

FUNDAMENTALS OF AI-AUGMENTED ENGINEERING: A CONVERSATIONAL APPROACH

Taner Onsay, Ph.D. & Sebnem Onsay
Authors
October 2026 · Revision 1.0

© 2026 AI First Engineering



The engineering workflow you will use throughout this book

CHAPTER - 9

PART 4 · ≈ 35 MIN

Is Jacobi the Only Way? What Happens at Engineering Scale?

Part 4 introduces Gauss-Seidel, compares it with Jacobi on computed evidence, grows the circuit into a sparse grid with 10,000 unknowns, and asks how engineers choose a solver when the problem becomes large.

4

Problem

›

Model

›

Compute

›

Analyze

›

Validate

›

Improve



CONTENTS OF CHAPTER 9:

Chapter 8: a few coupled unknowns, solved directly. **Chapter 9:** far too many — solved by repeated correction, and checked.

1

Why do we need another way to solve equations?

From two unknowns to 100,000 · mostly-empty matrices · the 80 GB surprise · direct vs. iterative



2

How does Jacobi actually work?

Repeated correction by hand · using only the previous guess · pseudocode and compact MATLAB



3

When do iterations converge — and when do they fail?

Convergence and divergence · equation order · change vs. residual vs. error · stopping safely



4

Is Jacobi the only way? What happens at engineering scale? · you are here

Gauss-Seidel · a growing sparse network · LU in concept · what industrial solvers do



5

Can we trust the solver — and the AI?

Auditing AI-written solver code · ACCEPT / MODIFY / REJECT · a verification checklist



Course spiral: Ch 4–5 — one unknown, improved step by step until a stopping test is met (bisection, Newton) · Ch 8 — several coupled unknowns, solved directly · Ch 9 — the same kind of system, solved by repeated correction, and judged by evidence.



Can We Improve on Jacobi?

TO CHANGE < 1 mV

109

Jacobi iterations on the strongly coupled circuit — still 9.57 mV from the reference.

TO REACH 1 mV ERROR

154

Jacobi iterations before the reference error falls below 1 mV.

A CLUE FROM PART 2

The "hidden mistake" code reused the **newest** value of V_1 inside an iteration.

It was not Jacobi — yet it reached (2, 1) V in **6** iterations instead of **10**.

ASK THE ROOM

If a newer value is already available halfway through an iteration, why wait until the next iteration to use it?

Today: a second classic method — and how engineers choose among many.

Problem

›

Model

›

Compute

›

Analyze

›

Validate

›

Improve



One Small Change: Use the Newest Value

JACOBI

$$V_1^{(k+1)} = (5 + V_2^{(k)}) / 3$$

$$V_2^{(k+1)} = V_1^{(k)} / 2$$

```
x_old = x;  
x(1) = (5 + x_old(2)) / 3;  
x(2) = x_old(1) / 2;
```

GAUSS-SEIDEL

$$V_1^{(k+1)} = (5 + V_2^{(k)}) / 3$$

$$V_2^{(k+1)} = V_1^{(k+1)} / 2$$

```
% no copy of the old vector  
x(1) = (5 + x(2)) / 3;  
x(2) = x(1) / 2;
```

- **Jacobi**: a snapshot of the previous iteration.
- **Gauss-Seidel**: reuses each value as soon as it exists.

- Both are legitimate; they can behave **differently**.
- For Gauss-Seidel, the **update order** is part of the method.

One change: the second update uses the V_1 computed a moment ago — Part 2's "hidden mistake", now on purpose.

Problem ›

Model ›

Compute ›

Analyze ›

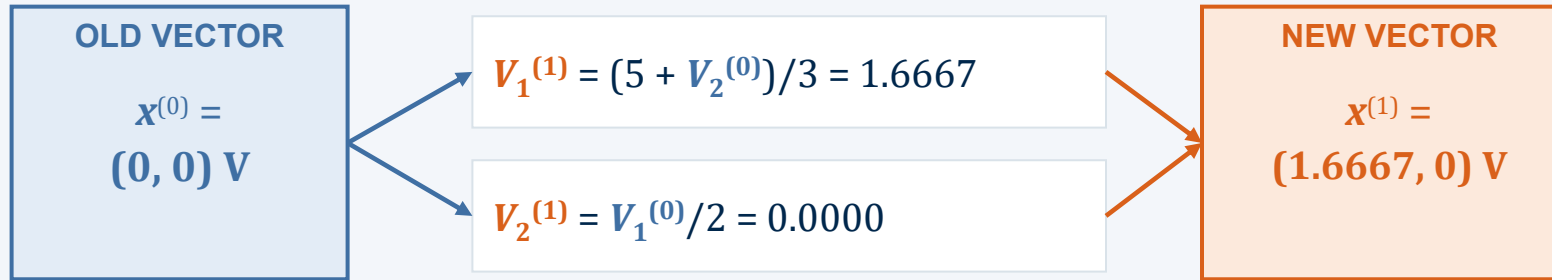
Validate ›

Improve

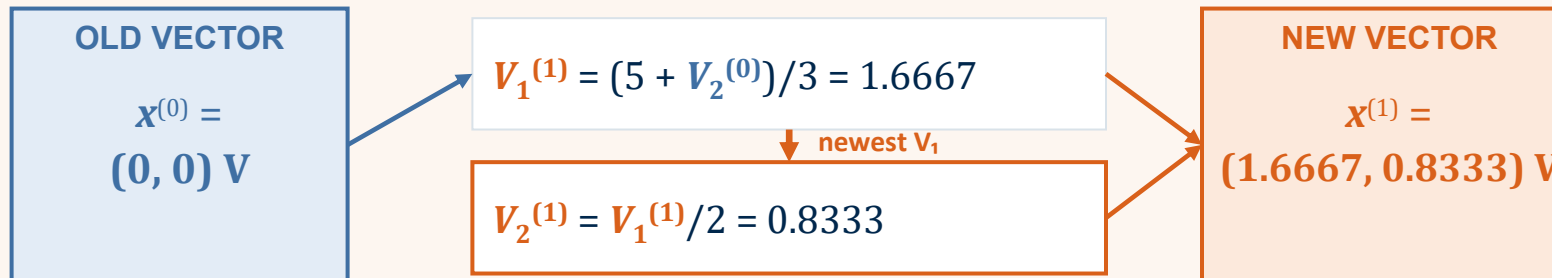


Jacobi vs. Gauss-Seidel: Side by Side

JACOBI · BOTH UPDATES READ THE OLD VECTOR



GAUSS-SEIDEL · EACH NEW VALUE IS USED AT ONCE



WHAT DIFFERS

Jacobi

- Keeps two vectors: old and new.
- Updates are independent — they can run at the same time.

Gauss-Seidel

- One vector, overwritten in place.
- Updates must run in order; a different order gives a different iteration.

Same equations, same starting guess — after one iteration Gauss-Seidel already has news about V_1 in V_2 .

Problem

**Model**

Compute



Analyze



Validate



Improve



Let's Calculate Two Iterations

ITERATION 1 · TOGETHER

$$V_1^{(1)} = (5 + V_2^{(0)}) / 3 = (5 + 0) / 3 = \mathbf{1.6667\ V}$$

$$V_2^{(1)} = V_1^{(1)} / 2 = 1.6667 / 2 = \mathbf{0.8333\ V}$$

The second line uses the value computed one line earlier.

ITERATION 2 · YOUR TURN (60 s)

$$V_1^{(2)} = (5 + \boxed{}) / 3 = \boxed{}\ \text{V}$$

$$V_2^{(2)} = \boxed{} / 2 = \boxed{}\ \text{V}$$

Which value goes in each box — old or newest?

THEN COMPARE

After two iterations Jacobi has **(1.6667, 0.8333) V**. Is your Gauss-Seidel vector closer to **(2, 1) V**? By how much?

Newest value, every time: V_1 uses the latest V_2 ; V_2 uses the V_1 you just computed.

Problem ›

Model ›

Compute ›

Analyze ›

Validate ›

Improve



How Much Faster Is It?

k	Jacobi V_1	Jacobi V_2	G-S V_1	G-S V_2
0	0.0000	0.0000	0.0000	0.0000
1	1.6667	0.0000	1.6667	0.8333
2	1.6667	0.8333	1.9444	0.9722
3	1.9444	0.8333	1.9907	0.9954
4	1.9444	0.9722	1.9985	0.9992
Ref.	2.0000	1.0000	2.0000	1.0000

Table 1. System A (familiar circuit) from (0, 0) V; values in V.

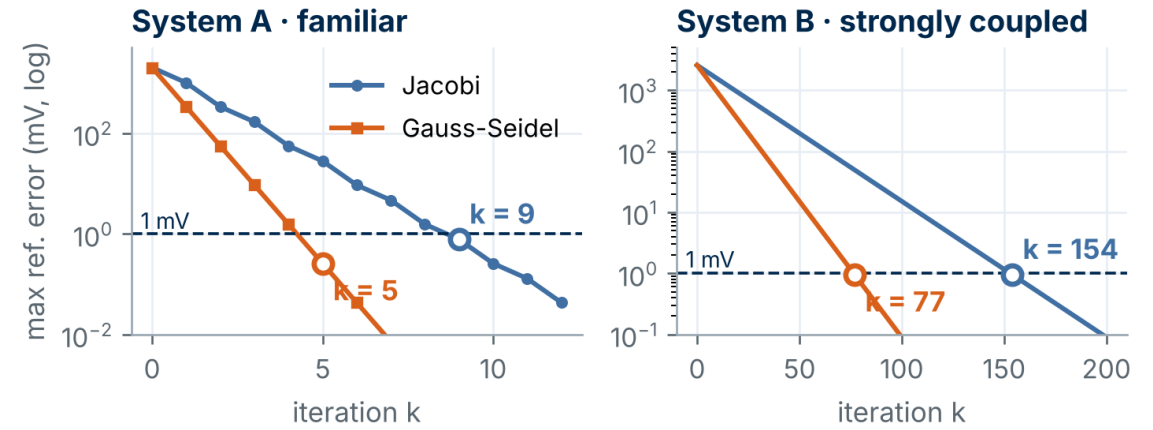


Figure 1. Computed max reference error; circles: first iterate below 1 mV.

System	Method	k at change < 1 mV	error then (mV)	max r then (mA)	k to error < 1 mV
A · familiar	Jacobi	10	0.257	0.00064	9
A · familiar	Gauss-Seidel	6	0.043	0.00011	5
B · strongly coupled	Jacobi	109	9.57	0.0187	154
B · strongly coupled	Gauss-Seidel	55	9.09	0.0177	77

About half the iterations here — but on System B both methods still stop about 9 mV from the reference.

Problem



Model



Compute



Analyze



Validate



Improve



Does Gauss-Seidel Always Win?

TWO EXAMPLES ARE NOT A LAW

On our two circuits Gauss-Seidel needed about half the iterations (5 vs 9, 77 vs 154).

That is evidence about these two systems — not a theorem about all systems.

ORDER STILL MATTERS

Part 3's swapped equations also defeat Gauss-Seidel — even faster:

$(0, -5) \rightarrow (-10, -35) \rightarrow (-70, -215) \rightarrow (-430, -1295) \text{ V}$

A COUNTEREXAMPLE

$$\begin{bmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} b = [1; 3; 5], \text{ solution } (1, 1, 1)$$
Jacobi: exact at $k = 3$
Gauss-Seidel: $(1, 2, -1), (-5, 9, -3), (-23, 29, -7) \dots$ runs away

FEWER ITERATIONS $\neq$ ALWAYS BETTER

- Gauss-Seidel updates must run one after another.
- Jacobi updates are independent — easy to run in parallel.
- Which wins depends on the system and the hardware.

Evidence, not reputation. Each method must earn trust on each problem.

Problem



Model



Compute



Analyze



Validate

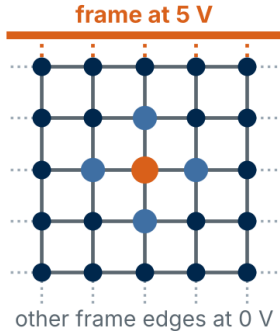


Improve



Now Grow the Engineering Problem

physical grid: one unknown per node,
linked to at most 4 neighbours



spy(A) for a 10 × 10 grid:
100 × 100 matrix

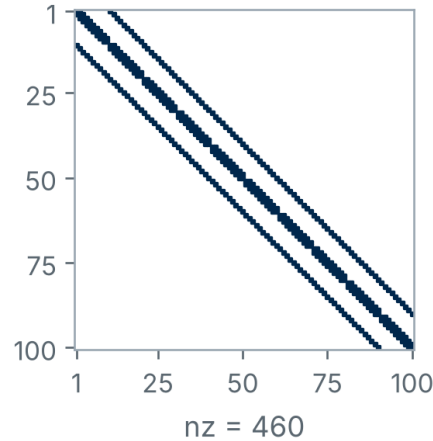


Figure 2. A 5 × 5 sketch of the mesh (left) and spy(A) for the 10 × 10 mesh (right): 1 kΩ resistors, frame at 5 V on top, 0 V elsewhere.

Unknowns ×100 (100 → 10,000). Dense entries ×10,000. Nonzeros only ×108 (460 → 49,600).

	10 × 10	30 × 30	100 × 100
Unknowns n	100	900	10,000
Dense entries n ²	10,000	810,000	100,000,000
Nonzeros	460	4,380	49,600
Nonzero fraction	4.6 %	0.54 %	0.050 %
Dense storage	80 KB	6.5 MB	800 MB
Sparse storage	≈ 8 KB	≈ 77 KB	≈ 0.87 MB

Table 2. Counted from the constructed matrices; 8 bytes per dense entry.

```

N = 30; e = ones(N,1);
T = spdiags([-e 2*e -e], -1:1, N, N);
A = kron(speye(N),T) + kron(T,speye(N));
spy(A), nnz(A)           % 4380

```

Each node talks to at most four neighbours, so each row has at most five nonzeros — at every size.

Problem



Model



Compute



Analyze



Validate



Improve



Sparse Does Not Mean Easy

STORAGE

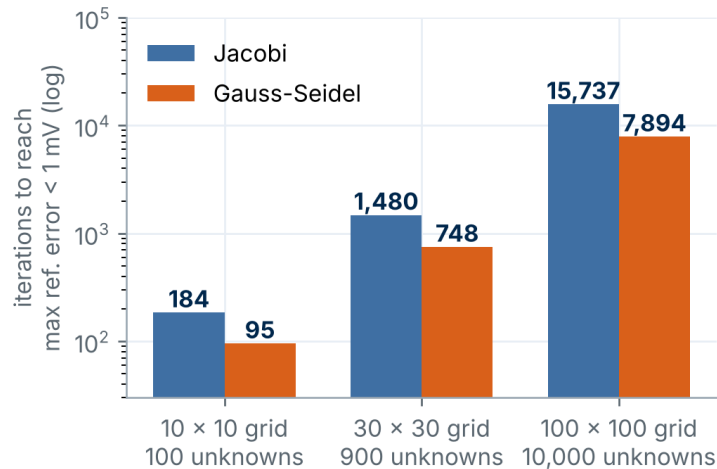
How efficiently can we represent the matrix?

800 MB $\rightarrow$ 0.87 MB for 10,000 unknowns. Sparsity answers this.

SOLUTION

How efficiently and reliably can we solve the equations?

Sparsity alone does not answer this.



THE TRAP GROWS TOO

100 × 100 grid, stop when change < 1 mV:

Jacobi stops at $k = 1,129$ — still **1.35 V** from the reference.

Gauss-Seidel stops at $k = 976$ — still **0.85 V** off.

WHAT ELSE MATTERS

- Coupling and matrix structure (Part 3: 2 unknowns, 109 iterations).
- How hard the problem is to solve accurately (its conditioning).
- The solver chosen — plain Jacobi is not the industrial tool for large grids.

Figure 3. Computed iterations (not timings).

Sparse storage makes large problems possible — it does not make them quick or easy to solve.

Problem



Model



Compute



Analyze



Validate



Improve



What About Direct Solvers?

FACTOR ONCE: $A = L U$

$$\begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1/3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 0 & 5/3 \end{bmatrix}$$

L: lower triangular (zeros above the diagonal)

U: upper triangular (zeros below the diagonal)

THEN SOLVE IN TWO EASY STEPS

1 Forward: $Ly = b \rightarrow y = (5, 5/3)$

2 Backward: $Ux = y \rightarrow x = (2, 1) V$

Each step solves one unknown at a time — no iteration, no stopping rule.

WHY FACTOR? REUSE

Same circuit, source doubled: $b = [10; 0]$ mA.

Reuse L and U: two quick triangular solves give $x = (4, 2) V$.

A SIMPLIFIED PICTURE

- Real solvers reorder rows and columns (pivoting, permutations): MATLAB's $[L, U, P] = \text{lu}(A)$ gives $PA = LU$.
- You will not be asked to compute L or U.

Direct solvers do the hard work once: factor A, then every new right-hand side is cheap.

Problem



Model



Compute



Analyze



Validate



Improve



What Do Engineers Actually Use?

Solver family	Examples	Typical use	Watch out for
Dense direct	Gaussian elimination, LU	small and medium dense systems	storage n^2 , work grows fast
Sparse direct	sparse LU; Cholesky (symmetric positive-definite)	large sparse systems; many right-hand sides	fill-in during factorisation
Basic iterative	Jacobi, Gauss-Seidel	teaching; building blocks inside other methods	slow on large grids; may not converge
Advanced iterative	Conjugate Gradient (SPD); GMRES (general)	very large sparse systems	usually needs preconditioning; monitor convergence

Table 3. Solver families. SPD = symmetric positive-definite (resistor-grid nodal matrices are SPD). Awareness level for Cholesky, CG and GMRES.

FILL-IN · 30 × 30 GRID

Upper half of A: **2,640** nonzeros. Cholesky factor: **27,029** — or **10,042** after reordering the nodes.

NO UNIVERSALLY BEST SOLVER

The right family depends on size, sparsity, symmetry, accuracy needed and how many right-hand sides you have.

Choosing a solver is an engineering decision — made from the problem's structure, not from an algorithm's popularity.

Problem



Model



Compute



Analyze



Validate



Improve



YOUR TURN

Choose the Solver

CASE A

Small circuit

- 3 unknowns, dense 3×3
- Priority: simple, reliable, done once

CASE B

Large resistor grid

- 1,000,000 unknowns, sparse
- Symmetric positive-definite
- Solved many times with new sources

CASE C

Controlled-source circuit

- 500,000 unknowns, sparse
- Nonsymmetric matrix
- Must be solved reliably

For each case: which solver family would you investigate first — and what evidence would you need before accepting its result?

Use the families on the previous slide. Names of advanced methods are not required — reasons are.

More than one answer can be defensible. An answer without a reason — or without evidence — is not.

Problem



Model



Compute



Analyze



Validate



Improve



The Best Solver Is the One You Can Trust

JACOBI



GAUSS-SEIDEL



SPARSE SCALE



SOLVER CHOICE

WE KNOW

Gauss-Seidel reuses the newest values; here it needed about half of Jacobi's iterations — but not always.

WE MEASURED

Sparse storage shrinks 800 MB to under 1 MB, yet iterations still grow with the grid.

WE CHOOSE

By size, sparsity, symmetry and accuracy — and we accept results only with evidence.

NEXT — PART 5 · CAN WE TRUST THE SOLVER — AND THE AI?

If an AI assistant writes your solver, how will you know it implements the method it claims — and that its answer is right?

The best solver is the one you can trust. Trust comes from evidence — never from RUNS alone.

Problem



Model



Compute



Analyze



Validate



Improve