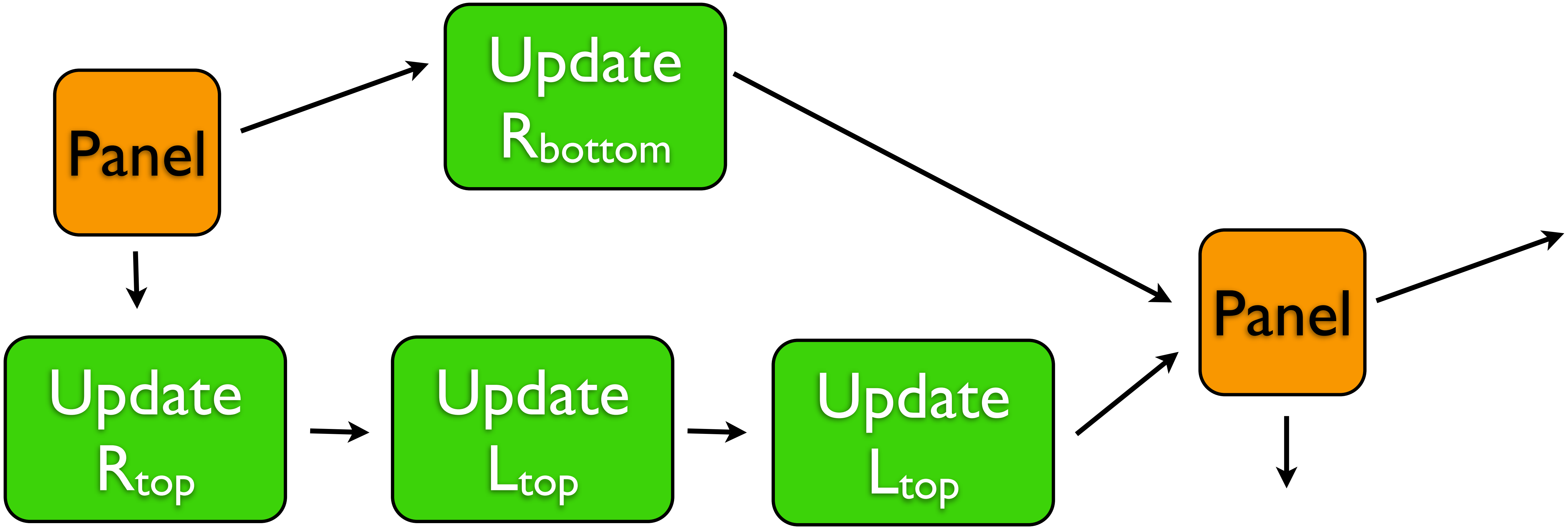
Look ahead

- Can we overlap panel factorization and update???



Look ahead

- Early Attempt



Blocking and xGEHRD

Panel

- Compute t_1, v_1 from 1st col. of A
- Compute t_2, v_2 from $A_1 e_2$
- Compute t_k, v_k from $A_{k-1} e_k$

(~ n^2k FLOPs and data transfers)

Update

- Combine into single update

$$A_k = (I - VTV^T)A(I - VTV^T)^T$$

(~ $4n^2k$ FLOPS and n^2 data)

Panel

- Compute t_{k+1}, v_{k+1} from 1st col. of A_k
- Compute t_2, v_2 from $A_1 e_2$
- Compute t_{2k}, v_{2k} from $A_{2k-1} e_k$

(~ n^2k FLOPs and data transfers)

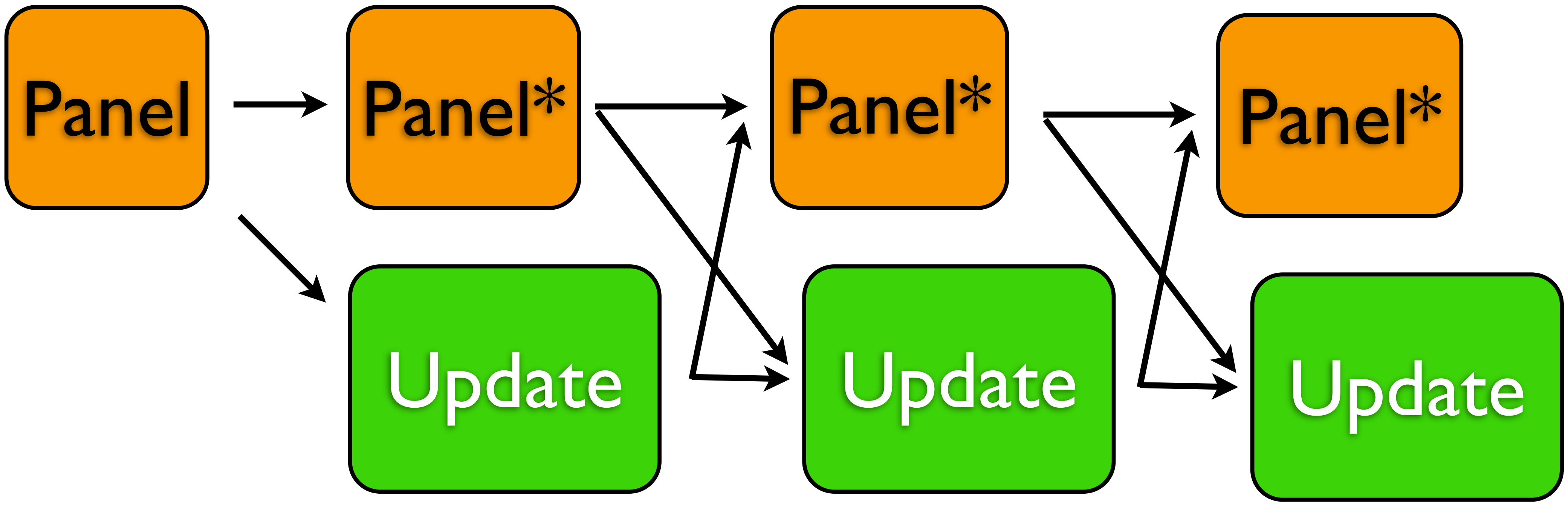
Update

- Combine into single update

$$A_{2k} = (I - VTV^T)A_k(I - VTV^T)^T$$

(~ $4n^2k$ FLOPS and n^2 data)

Look ahead



Conclusions / Further Work

- Need to study and model memory access/ transfers, cost of copy, cost of computation
- Explore potential in hybrid framework with specialized hardware for matrix vector product
- Opportunity for changing the data layout of matrix A during the copy