OPERATION KOMOREBI

REALIZATION CALCULUS

PAPER VIII

Packet Operator Algorithms

Exact compilation, structured arithmetic, precision, and GPU realization

Thirty-two worked pages follow exact structured operators, small-core recompression, support and cancellation, error budgets, compilation, and fair performance contracts.

Komorebi P.

Visual guide • 9 October 2026

How to use this volume

Follow each small operator through its factors and formulas, then read the cited general argument in the traditional volume. The Lean companion records the separate finite formal contract.

Routes through the guide

The labels V1 through V32 identify worked pages independently of PDF pagination. Each source line cites the printed page of the general statement in the traditional volume. The integrated atlas uses these same labels while the traditional text keeps its own page numbers.

Route	Mathematical question	Visual pages
Representations	Which exact data represent the required operator?	1–4, 27
Structured transforms	Which conventions preserve the same map?	5–8
Low-rank algebra	What remains exact after products and recompression?	9–13, 29
Support and solves	Which interactions can occur or cancel?	14–18, 30
Precision and compilation	Which finite costs and errors does the optimizer use?	19–23, 31–32
Performance contracts	Which one-time costs and outputs are compared?	24–26
Evidence	What is checked, proved, or still open?	2, 28

Keep the represented object visible

A dense array, factored operator, and execution plan may describe the same map while returning different objects. Representation savings are exact modeled counts. Measured speed requires a separate experiment with matching output and timing contracts.

Preservation and evidence

All 32 worked pages and 30 editable vector plates remain supplied. The monograph changes typography, reading notes, and source locators. Model and checking code remain separate from the drawing coordinates. Fresh replay records distinguish exact examples, symbolic calculations, bounded enumerations, and rejected controls.

The saved v0.6.0 formal record contains 56 manuscript-proved finite items, 392 named declarations in 40 modules, and 66 construction audits. Nineteen declarations are reused foundation results. The ordered-SVD hypotheses, four classical entries, historical timing corpus, and three broad research targets retain their separate scope. No fresh Lean compiler result is asserted here.

Read this as: Choose a question, calculate its small model, and check the hypotheses, output contract, and evidence level of the corresponding general result.

V1 / THREE VIEWS OF ONE COMPUTATION

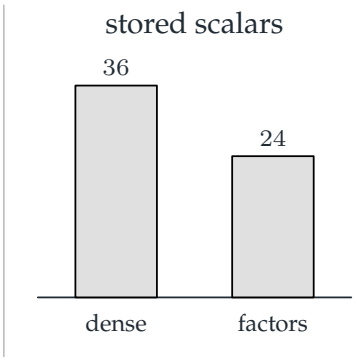
Compute the representation first

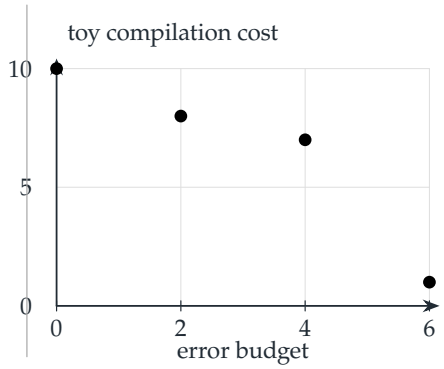
Traditional pp. 7, 10, 79 / Section 2, Section 3, Appendix F.

A dense array, a factored operator, and a compiled execution plan are different objects. They may denote the same linear map, but they need not store the same data, return the same output, or incur the same costs. This guide makes those distinctions visible through calculations rather than a pipeline of boxes.

A: 36 visible entries

4	2	1	1	1	1
2	4	1	1	1	1
1	1	4	2	1	1
1	1	2	4	1	1
1	1	1	1	4	2
1	1	1	1	2	4





Three models to carry through the paper

The first is a six-dimensional operator $A = D + uv^T$. Every entry of A is nonzero, yet three 2×2 diagonal blocks and two thin vectors suffice to apply it exactly. Pages V3, V9, V10, and V27 reuse this operator or its factors.

The second starts with redundant factors L and R. Actual range coordinates reduce a 4×3 product to a 2×3 core. Pages V11–V12 connect rank, singular tails, and the geometric reason a small-core optimum is also an ambient optimum.

The third is a seven-node candidate tree with exactly five prunings. Its budget optimum is not selected by any Lagrangian penalty. Pages V21–V23 let readers inspect every feasible choice and every changed subproblem.

How to use the guide

A source line beneath each title identifies the relevant final traditional result; the accompanying source crosswalk records its label and page. The preserved examples use rational, Gaussian-rational, symbolic, and finite-enumeration arithmetic. Portable checks use exact rational and Gaussian-rational arithmetic. The original SymPy suite is replayed for this edition, with its assertions and rejected controls recorded separately. Smooth drawing coordinates are illustrations of stated formulas, not additional proof evidence.

> **Read this as:** An exact representation can remove work before a device is chosen. It does not by itself supply a fast kernel, a good approximation, or a fair timing comparison.

V2 / A READER'S ATLAS

The objects and their contracts

Traditional pp. 7, 9, 26, 30, 36 / Definition 2.1, Definition 2.2, Definition 2.7, Section 8, Section 9, Section 11.

The same letter can refer to a mathematical operator or to a particular stored representation. Keep those levels separate when reading an algorithm or a benchmark row.

Symbol	Object	Question to ask
P_α	Coordinate projector	Which input or output coordinates does it select?
$A_{\beta\alpha}$	A block map	What are its source and target dimensions?
E	Active block addresses	Is this exact support, or only a structural upper bound?
$D + UV^*$	Factored operator	Is the desired output a vector, factors, or a dense array?
Q_L, G, Q_R	Range factors and core	Are the outer columns orthonormal, with exact reconstruction?
E_j	An error operator	Is it a factor perturbation or an additive error after a product?
s_e, ϵ_e	Sensitivity and tolerance	Is the budget input-dependent or uniform over unit inputs?
$(\kappa_v, \epsilon_v, \sigma_v)$	Candidate-tree data	Do costs, errors, and split overheads really add?
$F_v(b)$	A budget table	Does b mean exact weight, or an upper bound?
R	Reuse count	Which one-time costs have been included?

Conventions used in the examples

Matrices act on column vectors. A block $A_{\beta\alpha}$ goes from source α to target β. Products act from right to left. Real examples use transpose; complex examples use adjoint unless a vectorization identity explicitly requires transpose. Column-major vectorization stacks the columns of a matrix.

An arithmetic count follows the paper's modeled multiply/add convention. It is not a count of executed machine instructions. The tiny scheduling and amortization examples use declared synthetic units, not measured GPU times. The original benchmark corpus remains historical data in the preserved source tree.

Evidence levels

The original theorem ledger records its inherited finite formal scope. The new exact checks validate these particular examples. A plot samples a formula. A historical timing describes its recorded case and machine. None of these evidence types can silently substitute for another.

> **Read this as:** Before interpreting a number, identify its object, norm, output contract, and timing scope. Most apparent contradictions disappear once those four fields are explicit.

V3 / THE RUNNING SIX-DIMENSIONAL OPERATOR

A dense-looking matrix, one scalar coupling

Traditional pp. 7, 11 / Theorem 2.3, Theorem 3.4.

Let D have three identical blocks $\begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$ and put $u = v = \mathbf{1} \in \mathbb{Q}^6$. Then $A = D + uv^T$ has diagonal entries four, within-block off-diagonal entries two, and every cross-block entry one. All 36 entries are nonzero.

$$x = (1, -1, 2, 0, -2, 3)^T, \qquad v^T x = 3, \qquad Ax = Dx + 3u.$$

x		Dx		$u(v^T x)$		Ax
1		2		3		5
−1		−2		3		1
2		6		3		9
0	+	2	+	3	=	5
−2		−3		3		0
3		7		3		10

$v^T x = 3$ is shared by all six output coordinates.

What was avoided?

The global part does not require 36 independent coefficients. It depends on the input through one scalar, $v^T x$, which is broadcast through u. The diagonal part is computed by three separate local matrix products. Nothing is approximated and no channel is discarded.

The sixth output is $7 + 3 = 10$, while the fifth is $-3 + 3 = 0$. A zero result can arise through cancellation even when the corresponding row contains no zero coefficient. The output vector is therefore not a support map of the input matrix.

What has not been established?

This computation proves neither that the supplied factors are the cheapest representation nor that six small tasks outperform one dense kernel. For example, the repeated diagonal blocks and all-one vectors admit further special handling. The stated BDLR model deliberately treats the blocks as dense and the factors as stored vectors so that the paper's general formulas apply literally.

The checker constructs the complete matrix independently, multiplies it by x, and compares all six entries with the factored route. The same input returns in the product calculation on V27.

> **Read this as:** A matrix can have full entrywise support and still admit a small exact representation. Entry sparsity and algebraic compressibility are different properties.

V4 / ARITHMETIC, STORAGE, AND CAPACITY

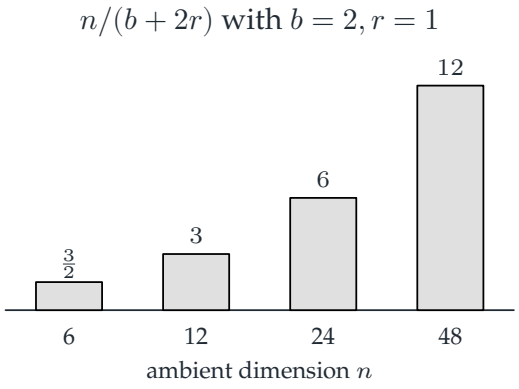
Work removed is not time measured

Traditional pp. 10, 11, 12, 13 / Theorem 3.1, Corollary 3.2, Theorem 3.4, Corollary 3.9, Proposition 4.1.

For q right-hand sides, the paper models dense application by $2n^2q$ operations. A block-diagonal-plus-rank-r representation with equal block width b uses

$$F = 2qn(b + 2r), \qquad M = n(b + 2r) \text{ stored scalars.}$$

These formulas count the dense block products and two thin products. They follow the manuscript convention rather than adding every scalar accumulation or index operation.



Six-dimensional running operator

dense: 36 scalars, $72q$ work

BDLR: 24 scalars, $48q$ work

ratio: $3/2$, not a timing result

For the running example, $n = 6, b = 2, r = 1$. The model reduces storage from 36 to 24 scalars and application work from $72q$ to $48q$. For fixed b, r, the ratio $n/(b + 2r)$ grows with n. If the rank instead grows proportionally with dimension, that apparent advantage need not grow.

Memory is another constraint

With scalar size s, the representation alone requires sM bytes. Inputs, outputs, workspace, allocator overhead, and stored indices are additional. A matrix-only capacity inequality is not a complete device-memory fit test.

For a block band of p packets and half-bandwidth w, the exact active-block count is

$$p(2w + 1) - w(w + 1).$$

The subtraction removes overcounted edge diagonals. The checker enumerates all addresses for $1 \le p \le 6$ and every admissible w, including $w = 0$ and $w = p - 1$.

> **Read this as:** An arithmetic or storage ratio explains a representation advantage. Kernel launch cost, traffic, utilization, and conversion determine whether that advantage becomes a time advantage.

V5 / CIRCULANT MULTIPLICATION

Four coefficients, four exact frequencies

Traditional p. 14 / Theorem 4.3.

Take $c = (2, 1, 0, -1)^T$ and define $C(c)_{jk} = c_{j-k \bmod 4}$. The unitary Fourier matrix is $F_{jk} = \mathrm{i}^{-jk}/2$. Its normalization is visible in the exact diagonalization:

$$C(c) = F^* \operatorname{diag}(2Fc) F, \qquad 2Fc = (2, 2 - 2\mathrm{i}, 2, 2 + 2\mathrm{i})^T.$$

$C(c)$

2	−1	0	1
1	2	−1	0
0	1	2	−1
−1	0	1	2

$\sqrt{4}\,Fc$

2
$2 - 2i$
2
$2 + 2i$

$$F_{jk} = \tfrac{1}{2}\,\mathrm{i}^{-jk}$$

$$C = F^* \operatorname{diag}(\sqrt{4}Fc) F$$

$\sqrt{4} = 2$ is essential.

A unitary transform changes the normalization of the convolution eigenvalues.

Follow an actual input

For $x = (1, 2, -1, 0)^T$, dense multiplication gives $(0, 6, 0, -2)^T$. Transforming x, multiplying by the four displayed eigenvalues, and applying F^* gives exactly the same vector. Every entry belongs to $\mathbb{Q}(\mathrm{i})$, so the check needs no approximate trigonometric values.

There is a common normalization trap. With an unnormalized discrete transform, the convolution multiplier has one convention. With the unitary transform used here, the eigenvalue vector is $\sqrt{n}\,Fc$, not Fc. Dropping the factor two reconstructs $C/2$ rather than C in this example. The invalid control deliberately tries that change and rejects it.

Where the fast algorithm enters

The identity specifies the represented operator. An FFT is an additional factorization that evaluates the transforms efficiently. The algebraic diagonalization does not certify a particular floating-point FFT, data layout, or launch policy. For a tiny four-point example, direct multiplication may be the simpler execution path even though the frequency representation is exact.

> **Read this as:** Transform coordinates change both the storage pattern and the normalization conventions. Verify the inverse transform and convolution scale before interpreting a frequency-domain product.

V6 / TOEPLITZ EMBEDDING WITHOUT WRAPAROUND

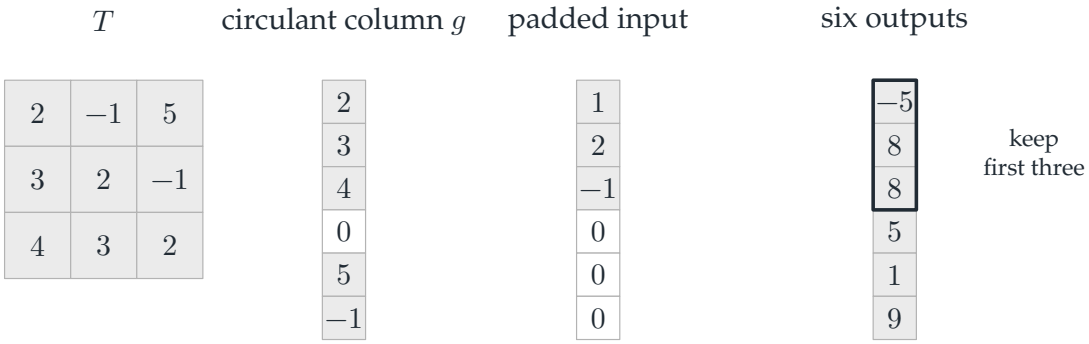
Padding is part of the theorem

Traditional p. 15 / Corollary 4.4.

Consider

$$T = \begin{pmatrix} 2 & -1 & 5 \\ 3 & 2 & -1 \\ 4 & 3 & 2 \end{pmatrix}, \qquad x = (1, 2, -1)^T.$$

The diagonal pattern is Toeplitz, not circulant. To use a circular convolution, embed it into dimension six with first column $g = (2, 3, 4, 0, 5, -1)^T$ and pad the input with three zeros.



Padding is part of the operator identity, not optional workspace.

Read the wrapped indices

For output index $j < 3$, an input index $k \le j$ uses the first-column coefficient c_{j-k}. An input index $k > j$ wraps to the stored first-row coefficient r_{k-j}. The padded input positions contribute nothing. Keeping the first three outputs therefore gives

$$Tx = (-5, 8, 8)^T.$$

The remaining circular outputs are workspace consequences of the embedding, not additional components of Tx.

A concrete failure mode

Using only the first column $(2, 3, 4)$ as a three-dimensional circulant changes the upper-triangular coefficients. That is a different matrix. The checker compares its output with Tx and rejects the unpadded shortcut.

The middle zero in g is not a discarded Toeplitz coefficient: it separates the lower and reversed upper diagonals in the chosen $2n$ embedding. Other embeddings require their own index identities. It is not sufficient for an array merely to have the right length or to produce a plausible plot.

> **Read this as:** Zero padding prevents the wrong circular interactions. It is an algebraic part of the operator representation, not an optional performance tweak.

V7 / KRONECKER ACTION AND ARRAY ORDER

A transpose, not an adjoint

Traditional p. 15 / Theorem 4.5, Corollary 4.6.

With column-major vectorization, the tensor identity is

$$(A \otimes B)\,\mathrm{vec}(X) = \mathrm{vec}(BXA^T).$$

Here the transpose records index order, not a Hermitian inner product. Use the explicitly complex matrix $A = \begin{pmatrix} 1 & \mathrm{i} \\ 2 & -1 \end{pmatrix}$, together with $B = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix}$ and $X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$.

X

1	2
3	4

$\mathrm{vec}(X)$

1
3
2
4

BXA^T

$5 + 8i$	2
$9 + 12i$	6

A^T not A^*

$$(A \otimes B)\,\mathrm{vec}(X) = \mathrm{vec}(BXA^T)$$

Check the coordinates

The stacked input is $(1, 3, 2, 4)^T$. Direct tensor multiplication agrees with stacking the columns of

$$BXA^T = \begin{pmatrix} 5 + 8\mathrm{i} & 2 \\ 9 + 12\mathrm{i} & 6 \end{pmatrix}.$$

Replacing A^T with A^* changes the signs of the imaginary terms. A real-only test would conceal that error, which is why the worked model uses an imaginary coefficient.

Why modewise contraction matters

For A of order m and B of order n, the explicit tensor matrix stores m^2n^2 entries. Its factors store $m^2 + n^2$. Applying through BXA^T costs $2n^2m+2nm^2$ modeled operations, rather than forming the full tensor matrix first.

A different flattening convention has a different coordinate formula. The statement is not independent of how an array is reshaped. In a software implementation, shape, stride, and flattening order belong to the correctness contract as well as the cost model.

> **Read this as:** Tensor structure gives exact savings only when the address convention is preserved. A silent transpose, conjugation, or reshape change implements a different map.

V8 / WALSH RECURSION IN EIGHT COORDINATES

See every butterfly value

Traditional p. 16 / Theorem 4.7.

The unnormalized Walsh matrix satisfies $H_1 = [1]$ and

$$H_{2n} = \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix}, \qquad H_n H_n^T = nI.$$

Apply the three add/subtract stages to $(1,2,3,4,5,6,7,8)^T$. The full intermediate ledger is displayed below.

input	1	2	3	4	5	6	7	8
stage 1	3	-1	7	-1	11	-1	15	-1
stage 2	10	-2	-4	0	26	-2	-4	0
stage 3	36	-4	-8	0	-16	0	0	0

$8 \log_2 8 = 24$ additions/subtractions; $H_8 H_8^T = 8I$.

Arithmetic can be counted exactly

Each stage visits eight coordinates, producing one sum or difference per coordinate. Three stages therefore use exactly 24 additions or subtractions. No multiplication by a dense 8×8 matrix is necessary.

The result $(36, -4, -8, 0, -16, 0, 0, 0)^T$ still reconstructs the input: the inverse is $H_8^T/8$. Because the displayed transform is unnormalized, its squared output norm is eight times the squared input norm. The normalized transform $H_8/\sqrt{8}$ would instead be orthogonal.

What the zeros mean

The zero coefficients arise from exact cancellation among the input values. They do not imply that the transform's defining matrix is sparse. The butterfly structure, not the final coefficient support, explains the arithmetic reduction.

The checker verifies the recursion, orthogonality, dense-versus-butterfly output, and operation count for sizes one, two, four, eight, and sixteen. These bounded tests exercise the implementation and boundary cases. The general recursion theorem remains the manuscript's mathematical statement.

> **Read this as:** Factorization can turn a dense matrix into a short exact arithmetic circuit. Keep its normalization separate from its recursive evaluation rule.

V9 / CLOSED LOW-RANK PRODUCT ALGEBRA

The cross term has somewhere to go

Traditional pp. 16, 17 / Theorem 5.1, Corollary 5.2.

Let $A = D_A + u_a v_a^T$ be the running operator. Take $D_B = \mathrm{diag}(1,2,3,4,5,6)$, $u_b = (1,0,1,0,1,0)^T$, and $v_b = (0,1,0,1,0,1)^T$. Define $B = D_B + u_b v_b^T$.

$$U_C = [D_A u_b + u_a(v_a^T u_b) \;\; u_a], \qquad V_C = [v_b \;\; D_B^T v_a].$$

U_C

6	1
4	1
6	1
4	1
6	1
4	1

V_C

0	1
1	2
0	3
1	4
0	5
1	6

$$AB = D_A D_B + U_C V_C^T$$

$$U_A(V_A^T U_B)V_B^T$$

must be retained

Two input rank-one corrections produce at most rank two, not three.

Why there are only two columns

Expanding AB gives three correction terms, but two share the right factor v_b^T:

$$D_A u_b v_b^T + u_a(v_a^T u_b)v_b^T = (D_A u_b + u_a(v_a^T u_b))v_b^T.$$

The remaining term is $u_a v_a^T D_B$. Thus $AB = D_A D_B + U_C V_C^T$ with at most two correction directions. Counting the three expanded summands as independent ranks would overestimate this representation.

The scalar $v_a^T u_b$ is three. Removing the quadratic term would therefore remove $3u_a v_b^T$, which is nonzero. The checker compares the complete product to the factor construction and rejects a calculation that omits this contribution.

Before and after recompression

A rank bound on stored factors is not always the actual matrix rank. Repeated products may introduce dependent columns. Recompression can remove those dependencies exactly, or remove small singular directions approximately under an explicit error budget.

> **Read this as:** Closure of a representation class is an algebraic statement about all product terms. It is not permission to drop an interaction merely because it is second order in the factors.

V10 / WOODBURY WITH BOTH INVERTIBILITY CHECKS

A large inverse, a small denominator

Traditional p. 18 / Theorem 5.3, Corollary 5.4.

For the running operator, each diagonal block has inverse $\frac{1}{8}\begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix}$. Hence $D^{-1}u = u/4$ and

$$S = 1 + u^T D^{-1} u = \frac{5}{2}.$$

The Woodbury identity reduces the entire inverse to

$$A^{-1} = D^{-1} - D^{-1} u S^{-1} u^T D^{-1} = D^{-1} - \frac{1}{40} u u^T.$$

one block of D^{-1}

$\frac{3}{8}$	$-\frac{1}{8}$
$-\frac{1}{8}$	$\frac{3}{8}$

$$D^{-1}u = \tfrac{1}{4}u$$

$$S = 1 + u^T D^{-1} u = \tfrac{5}{2}$$

$$A^{-1} = D^{-1} - \tfrac{1}{40} u u^T$$

$I - e_1 e_1^T$

0	0
0	1

invertible D
singular update

Two hypotheses, not one

Invertibility of D alone does not suffice. For $D = I_2$, $u = e_1$, and $v = -e_1$, the update is $\mathrm{diag}(0, 1)$. The small matrix S is zero, exactly where the correction formula would require an invalid division.

For the running model, the checker multiplies the proposed inverse by A in both orders and obtains the identity. It also checks the determinant relation $\det A = \det D \det S$. These tests verify the actual factors and sign of the correction, not just a numerical solve residual.

Conditioning survives the reduction

A small S can still be ill-conditioned. Reducing its dimension does not bound $\|S^{-1}\|$. The manuscript's inverse estimate keeps that term explicit, along with the norms of $D^{-1}u$ and $v^T D^{-1}$.

> **Read this as:** The hidden coupling has not vanished. Its solvability and sensitivity are concentrated in a smaller operator that still has to be invertible and numerically controlled.

V11 / ACTUAL RANGE COORDINATES

Stored width is not intrinsic rank

Traditional pp. 18, 79 / Theorem 5.6, Section F.1.

Choose the isometry

$$Q_L = \begin{pmatrix} 3/5 & 0 \\ 4/5 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad R_L = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad R_R = \begin{pmatrix} 3 & 0 & 2 \\ 0 & 2 & 0 \\ 1 & 0 & -1 \end{pmatrix}.$$

Set $L = Q_L R_L$ and $R = R_R$. The two equal columns of L do not create a third range direction. In contrast, R has rank three.

L: rank 2

$\frac{3}{5}$	0	$\frac{3}{5}$
$\frac{4}{5}$	0	$\frac{4}{5}$
0	1	0
0	0	0

R: rank 3

3	0	2
0	2	0
1	0	−1

G: 2×3

5	0	0
0	2	0

$$LR^T = Q_L G$$

The stored width is three; the left range has only two independent directions.

The actual core is

$$G = R_L R_R^T = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}, \qquad LR^T = Q_L G.$$

Its dimensions are the two actual input ranks, two and three. This is not inferred by merely discarding a column with a small norm. The range factor Q_L has orthonormal columns and reconstructs L exactly.

The preserved formal boundary

The sixth release constructs orthonormal bases of the actual column spaces. It then assumes exact ordered thin-SVD data for the core, as the manuscript instructs the reader to take an SVD. It does not assume an optimality or tail-error conclusion. Nor does it certify a floating-point QR or SVD implementation simply because those routines use similar notation.

This example has a diagonal rectangular core, so its singular directions are explicit. No iterative numerical factorization is needed to identify the two nonzero singular values, five and two.

> **Read this as:** Compactness is a statement about actual range spaces. The number of stored factor columns can exceed the dimensions that matter for recompression.

V12 / COMPRESSION, KERNEL WITNESSES, AND TAILS

Why a core optimum is global

Traditional pp. 18, 80, 81 / Theorem 5.6, Section F.2, Section F.3, Section F.4.

The product from the previous page is

$$A_0 = LR^T = \begin{pmatrix} 3 & 0 & 0 \\ 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Keeping only its first singular direction gives a rank-one approximation with spectral and Frobenius error exactly two. Keeping none gives errors five and $\sqrt{29}$; keeping both gives zero.

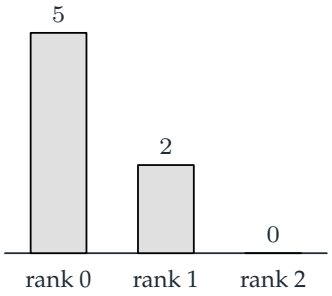
LR^T

3	0	0
4	0	0
0	2	0
0	0	0

rank-one output

3	0	0
4	0	0
0	0	0
0	0	0

spectral error



Frobenius errors: $\sqrt{29},\ 2,\ 0$. The rank-one choice loses only the second mode.

Why an outside competitor cannot do better

For isometric outer factors, compression $C(B) = Q_L^* B Q_R$ cannot increase rank or either matrix norm. Every ambient rank-k competitor therefore supplies an admissible core competitor. The lifted core residual has exactly the core residual norm. These two facts transfer a core optimum to the entire ambient matrix space, not just to competitors already lying in the supplied ranges.

For a d-dimensional diagonal core and $0 \le k < d$, the spectral lower bound has a concrete witness. A rank-at-most-k competitor has a nonzero kernel vector in the first $k+1$ coordinates. On that vector the diagonal matrix acts with norm at least σ_{k+1} times the vector norm. The competitor cannot cancel that direction.

The Frobenius proof uses an orthonormal basis of the competitor's kernel. Its row weights lie in $[0, 1]$ and sum to the kernel dimension. Weighting the decreasing squared singular values forces at least the discarded tail. The two norm arguments are distinct; one is not a graph of the other.

> **Read this as:** Optimality is not inferred from the appearance of the singular-value plot. It follows from compression and a lower bound valid against every rank-bounded competitor.

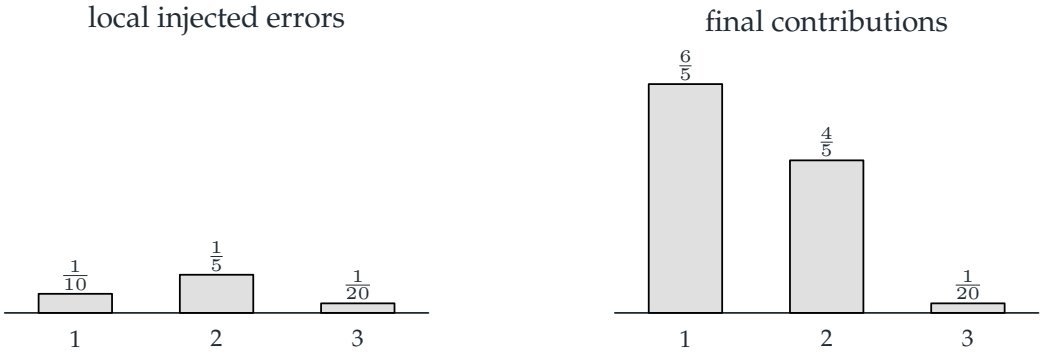
V13 / FACTOR PERTURBATION VERSUS RECOMPRESSION

Errors enter at different places

Traditional pp. 19, 20 / Theorem 5.8, Theorem 5.9.

Let the exact scalar factors be $(2, 3, 4)$ and start from $p_0 = 1$. If additive errors $(1/10, 1/5, 1/20)$ enter *after* each partial multiplication, then

$$\hat{p}_j = a_j \hat{p}_{j-1} + e_j, \qquad \hat{p}_3 - p_3 = 12e_1 + 4e_2 + e_3 = \frac{41}{20}.$$



$\hat{p}_j = a_j \hat{p}_{j-1} + e_j$, with $(a_1, a_2, a_3) = (2, 3, 4)$.

The same local numbers, a different model

If instead the factors themselves are perturbed, the computed product is $(4+e_3)(3+e_2)(2+e_1)$. Its error is $402/125$, not $41/20$. The factor-perturbation telescoping formula contains already-perturbed later factors; the additive recompression formula contains the original later factors.

The visual comparison uses the additive model. The first error is multiplied by both remaining factors. The last error is not amplified at all. Summing only the local error magnitudes gives $7/20$, which is smaller than the actual final error and therefore not a valid bound here.

In a matrix chain

The corresponding formulas replace scalar magnitudes with operator norms and respect multiplication order. A bound can be conservative when contributions cancel, but it must not omit possible amplification. Singular-tail errors from a recompression step become useful only after they are inserted into the correct global error model.

> **Read this as:** An error tolerance needs an entry point. A bound for perturbing factors and a bound for compressing partial products are not interchangeable.

V14 / SUPPORT IS AN INCLUSION

Two routes can cancel exactly

Traditional pp. 8, 21 / Theorem 2.5, Theorem 6.1, Corollary 6.3.

Let K have transfers $0 \to 1$, $0 \to 2$, $1 \to 3$, and $2 \to 3$, with coefficients $1, 1, 1, -1$. There are two length-two routes from packet zero to packet three, but their contributions sum to zero.

K

0	0	0	0
1	0	0	0
1	0	0	0
0	1	−1	0

path counts at length two

0	0	0	0
0	0	0	0
0	0	0	0
2	0	0	0

$$1 + (-1) = 0$$

$$K^2 = 0$$

Structural support is an upper bound. It does not encode destructive cancellation.

$$(K^2)_{30} = K_{31}K_{10} + K_{32}K_{20} = 1 - 1 = 0, \qquad K^2 = 0.$$

The matrix of path counts sees the two possibilities and forgets the signs. It therefore predicts possible support but not actual support. Replacing the inclusion $\operatorname{supp}(AB) \subseteq \operatorname{supp}(A) \circ \operatorname{supp}(B)$ by equality would be wrong even in this four-dimensional model.

What an acyclic graph does guarantee

If the longest strict directed path has h edges, then no term of K^{h+1} can survive. Thus $K^{h+1} = 0$. This is an upper bound on the nilpotence index, not an assertion that $K^h \neq 0$. Here the graph has paths of length two, while coefficient cancellation already gives $K^2 = 0$.

A structurally permitted block may also vanish because a product of rectangular maps is zero before any sum is taken. A support graph cannot distinguish that mechanism from cancellation among separate routes. The actual block coefficients are therefore part of a proof-bearing operator representation.

> **Read this as:** Use graph reachability to exclude impossible transfers. Use the algebra to determine which permitted transfers actually survive.

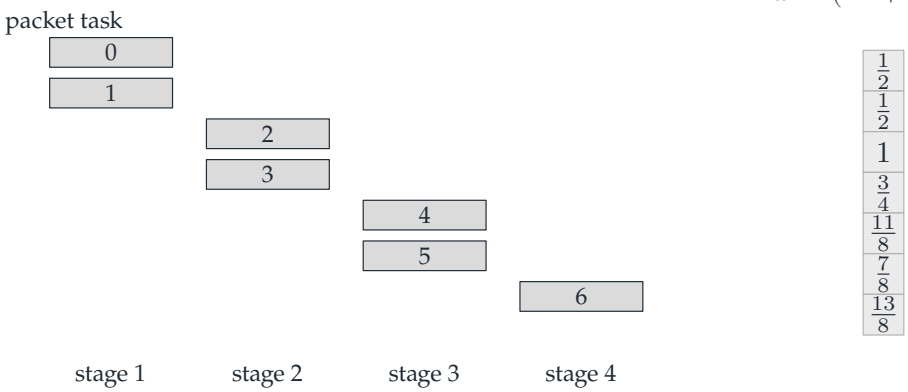
V15 / TOPOLOGICAL LEVELS WITH EXACT OUTPUTS

A solve has a dependency clock

Traditional pp. 22, 35 / Theorem 6.5, Corollary 6.6, Theorem 10.3.

For seven scalar packets, put two on the diagonal and -1 on each declared incoming edge. The edges are $0 \to 2$, $1 \to 2$, $1 \to 3$, $2 \to 4$, $3 \to 4$, $3 \to 5$, $4 \to 6$, and $5 \to 6$. Solve $(2I + K)x = \mathbf{1}$.

$$x = (2I + K)^{-1}\mathbf{1}$$



Read the stages as dependencies

Packets zero and one are ready first. Packets two and three use those values. Packets four and five then become available, followed by packet six. For instance,

$$x_4 = \frac{1 + x_2 + x_3}{2} = \frac{11}{8}, \qquad x_6 = \frac{1 + x_4 + x_5}{2} = \frac{13}{8}.$$

All entries in the displayed solution are checked by multiplying the original triangular matrix by the resulting vector.

When is four a minimum?

There are four vertices on a longest dependency path. Under the paper's unit-duration, unlimited-concurrency model, four stages are both necessary and attainable. If only one execution slot is available, seven tasks require seven slots of work. If task durations differ, the relevant critical path is weighted.

The theorem is not a universal GPU scheduling result. Transfers, synchronization, contention, and nonunit task costs change the model. The exact benefit here is that independent rows in one level need not wait for one another's mathematical results.

> **Read this as:** A topological schedule separates dependencies from resources. Its optimal depth is conditional on the task-duration and concurrency model that defines a stage.

V16 / SCHUR FILL AND ITS CANCELLATION

Elimination creates new interactions

Traditional p. 23 / Theorem 6.7, Proposition 6.8.

Retain the first two coordinates of

$$A = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}.$$

The third coordinate is hidden. Its invertible block is $[2]$, so eliminating it produces

$$S = A_{RR} - A_{RH}A_{HH}^{-1}A_{HR} = \begin{pmatrix} 3/2 & -1/2 \\ -1/2 & 3/2 \end{pmatrix}.$$

A before elimination

2	0	1
0	2	1
1	1	2

$S = A_{RR} - A_{RH}A_{HH}^{-1}A_{HR}$

$\frac{3}{2}$	$-\frac{1}{2}$
$-\frac{1}{2}$	$\frac{3}{2}$

cancelling variant

$\frac{3}{2}$	0
0	$\frac{3}{2}$

A hidden route creates $-1/2$; adding $+1/2$ in the original block cancels it.

The hidden route has a sign

The retained coordinates had no direct coupling, yet both touch the hidden coordinate. The resulting correction couples them by $-1/2$. This is fill: a new active retained block produced by an eliminated route.

Now change both original retained off-diagonal entries to $+1/2$. The same hidden correction cancels them, leaving the diagonal matrix $\frac{3}{2}I$. Thus a structural route can create or remove visible coupling depending on the direct coefficient already present.

A stronger inverse statement

For an invertible finite packet matrix, inverse support lies in the directed transitive closure. A reachable-coordinate subspace is invariant under the matrix. Injectivity of its restriction implies surjectivity in finite dimension, so the inverse preserves it as well. This proof uses finite-dimensional invertibility, not a convergent geometric series.

The checker verifies both Schur complements and, in each invertible example, equality between S^{-1} and the retained block of A^{-1}.

> **Read this as:** Removing coordinates does not remove their influence. It changes the retained operator, with possible fill, cancellation, and new conditioning constraints.

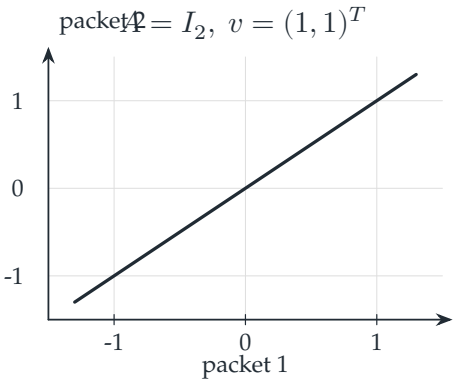
V17 / A STRICT KRYLOV INCLUSION

Invariant does not mean independently addressable

Traditional p. 24 / Proposition 7.2, Theorem 7.3.

Take $A = I_2$, split into the two coordinate packets, and start with $v = (1,1)^T$. Every packet is invariant. Nevertheless every polynomial applies the *same* scalar to both coordinates:

$$p(A)v = p(1)(1,1)^T, \qquad \dim \mathcal{K}_m(A,v) = 1 \quad (m \geq 1).$$



$$p(A)v = p(1)(1,1)^T$$

$$\dim \mathcal{K}_m(A,v) = 1$$

Independent local spans have dimension 2.

The direct sum of the two local Krylov spans is all of $\mathbb{Q}^2$. It permits independent coefficients in the two packets. The global Krylov space does not. This is why the manuscript states an inclusion, not an unconditional equality.

What breaks the linkage?

With $A = \mathrm{diag}(1,2)$, the vectors v and $Av = (1,2)^T$ are independent. Two polynomial samples now distinguish the packet labels. Alternatively, allowing the packet projectors as extra address operators separates the coordinates even when $A = I_2$. Both changes enlarge the addressing power, but in different ways.

When invariance is only approximate

If $A = D + K$ with D packet diagonal, then

$$\|A^j - D^j\| \leq jM^{j-1}\|K\|, \qquad M = \max\{\|A\|, \|D\|\}.$$

This controls departure from the decoupled iterate. It is not a claim that the dimension or numerical conditioning of a Krylov basis varies continuously. Near dependence can remain difficult even when the operator error is small.

> **Read this as:** A single global polynomial can lock otherwise invariant packets together. Independence of subspaces is not the same as independence of the allowed actions.

V18 / BLOCK JACOBI AS A GEOMETRIC CHANGE

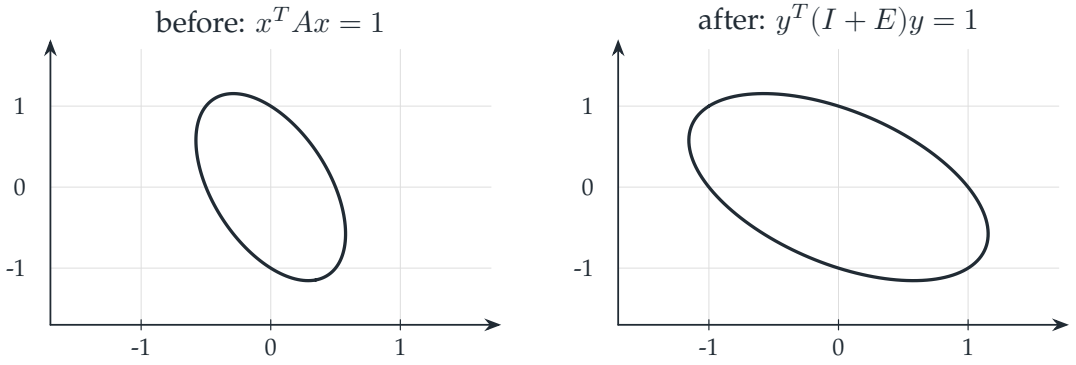
Measure coupling after normalization

Traditional p. 25 / Theorem 7.4, Corollary 7.6, Classical input 7.5.

Let

$$A = \begin{pmatrix} 4 & 1 \\ 1 & 1 \end{pmatrix}, \quad D = \operatorname{diag}(4, 1), \quad B = D^{-1/2} = \operatorname{diag}(1/2, 1).$$

Then $BAB = I + E = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 1 \end{pmatrix}$. Its exact eigenvalues are $1/2$ and $3/2$, giving condition number three.



Exact normalized eigenvalues: $1/2, 3/2$. Hence $\kappa_2 = 3$.

The unit circles of the energy norms

The left ellipse is $x^T Ax = 1$. Writing $x = By$ gives the right ellipse $y^T(BAB)y = 1$. The drawing samples these explicitly defined curves; the norm and eigenvalue equalities are checked symbolically.

Here $\|E\|_2 = 1/2$. The manuscript's general inequality

$$\kappa_2(D^{-1/2}AD^{-1/2}) \le \frac{1+\eta}{1-\eta}, \qquad \|E\|_2 \le \eta < 1,$$

is attained. The raw off-diagonal entry is one; its meaningful size is one-half only after normalization by the diagonal energy scales.

Keep the convergence input separate

The condition bound can be inserted into a conjugate-gradient convergence estimate under that method's positive-definite assumptions. The CG estimate is one of the separately tracked classical inputs. The algebraic normalization theorem does not certify every stopping rule, floating-point iteration, or arbitrary preconditioner.

> **Read this as:** A small raw coupling is not scale-invariant. Normalize against the diagonal blocks before deciding whether the coupling is weak enough for a condition bound.

V19 / DISCOVERY AND PERTURBATION CERTIFICATES

Twelve small blocks, one coherent error

Traditional pp. 26, 30 / Theorem 8.1, Proposition 9.2.

Split $\mathbb{Q}^4$ into scalar packets and let

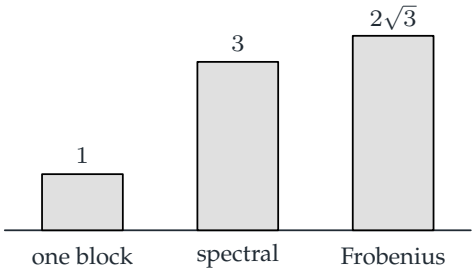
$$A = I + E, \qquad E = \frac{1}{100}(\mathbf{1}\mathbf{1}^T - I).$$

Dropping every off-diagonal block leaves I. Each discarded entry is only $1/100$, but the total error is not bounded by that single-entry size.

$100E$: twelve omitted entries

0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

error norms, in units of $1/100$



$\|E\|_F^2 = 12/10^4$, while $E\mathbf{1} = (3/100)\mathbf{1}$.

Two norms answer different questions

There are twelve omitted entries, so $\|E\|_F^2 = 12/10^4$. The all-one vector is an eigenvector with eigenvalue $3/100$, while the three orthogonal directions have eigenvalue $-1/100$. Hence $\|E\|_2 = 3/100$.

For orthogonal coordinate blocks, the squared Frobenius norms add exactly. Operator norms instead require a suitable bound or further spectral information. The maximum omitted block norm is generally insufficient.

Input sensitivity is additional information

The packet perturbation estimate is

$$\|(\widehat{A} - A)x\| \le \sum_e \|E_e\| \, \|P_{\alpha(e)}x\|.$$

It can be much tighter for an input that scarcely occupies an affected source packet. To obtain a uniform operator bound, the input dependence must be bounded uniformly over unit vectors. A certificate for one input is not automatically a certificate for every right-hand side.

These are a posteriori checks for supplied projectors. They do not discover the best packet system or prove that a thresholding heuristic is globally optimal.

> **Read this as:** A proposed compression needs a residual certificate in the requested norm and input scope. Small individual blocks can align into a larger collective error.

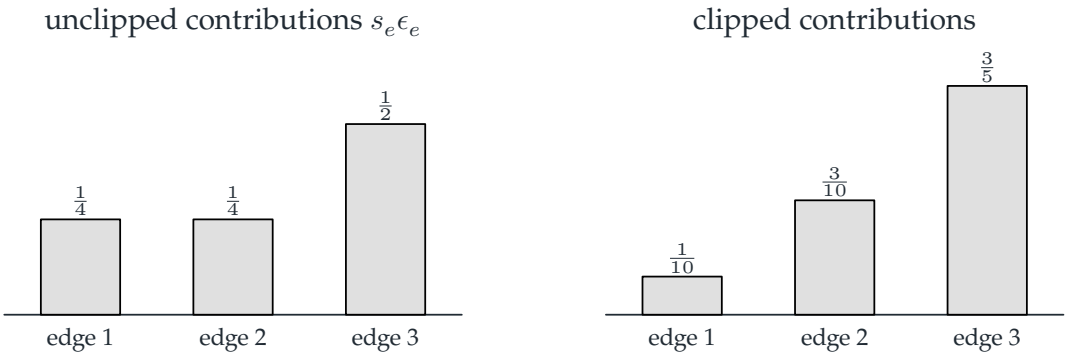
V20 / CONTINUOUS AND CLIPPED PRECISION

Spend error where it is cheapest

Traditional pp. 27, 28 / Theorem 8.4, Theorem 8.5.

Take sensitivities $s = (1, 2, 4)$, cost coefficients $c = (1, 1, 2)$, and budget $\tau = 1$. Minimizing $\sum_e c_e \log(1/\epsilon_e)$ subject to $\sum_e s_e \epsilon_e = 1$ gives

$$\epsilon_e = \frac{c_e}{4s_e}, \qquad \epsilon = (1/4, 1/8, 1/8).$$



Both allocations spend exactly one unit of error budget.

The budget is weighted

The three contributions to the error allowance are $(1/4, 1/4, 1/2)$. Equal cost coefficients give equal weighted contributions, not equal tolerances. The third edge has twice the cost coefficient of the second, which offsets its doubled sensitivity in this example.

Now impose upper tolerances $m = (1/10, 1/2, 1/2)$. The first edge must stop at its tighter cap. With multiplier $\lambda = 10/3$, the clipped solution is

$$\epsilon_e = \min\{m_e, c_e/(\lambda s_e)\}, \qquad \epsilon = (1/10, 3/20, 3/20).$$

Its weighted contributions are $(1/10, 3/10, 3/5)$ and still sum to one. The checker verifies the active cap, the multiplier equations, and the budget exactly.

From tolerance to a real data type

The continuous objective is a model. Actual kernels offer discrete precisions and may use different accumulation types. A tolerance below one does not identify a particular hardware format. The finite option program on the next page is the correct next step once admissible costs and error bounds have been supplied.

> **Read this as:** Precision allocation is an optimization over certified sensitivities and costs. It does not infer error bounds from a data-type name alone.

V21 / THREE EDGES, TWENTY-SEVEN ASSIGNMENTS

An exact budget is not a greedy choice

Traditional p. 29 / Theorem 8.7, Remark 8.8.

Each edge has three allowed $(\text{integer weight}, \text{cost})$ options:

$$1 : (0,7), (2,3), (4,1),$$
$$2 : (0,6), (1,4), (3,1),$$
$$3 : (0,9), (2,5), (5,1).$$

There are exactly $3^3 = 27$ full assignments. The table stores the minimum cost at each *exact* weight; columns zero through eight are displayed.

	0	1	2	3	4	5	6	7	8
edges 0	0	∞	∞	∞	∞	∞	∞	∞	∞
edges 1	7	∞	3	∞	1	∞	∞	∞	∞
edges 2	13	11	9	7	7	4	∞	2	∞
edges 3	22	20	18	16	14	12	12	9	8

Columns are exact weights; the budget optimum is the minimum across allowed columns.

Read the final row correctly

At budget four, minimize across columns zero through four. The optimum costs 14: choose weights $(2, 0, 2)$ with costs $(3, 6, 5)$. The zero-weight choice on edge two is its first, more expensive option. A locally cheap choice can consume too much of the shared budget.

The checker compares the recurrence with complete enumeration for every budget from zero through twelve. Infeasible states remain $+\infty$; they are not zero-cost states.

Quantization must be conservative

Suppose real errors are $3/50$ and $2/25$, and one integer unit represents $1/10$. Rounding down gives two zero weights, even though the true sum is $7/50 > 1/10$. Rounding up gives a certified upper envelope instead. The budget method is exact for its integers; it cannot repair unsafe integers supplied by a preprocessing step.

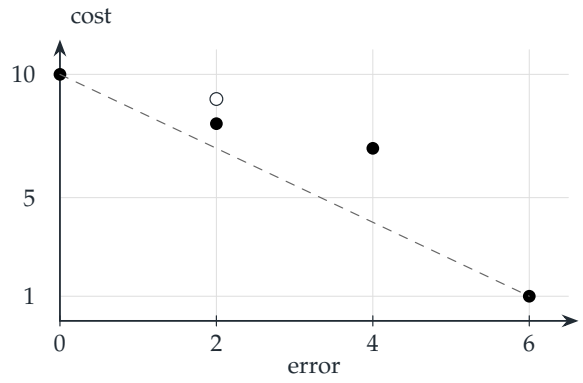
> **Read this as:** The table optimizes the declared option set exactly. Accuracy depends separately on the validity and conservativeness of the option weights.

V22 / FIVE PRUNINGS, ONE NONCONVEX FRONTIER

A penalty can miss the budget optimum

Traditional pp. 31, 32, 33 / Theorem 9.4, Corollary 9.7, Theorem 9.8.

A root r has children a, b, each with two terminal children. Keeping r costs one with error six; splitting it adds overhead one. Keeping either a or b costs three with error two. Splitting a costs four with zero error, and splitting b costs five with zero error.



Budget $B = 2$: cost 8

Penalty optimum: root or full split

$8 + 2\lambda \le 1 + 6\lambda$
and $8 + 2\lambda \le 10$
cannot both hold.

The five complete prunings have $(\text{cost}, \text{error})$ pairs

$$(1, 6),\ (7, 4),\ (9, 2),\ (8, 2),\ (10, 0).$$

At budget two the unique cheapest pruning is $\{a_0, a_1, b\}$, costing eight. It is Pareto optimal, but never minimizes cost $+\lambda$ err for any $\lambda \ge 0$.

See the incompatible inequalities

To beat the root, that pruning would require $8 + 2\lambda \le 1 + 6\lambda$, hence $\lambda \ge 7/4$. To beat the full split, it would require $8 + 2\lambda \le 10$, hence $\lambda \le 1$. No multiplier meets both conditions.

The exact budget recurrence retains the point because it enumerates admissible weight allocations. A sweep of Lagrangian parameters sees only supported frontier points. The two compiler formulations are therefore genuinely different, even on this tiny tree.

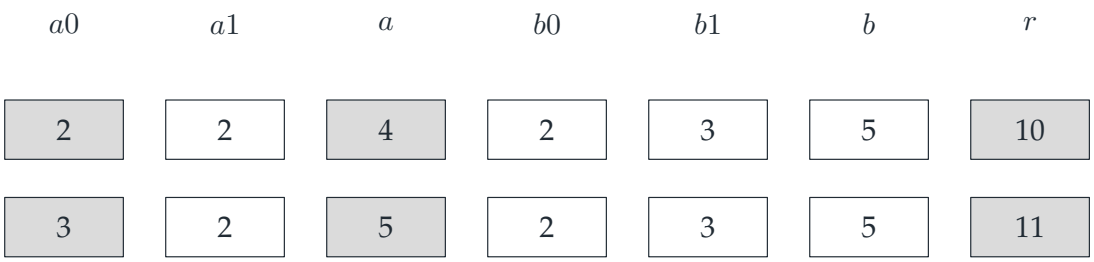
> **Read this as:** A penalty parameter is not a universal substitute for an explicit budget. Discrete, nonconvex frontiers can contain useful optima that no penalty selects.

V23 / LOCAL DEPENDENCIES AND STRICT MARGINS

Recompile the changed part, not everything

Traditional pp. 33, 34 / Theorem 9.11, Theorem 9.12.

Use the same tree with $\lambda = 2$. Increase only the leaf cost at a_0 from two to three. The exact subtree values change at a_0, its parent a, and the root r. Every value in the b subtree remains untouched.



Top: old values. Bottom: new values. Only a0, a, and r change.

Why the update is local

A dynamic-programming value depends only on the local keep/split data and its children's values. A changed leaf can therefore affect its ancestors but not a disjoint sibling subtree. Recomputing the dependency closure yields the same global optimum as a fresh traversal. In this example, the root value increases from ten to eleven.

Strict gaps protect decisions

Before the update, the root keep score is 13 and its split score is 10, giving gap three. If each score changes by less than $3/2$, their strict ordering is retained. Allowing exactly $3/2$ in both directions can produce a tie, so the strict inequality in the stability statement matters.

The bound applies to the completed scores after child subproblems have been evaluated. Bounds on raw leaf perturbations must first be propagated to those scores. Shared caches, overlapping subproblems, or global rank constraints can introduce dependencies outside a tree and invalidate this simple locality argument.

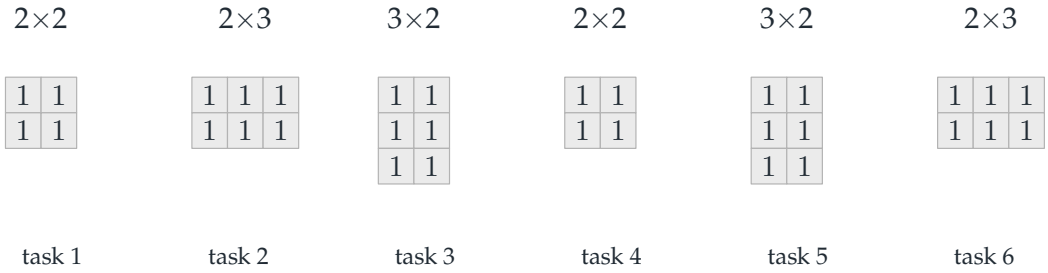
> **Read this as:** Local recompilation is exact because the model has local dependencies. The locality is a mathematical hypothesis, not an automatic property of every compiler optimization.

V24 / A LAUNCH MODEL IS NOT A BANDWIDTH MODEL

Batch shapes, account for traffic

Traditional pp. 11, 12, 35 / Proposition 3.6, Theorem 3.8, Proposition 10.2.

Six independent blocks have the displayed dimensions. When right-hand-side count, scalar type, accumulation type, and kernel contract also agree within each class, they can be grouped into three batches. That reduces the modeled launch count from six to three without changing their arithmetic.



Three shape classes, six independent tasks.

Grouping does not remove arithmetic, data transfer, or shared-output dependencies.

Independence must be real

The batching statement assumes compatible backend support and independent block tasks. If several blocks accumulate into one target, the reduction or accumulation policy is additional work. Equal matrix dimensions alone do not remove that dependency. Different accumulation types also define different shape classes under the paper's convention.

Count the data separately

For three diagonal 2×2 blocks, eight-byte scalars, and q right-hand sides, the stated one-pass model moves

$$B_{\text{stream}} = 8 \cdot 3(4 + 2q + 4q) = 96 + 144q \text{ bytes.}$$

It gives 240 bytes at $q = 1$ and 672 bytes at $q = 4$. It deliberately assumes no reuse across edges. A cache-aware implementation may have another traffic model, but it must state one.

Given operation capacity P and bandwidth W, $T \geq \max(F/P, B/W)$ is a lower bound, not an achieved runtime. Launch overhead, imperfect overlap, and utilization can make the actual time much larger.

> **Read this as:** Grouping can remove launches while leaving the same products and data movement. Arithmetic, traffic, and scheduling are related constraints, not one interchangeable cost.

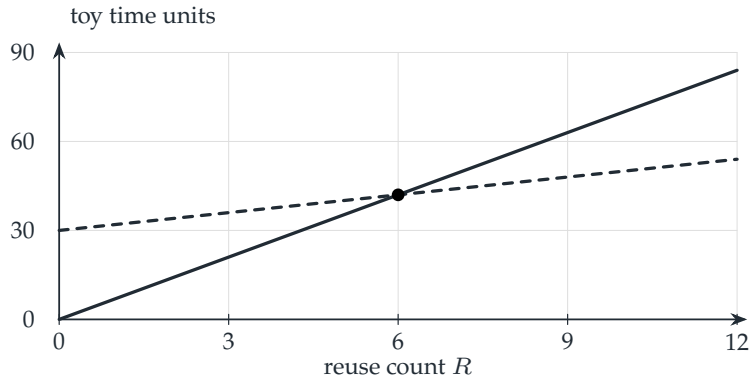
V25 / SETUP AND REUSE IN ONE LINEAR MODEL

The first win comes after the tie

Traditional pp. 35, 36 / Theorem 10.4, Corollary 10.5, Corollary 10.6.

Suppose the baseline costs seven units per use. A candidate costs 30 units to set up and two per use. These are synthetic constants chosen to expose the exact threshold:

$$T_b(R) = 7R, \qquad T_c(R) = 30 + 2R.$$



solid: $7R$

dashed: $30 + 2R$

tie at 6
first win at 7

At $R = 6$, both cost 42. The candidate becomes strictly faster at $R = 7$, where it costs 44 instead of 49. The minimum profitable integer reuse count is

$$\left\lfloor \frac{30}{7-2} \right\rfloor + 1 = 7.$$

Replacing the floor-plus-one rule with a ceiling would be wrong when the ratio is already an integer and strict improvement is required.

Which costs belong to setup?

Discovery, compilation, host/device transfer, graph capture, and factor preparation can have different reuse lifetimes. If a matrix changes but its shape does not, a captured schedule might remain reusable while its numerical factors do not. A single setup number is meaningful only after those lifetimes are fixed.

Materialization has the opposite tradeoff: pay once for an explicit representation, then use a potentially faster dense downstream operation. It pays off only when the explicit per-use cost is lower and the reuse count exceeds the corresponding conversion threshold.

> **Read this as:** A faster steady-state kernel can lose end-to-end. State the reuse regime and count the tie separately from the first strict win.

V26 / A BENCHMARK ROW IS A CONTRACT

Same output before comparing times

Traditional pp. 9, 36, 37 / Proposition 2.9, Section 11, Section 12.

Consider four synthetic timing rows for a product. Row B returns factors, while row A returns an explicit matrix. A ratio $12/2 = 6$ does not compare two implementations of the same returned object. Adding the decoding cost changes the candidate to row C.

row	output	scope	time
A	explicit matrix	materialization included	12
B	low-rank factors	factor product only	2
C	explicit matrix	factor product + decode	14
D	explicit matrix	different input precision	1

Synthetic times: A versus C is contract-aligned. A versus B is not a sixfold speedup.

What a fair match retains

The input values, dimensions, layout, source precision, correctness reference, returned output, and timing scope must agree. Row D changes the input precision; that can be a useful accuracy tradeoff, but it is not the same case merely because the output has the same shape.

If a downstream task consumes factors directly, row B has a valid compressed-output contract. It should be compared with another method that supplies that downstream result, not forced to materialize an unused array. Fairness does not mean every algorithm must use the same internal representation.

How the original measurements are treated here

The preserved manuscript contains its historical CPU/GPU corpus. Replaying its figure generator redraws those saved observations. It does not rerun the recorded device, establish present hardware performance, or transform logical avoided work into physically executed floating-point operations. No new GPU timing is added by this guide.

Exact representation gains, device gains, kernel gains, and amortization gains can be isolated only through appropriately matched rows. A total ratio that mixes them is still an observation, but not a decomposition theorem.

> **Read this as:** A timing number without its case and scope is incomplete data. Preserve the mathematical output contract before claiming a speedup.

V27 / FACTORS, APPLICATION, AND AN INDEPENDENT REFERENCE

Close the loop on the running example

Traditional pp. 7, 11, 16 / Theorem 2.3, Theorem 3.4, Theorem 5.1.

Return to the two operators from V9. The exact product representation has diagonal part $D_A D_B$ and two-column correction factors U_C, V_C. Apply it to the original $x = (1, -1, 2, 0, -2, 3)^T$.

$$V_C^T x = (2, 13)^T, \qquad (AB)x = D_A D_B x + U_C(V_C^T x).$$

$V_C^T x$	$D_A D_B x$	$U_C(V_C^T x)$	$(AB)x$
2	1	25	26
13	−5	21	16
	18	25	43
	6	21	27
	−12	25	13
	44	21	65

Two intermediate scalars and six output coordinates, with an exact dense reference.

A complete reader calculation

The local contribution is $(1, -5, 18, 6, -12, 44)^T$. The correction is $(25, 21, 25, 21, 25, 21)^T$. Their sum is

$$(26, 16, 43, 27, 13, 65)^T.$$

The checker forms the dense matrices A and B independently, multiplies them, applies the product to x, and compares all six components. The dense array is a validation reference, not a requirement of the factored application route.

Choose what happens next

For another input, reuse the factors. For a required explicit matrix, pay materialization. For a long product chain, track rank growth and decide whether recompression is useful. If approximation is introduced, transport its error through the later operators rather than reporting only a local singular tail.

This small example does not predict a speedup: at dimension six, implementation overhead can dominate. Its purpose is to make the semantics auditable. The same identity can then be used inside a larger, separately benchmarked workload.

> **Read this as:** The full route is: represent the exact operator, retain every interaction, return the requested object, and validate against an independent reference. Performance is a further experiment.

V28 / FORMAL, EXACT, NUMERICAL, AND EMPIRICAL EVIDENCE

What this edition certifies

Traditional pp. 41, 42, 83, 84 / Section 13, Section 14, Section F.6, Section F.7.

The preserved v0.6.0 release records 56 of 56 manuscript-proved finite items, 392 named theorem declarations in 40 modules, and 66 separately counted construction audits. Of the theorem declarations, 19 are reused foundation results. These are inherited records, not a new Lean run for this editorial edition.

Evidence	Its precise use here
Saved Lean records	Preserve the exact hypotheses and declared finite formalization boundary.
Fresh portable checks	Reproduce the guide's displayed matrices, identities, integer models, and finite optima. They are not extracted Lean proofs.
Vector drawings	Display explicit arrays, schedules, and sampled geometric formulas. Pixel accuracy is not algebraic certification.
Separate software checks	The release verification report records actual runs, tolerances, failures, and unavailable gates; these are separate from this guide's exact arithmetic.
Historical GPU corpus	Remains a record of the supplied original cases and environment. It is not a new measurement.

The recompression distinction

The sixth block derives best-rank and singular-tail conclusions from ordered thin-SVD equations, rather than assuming an optimality certificate. It does not independently establish general SVD existence, certify a floating-point QR/SVD algorithm, or identify every result with a canonical singular-values API. Those are separate statements.

The four classical entries remain visible

The manuscript separately tracks Eckart–Young–Mirsky, the conjugate-gradient energy estimate, the floating-point product model, and simultaneous spectral decomposition. The strengthened ordered-SVD proof does not silently promote all associated existence and numerical-algorithm claims.

Limits of exact compilation

The optimizer needs a finite candidate tree with additive data. Nonlocal cache effects, overlapping decompositions, global rank coupling, and unrestricted discovery can destroy its optimal substructure. No new generic dense-matrix exponent or universal accelerator claim follows from this guide.

> **Read this as:** This is a reader-facing reconstruction of the paper's finite concepts, with reproducible examples and preserved source evidence. It changes exposition, not the formal theorem contract.

V29 / THE INNER PRODUCT IS PART OF THE CONTRACT

A complex range needs an adjoint

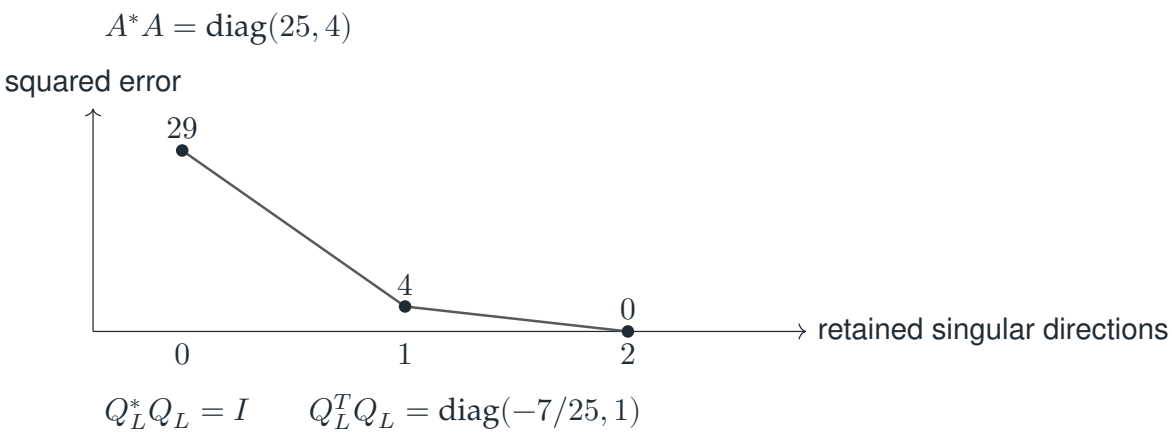
Traditional pp. 18, 79, 80 / Theorem 5.6, Section F.1, Section F.2.

The earlier compact example is real. Over $\mathbb{C}$, an isometry means $Q^*Q = I$, with conjugation. Take the exact Gaussian-rational factors

$$Q_L = \begin{pmatrix} 3/5 & 0 \\ 4\mathrm{i}/5 & 0 \\ 0 & 1 \end{pmatrix}, \qquad Q_R = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{i} \end{pmatrix}, \qquad G = \mathrm{diag}(5, 2).$$

Their product is a genuinely complex rectangular operator:

$$A = Q_L G Q_R^* = \begin{pmatrix} 3 & 0 \\ 4\mathrm{i} & 0 \\ 0 & -2\mathrm{i} \end{pmatrix}, \qquad A^*A = \mathrm{diag}(25, 4).$$



The small calculation exposes a wrong convention

The first column of Q_L has Hermitian squared length $9/25 + 16/25 = 1$. Its bilinear square without conjugation is $9/25 - 16/25 = -7/25$. Thus $Q_L^TQ_L$ is not the identity. The portable negative control replaces the adjoint by transpose and rejects the resulting isometry claim.

Keeping the first diagonal entry of G gives A_1 with its second column zero. The residual has squared Frobenius norm four and Gram matrix $\mathrm{diag}(0, 4)$, hence spectral norm two. Both errors agree with the discarded core singular value. Every displayed arithmetic operation occurs in $\mathbb{Q}(\mathrm{i})$.

What the finite check establishes

It checks both isometries, the reconstructed ambient matrix, its Gram matrix, and the explicit rank-one residual. Global best-rank optimality still comes from the manuscript's compression and lower-bound proof; this two-column calculation is a worked instance of that argument. No existence theorem or numerical SVD routine is executed.

> **Read this as:** Conjugation preserves the complex Euclidean norm. A transpose is correct in the vectorization identity on V7, but an adjoint is required for an isometric range lift.

V30 / A FINITE INVERSE WITH A CANCELLED ROUTE

Inverse reachability is still an inclusion

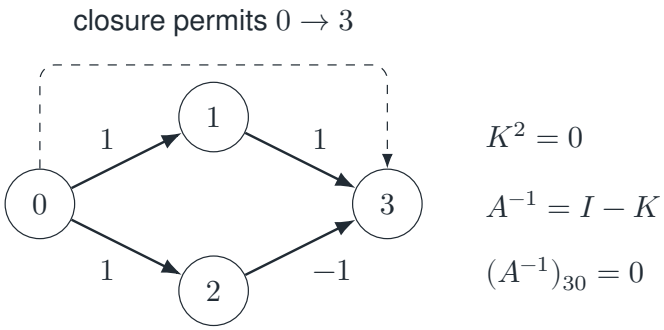
Traditional pp. 21, 23 / Theorem 6.7, Theorem 6.1, Corollary 6.3.

Reuse the signed diamond from V14, now as a complete operator:

$$A = I + K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 1 \end{pmatrix}.$$

The two routes from coordinate zero to coordinate three cancel, so $K^2 = 0$. Consequently

$$A^{-1} = I - K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 1 \end{pmatrix}.$$



What the closure permits

The directed transitive closure includes $0 \to 3$, because it records existence of a route rather than the signed sum of its coefficients. Nevertheless $(A^{-1})_{30} = 0$. Reachability is therefore an upper envelope for inverse support, just as it is for product support.

The portable checker computes the inverse independently by rational row elimination, compares it with $I - K$, constructs Boolean reachability, and checks every inverse entry outside that closure. It separately rejects the assertion that every reachable transfer is nonzero.

Why the general proof is broader

This example happens to have a terminating Neumann polynomial. The general finite-dimensional theorem does not require nilpotence or a small norm. For a fixed source coordinate, its reachable-coordinate subspace is invariant under A. If A is invertible, the restricted map is injective. Equal finite domain and codomain dimensions make it surjective, so the inverse preserves the same subspace.

The finite-dimensional and invertibility hypotheses matter. In an infinite-dimensional space, an injective restriction can fail to be onto. A singular matrix does not have an inverse whose support can be tested at all.

> **Read this as:** The inverse can never transmit outside the reachable subspace under the theorem's hypotheses. Inside that subspace, coefficient cancellation can still remove a structurally permitted interaction.

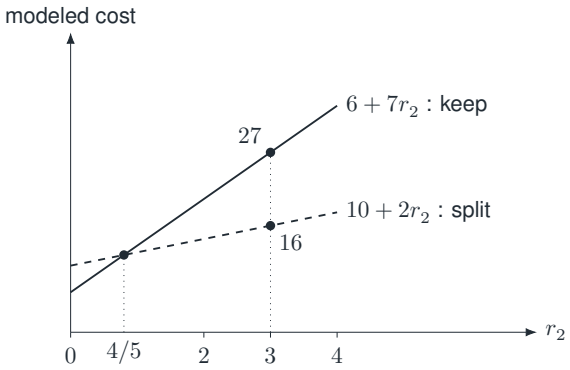
V31 / SHARED COMPILATION HAS ITS OWN OBJECTIVE

One tree can serve a weighted family

Traditional p. 31 / Corollary 9.5, Theorem 9.4.

Use a tree with one root and two terminal children. There are exactly two admissible prunings: keep the root, or split into both children. All declared errors and the split overhead are zero. Their additive total costs are:

Cost component	Keep	Split
Setup paid once	4	2
Operator 1, per use	2	8
Operator 2, per use	7	2



With reuse weights $(r_1, r_2) = (1, 3)$, the aggregated objectives are

$$J_{\text{keep}} = 4 + 2 + 3 \cdot 7 = 27, \qquad J_{\text{split}} = 2 + 8 + 3 \cdot 2 = 16.$$

The exact shared optimum is split. Operator 1 alone prefers keep on its per-use costs, while operator 2 prefers split. Choosing independently and trying to combine those decisions would not solve the declared one-pruning problem.

A hand-computable decision threshold

Fix $r_1 = 1$ and vary $r_2 = r \geq 0$. Keeping costs $6 + 7r$ and splitting costs $10 + 2r$. Splitting is strictly cheaper when $r > 4/5$; its first integer reuse win is $r = 1$. The checker evaluates both admissible choices for integers zero through four and checks the displayed totals at three.

Where the common-tree proof applies

The family corollary aggregates local keep costs and errors, then uses the same exact subtree recurrence. In this example, each row denotes the sum of the selected local leaf terms; it does not conceal a nonlocal discount. A setup shared by distant unrelated nodes, overlapping candidates, or a global resource constraint would need an enlarged state or a different model.

> **Read this as:** Reuse weights change the objective before compilation. Shared optimality is exact for the supplied common tree and additive family costs, rather than for an unspecified set of future workloads.

V32 / QUANTIZATION AND THE SIZE OF THE BUDGET TABLE

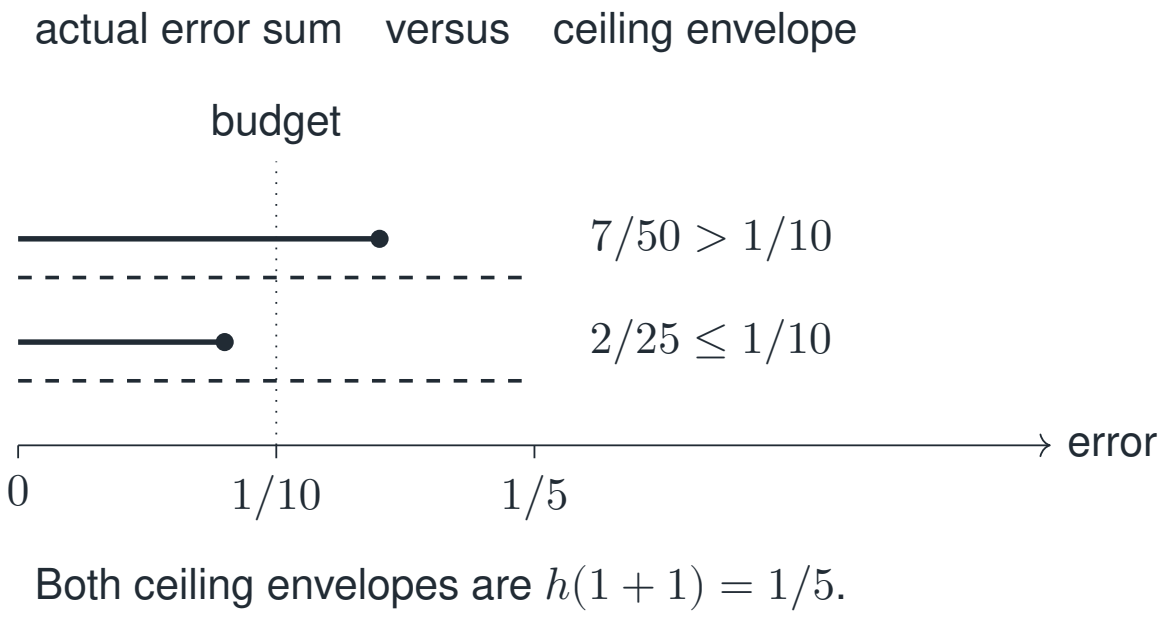
Safe integers can still be conservative

Traditional pp. 29, 33 / Theorem 8.7, Theorem 9.8, Corollary 9.9.

Let one integer unit represent $h = 1/10$ of certified error. For an individual nonnegative error e, choosing

$$w = \lceil e/h \rceil \qquad \text{ensures} \qquad e \le hw.$$

Thus integer feasibility $\sum w \le B$ implies the real bound $\sum e \le hB$. This implication is checked before applying the exact option or tree recurrence.



A false certificate and a conservative rejection

For errors $(3/50, 2/25)$, rounding down assigns weights $(0, 0)$ although their real sum is $7/50 > 1/10$. Rounding up gives $(1, 1)$ and correctly excludes the assignment from budget one.

Now change the errors to $(3/50, 1/50)$. Their sum is $2/25 \le 1/10$, but separate ceilings still give $(1, 1)$. The integer model rejects a genuinely real-feasible assignment. Conservative rounding protects soundness; it need not preserve every optimum of the unquantized problem.

Why the complexity is pseudo polynomial

For a declared integer budget B, a table has $B + 1$ possible exact-weight states. A direct merge checks at most $(B + 1)^2$ state pairs. At $B = 255$, this means 256 states and 65,536 candidate pairs at a merge, even though the budget fits in eight binary digits.

With a bounded number of children, the paper's direct tree implementation therefore has quadratic budget dependence at each node. Polynomial dependence on the numerical value of B does not mean polynomial dependence on the binary input length. These are counts of declared states and pairs, not measured runtimes or claims that every state is reachable.

> **Read this as:** The dynamic program is exact for its integer data. Conservative conversion and the numerical size of the budget are separate obligations that determine its accuracy and practical scale.