

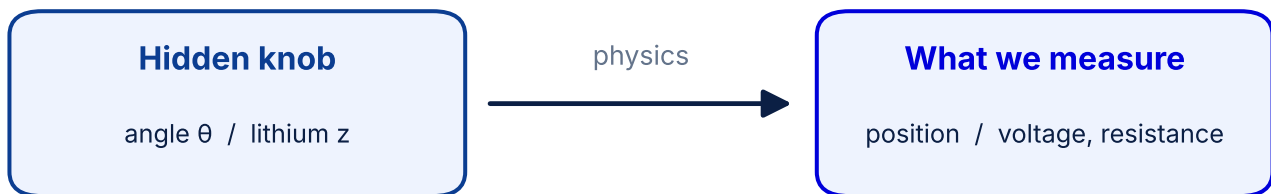
The Riemannian Pullback, ultra simple

First-year engineering maths only: one derivative rule at a time

The big idea in one picture

Every physical system has a hidden knob and things we can measure. A pendulum's knob is its angle. A battery's knob is how much usable lithium is left.

We usually can't see the knob directly. We only see its effects: a position, a voltage, a resistance.



The one question a pullback asks

If I turn the hidden knob a tiny bit,
how much do my measurements move?
The answer is one number, called the metric G .

Why this matters

General Relativity uses exactly this maths, Riemann's geometry, to describe curved spacetime. The MW Framework borrows it for all physics, not just gravity.

Instead of re-solving hard equations (PDEs) every time on GPUs in the cloud, we learn the shape of the physics once. After that, a CPU can do the work.

What you need for this PDF

Only three derivative rules. Each one is shown at the moment we use it.

$$\frac{d}{d\theta} \sin \theta = \cos \theta \quad \frac{d}{d\theta} \cos \theta = -\sin \theta$$

$$\frac{d}{dz} \ln z = \frac{1}{z} \quad \frac{d}{dz} z^n = n z^{n-1}$$

school fact: $\sin^2 \theta + \cos^2 \theta = 1$

And one matrix step: a row times a column means multiply the pairs, then add.

$$\begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = ac + bd$$

Example 1: a pendulum

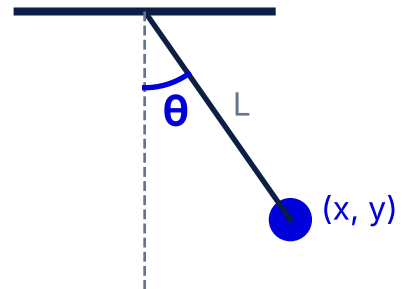
Knob = angle θ . Measurement = where the bob is.

Step 1 · Write the physics

A bob on a string of length L :

$$x = L \sin \theta$$

$$y = -L \cos \theta$$



Step 2 · Differentiate each line with respect to θ

How fast does each measurement change as the knob turns? (L is a constant.)

$$\frac{dx}{d\theta} = L \frac{d}{d\theta} \sin \theta = L \cos \theta$$

rule: $d/d\theta \sin \theta = \cos \theta$

$$\frac{dy}{d\theta} = -L \frac{d}{d\theta} \cos \theta = -L(-\sin \theta) = L \sin \theta$$

rule: $d/d\theta \cos \theta = -\sin \theta$

Stack these two rates in a column. That column is the Jacobian, J .

$$J = \begin{bmatrix} \frac{dx}{d\theta} \\ \frac{dy}{d\theta} \end{bmatrix} = \begin{bmatrix} L \cos \theta \\ L \sin \theta \end{bmatrix}$$

Step 3 · Build the metric: $G = J^T W J$

J^T (J transpose) = the same numbers laid out as a row. W = one weight per measurement. x and y are both in metres, so both weights are 1.

$$J^T = [L \cos \theta \quad L \sin \theta] \quad W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$G = [L \cos \theta \quad L \sin \theta] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} L \cos \theta \\ L \sin \theta \end{bmatrix} = L^2 \cos^2 \theta + L^2 \sin^2 \theta$$

$$G = L^2 (\cos^2 \theta + \sin^2 \theta) = L^2$$

school fact: $\sin^2 + \cos^2 = 1$

Step 4 · Put in a number

$$L = 2 \text{ m} \rightarrow G = 4, \sqrt{G} = 2$$

Turn the knob 0.01 rad $\rightarrow$ the bob moves $2 \times 0.01 = 0.02$ m, at every angle.



Twist: measure only the height

Same pendulum, poorer sensor. Now G changes with angle.

Step 1 · Write the physics

Height of the bob above its lowest point:

$$h = L - L \cos \theta$$

Step 2 · Differentiate with respect to θ

L is a constant, so its derivative is 0.

$$\frac{dh}{d\theta} = 0 - L \frac{d}{d\theta} \cos \theta = -L(-\sin \theta) = L \sin \theta$$

rule: $d/d\theta \cos \theta = -\sin \theta$

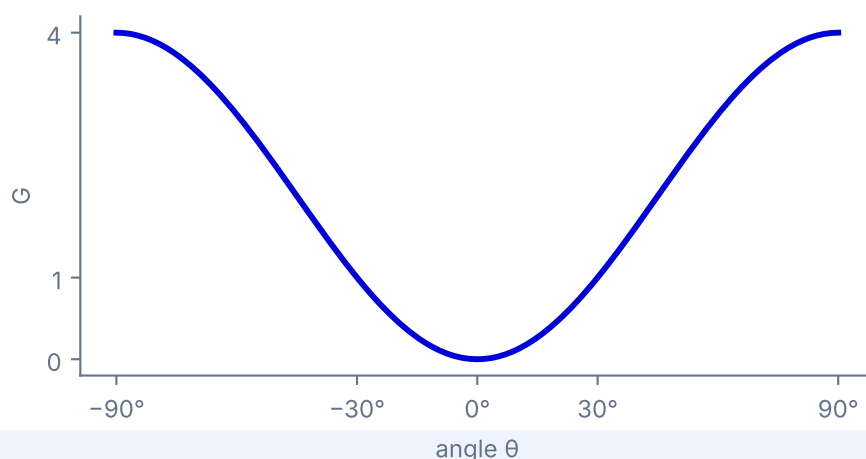
Step 3 · Build the metric

Same formula, $G = J^T W J$. With only one measurement, J , J^T and W are single numbers (weight = 1):

$$G = J^T W J = (L \sin \theta) \cdot 1 \cdot (L \sin \theta) = L^2 \sin^2 \theta$$

Step 4 · Put in numbers ($L = 2$ m, so $L^2 = 4$)

angle θ	$\sin \theta$	$G = 4 \sin^2 \theta$	height change for a 0.01 turn
0°	0	0	0 m
30°	0.5	1	0.010 m
90°	1	4	0.020 m



At the bottom, $G = 0$.

Height can't see
small swings there.

At the side, G is big.

Height sees θ clearly.

The lesson

G tells you how loudly the hidden knob shows up in your data.

Small G : the knob is hard to see. Large G : every tiny turn is obvious.



Example 2: a battery cell

Same four steps. Knob = z , the usable lithium left (1 = new, 0 = dead).

The full model reads 16 measurements. Here we use just 2:

x_1 = open-circuit voltage (volts) x_2 = SEI film resistance (ohms)

Step 1 · Write the physics (simplified, DFN-inspired)

$$x_1 = U_0 - \alpha \ln z \qquad x_2 = R_0 + \beta z^{-1/2} \qquad (\beta/\sqrt{z} \text{ written as } \beta z^{(-1/2)})$$

Step 2 · Differentiate each line with respect to z

$$\frac{dx_1}{dz} = 0 - \alpha \cdot \frac{1}{z} = -\frac{\alpha}{z} \qquad \text{rule: } d/dz \ln z = 1/z$$

$$\frac{dx_2}{dz} = 0 + \beta \cdot \left(-\frac{1}{2}\right) z^{-3/2} = -\frac{\beta}{2z^{3/2}} \qquad \text{rule: } d/dz z^n = n z^{n-1}, \text{ with } n = -1/2$$

Step 3 · Build the metric: $G = J^T W J$

Volts and ohms are different units, so W gives each its own weight: one for volts, one for ohms.

$$J = \begin{bmatrix} -\frac{\alpha}{z} \\ \beta \\ -\frac{\beta}{2z^{3/2}} \end{bmatrix} \qquad W = \begin{bmatrix} w_V & 0 \\ 0 & w_R \end{bmatrix}$$
$$G(z) = \begin{bmatrix} -\frac{\alpha}{z} & -\frac{\beta}{2z^{3/2}} \end{bmatrix} \begin{bmatrix} w_V & 0 \\ 0 & w_R \end{bmatrix} \begin{bmatrix} -\frac{\alpha}{z} \\ \beta \\ -\frac{\beta}{2z^{3/2}} \end{bmatrix} = \frac{w_V \alpha^2}{z^2} + \frac{w_R \beta^2}{4z^3}$$

Step 4 · Put in numbers ($\alpha = 0.1$, $\beta = 0.02$, both weights = 1)

cell state	z	$\alpha^2/z^2 + \beta^2/(4z^3)$	G
new	0.9	0.0123 + 0.0001	0.0125
mid-life	0.6	0.0278 + 0.0005	0.0282
near the knee	0.3	0.1111 + 0.0037	0.1148

New → near the knee: G grows 9.2×

Lose 0.01 of lithium: readings move $\sqrt{G} \times 0.01$

New cell: 0.0011 Near the knee: 0.0034 (about 3× more)

MW does NOT solve these equations

The examples show the meaning. Here is what actually runs.

Why not just solve them?

Pages 2 to 4 used 1 knob and 2 measurements. The real battery model reads 16 measurements and has about 15 uncertain physics parameters: SEI growth, plating, swelling, cracking, temperature sensitivity and more.

Brute force means trying combinations. Even 10 values per parameter gives:

$$10 \times 10 \times \cdots \times 10 \text{ (15 times)} = 10^{15} \text{ combinations}$$

Each one would need its own solver run. That is the wall today's approach hits.

What MW does instead

1 · Physics enters as priors

Each physics parameter gets a prior: a likely range, not a fixed number. Simple ageing laws link them to SoH. No PDE is solved.

2 · A Bayesian VAE learns by Monte Carlo

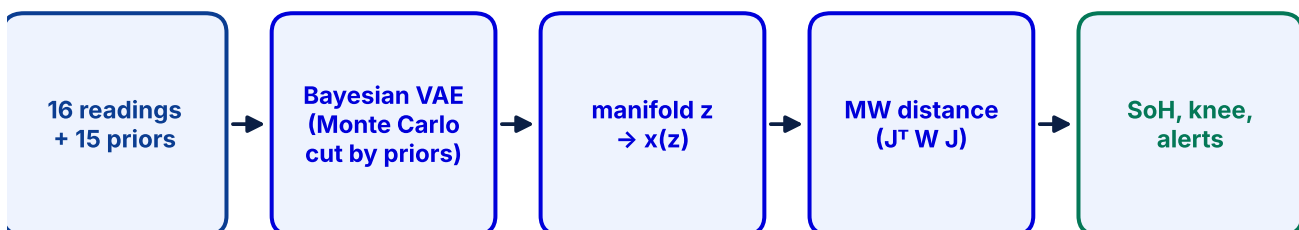
It draws random samples of parameters and hidden states. The priors cut away what physics forbids, so it only explores the physically possible region.

3 · Physics is tied to the manifold

The SoH the decoder produces must agree with the physics. So the learned surface $x(z)$, the Step 1 of earlier pages, obeys the physics.

4 · The pullback, in practice

MW distance compares decoded states with weights W (volts, ohms, days on one scale). For tiny steps this is exactly $G = J^T W J$.



The differentiator

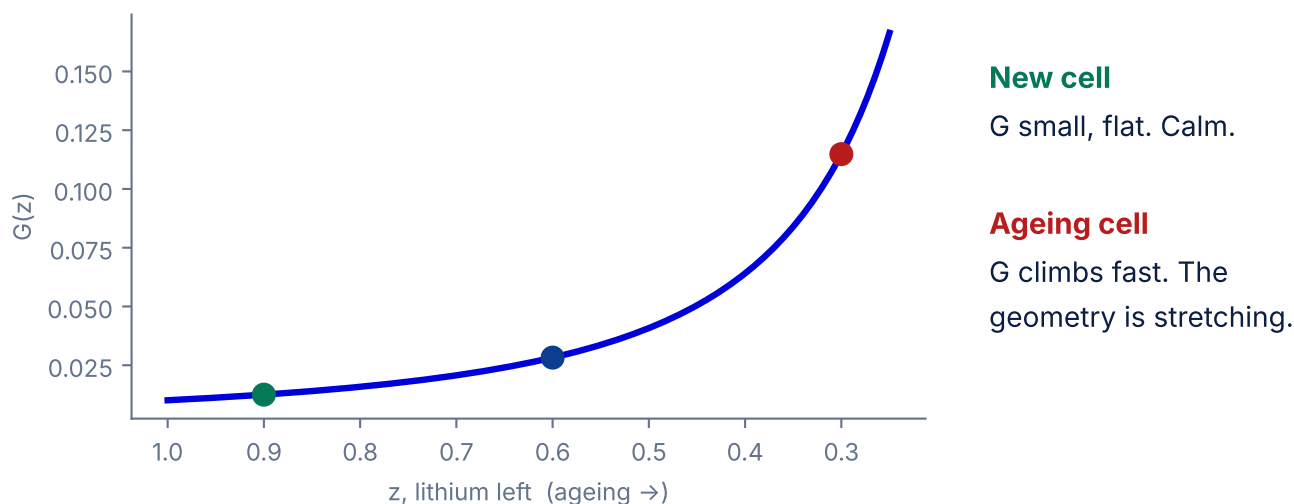
Solvers fight the PDEs every time. MW never solves them.

Priors cut the search; the geometry is measured on a CPU.



What it means, and how to use it

From one formula to an early warning



The curve bends up before the cell falls off its capacity cliff (the 'knee'). That bend is lead time: replace, re-rate or re-price the warranty before failure.

The recipe, for any physics

1. Pick what you measure, and what physics allows (the priors).
2. Generate data: solver runs (COMSOL / Ansys) and field telemetry.
3. Train the Bayesian VAE: Monte Carlo cut by priors learns $x(z)$.
4. Measure MW distance ($G = J^T W J$). No PDE solved.
5. Watch how G changes. The geometry tells the story.

With more knobs, G becomes a small table (a matrix). The idea is the same.

Why not LLMs?

This works because physics obeys rules written as PDEs. English grammar has no PDEs, so this geometry won't help a chatbot. This is physics AI, not language AI.

Learn it, use it

You need differential geometry (O'Neill, do Carmo) and probabilistic programming (Pyro). The full framework is open source:

github.com/RahulModak74/MW_Geometric_Intelligence

Bayesian's motto

Use whatever little maths we know to democratise, not to build moats.



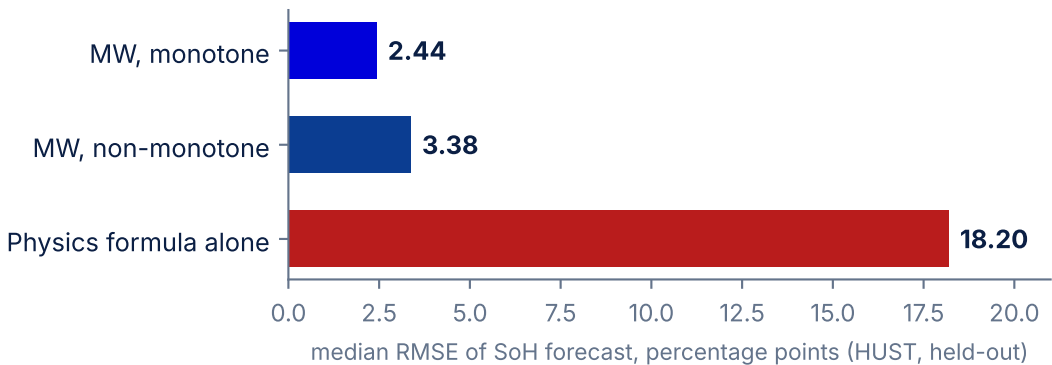
Proven on public data

95 LFP cells, two open datasets, held-out testing

	Sandia (Preger et al., 2020)	HUST (Ma et al., 2022)
Cells	18	77
Temperature	15, 25, 35 °C	30 °C
Shape of fade	steady, near-linear	slow, then a knee

Same A123 LFP 18650 cell in both. The task: from one measurement at 100 cycles, forecast the whole SoH path. Every score is on held-out cells.

Result: within 2.4 to 3.4 points, vs 18 to 21 for physics alone



Where the geometry earns its keep

The monotone model carries one strong prior: SoH can only fall. That keeps its forecast on a physically possible path, even when the inputs are unfamiliar.

Outside training data	Monotone	Unconstrained
Sandia	3.0	20.4
HUST (high C-rate cells)	3.5	5.7

Near the knee the picture flips: the unconstrained model bends faster and dates end of life better. So the engine runs both, routes by situation, and flags a pack when they disagree by more than about 5 points.

Full method and results: MW Battery Engine validation report, Sandia + HUST open data (October 2026).



The COMSOL / Ansys bottleneck

Keep the solvers. Use them differently.

The pain of today's approach

■ Stitching physics together

A battery is electrochemistry + heat + mechanics, each its own PDE. They must be coupled so the 3D model stays consistent. Coupled, stiff equations are hard to converge.

■ Meshing and setup

Every new geometry or design change means re-meshing and re-tuning. Experts spend days before the first useful run.

■ Compute and licence time

One detailed 3D run can take hours to days on workstations, clusters or GPUs. Licence seats and HPC capacity are tied up the whole time.

■ Every question is a new run

A what-if, a new load profile, a new temperature: each needs a fresh solve. Real-time monitoring of thousands of cells is out of reach.

The MW way: solvers teach, the manifold scores



	Solver alone	Solver + MW
Each new question	full re-solve	measure G on the manifold
Hardware per query	workstation / HPC / GPU	ordinary CPU
Licence time	every query	data generation only
Real-time fleet use	impractical	practical
Bad or faked data	not checked	off-manifold → flagged

Honest limit: the manifold knows only what the solver showed it. New regimes need new runs.

If your team runs COMSOL or Ansys

Count the runs that repeat the same physics with new inputs. They can train a manifold.