Dominant feature extraction

Francqui Lecture 7-5-2010

Paul Van Dooren
Université catholique de Louvain
CESAME, Louvain-la-Neuve, Belgium

Goal of this lecture

Develop basic ideas for large scale dense matrices

Recursive procedures for

- Dominant singular subspace
- Multipass iteration
- Subset selection
- Dominant eigenspace of positive definite matrix
- Possible extensions

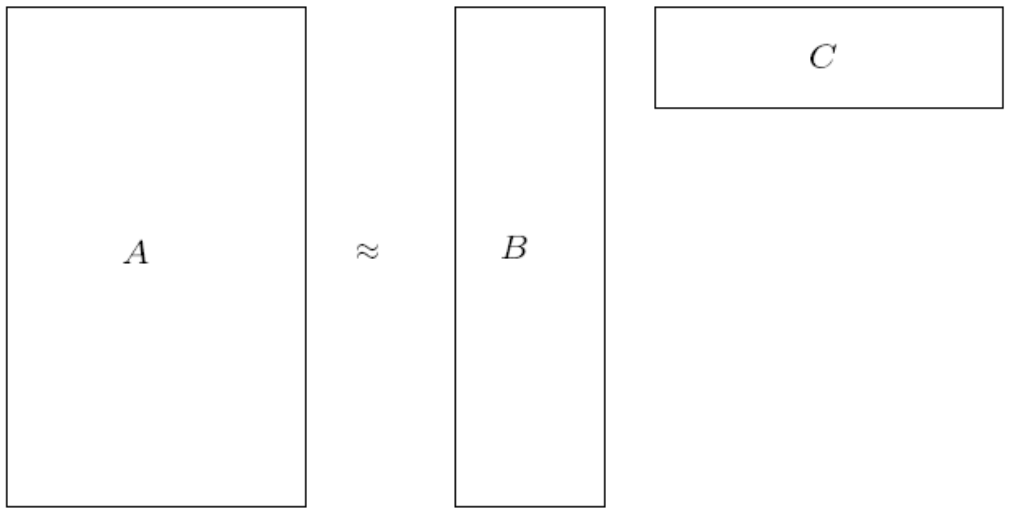
which are all based on solving cheap subproblems

Show accuracy and complexity results

Dominant singular subspaces

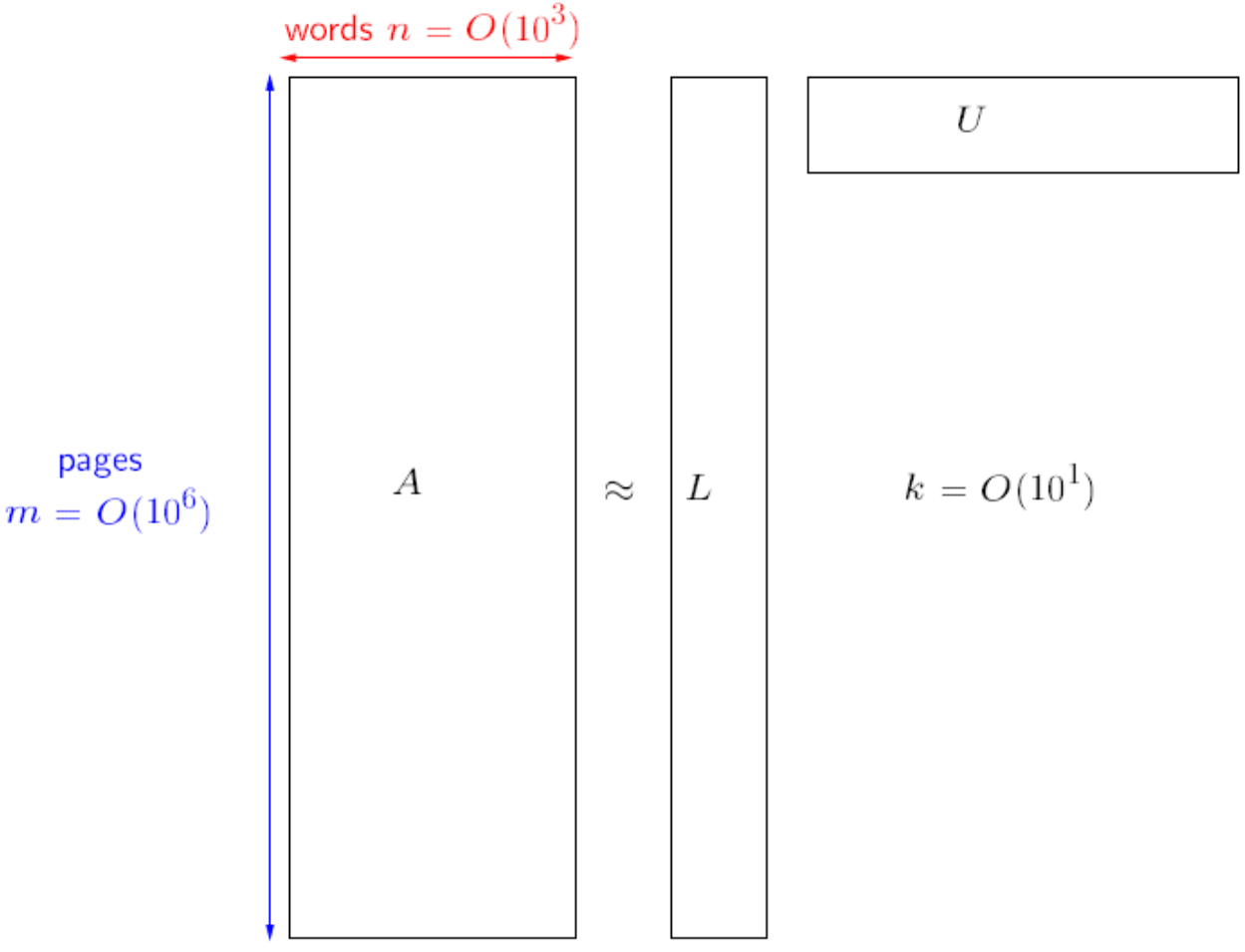
Given $A_{m\times n}$, approximate it by a rank k factorization $B_{m\times k}C_{k\times n}$ by solving

$$\min \|A - BC\|_2, \quad k \ll m, n$$



This has several applications in Image compression, Information retrieval and Model reduction (POD)

Information retrieval

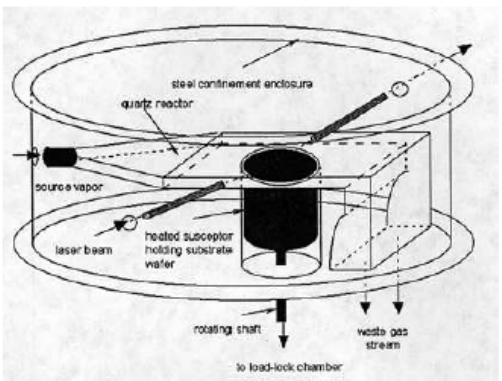


- Low memory requirement $0(k(m+n))$
- Fast queries $Ax \approx L(Ux)$ $0(k(m+n))$ time
- Easy to obtain $0(kmn)$ flops

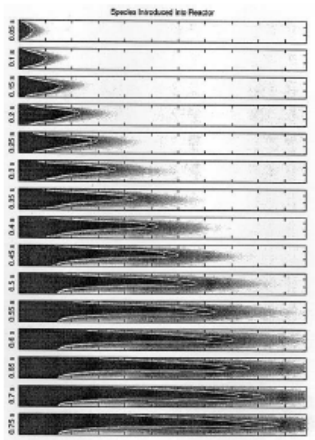
Proper Orthogonal decomposition (POD)

Compute a state trajectory for one “typical" input

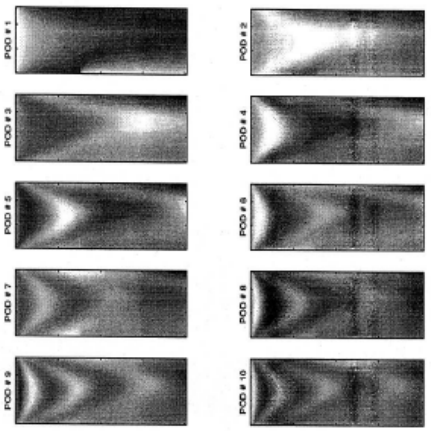
Collect the principal directions to project on



Quartz reactor



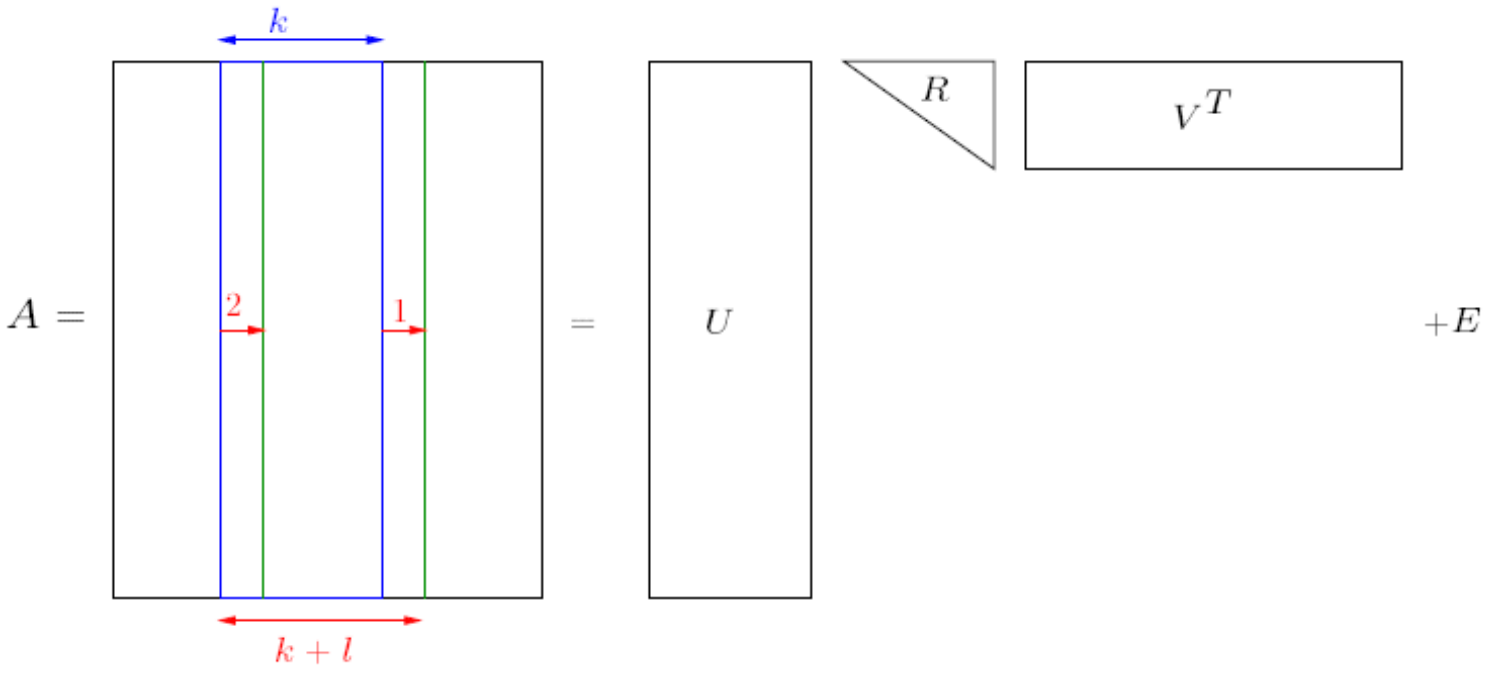
Snap shots of "typical" states



Ten dominant "states"

Recursivity

We pass once over the data with a window of length k and perform along the way a set of windowed SVD's of dimension $m \times (k + \ell)$



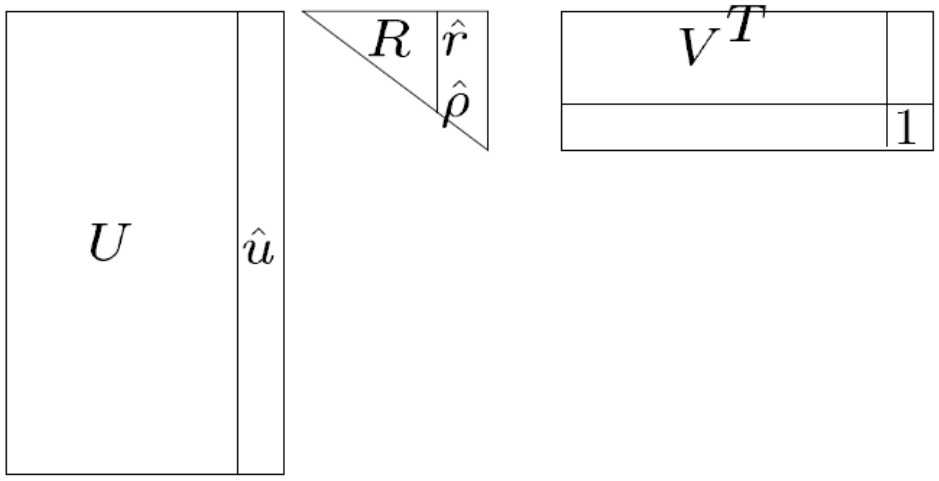
Step 1 : expand by appending ℓ columns (Gram Schmidt)
Step 2 : contract by deleting the ℓ least important columns (SVD)

Expansion (G-S)

Append column a_+ to the current approximation URV^T to get

$$\begin{bmatrix} URV^T & a_+ \end{bmatrix} = \begin{bmatrix} U & a_+ \end{bmatrix} \begin{bmatrix} R & 0 \\ & 1 \end{bmatrix} \begin{bmatrix} V^T & \\ & 1 \end{bmatrix}$$

Update with Gram Schmidt to recover a new decomposition $\hat{U}\hat{R}\hat{V}^T$:

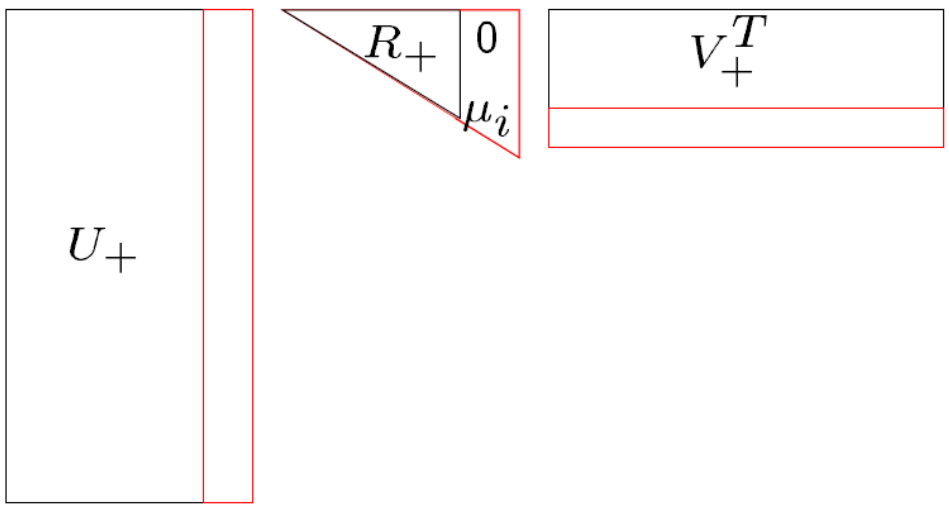


using $\hat{r} = U^T a_+$, $\hat{a} = a_+ - U\hat{r}$, $\hat{a} = \hat{u}\hat{\rho}$ (since $a_+ = U\hat{r} + \hat{u}\hat{\rho}$)

Contraction (SVD)

Now remove the ℓ smallest singular values of this new $\hat{U}\hat{R}\hat{V}^T$ via

$$\hat{U}\hat{R}\hat{V}^T = (\hat{U}G_u)(G_u^T\hat{R}G_v)(G_v^T\hat{V}^T) =$$



and keeping $U_+R_+V_+^T$ as best approximation of $\hat{U}\hat{R}\hat{V}^T$
(just delete the ℓ smallest singular values)

Complexity of one pair of steps

The Gram Schmidt update (expansion) requires $4mk$ flops per column (essentially for the products $\hat{r} = U^T a_+,\ \hat{a} = a_+ - U\hat{r}$)

For $G_u \hat{R} G_v = \begin{bmatrix} R_+ & 0 \\ & \mu_i \end{bmatrix}$ one requires the left and right singular vectors of $\hat{R}$ which can be obtained in $O(k^2)$ flops per singular value (using inverse iteration)

Multiplying $\hat{U}G_u$ and $\hat{V}G_v$ requires $4mk$ flops per deflated column

The overall procedure requires $8mk$ flops per processed column and hence $8mnk$ flops for a rank k approximation to a $m \times n$ matrix A

One shows that $A = U \begin{bmatrix} R & A_{12} \\ 0 & A_{22} \end{bmatrix} V^T$ where $\| \begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix} \|_F^2$ is known

Error estimates

Let $E := A - \hat{A} = U\Sigma V^T - \hat{U}\hat{\Sigma}\hat{V}^T$ and $\mu := \|E\|_2$

Let $\hat{\mu} := \max \mu_i$ where μ_i is the neglected singular value at step i

One shows that the error norm

$$\hat{\mu} \leq \sigma_{k+1} \leq \mu \leq \sqrt{n-k}\hat{\mu} \approx c\hat{\mu}$$

$$\hat{\sigma}_i \leq \sigma_i \preceq \hat{\sigma}_i + \hat{\mu}^2/2\hat{\sigma}_i$$

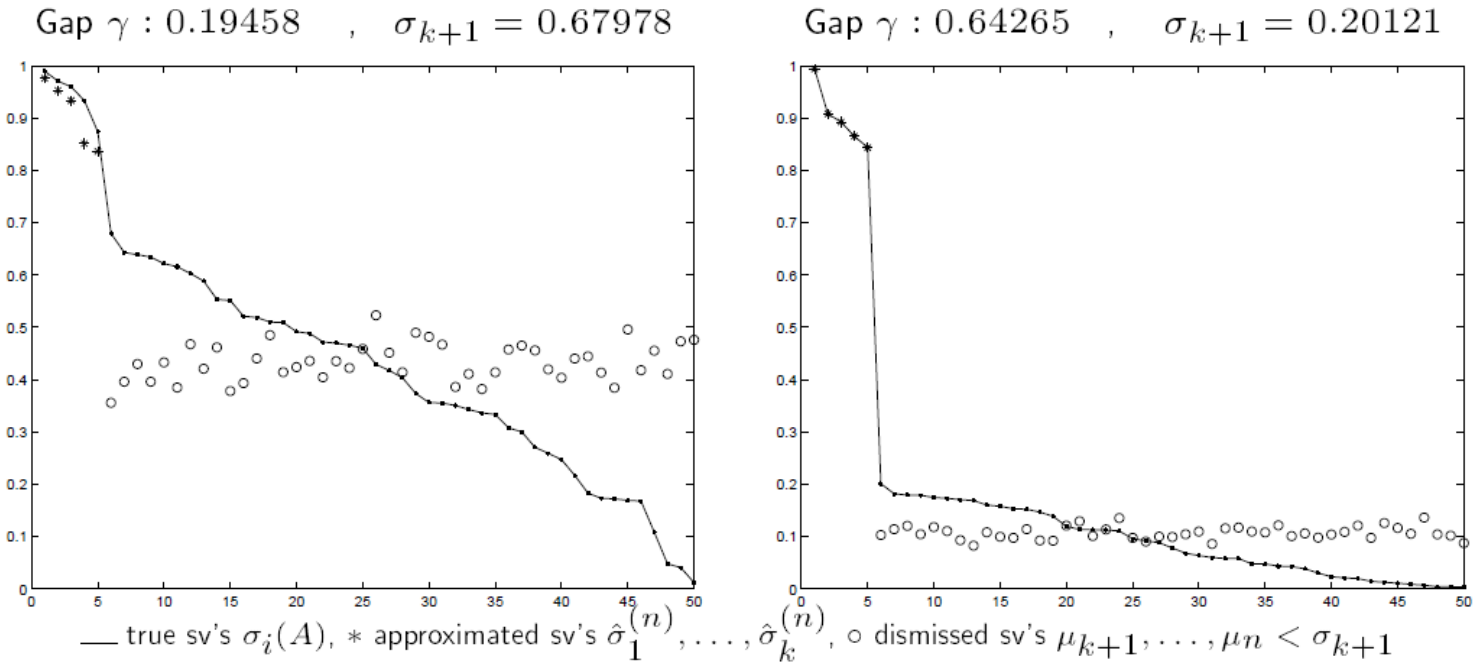
$$\tan\theta_k \preceq \tan\hat{\theta}_k := \hat{\mu}^2/(\hat{\sigma}_k^2 - \hat{\mu}^2), \quad \tan\phi_k \preceq \tan\hat{\phi}_k := \hat{\mu}\hat{\sigma}_1/(\hat{\sigma}_k^2 - \hat{\mu}^2)$$

where θ_k, ϕ_k are the canonical angles of dimension k :

$$\cos\theta_k := \|U^T(:,k)\hat{U}\|_2, \quad \cos\phi_k := \|V^T(:,k)\hat{V}\|_2$$

Examples

The bounds get much better when the gap $\sigma_k - \sigma_{k+1}$ is large

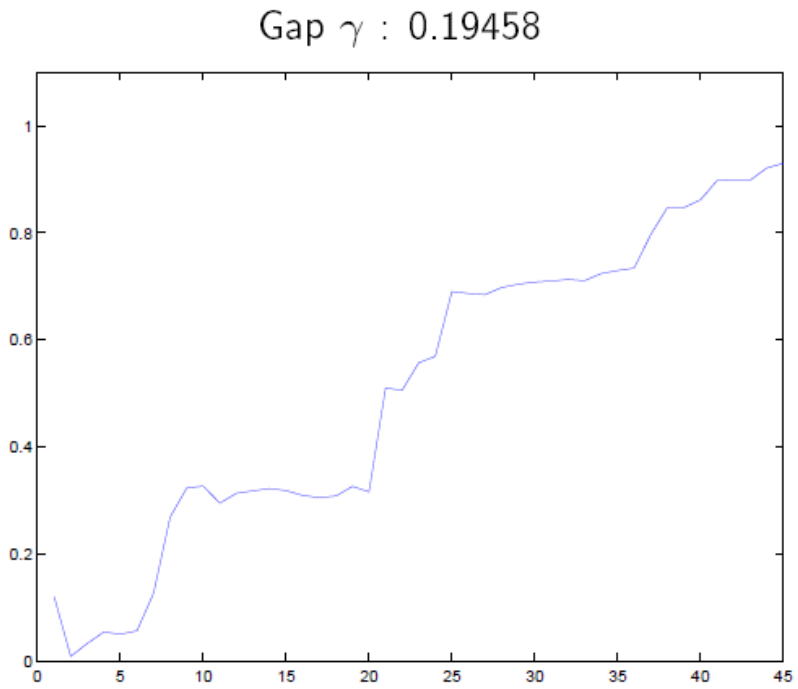


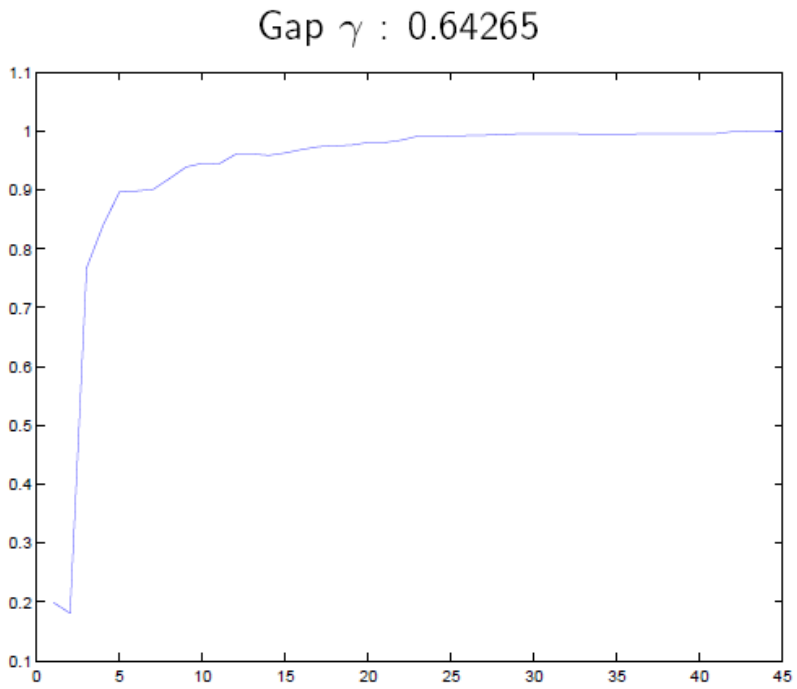
$\sigma_1 = 0.99008$	$\hat{\sigma}_1 = 0.97613$
$\sigma_2 = 0.97084$	$\hat{\sigma}_2 = 0.95301$
$\sigma_3 = 0.96010$	$\hat{\sigma}_3 = 0.93379$
$\sigma_4 = 0.93338$	$\hat{\sigma}_4 = 0.85142$
$\sigma_5 = 0.87437$	$\hat{\sigma}_5 = 0.83675$
$\mu = 0.73768$	$\hat{\mu} = 0.52330$
$\cos\theta_k = 0.93000$	$\cos\hat{\theta}_k = 0.82233$
$\cos\phi_k = 0.83881$	$\cos\hat{\phi}_k = 0.71038$

$\sigma_1 = 0.99430$	$\hat{\sigma}_1 = 0.99418$
$\sigma_2 = 0.90840$	$\hat{\sigma}_2 = 0.90815$
$\sigma_3 = 0.89284$	$\hat{\sigma}_3 = 0.89250$
$\sigma_4 = 0.86560$	$\hat{\sigma}_4 = 0.86551$
$\sigma_5 = 0.84387$	$\hat{\sigma}_5 = 0.84357$
$\mu = 0.20140$	$\hat{\mu} = 0.13631$
$\cos\theta_k = 0.99998$	$\cos\hat{\theta}_k = 0.99459$
$\cos\phi_k = 0.99935$	$\cos\hat{\phi}_k = 0.94334$

Convergence

How quickly do we track the subpaces ?





How $\cos\theta_k^{(i)}$ evolves with the time step *i*

Example

Find the dominant behavior in an image sequence

Images can have up to 10^6 pixels

Each column of A is one image

Original : $m = 28341,\ n = 100$

Approximation : $k = 6$

Multipass iteration

Low Rank Incremental SVD can be applied in several passes, say to

$$\frac{1}{\sqrt{k}} \begin{bmatrix} A & A & \dots & A \end{bmatrix}$$

After the first block (or "pass") a good approximation of the dominant space $\hat{U}$ has already been constructed

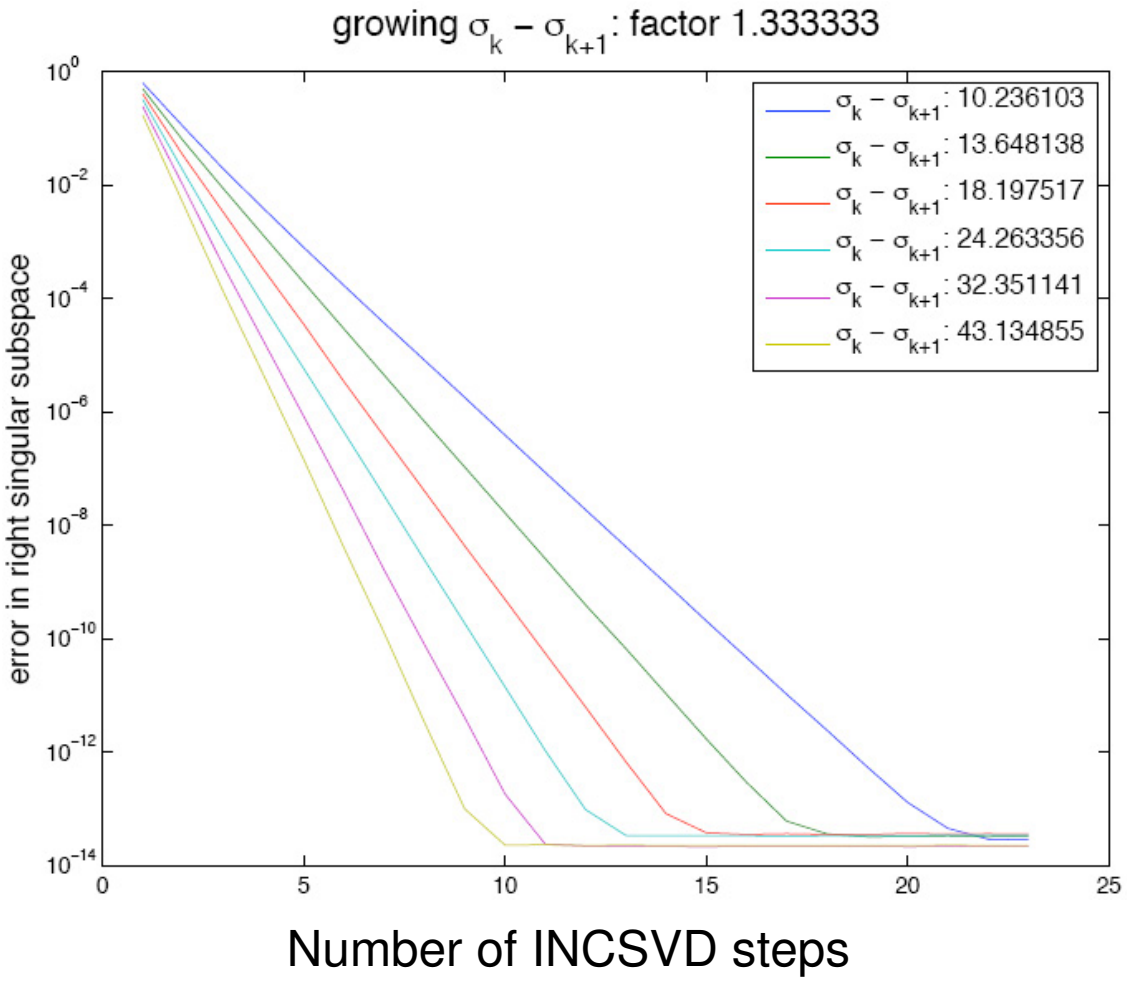
Going over to the next block (second "pass") will improve it, etc.

Theorem Convergence of the multipass method is linear, with approximate ratio of convergence $\psi/(1-\kappa^2) < 1$, where

- ψ measures orthogonality of the residual columns of A
- κ is the ratio σ_k/σ_{k+1} of A

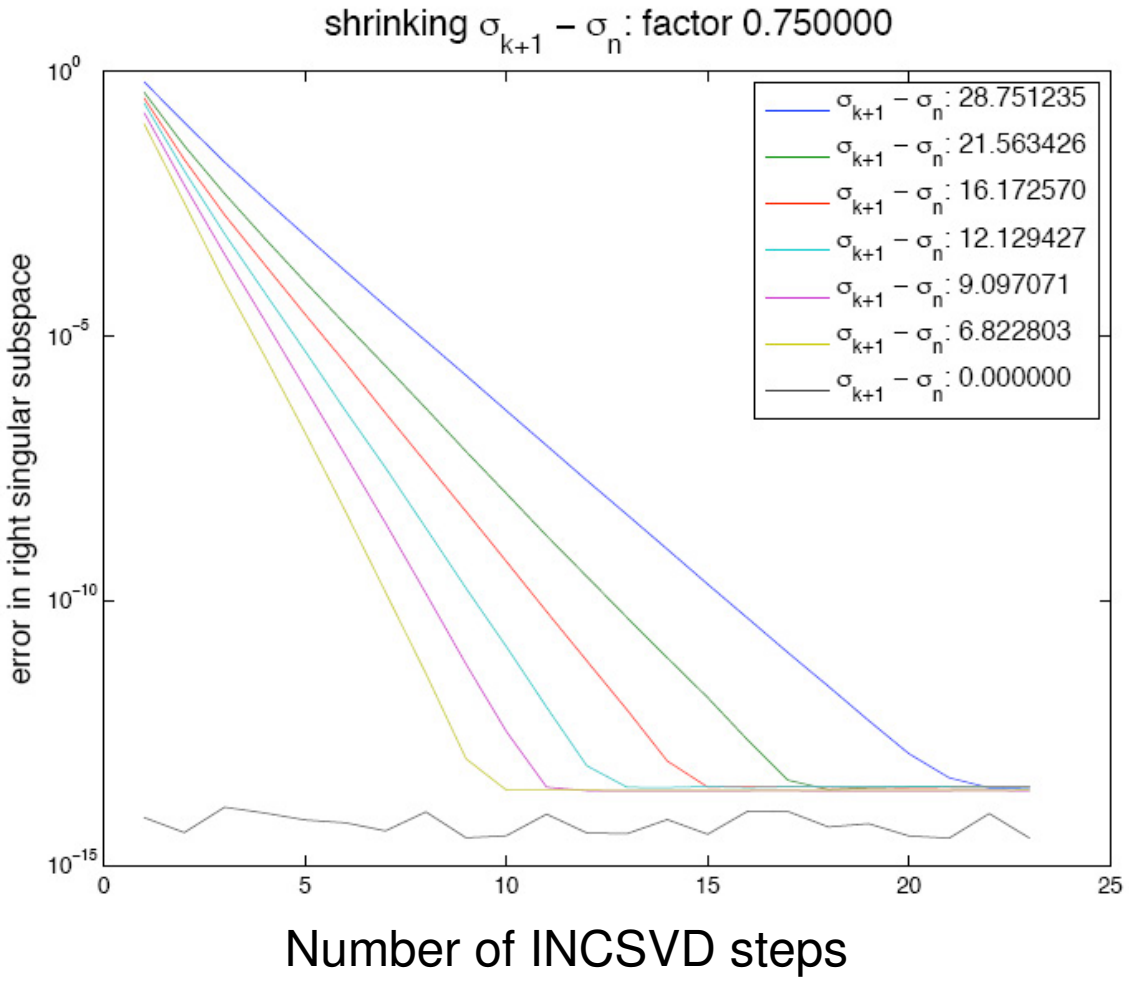
Convergence behavior

for increasing gap between “signal" and “noise"



Convergence behavior

for increasing orthogonality between “residual vectors"



Eigenfaces analysis

Ten dominant left singular vectors of ORL Database of faces
(40 images, 10 subjects, 92×112 pixels = 10304×400 matrix)

Using MATLAB' SVD function

Using one pass of incremental SVD

Maximal angle : 16.3°, maximum relative error in sing. values : 4.8%

Conclusions Incremental SVD

A useful and economical SVD approximation of $A_{m,n}$

For matrices with columns that are very large or “arrive" with time

Complexity is proportional to mnk and the number of “passes"

Algorithms due to
[1] Manjunath-Chandrasekaran-Wang (95)
[2] Levy-Lindenbaum (00)
[3] Chahlaoui-Gallivan-VanDooren (01)
[4] Brand (03)
[5] Baker-Gallivan-VanDooren (09)

Convergence analysis and accuracy in refs [3],[4],[5]

Subset selection

We want a “good approximation" of A_{mn} by a product $B_{mk}P^T$ where P_{nk} is a “selection matrix" i.e. a submatrix of the identity I_n

This seems connected to

$$\min \|A - BP^T\|_2$$

and maybe similar techniques can be used as for incremental SVD

Clearly, if $B = AP$, we just select a subset of the columns of A

Rather than minimizing $\|A - BP^T\|_2$ we maximize $\mathrm{vol}(B)$ where

$$\mathrm{vol}(B) = \det(B^T B)^{\frac{1}{2}} = \prod_{i=1}^{k} \sigma_i(B), \qquad m \geq k$$

There are $\begin{pmatrix} n \\ k \end{pmatrix}$ possible choices and the problem is NP hard

and there is no polynomial time approximation algorithm

Heuristics

Gu-Eisenstat show that the Strong Rank Revealing QR factorization (SRRQR) solves the following simpler problem

B is sub-optimal if there is no swapping of a single column of A (yielding $\hat{B}$) that has a larger volume (constrained minimum)

Here, we propose a simpler “recursive updating" algorithm that has complexity $O(mnk)$ rather than $O(mn^2)$ for Gu-Eisenstat

The idea is again based on a sliding window of size $k+1$ (or $k+\ell$)

Sweep through columns of A while maintaining a “best" subset B

- Append a column of A to B, yielding B_+
- Contract B_+ to $\hat{B}$ by deleting the “weakest" column of B_+

Deleting the weakest column

Let $B = A(:, 1:k)$ to start with and let $B = QR$ where R is $k \times k$

Append the next column a_+ of A to form B_+ and update its decomposition using Gram Schmidt

$$B_+ := \begin{bmatrix} QR & a_+ \end{bmatrix} = \begin{bmatrix} Q & a_+ \end{bmatrix} \begin{bmatrix} R & 0 \\ & 1 \end{bmatrix} = \begin{bmatrix} Q & \hat{q} \end{bmatrix} \begin{bmatrix} R & \hat{r} \\ & \hat{\rho} \end{bmatrix} = Q_+ R_+$$

with $\hat{r} = Q^T a_+,\ \hat{a} = a_+ - Q\hat{r},\ \hat{a} = \hat{q}\hat{\rho}$ (since $a_+ = Q\hat{r} + \hat{q}\hat{\rho}$)

Contract B_+ to $\hat{B}$ by deleting the “weakest" column of R_+

This can be done in $O(mk^2)$ using Gu-Eisenstat’s SRRQR method but an even simpler heuristic uses only $O((m+k)k)$ flops

Golub-Klema-Stewart heuristic

Let $R_+ v = \sigma_{k+1} u$ be the singular vector pair corresponding to the smallest singular value σ_{k+1} of R_+ and let v_i be the components of v

Let R_i be the submatrix obtained by deleting column i from R_+ then

$$\frac{\sigma_{k+1}^2}{\sigma_1^2} + \left(1 - \frac{\sigma_{k+1}^2}{\sigma_1^2}\right) |v_i|^2 \leq \frac{\mathrm{vol}^2(R_i)}{\prod_{j=1}^{k} \sigma_j^2} \leq \frac{\sigma_{k+1}^2}{\sigma_k^2} + \left(1 - \frac{\sigma_{k+1}^2}{\sigma_k^2}\right) |v_i|^2$$

Maximizing $|v_i|$ maximizes thus a lower bound on $\mathrm{vol}^2(R_i)$
In practice this is almost always optimal and guaranteed to be so if

$$\frac{\sigma_{k+1}^2}{\sigma_k^2} + \left(1 - \frac{\sigma_{k+1}^2}{\sigma_k^2}\right) |v_i|^2 \leq \frac{\sigma_{k+1}^2}{\sigma_1^2} + \left(1 - \frac{\sigma_{k+1}^2}{\sigma_1^2}\right) |v_j|^2 \qquad \forall j \neq i$$

GKS method

Start with $B = A(:, 1:k) = QR$ where R is $k \times k$

For $j = k+1 : n$

- append column $a_+ := A(:, j)$ to get B_+
- update its QR decomposition to $B_+ = Q_+ R_+$
- contract B_+ to yield a new $\hat{B}$ using the GKS heuristic
- update its QR decomposition to $\hat{B} = \hat{Q}\hat{R}$

One can verify the optimality by performing a second pass

Notice that GKS is optimal when $\sigma_{k+1} = 0$ since then

$$\mathrm{vol}(R_i) = |v_i| \prod_{j=1}^{k} \sigma_j$$

Dominant eigenspace of PSD matrix

Typical applications

- Kernel Matrices (Machine Learning)
- Spectral Methods (Image Analysis)
- Correlation Matrices (Statistics and Signal Processing)
- Principal Component Analysis
- Karhunen-Loeve
- $\vdots$

We use K_N to denote the full $N \times N$ positive definite matrix

Sweeping through K

- Suppose a rank m approximation of the dominant eigenspace of $K_n \in \mathbb{R}^{n\times n},\ n \gg m$, is known,

$$K_n \approx A_n := U_n M_m U_n^T,$$

$M_m \in \mathbb{R}^{m\times m}$ an spd matrix and $U_n \in \mathbb{R}^{n\times m}$ with $U_n^T U_n = I_m$

Sweeping through K

- Suppose a rank m approximation of the dominant eigenspace of $K_n \in \mathbb{R}^{n\times n},\ n \gg m$, is known,

$$K_n \approx A_n := U_n M_m U_n^T,$$

  $M_m \in \mathbb{R}^{m\times m}$ an spd matrix and $U_n \in \mathbb{R}^{n\times m}$ with $U_n^T U_n = I_m$

- Obtain the $(n+1)\times m$ eigenspace U_{n+1} of the $(n+1)\times(n+1)$ kernel matrix K_{n+1}

$$K_{n+1} = \begin{bmatrix} K_n & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \approx \tilde{U}_{n+1,m+2}\tilde{M}_{m+2}\tilde{U}_{n+1,m+2}^T$$

Sweeping through K

- Suppose a rank m approximation of the dominant eigenspace of $K_n \in \mathbb{R}^{n\times n},\ n \gg m$, is known,

$$K_n \approx A_n := U_n M_m U_n^T,$$

  $M_m \in \mathbb{R}^{m\times m}$ an spd matrix and $U_n \in \mathbb{R}^{n\times m}$ with $U_n^T U_n = I_m$

- Obtain the $(n+1)\times m$ eigenspace U_{n+1} of the $(n+1)\times(n+1)$ kernel matrix K_{n+1}

$$K_{n+1} = \begin{bmatrix} K_n & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \approx \tilde{U}_{n+1,m+2}\tilde{M}_{m+2}\tilde{U}_{n+1,m+2}^T$$

- Downdate $\tilde{M}_{m+2}$ to get back to rank m

Sweeping through K

- Suppose a rank m approximation of the dominant eigenspace of $K_n \in \mathbb{R}^{n\times n},\ n \gg m$, is known,

$$K_n \approx A_n := U_n M_m U_n^T,$$

  $M_m \in \mathbb{R}^{m\times m}$ an spd matrix and $U_n \in \mathbb{R}^{n\times m}$ with $U_n^T U_n = I_m$

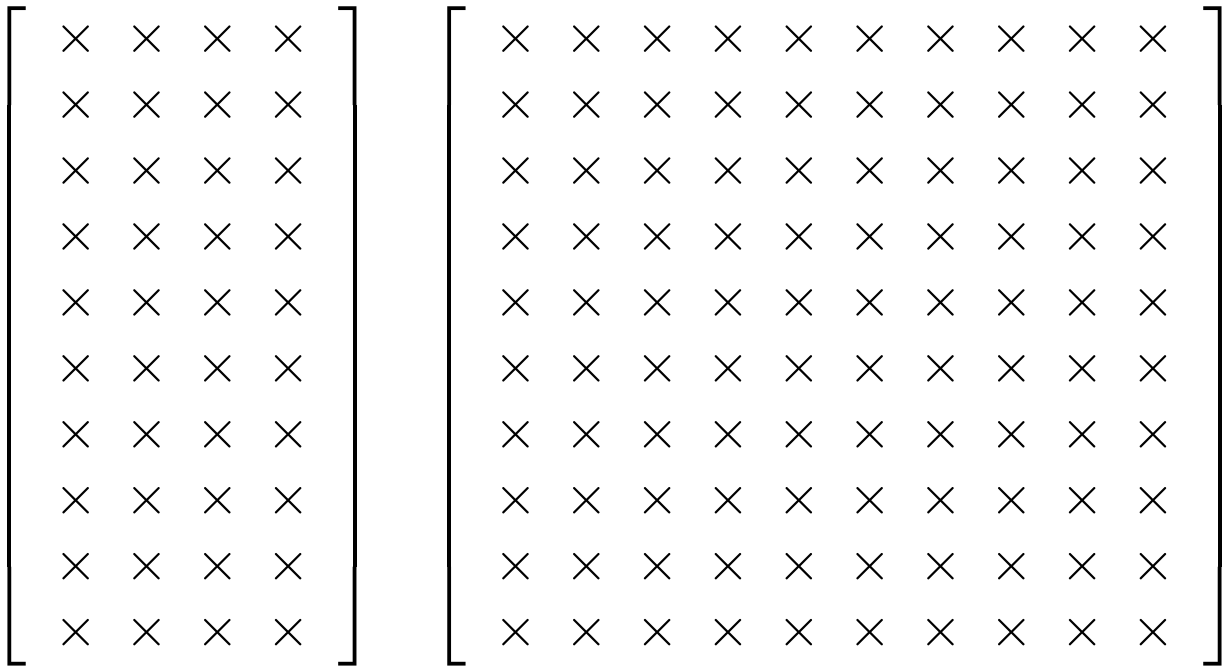
- Obtain the $(n+1)\times m$ eigenspace U_{n+1} of the $(n+1)\times(n+1)$ kernel matrix K_{n+1}

$$K_{n+1} = \begin{bmatrix} K_n & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \approx \tilde{U}_{n+1,m+2}\tilde{M}_{m+2}\tilde{U}_{n+1,m+2}^T$$

- Downdate $\tilde{M}_{m+2}$ to get back to rank m
- Downsize $\tilde{U}_{n+1}$ to get back to size n

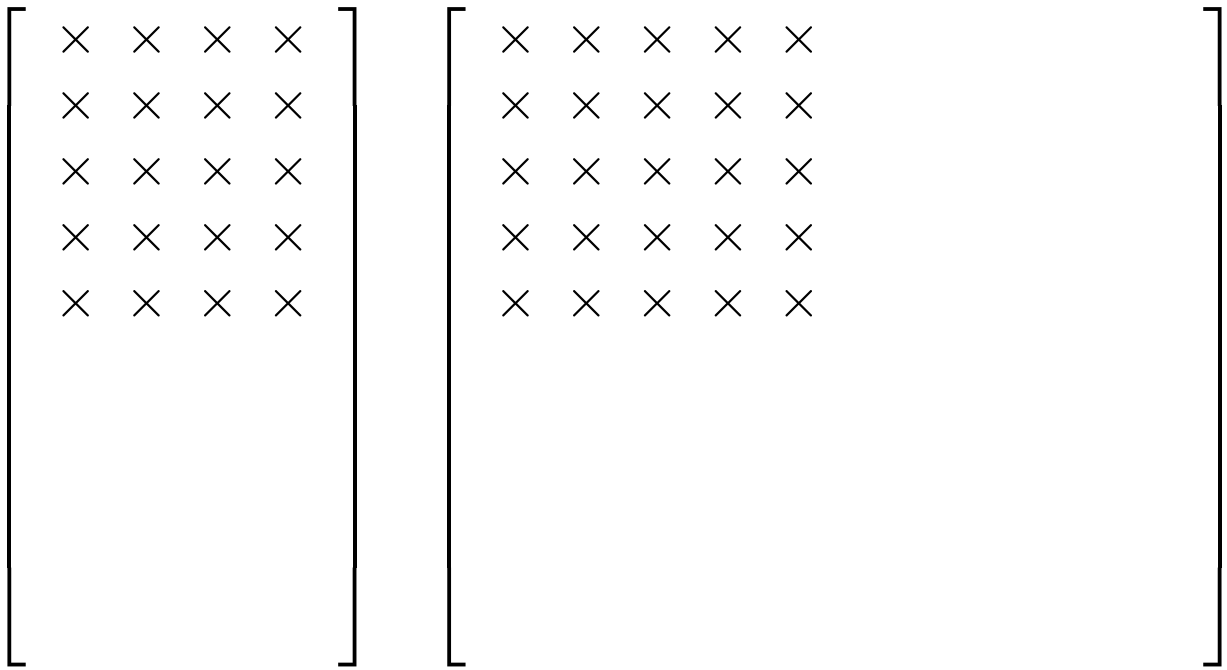
Sweeping through K

We show only the columns and rows of U and K that are involved



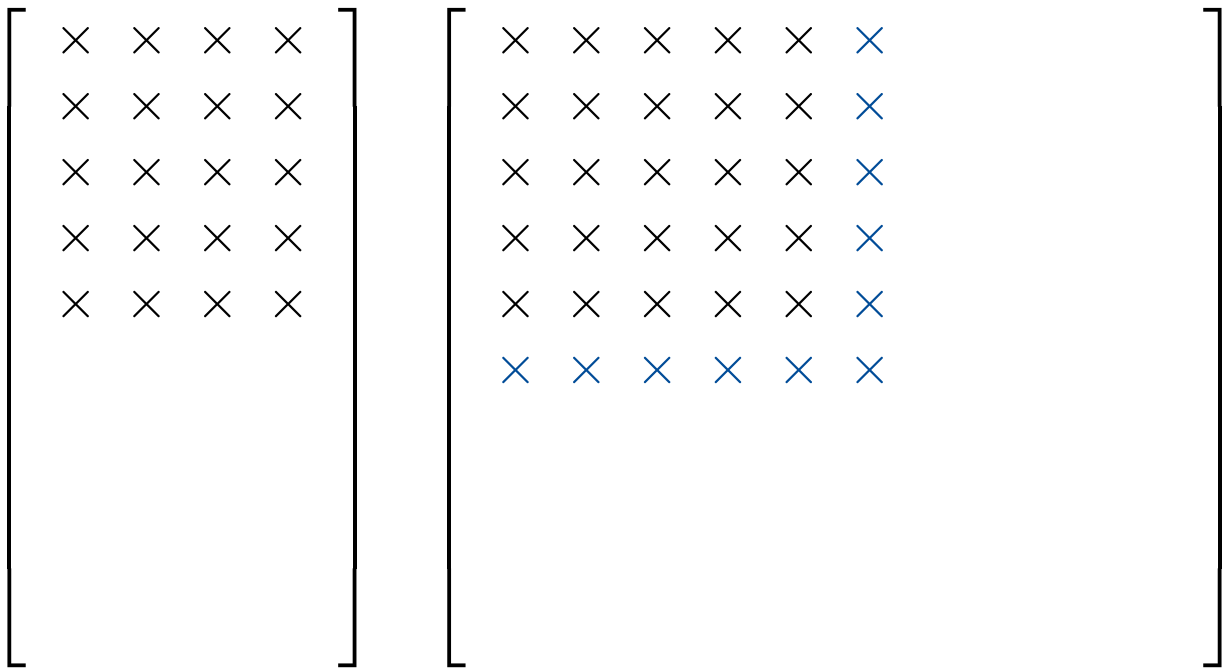
Window size $n = 5$, rank $k = 4$

step 1



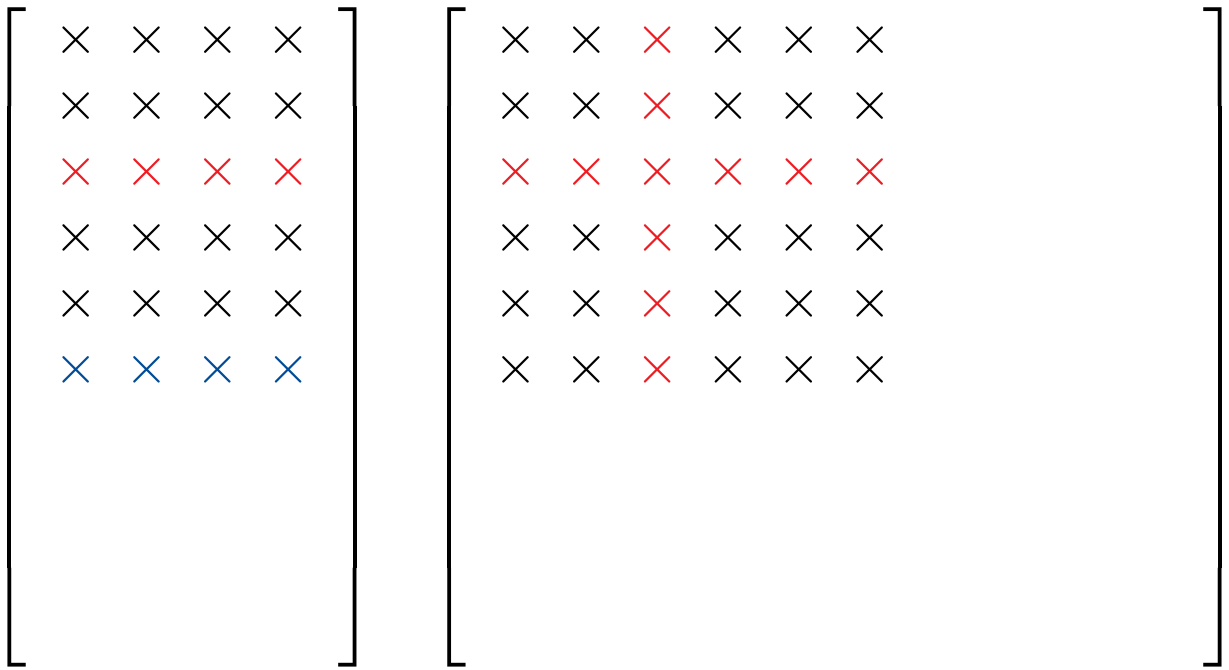
Start with leading $n \times n$ subproblem

step 2



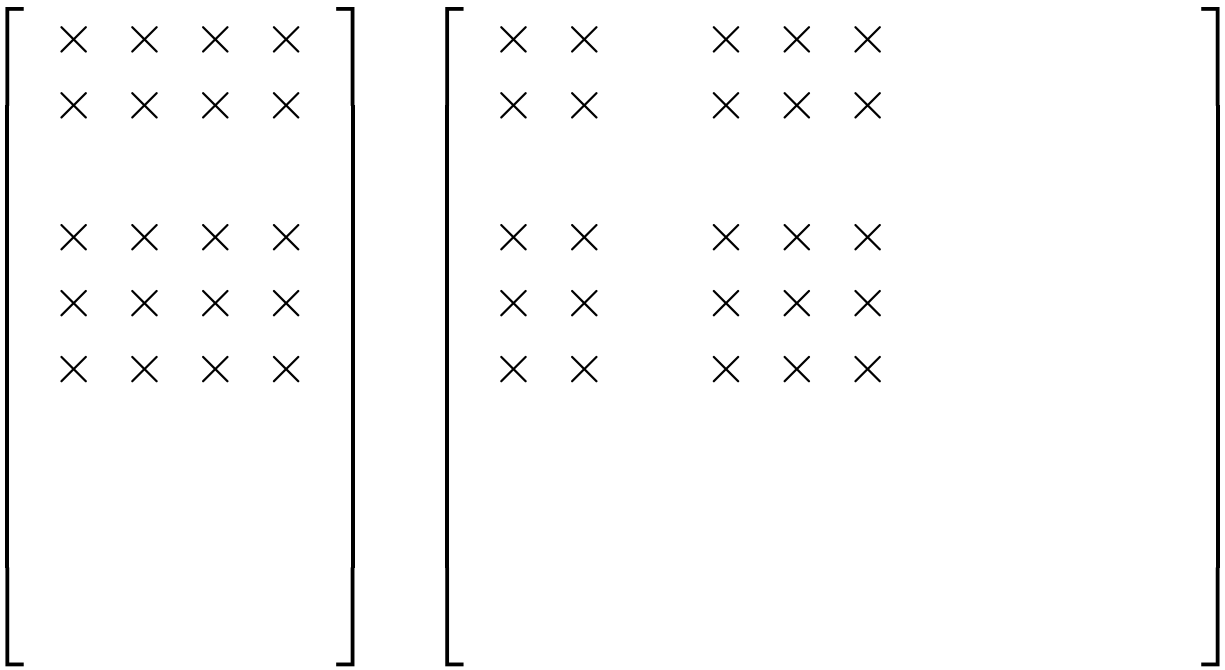
Expand

step 3

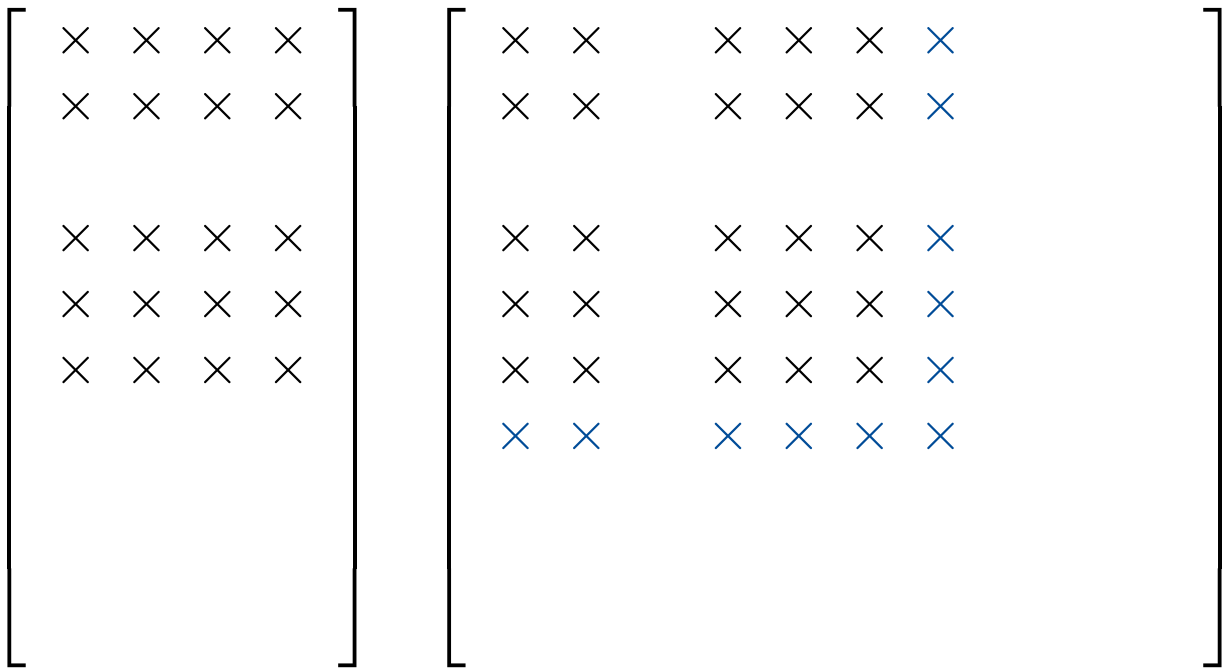


Downdate and downsize

step 4

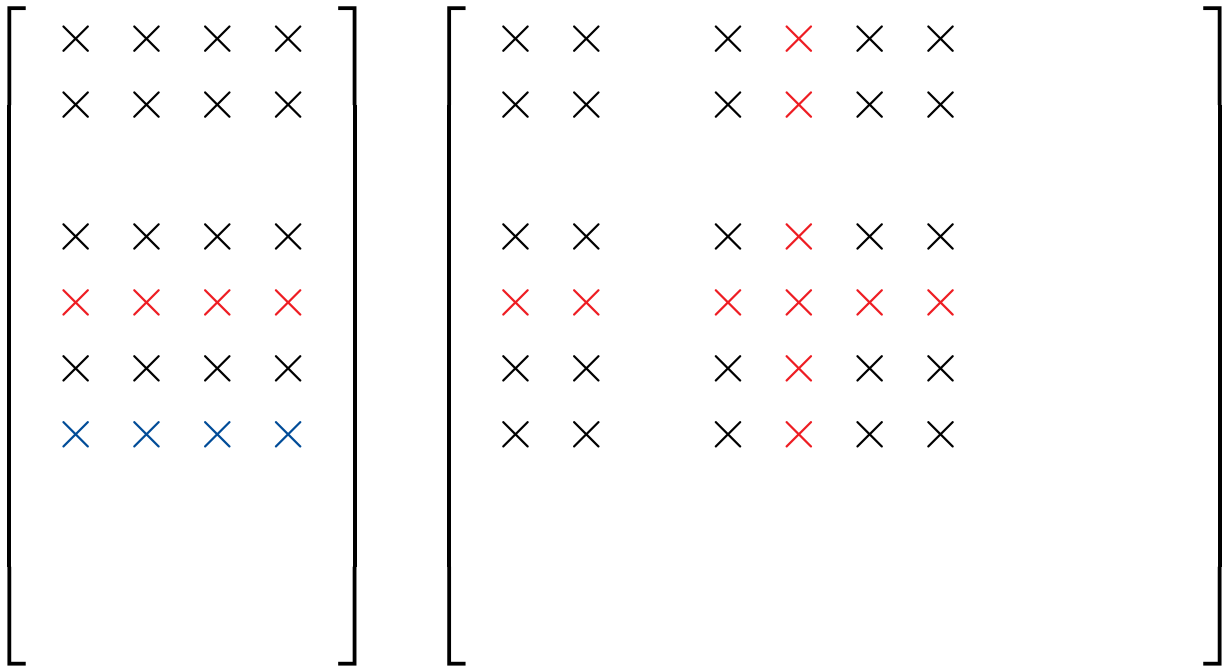


step 5



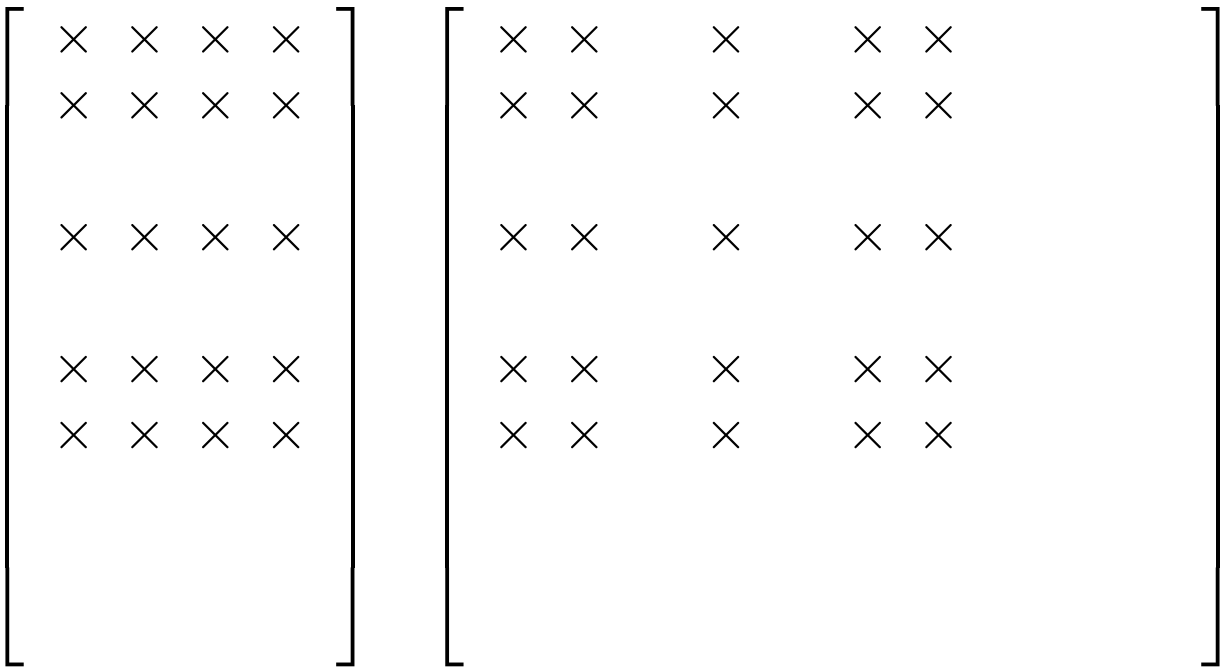
Expand

step 6

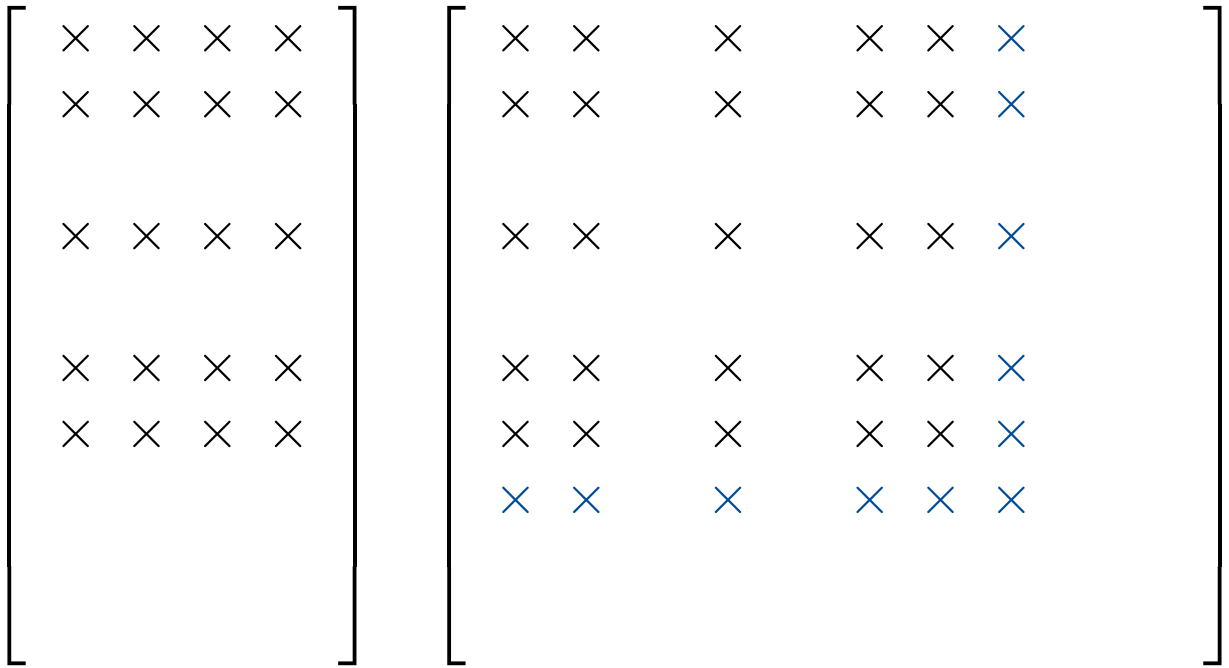


Downdate and downsize

step 7

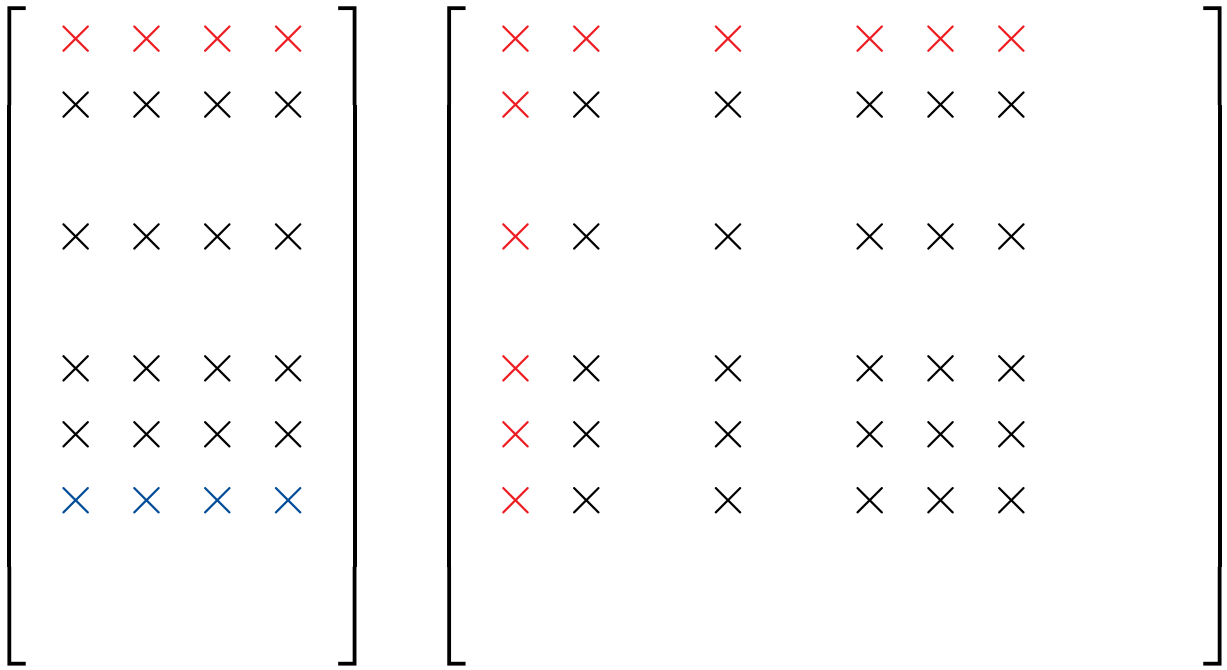


step 8



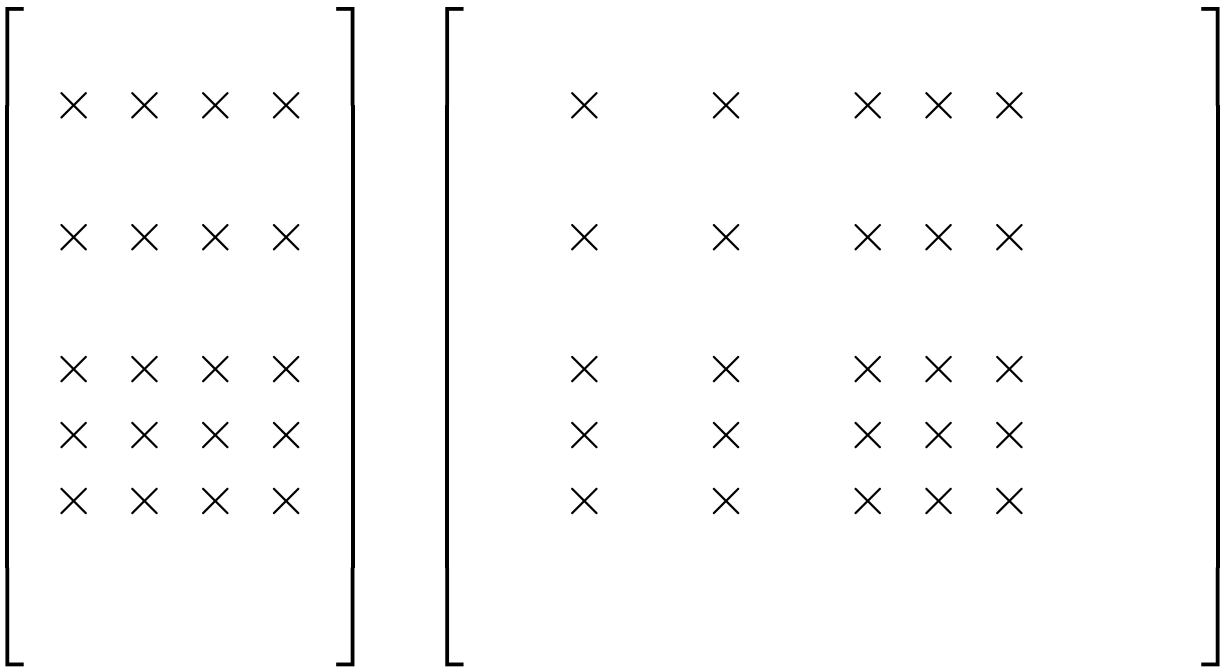
Expand

step 9

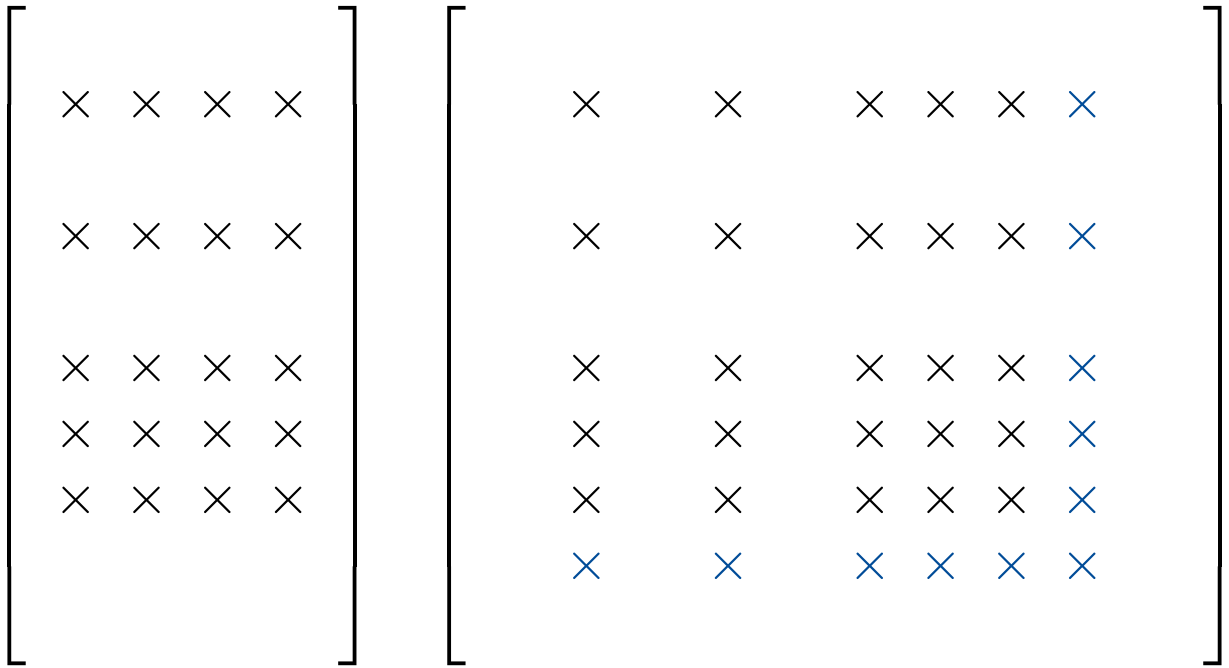


Downdate and downsize

step 10

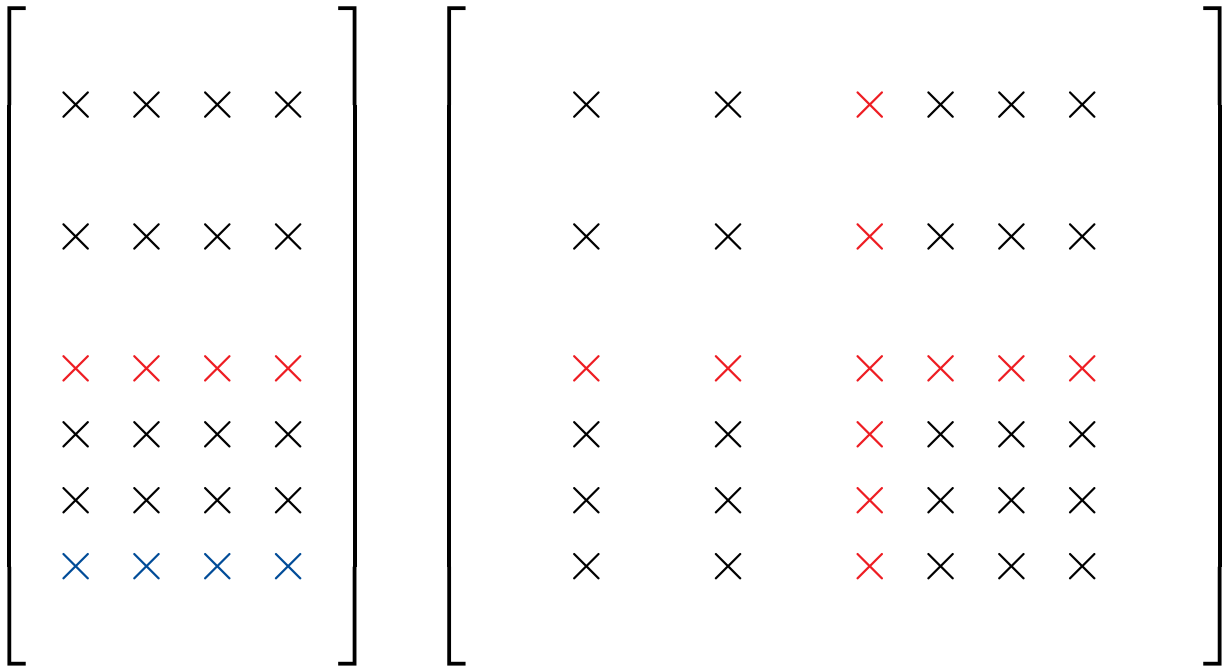


step 11



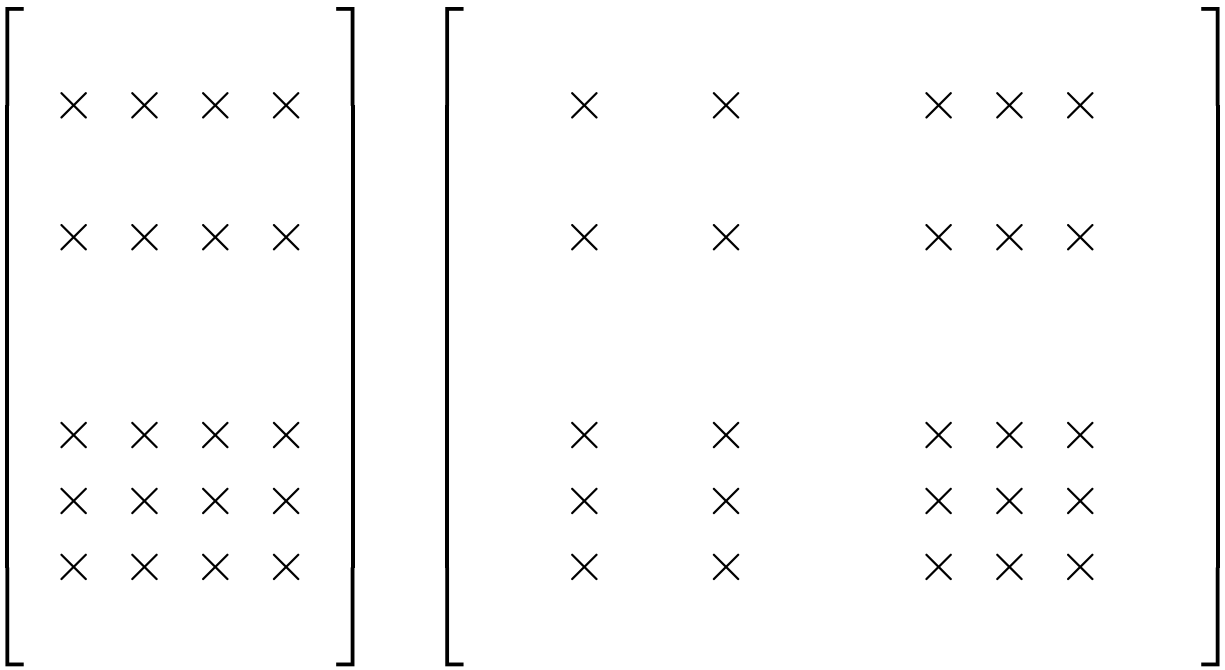
Expand

step 12

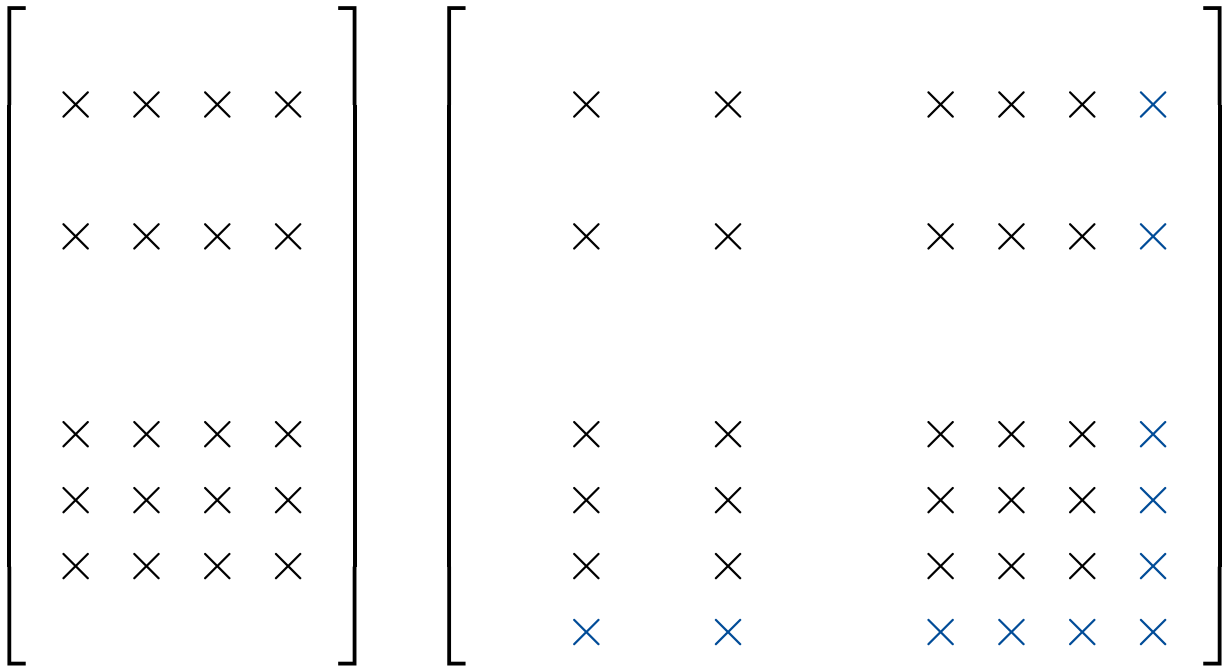


Downdate and downsize

step 13

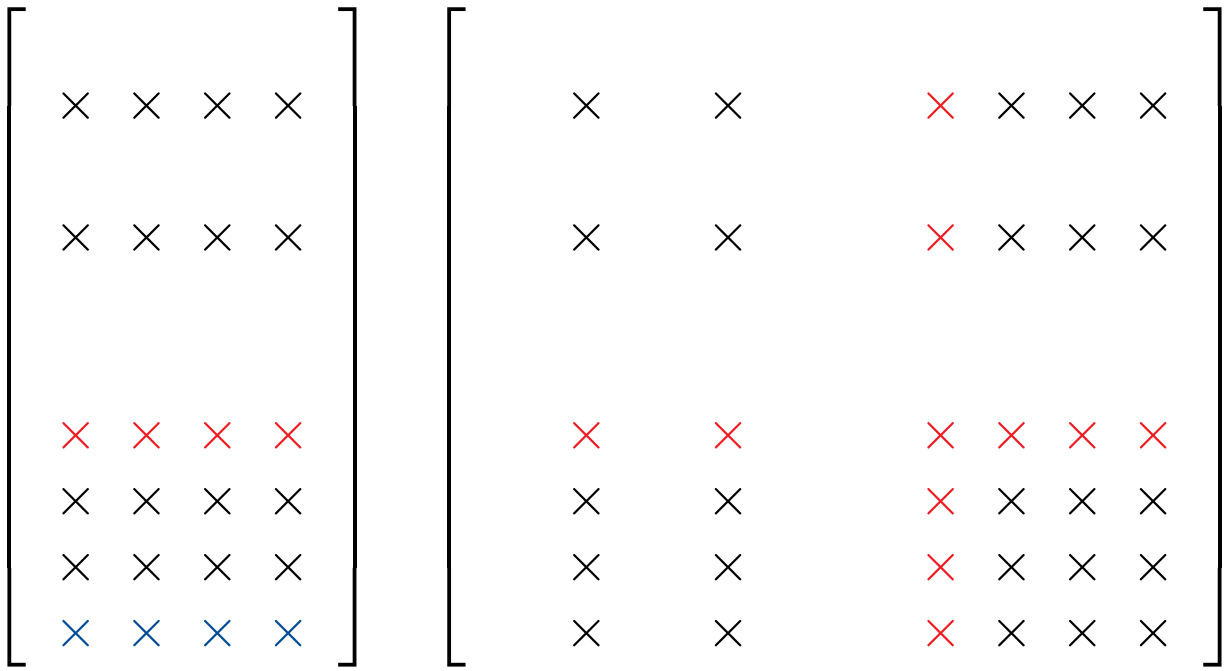


step 14



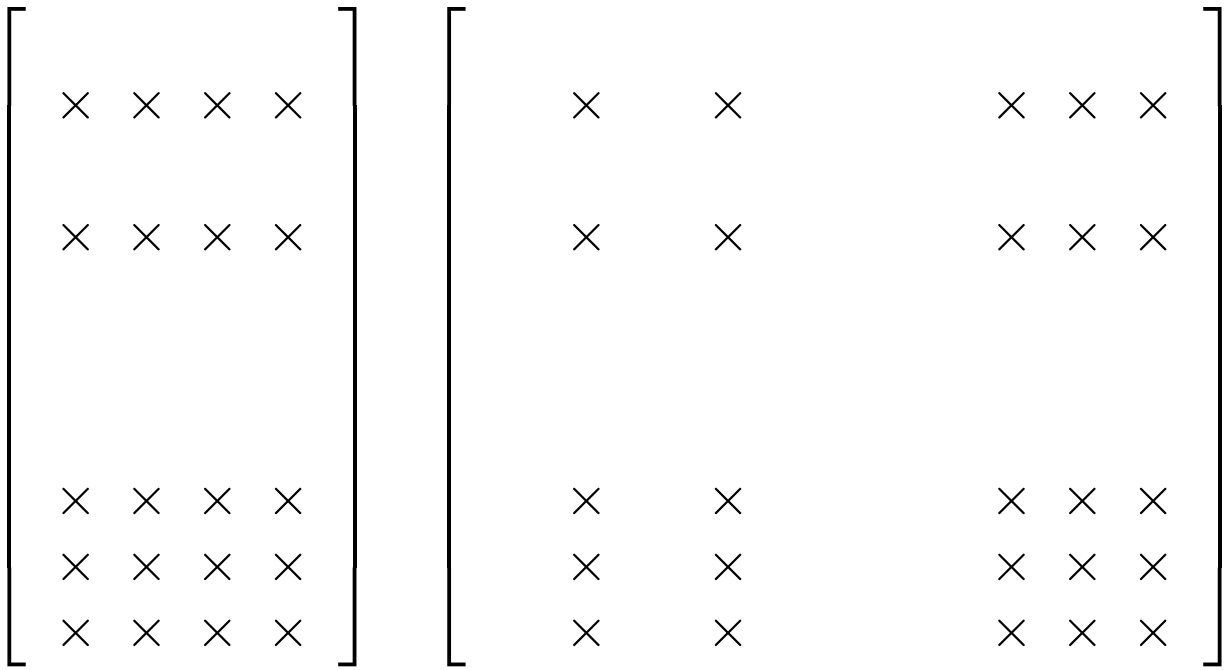
Expand

step 15



Downdate and downsize

step 16



Downdating K to fixed rank m

- Suppose a rank m approximation of the dominant eigenspace of $K_n \in \mathbb{R}^{n\times n},\ n \gg m$, is known,

$$K_n \approx A_n := U_n M_m U_n^T,$$

$M_m \in \mathbb{R}^{m\times m}$ an spd matrix and $U_n \in \mathbb{R}^{n\times m}$ with $U_n^T U_n = I_m$

- Obtain the $(n+1)\times m$ eigenspace U_{n+1} of the $(n+1)\times(n+1)$ kernel matrix K_{n+1}

$$K_{n+1} = \begin{bmatrix} K_n & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \approx \tilde{U}_{n+1,m+2}\tilde{M}_{m+2}\tilde{U}_{n+1,m+2}^T$$

- Downdate $\tilde{M}_{m+2}$ to delete the “smallest" two eigenvalues

Updating: Proposed algorithm

Since

$$\begin{aligned}\mathbf{a} &= U_n U_n^T \mathbf{a} + (I_m - U_n U_n^T)\mathbf{a} \\ &= U_n \mathbf{r} + \rho \mathbf{u}_\perp,\end{aligned}$$

with $\mathbf{q} = (I_m - U_n U_n^T)\mathbf{a}$, $\rho = \|\mathbf{q}\|_2$, $\mathbf{u}_\perp = \mathbf{q}/\rho$, we can write

$$\begin{aligned}A_{n+1} &= \left[\begin{array}{c|c} A_n & \mathbf{a} \\ \hline \mathbf{a}^T & b \end{array}\right] \\ &= \left[\begin{array}{c|c|c} U_n & \mathbf{u}_\perp & \\ \hline & & 1 \end{array}\right] \left[\begin{array}{c|c|c} M_m & & \mathbf{r} \\ \hline & 0 & \rho \\ \hline \mathbf{r}^T & \rho & b \end{array}\right] \left[\begin{array}{c|c|c} U_n^T & & \\ \hline \mathbf{u}_\perp{}^T & & \\ \hline & & 1 \end{array}\right] \\ &= \left[\begin{array}{c|c|c} U_n & \mathbf{u}_\perp & \\ \hline & & 1 \end{array}\right] \tilde{M}_{m+2} \left[\begin{array}{c|c|c} U_n^T & & \\ \hline \mathbf{u}_\perp{}^T & & \\ \hline & & 1 \end{array}\right].\end{aligned}$$

Updating: Proposed algorithm

- Let

$$M_m = Q_m \Lambda_m Q_m^T$$

where

$\Lambda_m = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_m), \quad \mu_1 \geq \cdots \geq \mu_m > 0, \quad Q_m^T Q_m = I_m.$

Updating: Proposed algorithm

- Let
$$M_m = Q_m \Lambda_m Q_m^T$$
where
$\Lambda_m = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_m), \quad \mu_1 \geq \cdots \geq \mu_m > 0, \quad Q_m^T Q_m = I_m.$

- Let
$$\tilde{M}_{m+2} = \tilde{Q}_{m+2} \tilde{\Lambda}_{m+2} \tilde{Q}_{m+2}^T$$
where
$\tilde{\Lambda}_{m+2} = \mathrm{diag}(\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_{m+1}, \tilde{\lambda}_{m+2}), \quad \tilde{Q}_{m+2}^T \tilde{Q}_{m+2} = I_{m+2}$

Updating: Proposed algorithm

- Let
$$M_m = Q_m \Lambda_m Q_m^T$$
where
$\Lambda_m = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_m), \quad \mu_1 \geq \cdots \geq \mu_m > 0, \quad Q_m^T Q_m = I_m.$

- Let
$$\tilde{M}_{m+2} = \tilde{Q}_{m+2} \tilde{\Lambda}_{m+2} \tilde{Q}_{m+2}^T$$
where
$\tilde{\Lambda}_{m+2} = \mathrm{diag}(\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_{m+1}, \tilde{\lambda}_{m+2}), \quad \tilde{Q}_{m+2}^T \tilde{Q}_{m+2} = I_{m+2}$

- By the interlacing property, we have
$$\tilde{\lambda}_1 \geq \mu_1 \geq \tilde{\lambda}_2 \geq \mu_2 \geq \cdots \geq \mu_m \geq \tilde{\lambda}_{m+1} \geq 0 \geq \tilde{\lambda}_{m+2}$$

Updating: Proposed algorithm

We want a simple orthogonal transformation H such that

$$H\,[v_{m+1}\ v_{m+2}] = \left[\begin{array}{cc} 0 & 0 \\ \hline \times & 0 \\ 0 & \times \end{array}\right], \qquad H\tilde{M}_{m+2}H^T = \left[\begin{array}{c|c} \bar{M}_m & \\ \hline & \begin{array}{cc} \lambda_{m+1} & \\ & \lambda_{m+2} \end{array} \end{array}\right],$$

with $\bar{M}_m \in \mathbb{R}^{m\times m}$. Therefore

$$A_{n+1} = \left[\begin{array}{c|c|c} U_n & \mathbf{u}_\perp & \\ \hline & & 1 \end{array}\right] H^T \left[\begin{array}{c|c} \bar{M}_m & \\ \hline & \begin{array}{cc} \lambda_{m+1} & \\ & \lambda_{m+2} \end{array} \end{array}\right] H \left[\begin{array}{c|c} U_n^T & \\ \mathbf{u}_\perp{}^T & \\ \hline & 1 \end{array}\right].$$

and the new updated decomposition is given by

$$\hat{A}_{n+1} = U_{n+1}\bar{M}_m U_{n+1}^T,$$

with U_{n+1} given by the first m columns of $\left[\begin{array}{c|c|c} U_n & \mathbf{u}_\perp & \\ \hline & & 1 \end{array}\right] H^T$

Cholesky factorization

- It is not needed to compute the whole spectral decomposition of the matrix $\tilde{M}_{m+2}$

Cholesky factorization

- It is not needed to compute the whole spectral decomposition of the matrix $\tilde{M}_{m+2}$
- To compute H only the eigenvectors $v_{m+1},\ v_{m+2}$ corresponding to $\lambda_{m+1},\ \lambda_{m+2}$ are needed

Cholesky factorization

- It is not needed to compute the whole spectral decomposition of the matrix $\tilde{M}_{m+2}$
- To compute H only the eigenvectors $v_{m+1},\ v_{m+2}$ corresponding to $\lambda_{m+1},\ \lambda_{m+2}$ are needed
- To compute these vectors cheaply, one needs to maintain (and update) the Cholesky factorization of

$$M_m = L_m L_m^T$$

Cholesky factorization

- It is not needed to compute the whole spectral decomposition of the matrix $\tilde{M}_{m+2}$
- To compute H only the eigenvectors $v_{m+1},\ v_{m+2}$ corresponding to $\lambda_{m+1},\ \lambda_{m+2}$ are needed
- To compute these vectors cheaply, one needs to maintain (and update) the Cholesky factorization of

$$M_m = L_m L_m^T$$

- The eigenvectors are then obtained via inverse iteration and

Cholesky factorization

- It is not needed to compute the whole spectral decomposition of the matrix $\tilde{M}_{m+2}$
- To compute H only the eigenvectors $v_{m+1},\ v_{m+2}$ corresponding to $\lambda_{m+1},\ \lambda_{m+2}$ are needed
- To compute these vectors cheaply, one needs to maintain (and update) the Cholesky factorization of

$$M_m = L_m L_m^T$$

- The eigenvectors are then obtained via inverse iteration and
- H can then be computed as a product of Householder or Givens transformations

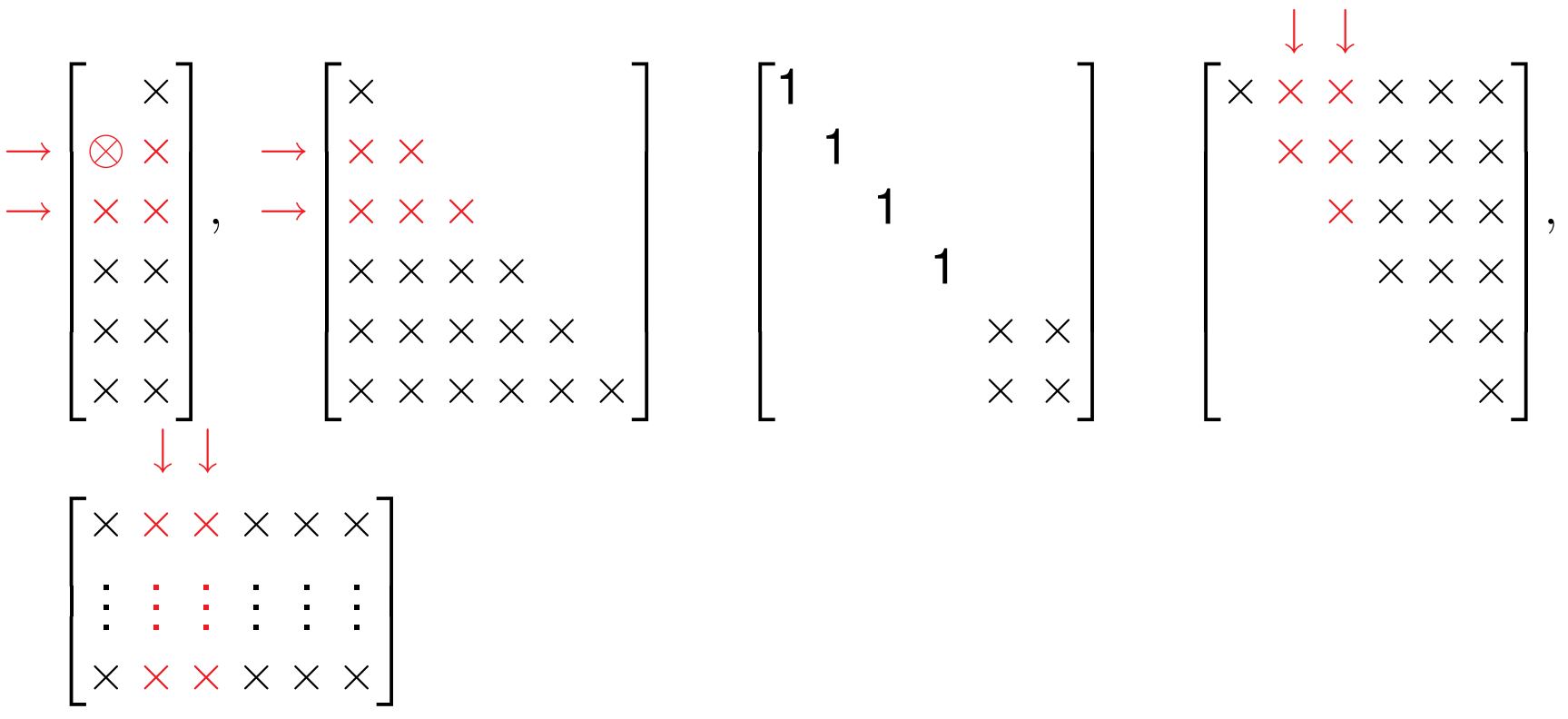
Updating Cholesky

$$
\begin{aligned}
\tilde{M}_{m+2} &= \left[\begin{array}{c|c|c} M_m & & \mathbf{r} \\ \hline & 0 & \rho \\ \hline \mathbf{r}^T & \rho & b \end{array}\right] \\
&= \left[\begin{array}{c|c|c} L_m L_m^T & & \mathbf{r} \\ \hline & 0 & \rho \\ \hline \mathbf{r}^T & \rho & b \end{array}\right] \\
&= \left[\begin{array}{c|c} L_m & \\ \hline \mathbf{0}_m^T & \\ \hline \mathbf{t}^T & I_2 \end{array}\right] \left[\begin{array}{c|c} I_m & \\ \hline & S_c \end{array}\right] \left[\begin{array}{c|c|c} L_m^T & \mathbf{0}_m & \mathbf{t} \\ \hline & I_2 & \end{array}\right] \\
&= \tilde{L}_{m+2} \tilde{D}_{m+2} \tilde{L}_{m+2}^T,
\end{aligned}
$$

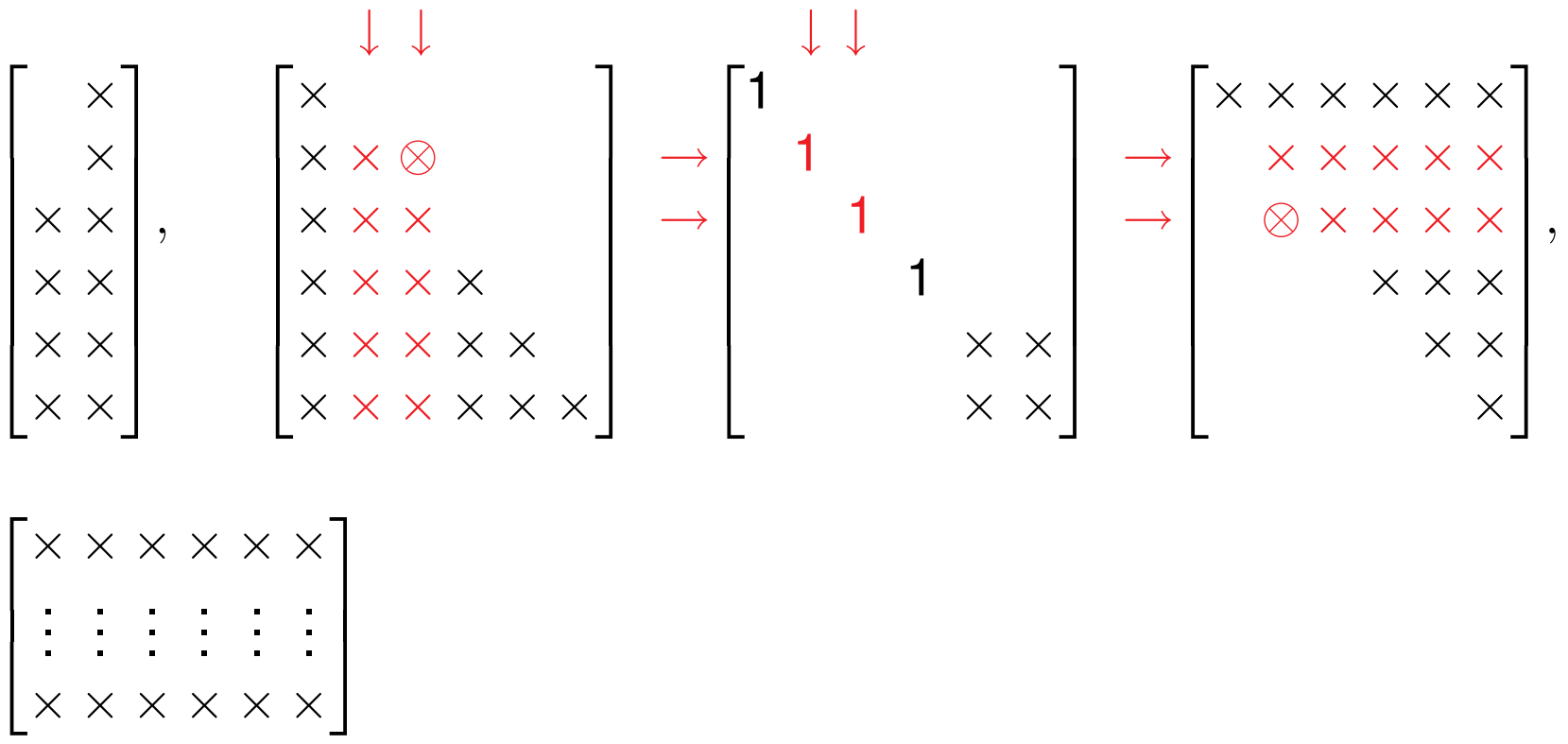
where

$$
\mathbf{t} = L_m^{-1}\mathbf{r} \quad \text{and} \quad S_c = \left[\begin{array}{c|c} 0 & \rho \\ \hline \rho & b - \mathbf{t}^T\mathbf{t} \end{array}\right].
$$

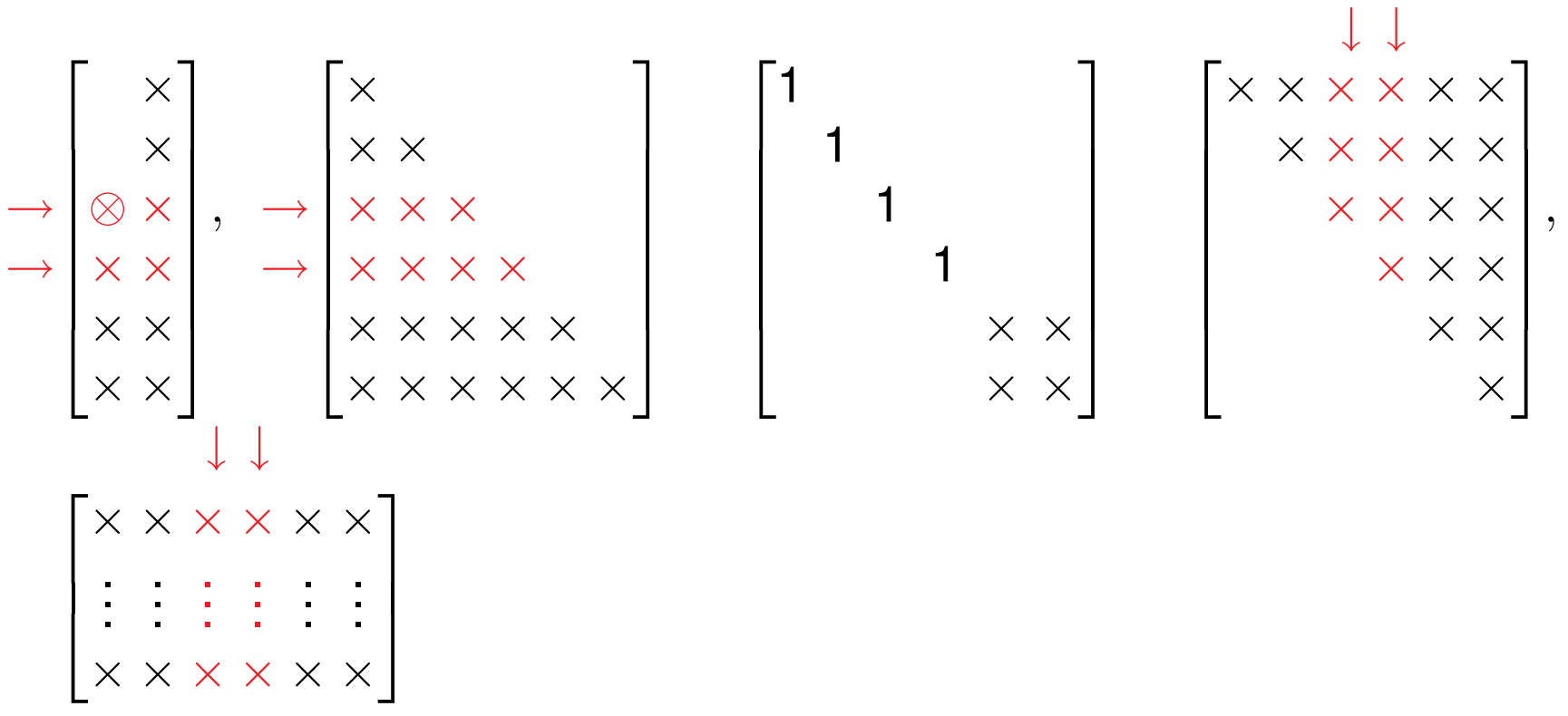
step 1



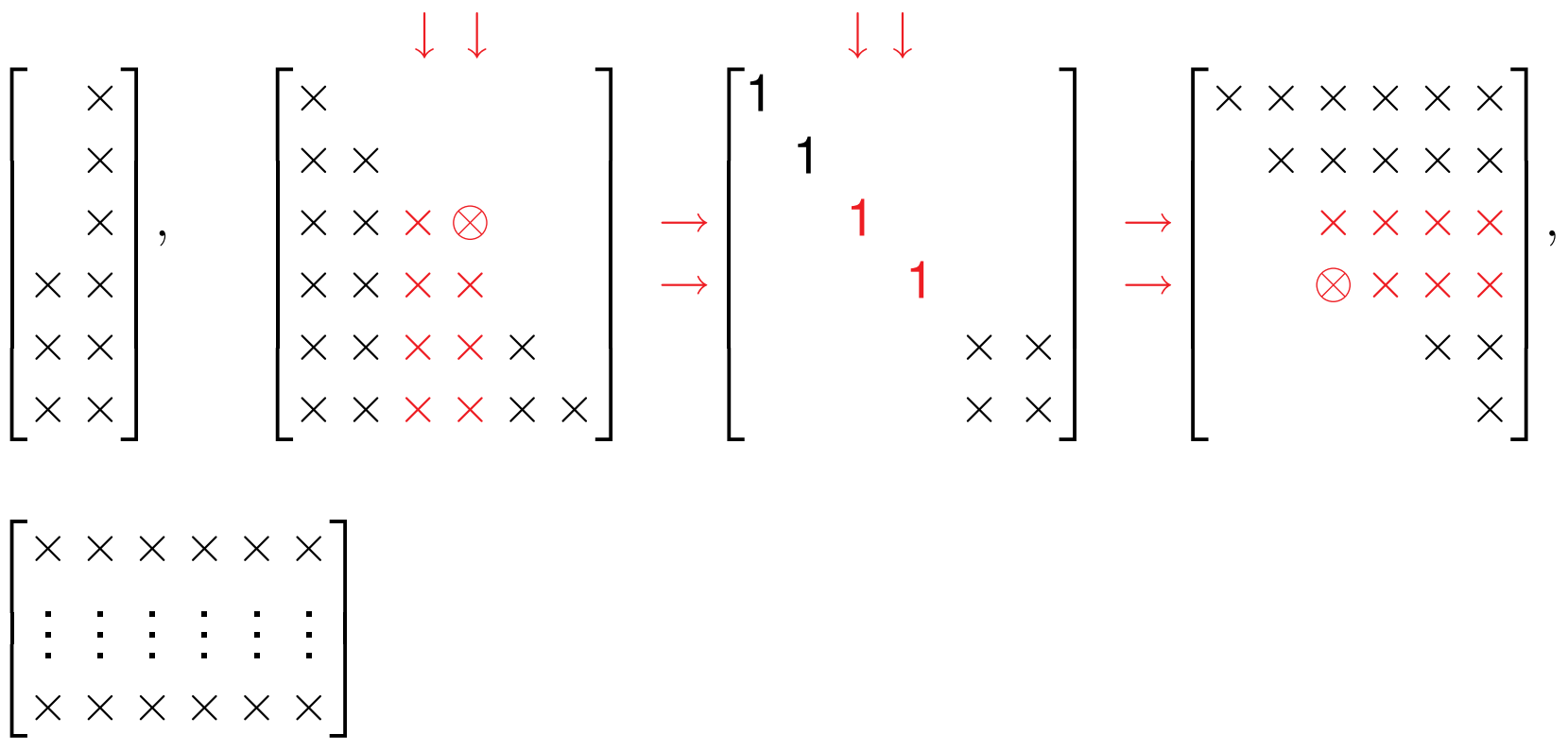
step 2



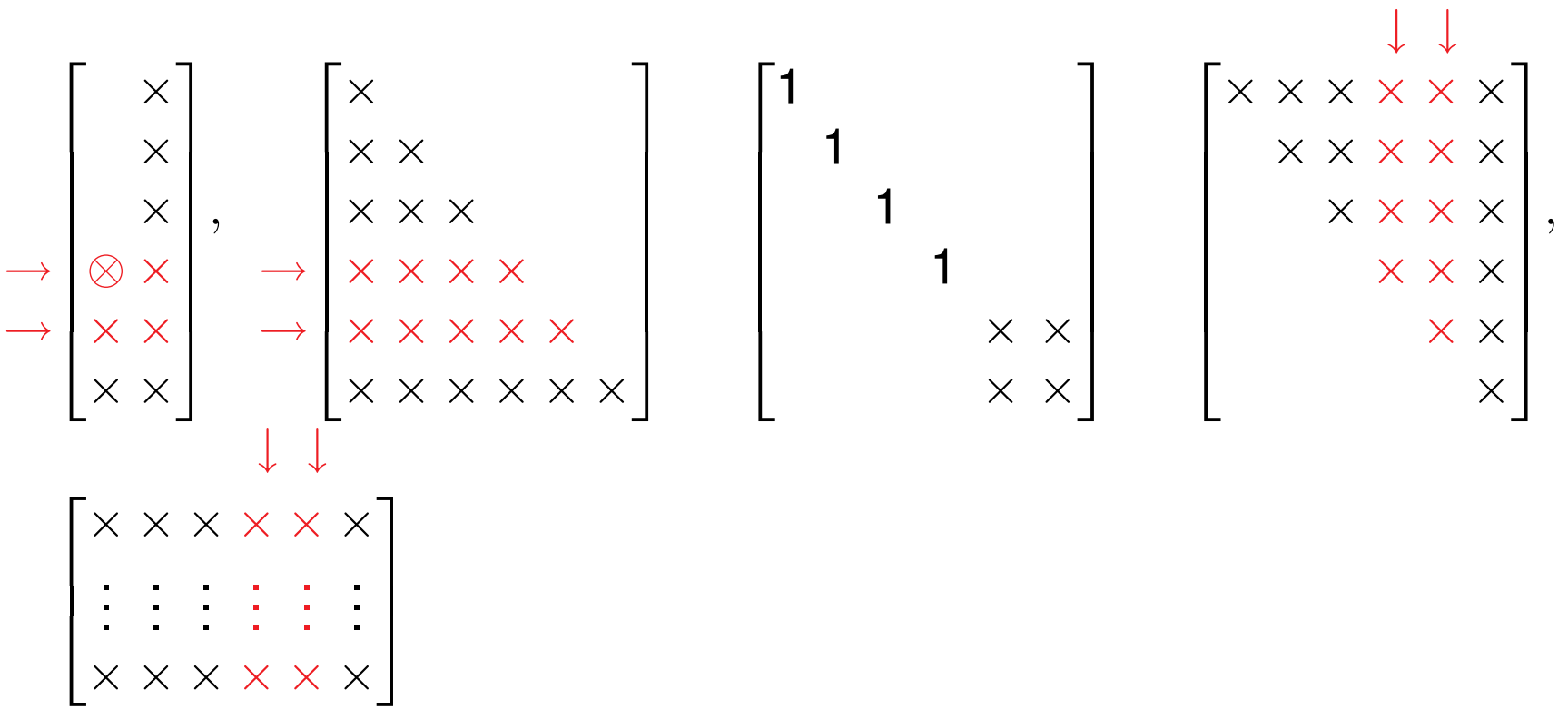
step 3



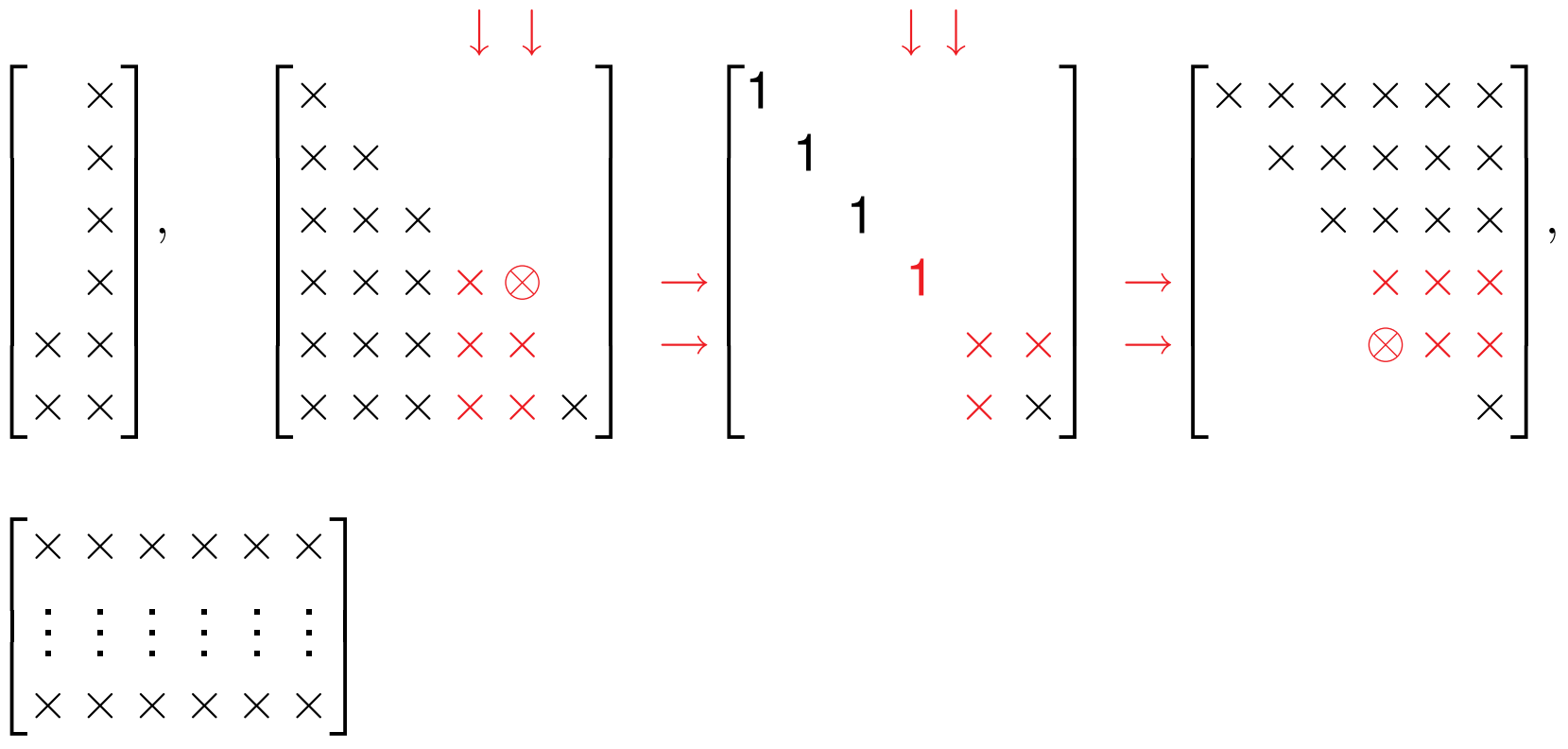
step 4



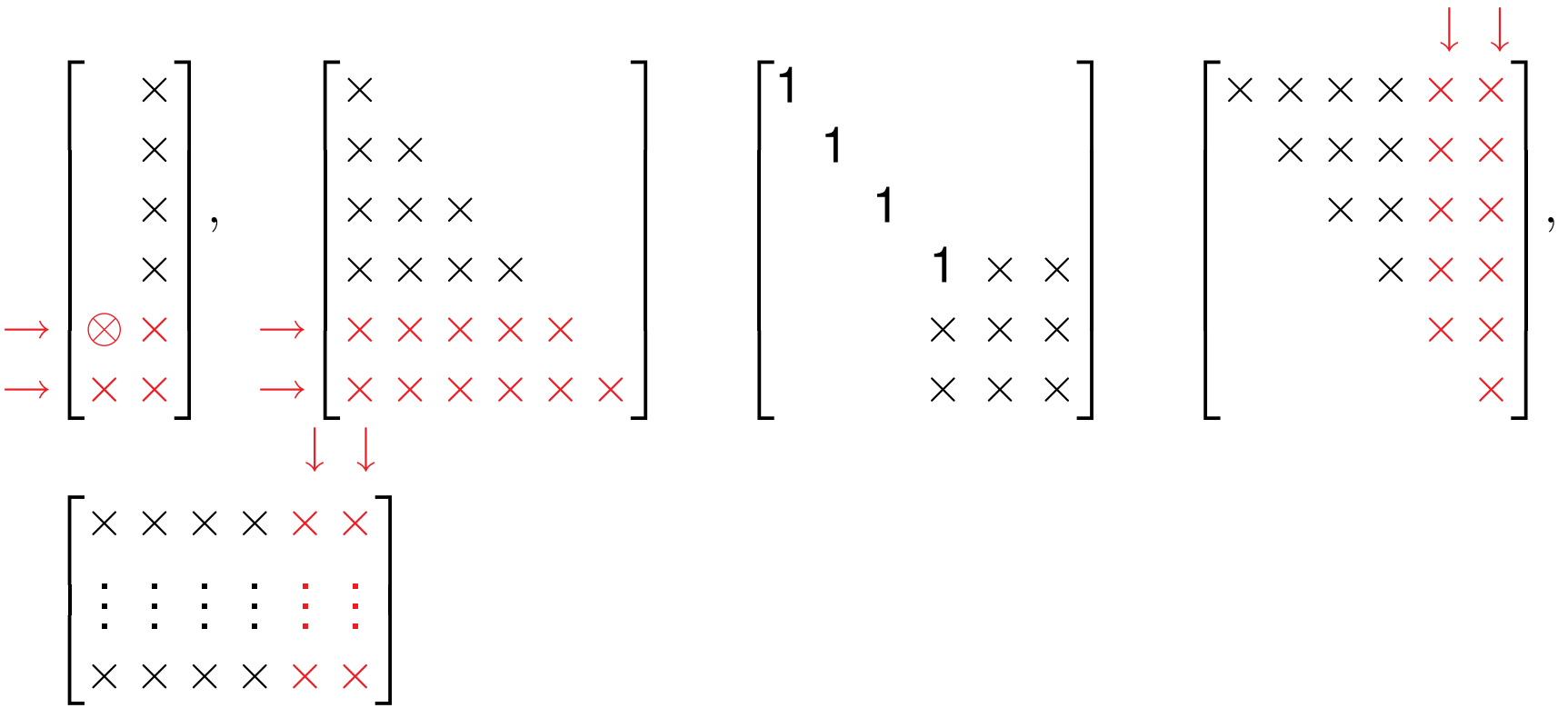
step 5



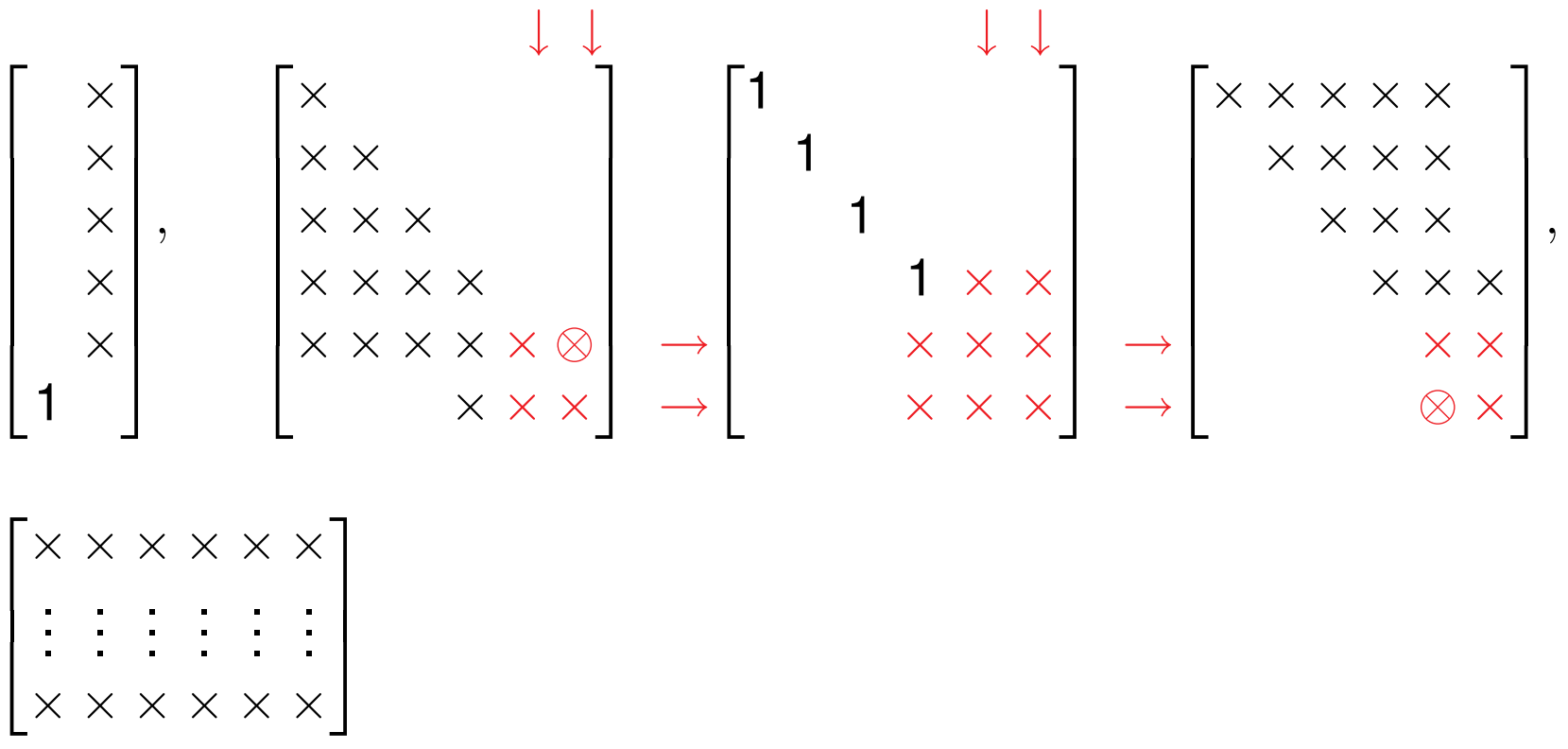
step 6



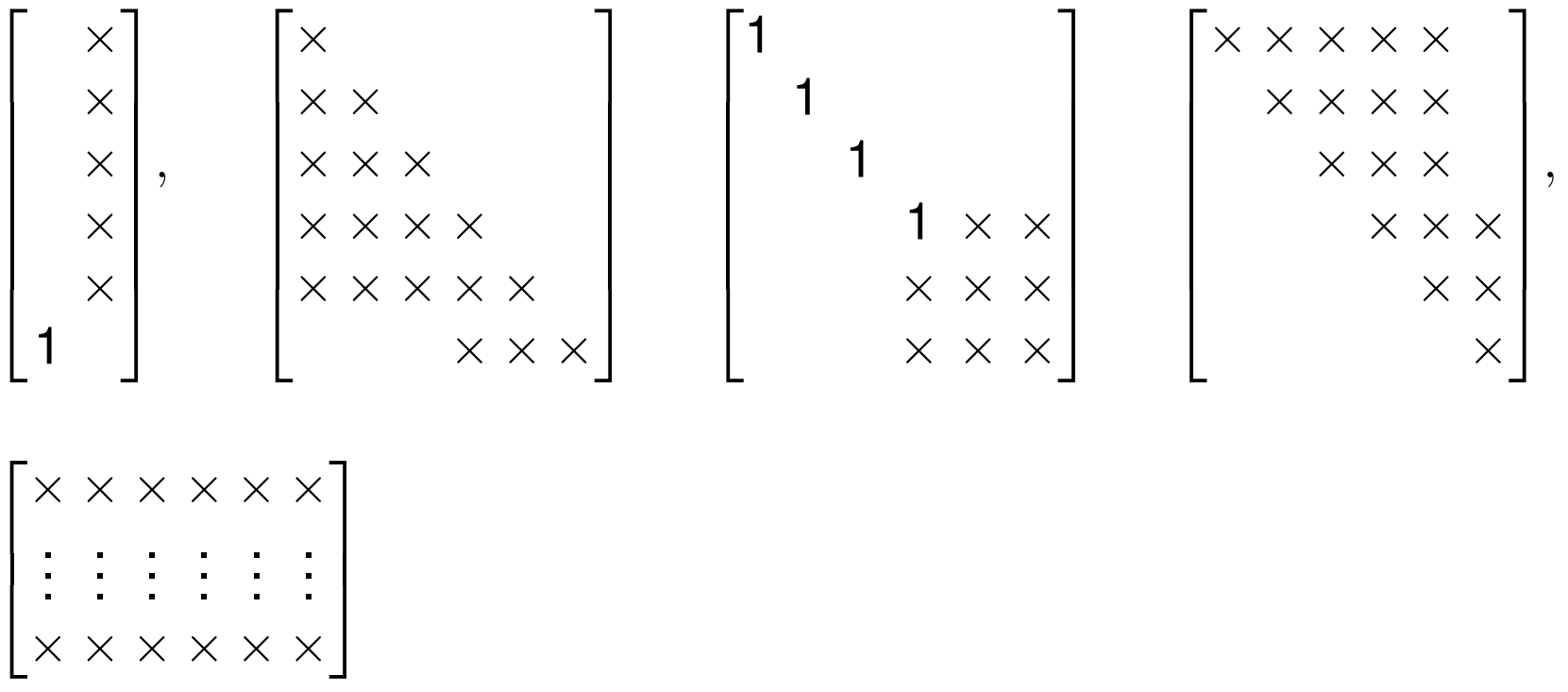
step 7



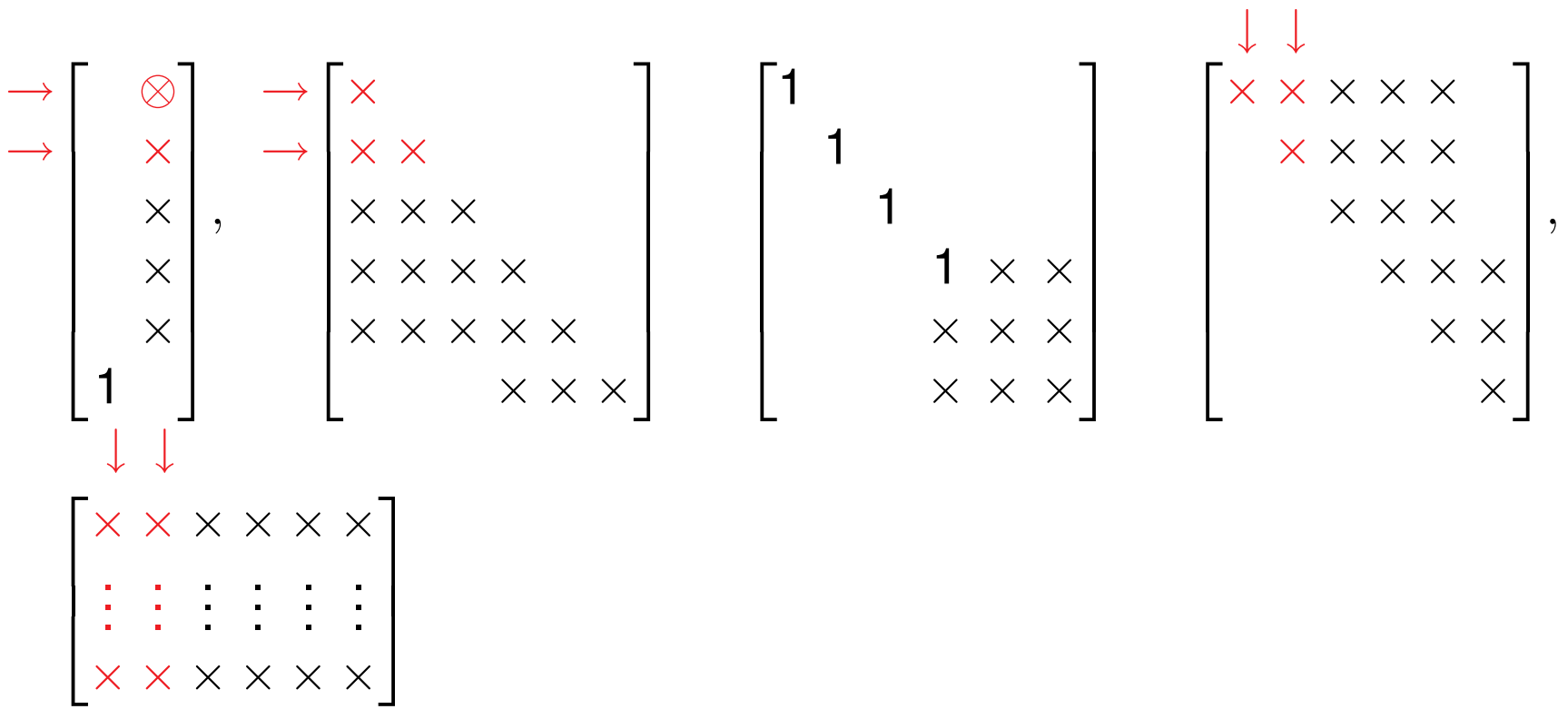
step 8



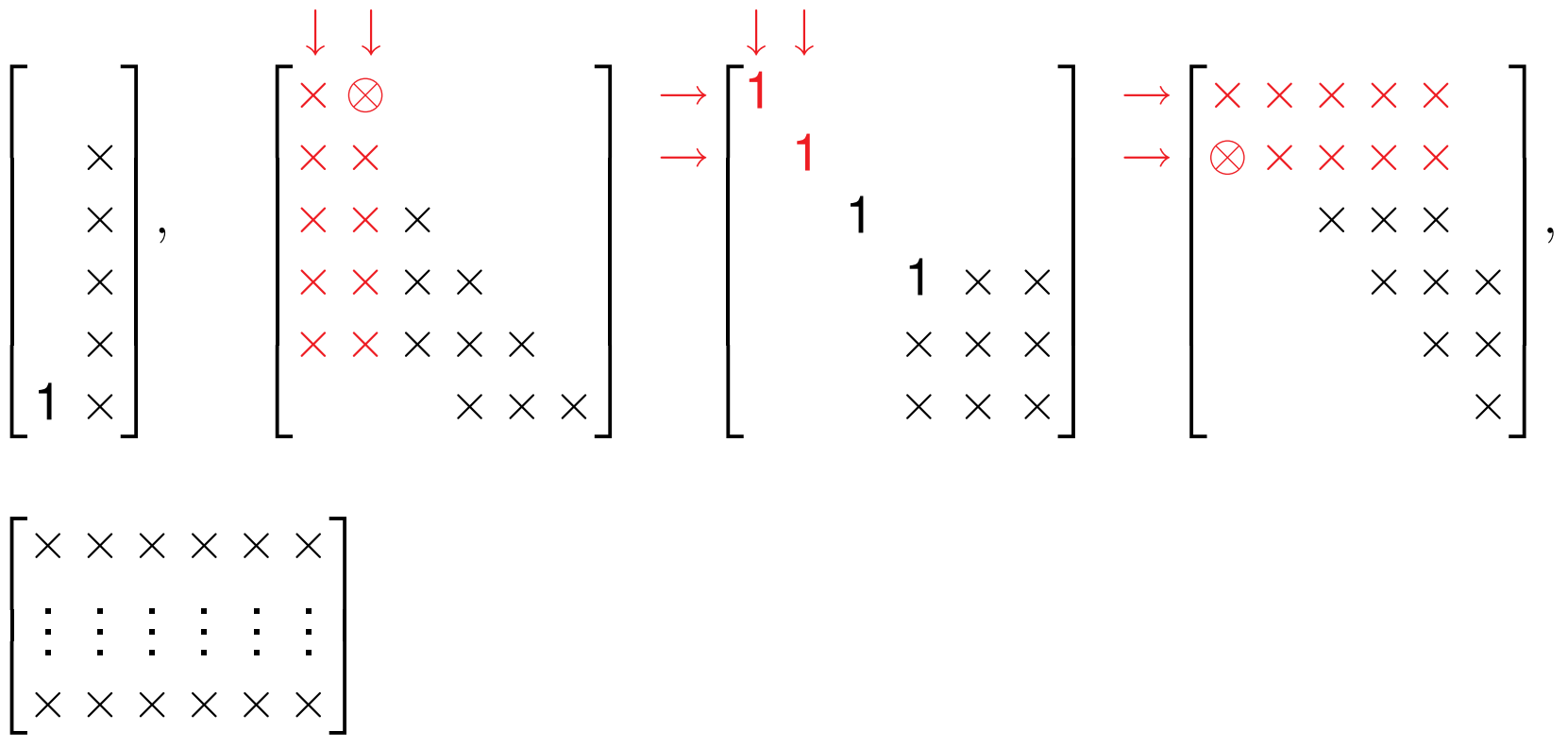
step 9



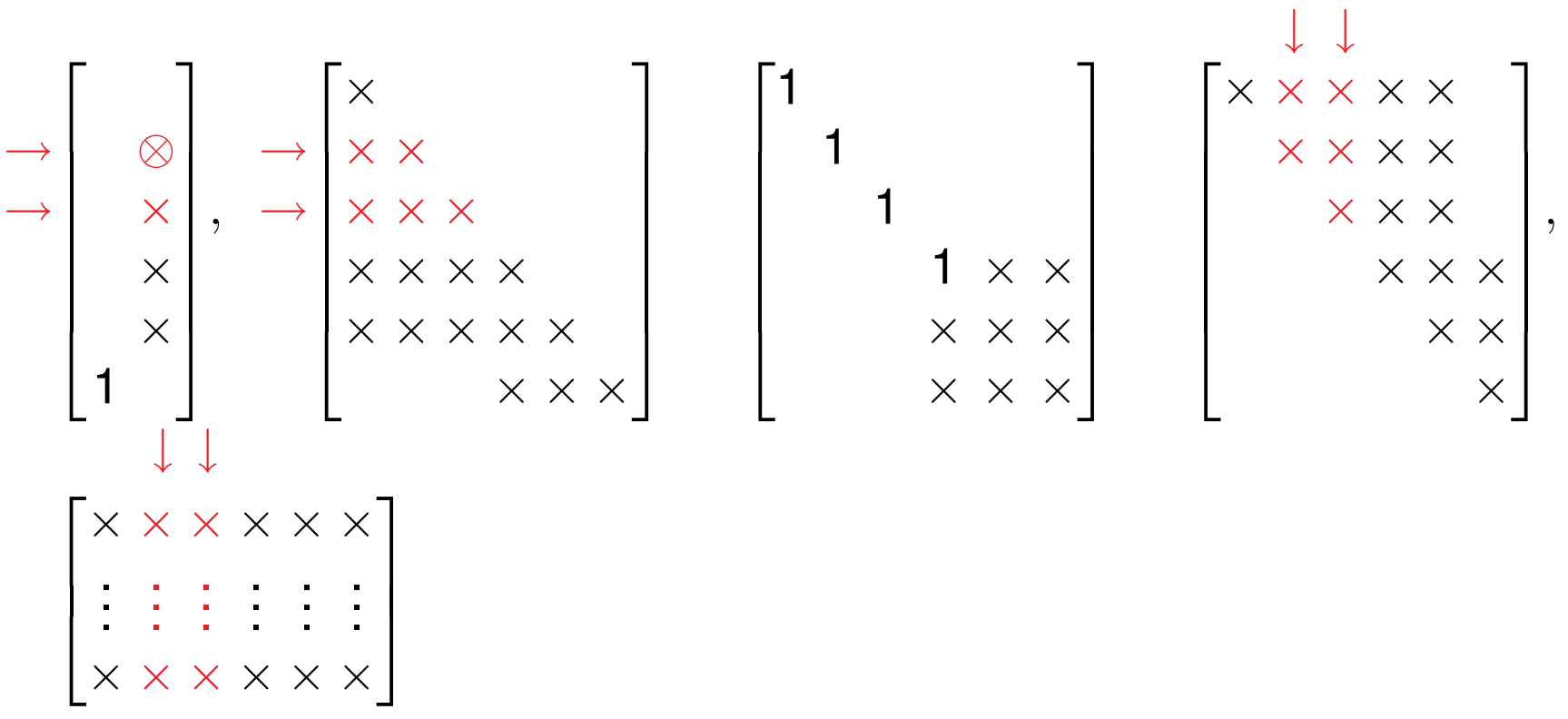
step 9



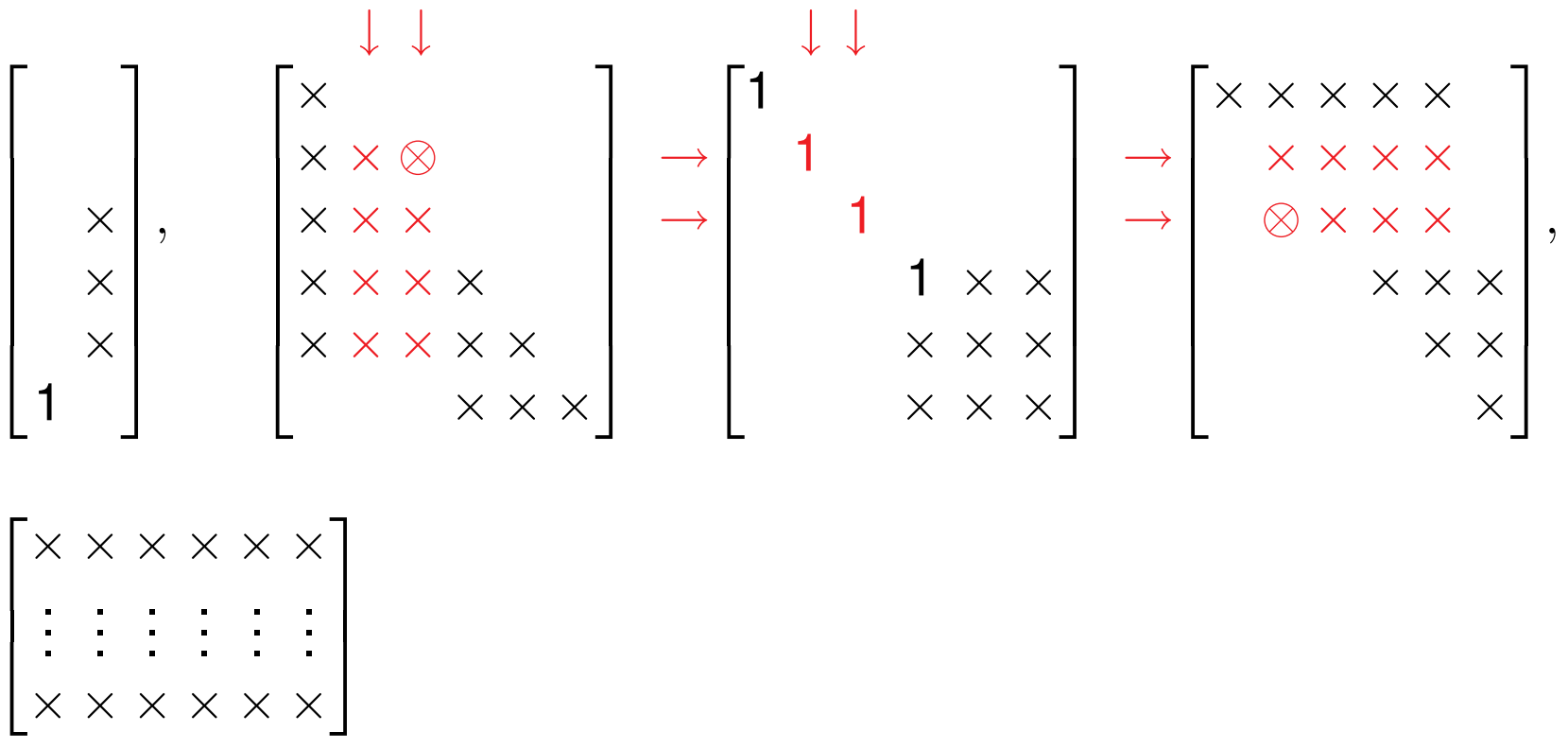
step 10



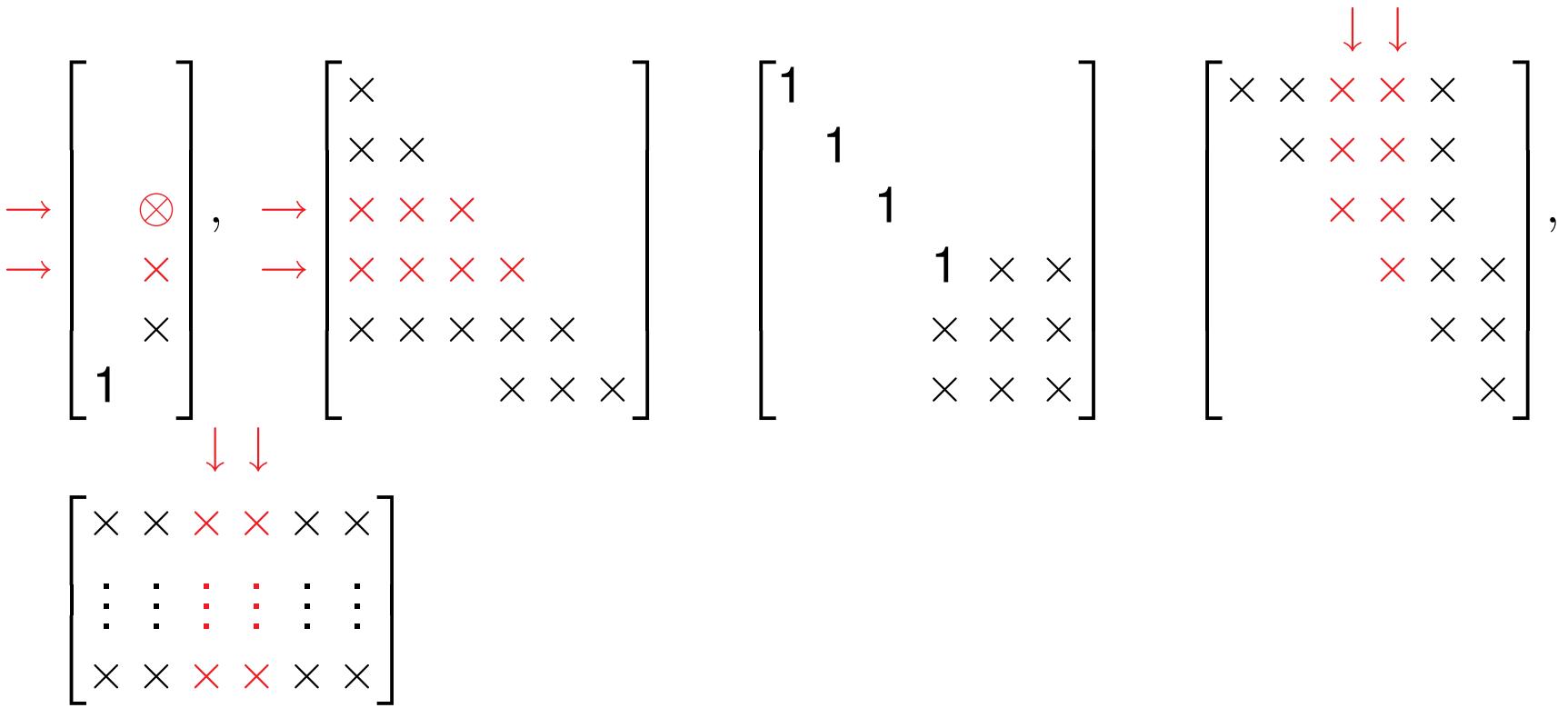
step 11



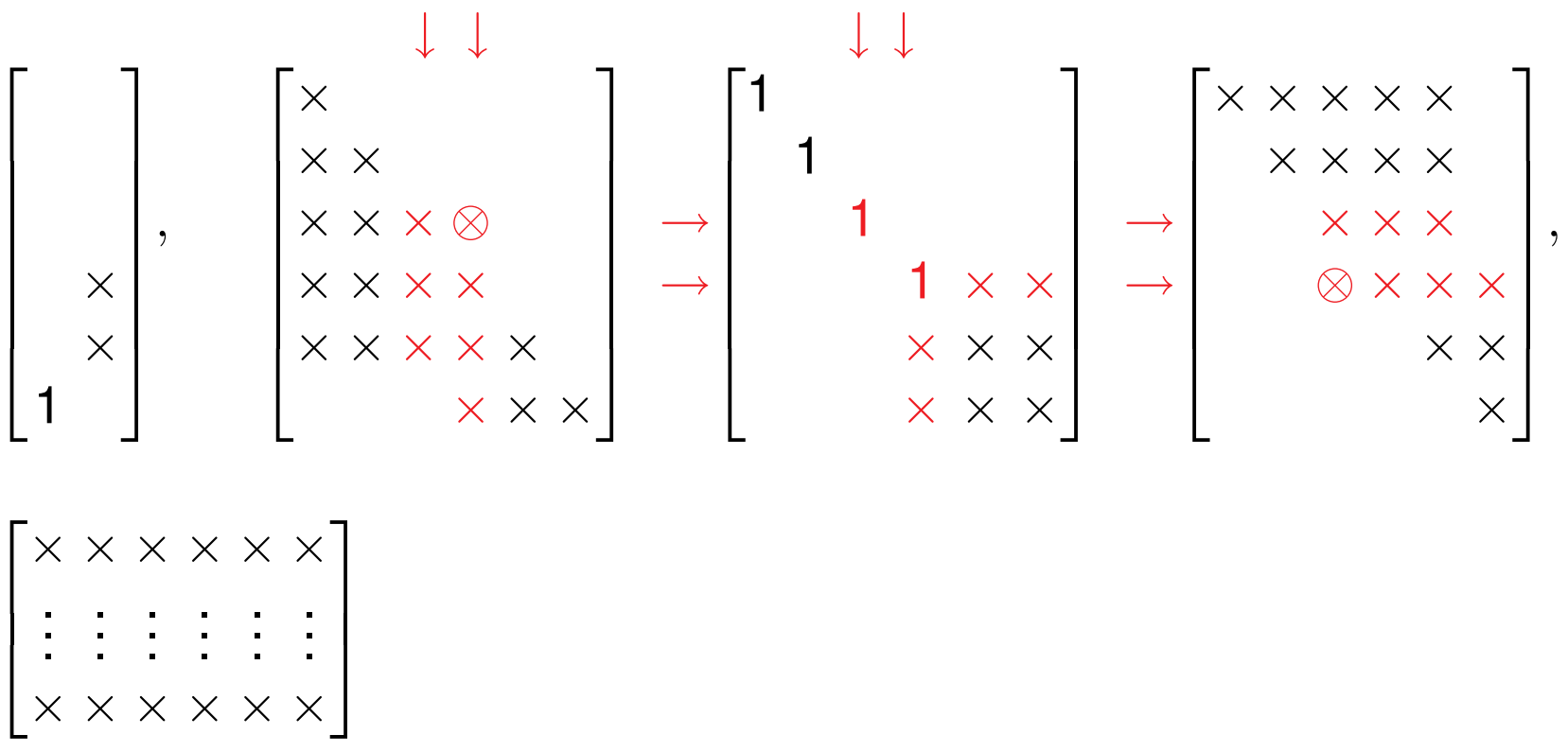
step 12



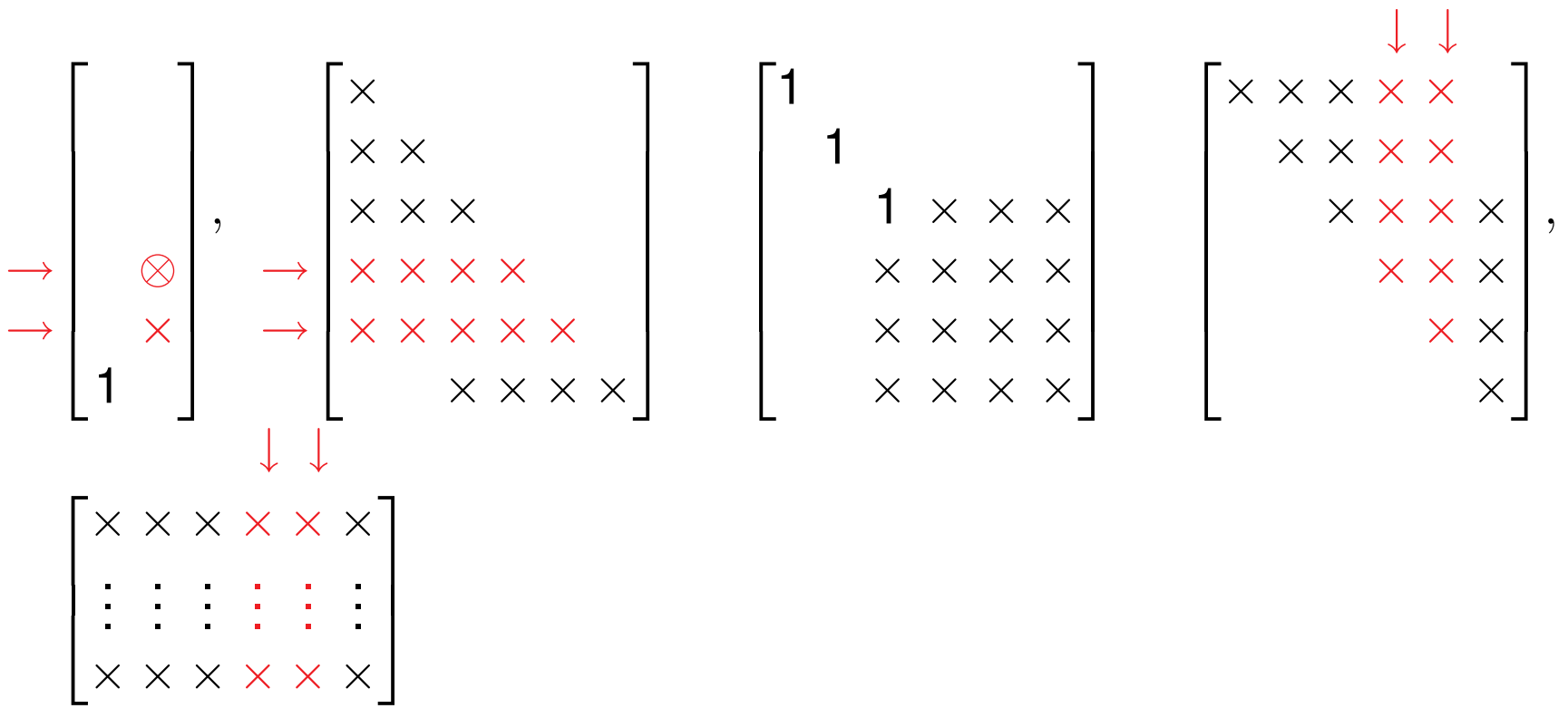
step 13



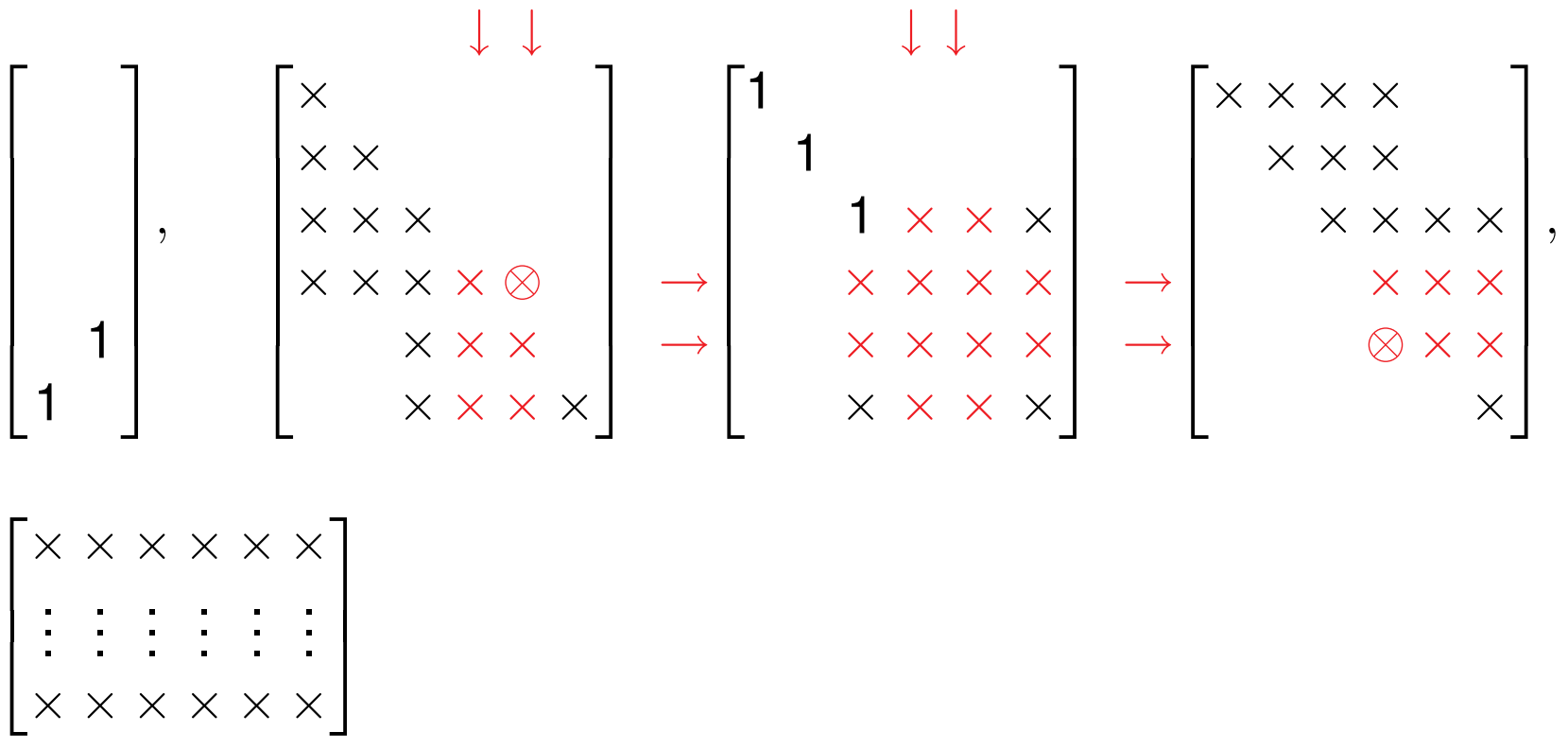
step 14



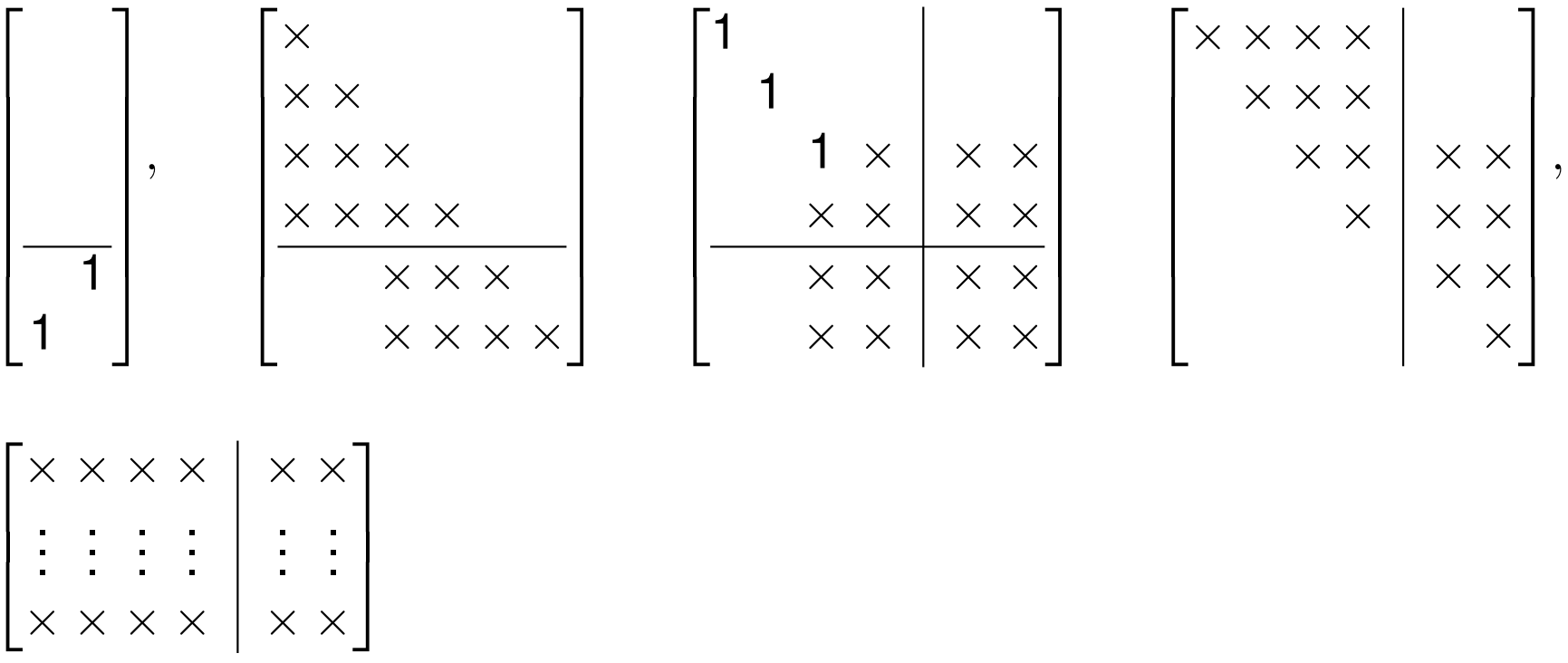
step 15



step 16



step 17



Downsizing algorithm

- Let H_v be orthogonal such that

$$H_v \mathbf{v} = \upsilon\, \mathbf{e}_{1,m}, \qquad \upsilon = \mp \|\mathbf{v}\|_2, \quad U_{n+1} = \begin{bmatrix} \mathbf{v}^T \\ V \end{bmatrix}.$$

Downsizing algorithm

- Let H_v be orthogonal such that

$$H_v \mathbf{v} = \upsilon\, \mathbf{e}_{1,m}, \qquad \upsilon = \mp \|\mathbf{v}\|_2, \quad U_{n+1} = \begin{bmatrix} \mathbf{v}^T \\ V \end{bmatrix}.$$

- Then

$$U_{n+1} H_v = \left[\begin{array}{c|ccc} \upsilon & 0 & \cdots & 0 \\ \hline & VH_v & & \end{array}\right].$$

Downsizing algorithm

- Let H_v be orthogonal such that

$$H_v \mathbf{v} = \upsilon\, \mathbf{e}_{1,m}, \qquad \upsilon = \mp \|\mathbf{v}\|_2, \quad U_{n+1} = \begin{bmatrix} \mathbf{v}^T \\ V \end{bmatrix}.$$

- Then

$$U_{n+1} H_v = \left[\begin{array}{c|ccc} \upsilon & 0 & \cdots & 0 \\ \hline & VH_v & & \end{array}\right].$$

- To retrieve the orthonormality of VH_v, it is sufficient to divide its first column of by $\sqrt{1-\upsilon^2}$ and therefore multiply the first column and row of M_m by the same quantity

Downsizing algorithm

- Let H_v be orthogonal such that

$$H_v \mathbf{v} = \upsilon\, \mathbf{e}_{1,m}, \qquad \upsilon = \mp \|\mathbf{v}\|_2, \quad U_{n+1} = \begin{bmatrix} \mathbf{v}^T \\ V \end{bmatrix}.$$

- Then

$$U_{n+1} H_v = \left[\begin{array}{c|ccc} \upsilon & 0 & \cdots & 0 \\ \hline & VH_v & & \end{array}\right].$$

- To retrieve the orthonormality of VH_v, it is sufficient to divide its first column of by $\sqrt{1-\upsilon^2}$ and therefore multiply the first column and row of M_m by the same quantity
- If the matrix M_m is factored as $L_m L_m^T$ this reduces to multiplying the first entry of L_m by $\sqrt{1-\upsilon^2}$

Downsizing algorithm

- Let H_v be orthogonal such that

$$H_v \mathbf{v} = \upsilon\, \mathbf{e}_{1,m}, \qquad \upsilon = \mp \|\mathbf{v}\|_2, \quad U_{n+1} = \begin{bmatrix} \mathbf{v}^T \\ V \end{bmatrix}.$$

- Then

$$U_{n+1} H_v = \left[\begin{array}{c|ccc} \upsilon & 0 & \cdots & 0 \\ \hline & VH_v & & \end{array}\right].$$

- To retrieve the orthonormality of VH_v, it is sufficient to divide its first column of by $\sqrt{1-\upsilon^2}$ and therefore multiply the first column and row of M_m by the same quantity
- If the matrix M_m is factored as $L_m L_m^T$ this reduces to multiplying the first entry of L_m by $\sqrt{1-\upsilon^2}$
- Any row of U_{n+1} can be chosen to be removed this way

Accuracy bounds

When there is no downsizing

$$\|K_n - A_n\|_F^2 \quad \leq \quad \eta_n := \sum_{i=m+1}^{\tilde{n}} \lambda_i^{(\tilde{n})^2} + \sum_{i=\tilde{n}+1}^{n} \delta_i^{(+)^2} + \sum_{i=\tilde{n}+1}^{n} \delta_i^{(-)^2},$$

$$\|K_n - A_n\|_2 \quad \leq \quad \zeta_n := \lambda_{m+1}^{(\tilde{n})} + \sum_{i=\tilde{n}+1}^{n} \max\left\{\delta_i^{(+)}, \delta_i^{(-)}\right\},$$

where

$$A_n := U_n M_m U_n^T, \quad \delta_i^{(+)} = \lambda_i^{(m+1)}, \quad \delta_i^{(-)} = \lambda_i^{(m+2)},$$

$$\|K_{\tilde{n}} - A_{\tilde{n}}\|_F^2 = \sum_{i=m+1}^{\tilde{n}} \lambda_i^{(\tilde{n})^2}, \quad \|K_{\tilde{n}} - A_{\tilde{n}}\|_2 = \lambda_{m+1}^{(\tilde{n})}.$$

Accuracy bounds

In relation to the original spectrum K_N one obtains the approximate bounds

$$\sum_{i=m+1}^{N} \lambda_i^2 \leq \|K_N - A_N\|_F^2 \lessapprox (N-m)\, \lambda_{m+1}^2.$$

and

$$\lambda_{m+1} \leq \|K_N - A_N\|_2 \lessapprox c\lambda_{m+1},$$

When donwsizing the matrix as well, there are no guaranteed bounds

Example 1 (no downsizing)

- The matrix considered in this example is a Kernel Matrix constructed from the Abalone benchmark data set `http://archive.ics.uci.edu/ml/support/Abalone`, with radial basis kernel function

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{100}\right),$$

Example 1 (no downsizing)

- The matrix considered in this example is a Kernel Matrix constructed from the Abalone benchmark data set `http://archive.ics.uci.edu/ml/support/Abalone`, with radial basis kernel function

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{100}\right),$$

- This data set has 4177 training instances

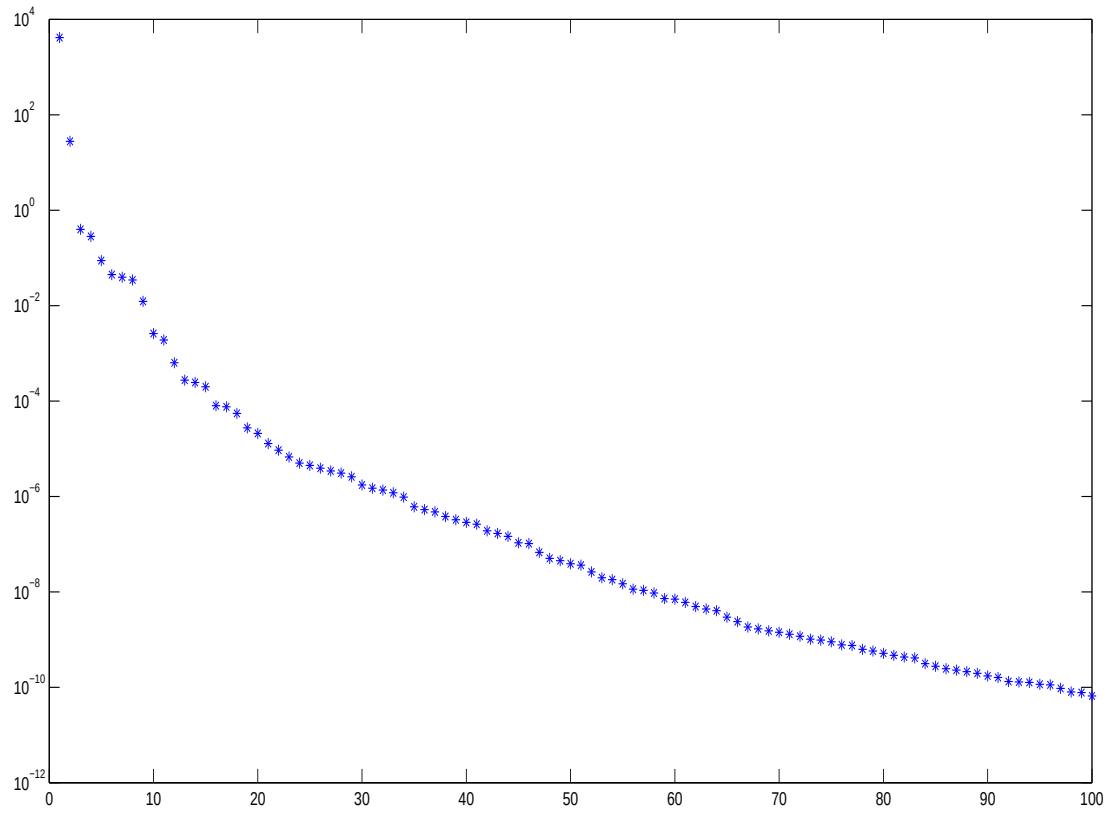


Figure: Distribution of the largest 100 eigenvalues of the Abalone matrix in logarithmic scale.

Table: Largest 9 eigenvalues of the Abalone matrix K_N (second column), of A_N obtained with updating with $\tilde{n} = 500$, $m = 9$ (third column) and with $\tilde{n} = 500$, $m = 20$ (fourth column), respectively.

λ_i	$\mu_i, \tilde{n} = 500, m = 9$	$\mu_i, \tilde{n} = 500, m = 20$
4.1483810825580 8e+3	4.1483810825581 2e+3	4.1483810825580 5e+3
2.7714246712392 6e+1	2.7714246712393 5e+1	2.7714246712390 8e+1
3.9694648635460 3e-1	3.9694648517433 9e-1	3.9694648635457 5e-1
2.8282783860038 4e-1	2.8282783824074 7e-1	2.8282783860179 4e-1
8.7635493872957 1e-2	8.7635489366471 4e-2	8.7635493873007 8e-2
4.4819176653871 7e-2	4.4819100229620 2e-2	4.4819176653746 2e-2
3.9500582114924 9e-2	3.9500503308202 8e-2	3.9500582114582 7e-2
3.4491659420644 3e-2	3.4491574649647 3e-2	3.4491659420696 3e-2
1.2275195012345 6e-2	1.2275093239400 3e-2	1.2275195011685 2e-2

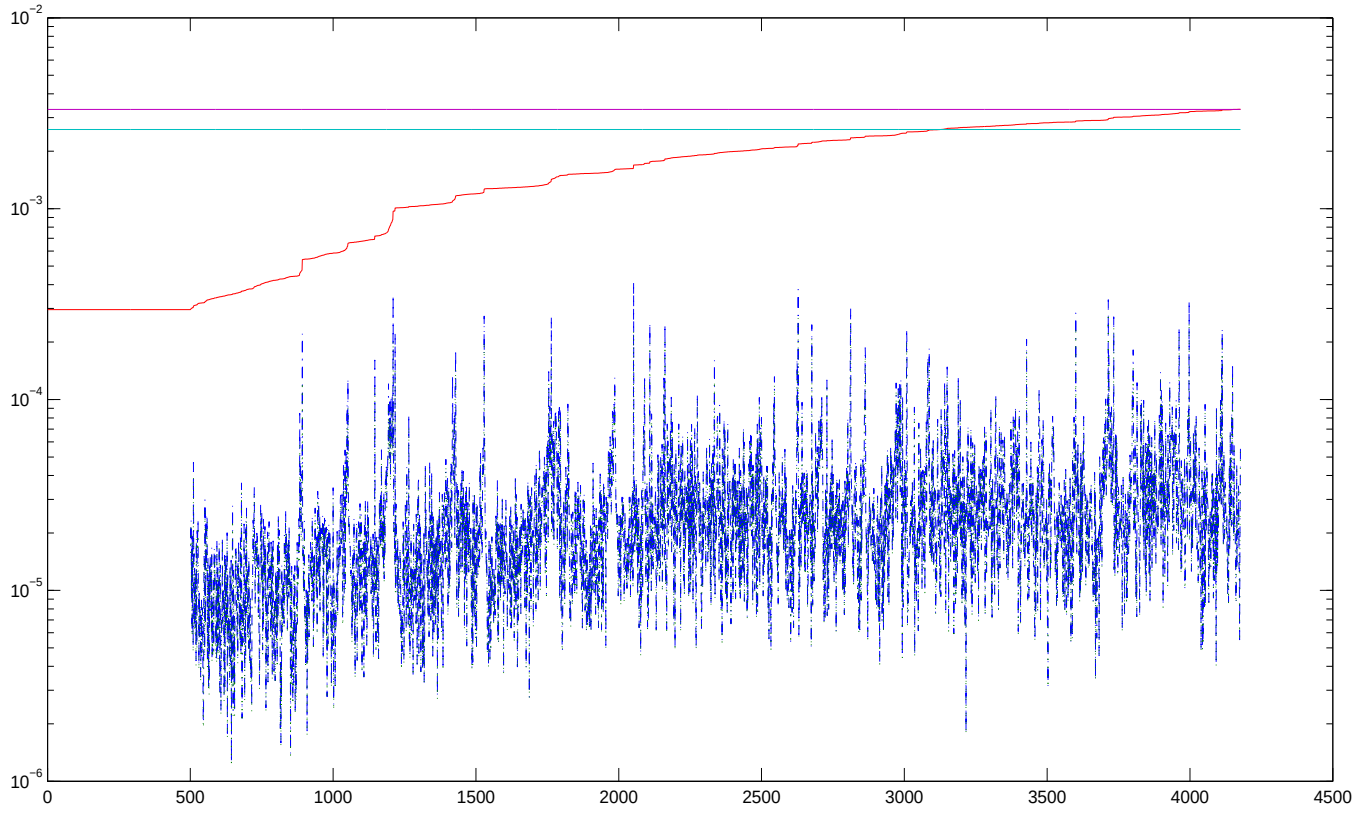


Figure: Plot of the sequences of $\delta_n^{(+)}$ (blue line), $\delta_n^{(-)}$ (green line), η_n (red solid line), λ_{m+1} (cyan solid line) and $\|K_N - A_N\|_F$ (magenta solid line).

Table: Angles between the eigenvectors corresponding to the largest 9 eigenvalues of K_N, computed by the function `eigs` of `matlab` and those computed by the proposed algorithm for $\tilde{n} = 500$, $m = 9$ (second column) and $\tilde{n} = 500$, $m = 20$ (third column).

i	$\angle(\mathbf{x}_i, \tilde{\mathbf{x}}_i)$	$\angle(\mathbf{x}_i, \hat{\mathbf{x}}_i)$
1	3.6500e-08	8.4294e-08
2	3.9425e-08	2.9802e-08
3	2.3774e-06	5.1619e-08
4	2.5086e-06	2.9802e-08
5	3.0084e-05	1.1151e-07
6	2.0446e-04	4.2147e-08
7	2.0213e-04	1.4901e-08
8	3.4670e-04	8.1617e-08
9	5.9886e-04	2.1073e-08

Example 2 (downsizing)

The following matrix has rank 3

$$F(i,j) = \sum_{k=1}^{3} \exp\left(-\frac{(i-\mu_k)^2 + (j-\mu_k)^2}{2\sigma_k}\right), \qquad i,j = 1,\ldots,100,$$

with

$$\mu = \begin{bmatrix} 4 & 18 & 76 \end{bmatrix}, \quad \sigma = \begin{bmatrix} 10 & 20 & 5 \end{bmatrix}.$$

Let $F = Q\Lambda Q^T$ be its spectral decomposition and let $\tilde{\Delta} \in \mathbb{R}^{100\times 100}$ be a matrix of random numbers generated by the `matlab` function `randn`, and define $\Delta = \tilde{\Delta}/\|\tilde{\Delta}\|_2$.
For this example, the considered SPD matrix is

$$K_N = F + \varepsilon\Delta\Delta^T, \qquad \varepsilon = 1.0e-5$$

Example 2

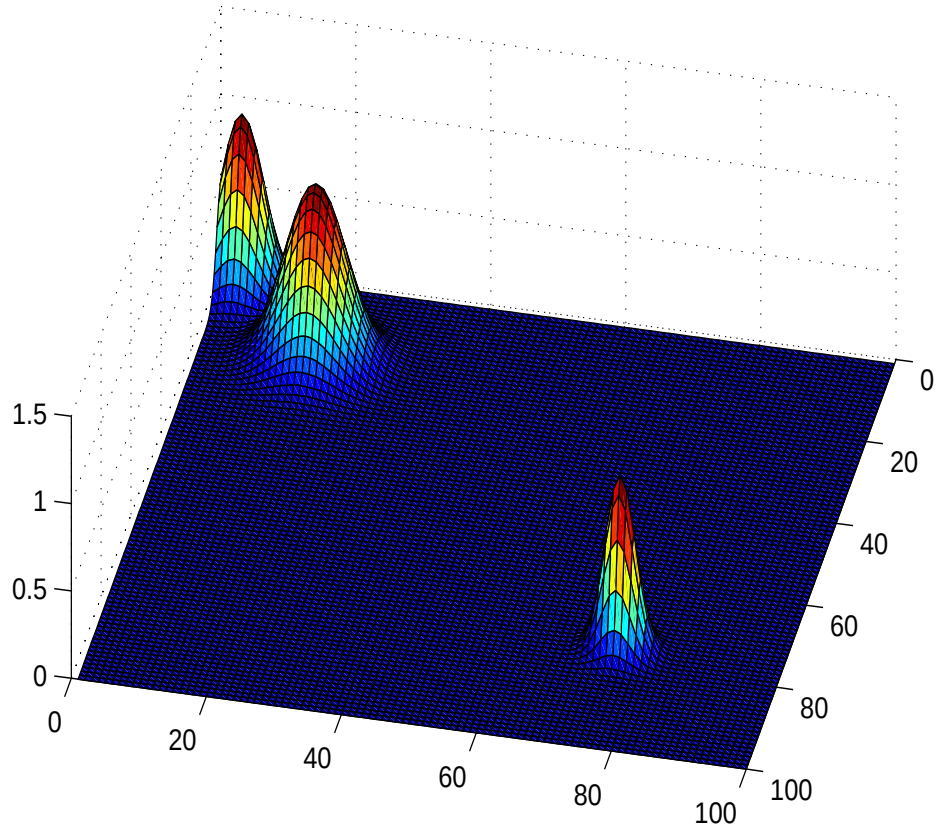


Figure: Graph of the size of the entries of the matrix K_N.

Example 2

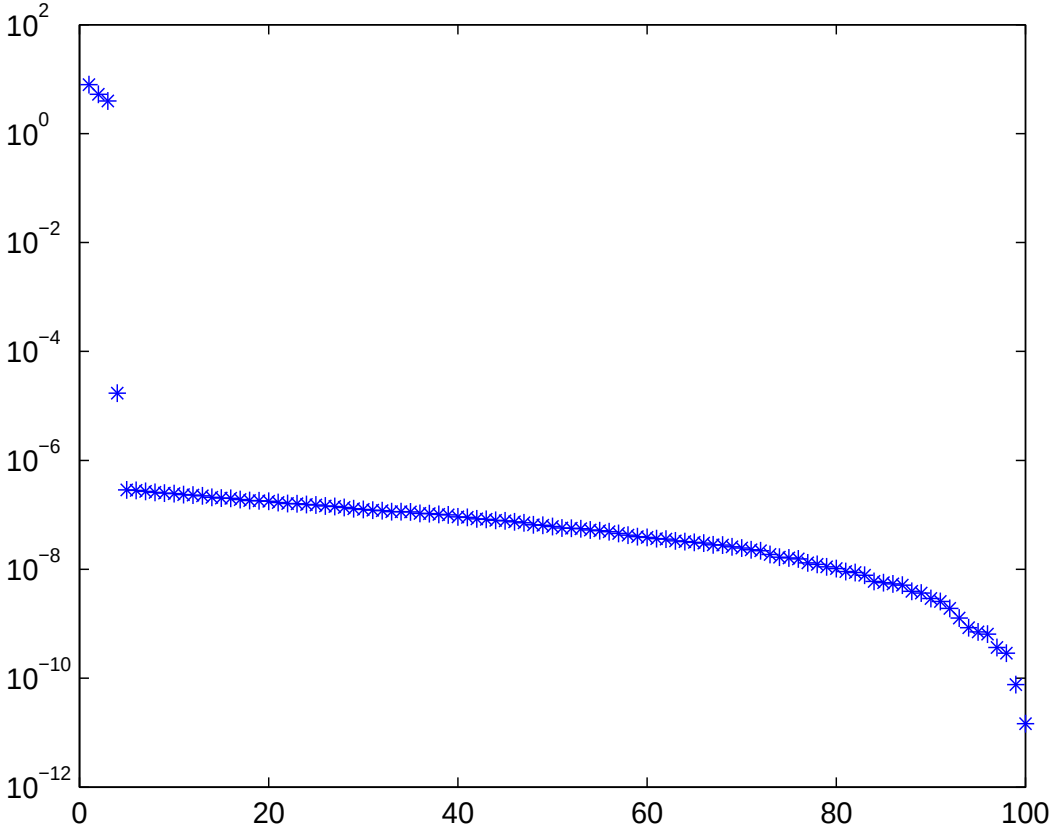


Figure: Distribution of the eigenvalues of the matrix K_N in logarithmic scale.

Example 2

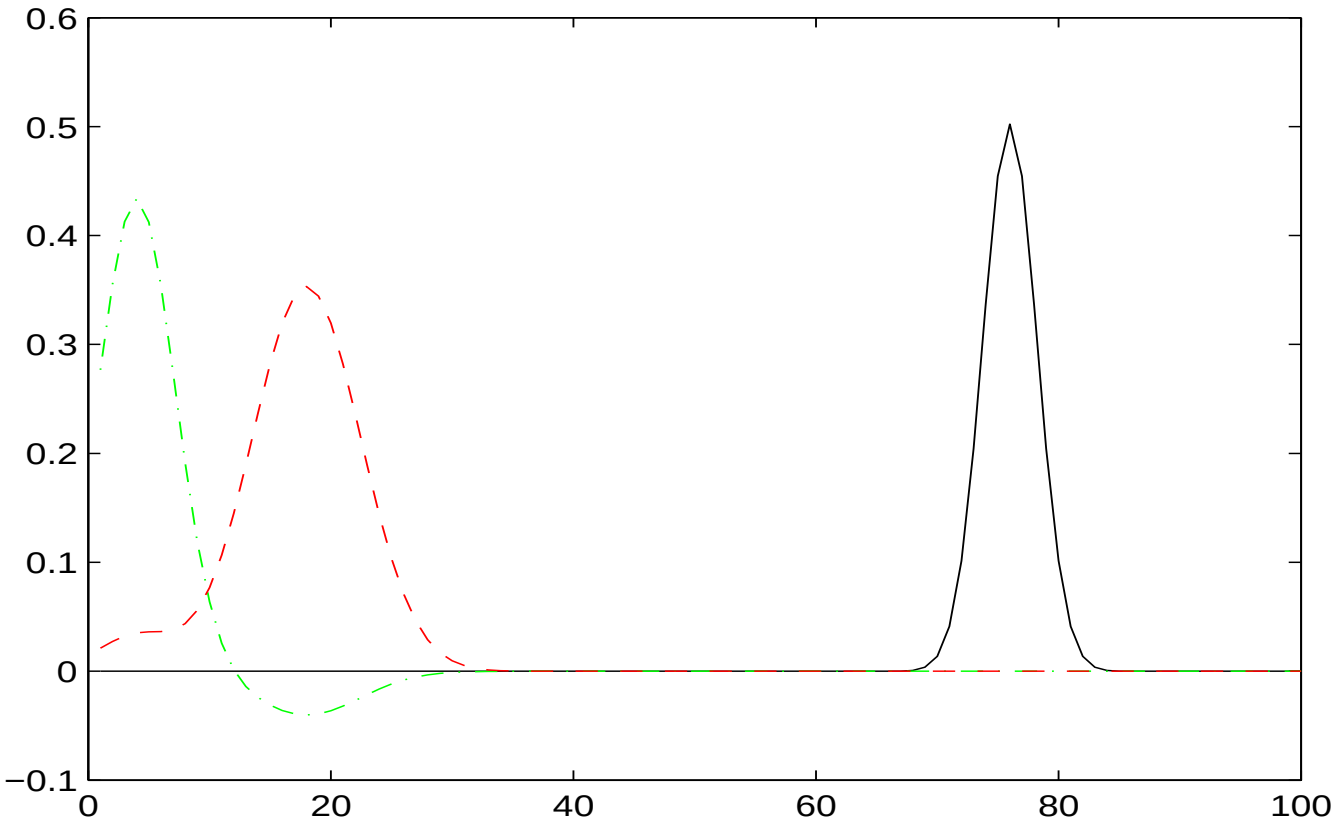


Figure: Plot of the three dominant eigenvectors of K_N

Example 2

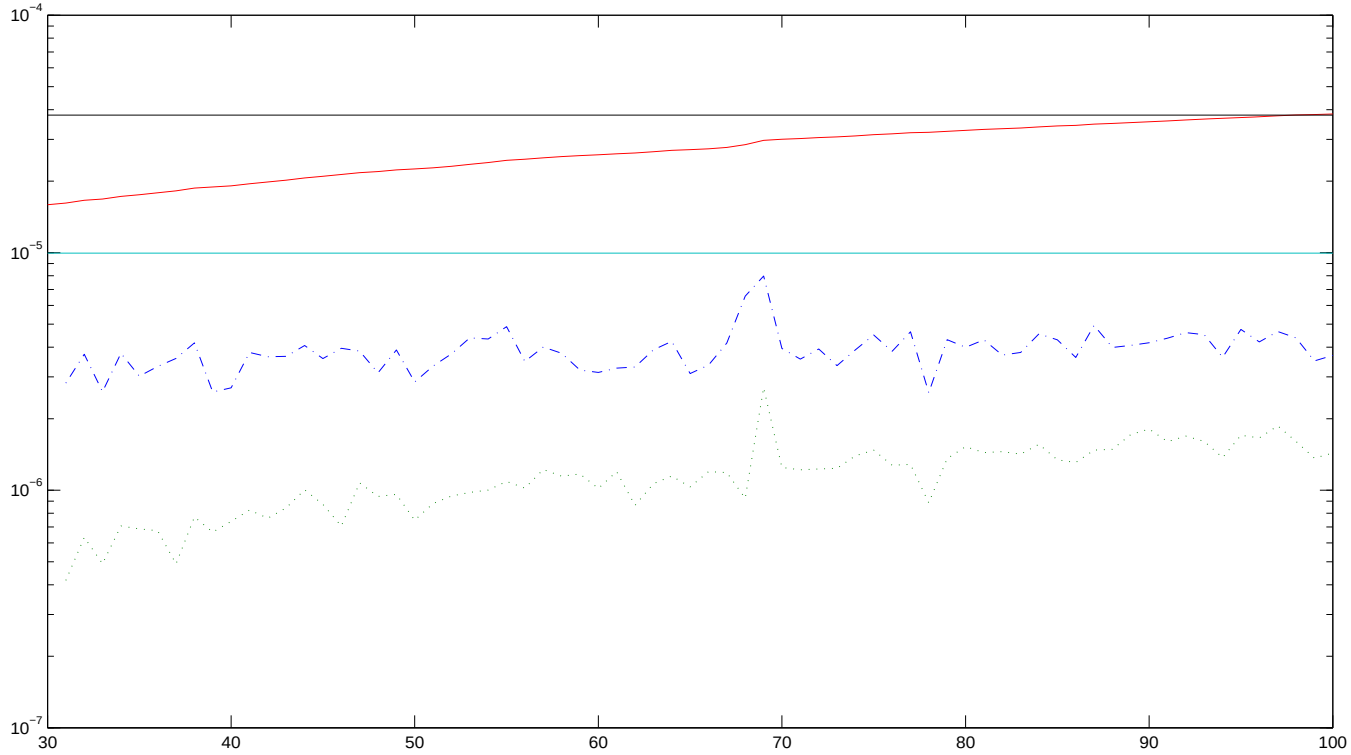


Figure: Plot of the sequences of $\delta_n^{(+)}$ (blue dash dotted line), $\delta_k^{(-)}$ (green dotted line), η_n (red solid line), λ_{m+1} (cyan solid line) and $\|K_N - A_N\|_F$ (black solid line).

Example 2

Table: Largest three eigenvalues of the matrix K_N (first column). Largest three eigenvalues computed with downdating procedure with minimal norm, with $m = 3$ and $\tilde{n} = 30, 40, 50$ (second, third and fourth column), respectively. Largest three largest eigenvalues of K_N computed with the " former" downdating procedure (fifth column).

λ_i	$\mu_i, \tilde{n} = 30$	$\mu_i, \tilde{n} = 40$	$\mu_i, \tilde{n} = 50$	$\mu_i, \tilde{n} = 50$
7.949478e0	7.375113e0	7.820407e0	7.947127e0	3.963329e0
5.261405e0	5.255163e0	5.260243e0	5.261384e0	5.417202e-6
3.963329e0	3.948244e0	3.963213e0	3.963329e0	4.824060e-6

Conclusions

- A fast algorithm for to compute incrementally the dominant eigenspace of a positive definite matrix

Conclusions

- A fast algorithm for to compute incrementally the dominant eigenspace of a positive definite matrix
- Improvement on Hoegaerts, L., De Lathauwer, L., Goethals I., Suykens, J.A.K., Vandewalle, J., & De Moor, B. *Efficiently updating and tracking the dominant kernel principal components.* Neural Networks, 20, 220–229, 2007.

Conclusions

- A fast algorithm for to compute incrementally the dominant eigenspace of a positive definite matrix
- Improvement on Hoegaerts, L., De Lathauwer, L., Goethals I., Suykens, J.A.K., Vandewalle, J., & De Moor, B. *Efficiently updating and tracking the dominant kernel principal components.* Neural Networks, 20, 220–229, 2007.
- The overall complexity of the incremental updating technique to compute an $N \times m$ basis matrix U_N for the dominant eigenspace of K_N, is reduced from $(m+4)N^2m + O(Nm^3)$ to $6N^2m + O(Nm^2)$.

Conclusions

- A fast algorithm for to compute incrementally the dominant eigenspace of a positive definite matrix
- Improvement on Hoegaerts, L., De Lathauwer, L., Goethals I., Suykens, J.A.K., Vandewalle, J., & De Moor, B. *Efficiently updating and tracking the dominant kernel principal components.* Neural Networks, 20, 220–229, 2007.
- The overall complexity of the incremental updating technique to compute an $N \times m$ basis matrix U_N for the dominant eigenspace of K_N, is reduced from $(m+4)N^2m + O(Nm^3)$ to $6N^2m + O(Nm^2)$.
- When using both incremental updating and downsizing to compute the dominant eigenspace of $\hat{K}_{\tilde{n}}$ (an $\tilde{n} \times \tilde{n}$ principal submatrix of K_N), the complexity is reduced $(12m+4)N\tilde{n}m + O(Nm^3)$ to $16N\tilde{n}m + O(Nm^2)$.

Conclusions

- A fast algorithm for to compute incrementally the dominant eigenspace of a positive definite matrix
- Improvement on Hoegaerts, L., De Lathauwer, L., Goethals I., Suykens, J.A.K., Vandewalle, J., & De Moor, B. *Efficiently updating and tracking the dominant kernel principal components.* Neural Networks, 20, 220–229, 2007.
- The overall complexity of the incremental updating technique to compute an $N \times m$ basis matrix U_N for the dominant eigenspace of K_N, is reduced from $(m+4)N^2m + O(Nm^3)$ to $6N^2m + O(Nm^2)$.
- When using both incremental updating and downsizing to compute the dominant eigenspace of $\hat{K}_{\tilde{n}}$ (an $\tilde{n} \times \tilde{n}$ principal submatrix of K_N), the complexity is reduced $(12m+4)N\tilde{n}m + O(Nm^3)$ to $16N\tilde{n}m + O(Nm^2)$.
- This is in both cases essentially a reduction by a factor m.

References

Gu, Eisenstat, *An efficient algorithm for computing a strong rank revealing QR factorization*, SIAM SCISC, 1996

Chahlaoui, Gallivan, Van Dooren, *An incremental method for computing dominant singular subspaces*, SIMAX, 2001

Hoegaerts, De Lathauwer, Goethals, Suykens, Vandewalle, De Moor, *Efficiently updating and tracking the dominant kernel principal components* Neural Networks, 2007.

Mastronardi, Tyrtishnikov, Van Dooren, *A fast algorithm for updating and downsizing the dominant kernel principal components*, SIMAX, 2010

Baker, Gallivan, Van Dooren, *Low-rank incrmenetal methods for computing dominant singular subspaces*, submitted, 2010

Ipsen, Van Dooren, *Polynomial Time Subset Selection Via Updating*, in preparation, 2010