

# 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 1 · ≈ 25 MIN

# Why Do We Need Another Way to Solve Equations?

Chapter 8 found two coupled node voltages directly. Engineering models can have 100,000 unknowns or more — too many to store as a full table, yet each equation involves only a few neighbours. Part 1 asks why that calls for a different way of solving.



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? · you are here

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?

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.



# 3 Equations ( $A \rightarrow 3 \times 3$ ) -- Were Easy. What About 100,000?

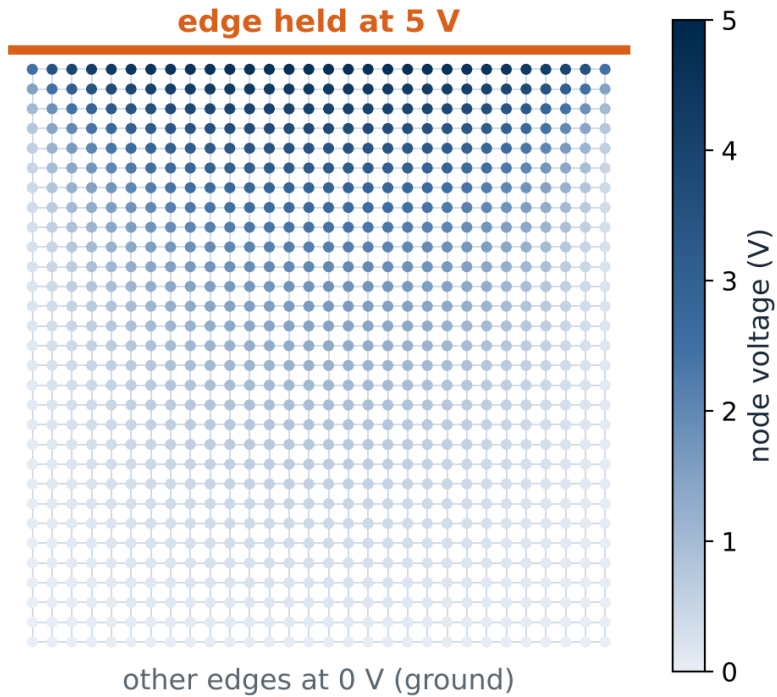


**Figure 1.** Computed voltages of a  $30 \times 30$  resistor mesh: 900 unknown node voltages, each coupled to its neighbours. Chip power grids and heat-sink models are far larger.

## CHAPTER 8

# 2

unknown node voltages  
4 matrix entries

## LARGE MODELS

# 100,000+

unknowns  
 $10^{10}$  matrix entries

## ASK THE ROOM

Chapter 8 solved two unknowns in an instant — Gauss-Jordan by hand,  $A \setminus b$  in MATLAB.

What could go wrong if the same circuit had **100,000** unknown nodes?

**The physics does not change. The size does.** Today: why size calls for a different way of solving.

Problem

›

Model

›

Compute

›

Analyze

›

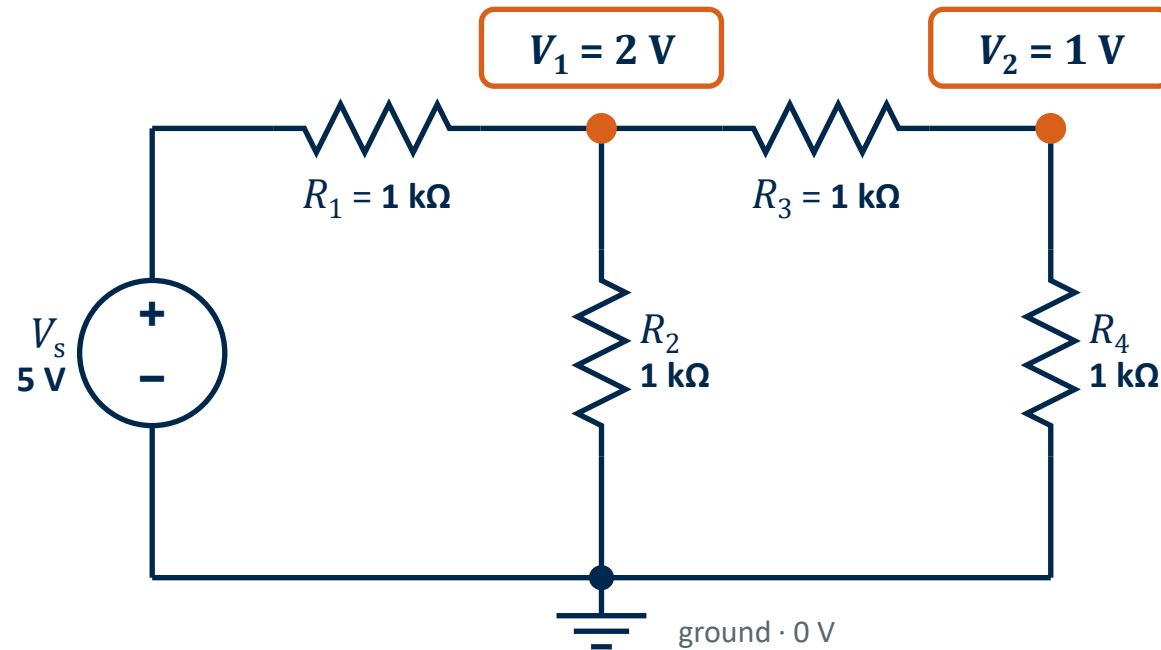
Validate

›

Improve



# What We Already Know



**Chapter 8 circuit:** every resistor is 1 kΩ, so every conductance is 1 mS; the 5 V source behind R1 contributes  $5\text{ V} \div 1\text{ k}\Omega = 5\text{ mA}$  at node V1.

**Kirchhoff Current Law**  $\sum I_{\text{leaving } V1} = 0$   
 $I_i = \Delta V / R_i$

## THE MODEL · KCL AT EACH NODE

$$3V_1 - V_2 = 5$$

$$-V_1 + 2V_2 = 0$$

$$\begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

$A$  in mS  
 $x$  in V,  $b$  in mA

## SOLVED DIRECTLY — AND CHECKED

Gauss-Jordan by hand, or  $\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$  in MATLAB:

**Result:**  $V_1 = 2\text{ V}$ ,  $V_2 = 1\text{ V}$

**Substitution check:**  $3(2) - 1 = 5 \checkmark$   $-2 + 2(1) = 0 \checkmark$

**A direct method:** a fixed sequence of steps leads to the answer — no guessing, no stopping decision.

Problem



**Model**



Compute



Analyze



Validate



Improve



# Where Do Thousands of Unknowns Come From?

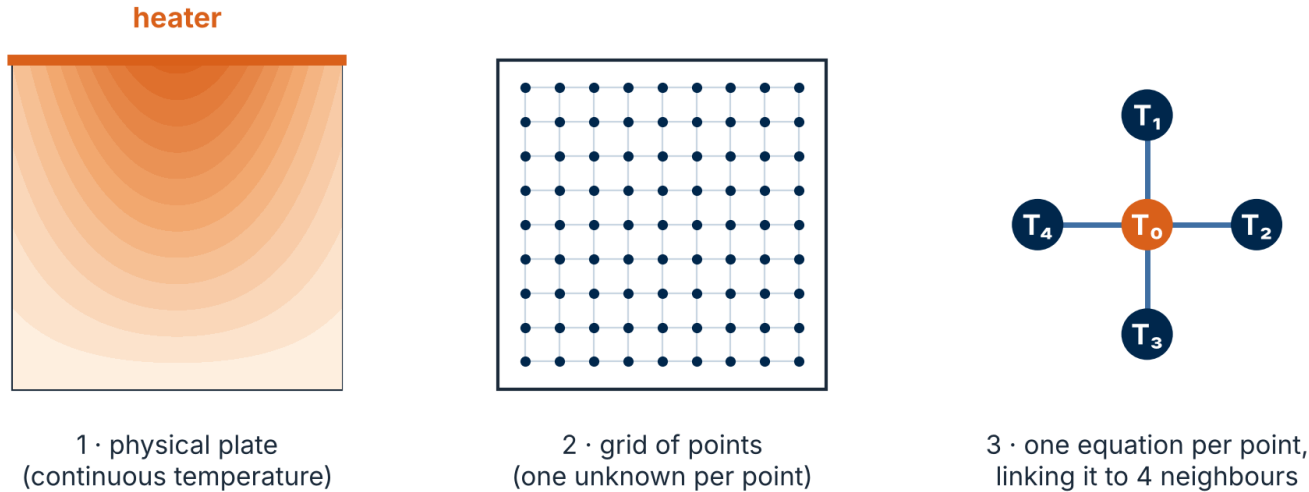


**Figure 2.** A continuous temperature field becomes a grid of unknown temperatures. A  $300 \times 300$  grid has 90,000 unknowns.

**Steady heat flow, uniform plate, equal spacing:**  $4T_0 - T_1 - T_2 - T_3 - T_4 = 0$

Same shape as a node equation in a resistor mesh: one unknown, its four neighbours, nothing else.

## CIRCUITS & POWER GRIDS

One unknown voltage per node. A chip's power-delivery grid can have millions of nodes.

## HEAT & FLUID FIELDS

One unknown temperature or pressure per grid point — 90,000 for a  $300 \times 300$  grid.

## STRUCTURES

Unknown displacements at every point of a finite-element mesh of a beam, frame or bridge.

**Many unknowns — but each one talks only to its neighbours.** That single fact shapes the rest of this chapter.

Problem



**Model**



Compute



Analyze



Validate



Improve

# A Matrix Can Be Mostly Empty

## CHAPTER 8

|    |    |
|----|----|
| 3  | -1 |
| -1 | 2  |

$2 \times 2$ : 4 of 4 entries nonzero  
— dense

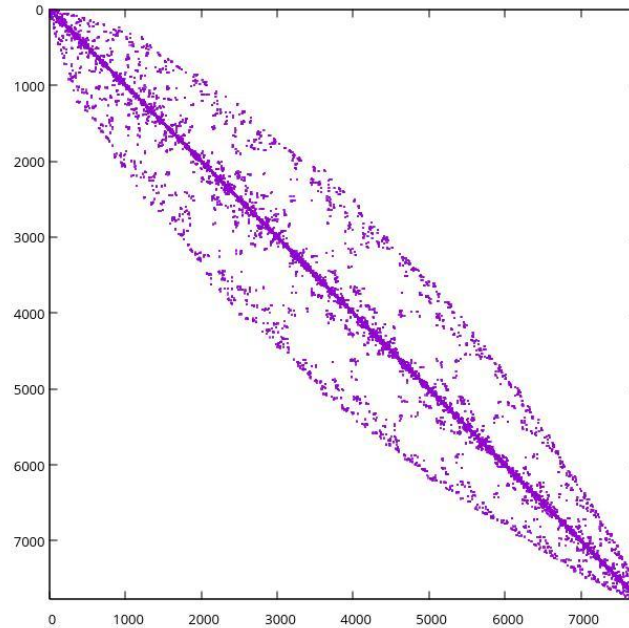
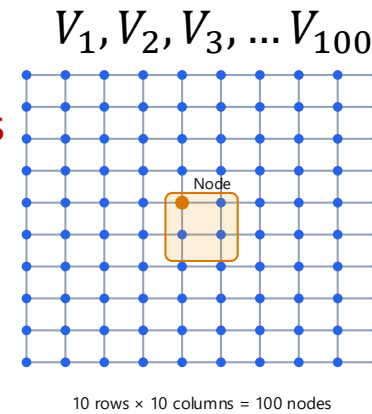


Figure 3-A. Finite-element *sparse* matrix visualization.

Nodes



## INSTRUCTOR DEMO MATLAB

```
N = 10; 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)           % 460
```

### For the Resistor Grid on the top

- **A  $10 \times 10$  resistor mesh:** 100 unknowns, 10,000 entries — only 460 nonzero (4.6 %).
- At most 5 nonzeros per row: a node and its 4 neighbours.
- The bigger the mesh, the smaller the nonzero fraction.

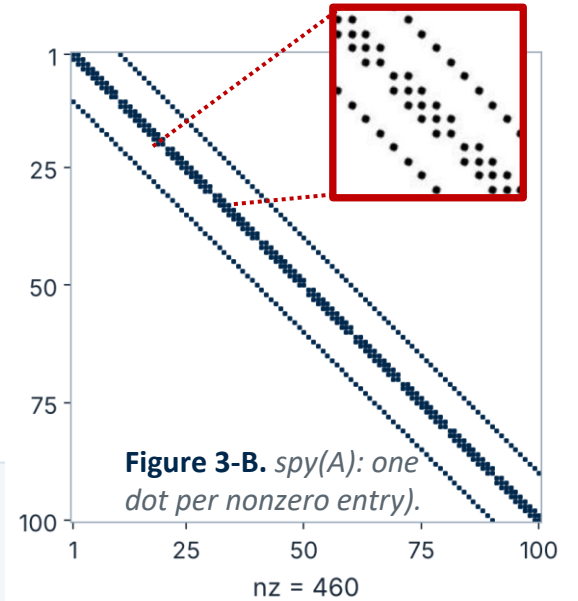


Figure 3-B. *spy(A)*: one dot per nonzero entry).

**Dense: almost every entry matters. Sparse: almost every entry is zero.** Store and use only the nonzeros.

Problem

›

**Model**

›

Compute

›

Analyze

›

Validate

›

Improve



# The 80 GB Surprise

3 × 3 · DENSE

## 72 bytes

9 entries × 8 bytes



100,000 × 100,000 · DENSE

## 80 GB

$10^{10}$  entries × 8 bytes  
=  $8 \times 10^{10}$  bytes ( $\approx 74.5$  GiB)



100,000 UNKNOWNNS · SPARSE

## $\approx 9$ MB

$\approx 500,000$  nonzeros ( $\leq 5$  per row)  
value + position stored

### WHY IT MATTERS

- A laptop typically has 8–32 GB of memory: the dense table alone would not fit — before any solving starts.
- Stored sparsely, the same matrix needs roughly 9,000 times less memory.

### ASK THE ROOM

Is every large engineering system sparse?

If a method needs the full table — or fills in many zeros as it works — what happens at 100,000 unknowns?

**Do not store what is zero.** Dense storage is a choice, not a law — for large sparse systems it is the wrong one.

Problem



Model



**Compute**



Analyze



Validate



Improve



# Two Different Ways to Solve

## DIRECT METHODS

*e.g. Gauss-Jordan, Gauss elimination, LU-type factorizations*

**How:** a fixed, finite sequence of operations produces the answer.

**Strengths:** predictable; exact apart from round-off; no stopping decision. The standard choice for small and medium systems — and, as sparse direct solvers, for many large ones.

**Costs:** work and memory grow quickly with size; elimination can fill in zeros.

1 › 2 › 3 → answer, after a known number of steps

## ITERATIVE METHODS

*e.g. Jacobi, Gauss-Seidel — and more advanced methods*

**How:** start from a guess and improve it, step after step.

**Strengths:** each step uses only the nonzeros, so memory stays small; you can stop as soon as the answer is accurate enough.

**Costs:** the answer is approximate; you must decide when to stop; it can converge slowly — or not at all.



guess → improve → check → repeat ... until good enough

**Neither family is always better.** The choice depends on size, sparsity and the accuracy needed — and both need evidence.

Problem ›

Model ›

**Compute** ›

Analyze ›

Validate ›

Improve



# We've Seen Iteration Before



## BISECTION · CH 4

**Unknown:** one unknown: the speed  $v$

**Update:** halve the bracket that holds the root

**Stop when:** the bracket is narrow enough

## NEWTON-RAPHSON · CH 5

**Unknown:** one unknown: the speed  $v$

**Update:** follow the tangent:  $20.0 \rightarrow 20.3704 \rightarrow 20.3679$  m/s

**Stop when:** the correction is smaller than required

## THIS CHAPTER

**Unknown:** many unknowns: every node voltage at once

**Update:** *Part 2*

**Stop when:** *Part 3*

**Same loop, new scale.** What changes is WHAT we update: one number — or a whole vector of them.

Problem



Model



Compute



Analyze



Validate

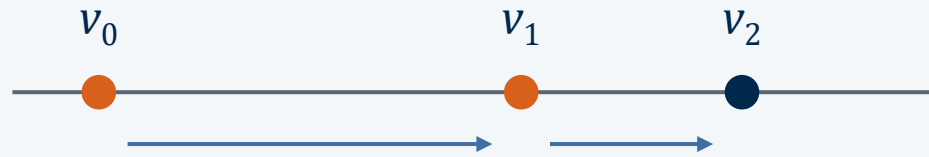


Improve



# What Changes When There Are Many Unknowns?

## ONE UNKNOWN · CH 4–5



Each estimate is **one number**. "How much did it change?" is one subtraction.

## MANY UNKNOWN · CH 9

$$\mathbf{x}^{(k)} = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

Each estimate is a **whole vector** of node voltages, in volts.

Chapter 8 example: start at  $[0; 0]$  V, target  $[2; 1]$  V.

## THREE NEW QUESTIONS

- **Update:** how do we improve every component — and which values may each update use? → **Part 2**
- **Change:** how do we measure the change of a whole vector with one number, in volts? → **Part 3**
- **Trust:** how do we know the vector is close to the true solution — not just changing slowly? → **Parts 3 & 5**

**An estimate is now a vector — with units.** Every component is a node voltage in volts, and every check must respect that.

Problem



Model



**Compute**



Analyze



Validate



Improve



## YOUR TURN

# Could Repeated Correction Work?

### THE SITUATION

$$3V_1 - V_2 = 5$$

$$-V_1 + 2V_2 = 0$$

Exact answer (Chapter 8): **(2, 1) V**.

Start from a guess: **(0, 0) V**.

A method uses the two equations to **correct the guess** — then corrects the corrected guess, over and over.

**No calculation. Predict.**

### What will repeated correction do?

**A** reach (2, 1) V exactly after a few corrections

**B** get closer and closer to (2, 1) V

**C** move further and further away

**D** it depends on how the correction is made

**Write your letter and one reason.** Then: what evidence would convince you the corrections are working?

**Commit now — and keep your answer.** Parts 2 and 3 put it to the test.

Problem



Model



Compute



**Analyze**



Validate



Improve



# The Engineering Question That Remains

LARGE MODELS ›

MOSTLY ZEROS ›

DENSE STORAGE FAILS ›

IMPROVE A GUESS

## WE KNOW

Chapter 8's direct methods solve small systems exactly:  $(2, 1) V$ , checked by substitution.

## WE FOUND

Real models have 100,000+ unknowns, but each equation links only a few neighbours. Stored as a full table: 80 GB.

## WE NEED

A method that improves a guess using only the nonzeros — and evidence that it is working.

## NEXT — PART 2 · HOW DOES JACOBI ACTUALLY WORK?

**Given a guess for every node voltage, how can the circuit's own equations produce a better guess — and how would we know it is better?**

**Numbers on a screen are not evidence.** Every iterative answer must earn trust — that is the thread of this chapter.

Problem ›

Model ›

Compute ›

Analyze ›

**Validate** ›

Improve