Bayesian Cybersecurity

THE MW FRAMEWORK · AN EBOOK · EDITION 1

Bayesian Geometry vs Gigawatts

Mathematics as an Alternative to GPU Capex

Physics, robotics and quantum-compute AI for teams without a data centre: the maths, the evidence from four open repositories, and what a laptop can already do.

THEORY · ROBOTICS · PHYSICS · QUANTUM · BATTERY

Rahul Modak
Dr Rahul Walawalkar
Co-developers of the Modak–Walawalkar (MW) framework

FOUR OPEN REPOSITORIES + ONE LIVE PRODUCTION APP · tap a name to open it

Theory	**MW_Geometric_Intelligence**
Robotics	**MW_Robotics**
Physics	**QUANTUM_GRAVITY_WITH_MW**
Quantum	**Quantum_Compute_MW**
Live app	**batteryanalytics.bayesiancybersecurity.com**

Open code, measured numbers. Please read the note on page 2.

Bayesian Cybersecurity Pvt Ltd, Thane · Edition 1, November 2026

WELCOME

Welcome

One idea, four repositories, and a laptop.

For two decades, the price of entry to frontier physics and AI has been a GPU data centre. **This book is a bet that mathematics can lower that price to a laptop.**

At Bayesian Cyber we firmly believe in using whatever little maths we know to democratise, not to build moats. Our tools are Pyro, PyTorch and a belief that the laws of physics belong to everyone. This book is our blueprint. **The code is open. The numbers are measured. We invite you to use it, break it, and build on it.**

Clusters and GPU fleets are wonderful tools, and many teams have them. Many brilliant teams do not: university labs without an allocation, start-ups whose runway cannot buy GPU hours, public institutions, and engineers in every region who wait in queues or cannot import the hardware. For all of them, this book is a different way in: spend mathematics instead of hardware.

The Modak–Walawalkar (MW) framework writes the physics you know as Bayesian priors, learns the surface of states that satisfy them, and measures that surface with geometry. The idea is old mathematics (Riemann's curved spaces, Bayes' rule) used in a new way, and it is tested in four open repositories that this book walks through, one by one, with ideas and worked examples.

Why the title? “Bayesian geometry” names the two ingredients. **Bayesian statistics** (the models are written in the probabilistic programming library Pyro and trained with Monte Carlo) says what we believe and how sure we are: physics enters as priors, and every answer carries uncertainty. **Riemannian geometry** says how to measure: the learned surface of valid states has a metric, so distances, speeds and curvature mean something. Together they replace brute-force compute with structure. “Gigawatts” stands for the data-centre side of the bargain.

The numbers at a glance

7 vs 130,000
core-hours: learn the Kerr family vs one typical numerical-relativity run

58 / 58
simulated humanoid tests, student at 2.3 ms per step on one CPU thread

2.3 ms
per battery forecast on a laptop CPU, in production

1000 qubits
low-entanglement circuits on one laptop, with an error bar

Where each repository stands

Repository	Role	Status in one line
MW_Geometric_Intelligence	Theory	Formal framework, primers, and the 1000-qubit logs. The mathematics lives here.
MW_Robotics	Robotics	A small CPU policy plus a physics “flight computer”: 58 of 58 simulated humanoid tests.
QUANTUM_GRAVITY_WITH_MW	Physics	Gravity and quantum mechanics in one learned object, and Kerr black-hole geometry and waveforms against exact answers.
Quantum_Compute_MW	Quantum computing	A CPU toolkit, `mwqc`: 1000-qubit low-entanglement circuits on a laptop, with an error bar on every result.
Battery analytics (live)	Production use	The Xtreme Analytics platform: battery health and life forecasts from ordinary BMS logs, in use on fleet packs.

How this book is organised

Chapter	What it does
1 · Maths for GPU-constrained labs	Why MW is different, and why Bayesian statistics and differential geometry are the leverage.
2 · From an equation to MW	A battery-led walk from one equation to a heat-rod PDE, the mesh, then MW.
3 · Use MW on your own physics	A five-step recipe worked on a wind turbine, and the live battery platform.
4 · Commercial: battery analytics, battery cyber, RF warfare	Live battery platform with clients, plus battery cybersecurity and RF electronic warfare, open for pilots.
5 · A humanoid on a laptop	A distilled student that runs on a CPU, and how to reproduce it.
6–7 · Quantum	What a laptop lets you study: Dr Zickert's books first, then litho, chemistry and spin chains.
8 · The compute gain, join in, open core	The arithmetic with Kerr black holes, every headline number, a bounty board, and how the work is sustained.

OUR OPEN SCIENCE PLEDGE

The code is open. Four repositories, linked on the cover, with a fifth live in production.

The numbers are measured. Every figure in this book traces to a file you can open and a command you can run.

The frontiers are marked. Where a result is proven in simulation, or the next milestone is still ahead, we say so once and invite you to help cross it. Part VII is the bounty board.

Your work is credited. Reproductions, improvements and solved problems are named in the next edition.

HOW TO READ THE NUMBERS

Every number comes from a file in one of the repositories, and the file is named. Chapter 8 collects them in one place.

Contents

Welcome 2
Contents 4

Part I · Why this book **5**
Maths for GPU-constrained labs 6

Part II · The maths **9**
From an equation to MW 10

Part III · Physics AI **16**
Use MW on your own physics 17
Commercial: battery analytics, battery cyber and RF warfare 20

Part IV · Robotics **28**
MW_Robotics: a humanoid on a laptop 29

Part V · Quantum **34**
Quantum on a laptop: the idea 35
Use cases: from the books to litho 37

Part VI · The compute gain, and joining in **41**
The compute gain, joining in, and the open-core model 42
Glossary 51
Sources and links 52

PART I

Why this book

For labs and teams that have brilliant ideas and no GPU cluster: what is different about MW, and what the maths adds.

1 · Maths for GPU-constrained labs

PART I · CHAPTER 1

Maths for GPU-constrained labs

Why MW is different, what the maths adds, and why Bayesian statistics and differential geometry are the two pieces that matter.

QUICK ANSWER

Physics AI usually means a solver on a cluster plus a big network on GPUs. MW swaps part of that bill for mathematics: write the physics as Bayesian priors, learn the surface of valid states once, and read answers off it in milliseconds on a laptop.

> **“Compute is a good tool. It should not be the only door.”**

1.1 The compute gate

Frontier physics and robotics AI is built in clusters and GPU data centres. If a problem is hard, the standard answer is a bigger solver run on a bigger machine. That works if you have the machine. Many teams do not: university labs without an allocation, start-ups whose runway cannot buy GPU hours, public institutions, and engineers in every region who wait in queues or cannot import the hardware.

We call this the compute gate. This book is about a second way in. It does not argue against GPUs, which are excellent at what they do. It adds a choice: before you buy compute, see how much a better description of the physics can do for free.

1.2 What is different about MW

Almost every physics-AI recipe has two expensive stages: a numerical solver that generates answers, and a large network trained on those answers. MW changes the shape of the problem in three ways:

- **The physics goes in as priors,** not as more data. A law becomes a score the model must satisfy, so a few hundred solver runs are enough to learn the surface.
- **Laws add.** Two theories from different fields become two terms in one objective. Coupling two PDE solvers means writing an interface; coupling two priors means writing one more line.
- **Answers come from a surface, not a solve.** After one-time training, every question is a single pass through a small network, in milliseconds, on a CPU.

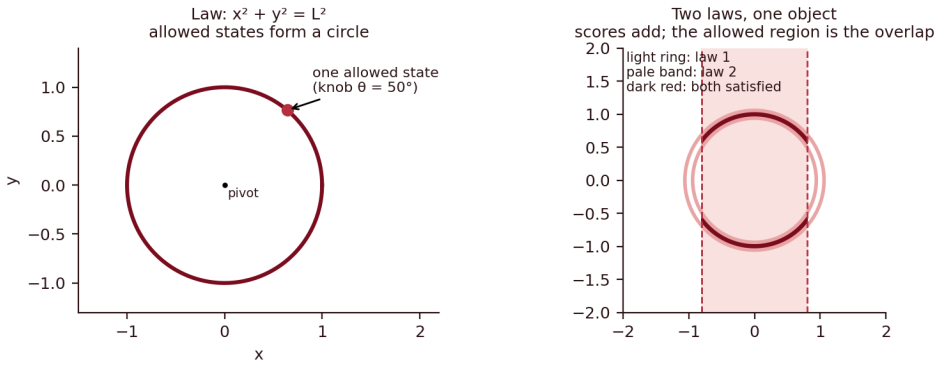


Figure 1. Left: one law carves out a circle of allowed states, and one knob (θ) walks along it. Right: two laws overlap, and the states obeying both are the dark region.

1.3 What the maths adds

Most physics-AI writing treats the network as a black box. Two pieces of mathematics open the box, and together they are the leverage behind everything in this book:

	Bayesian statistics	Differential geometry
The question it answers	What do we believe, and how sure are we?	What shape is the set of valid states, and how do we measure on it?
In MW	Physics enters as priors; every answer carries a likely range	The valid states form a smooth surface (a manifold); distances, speeds and curvature mean something
Tool	Pyro and Monte Carlo on a laptop	The decoder of the trained model is a smooth map, so its slopes exist and can be used
What you gain	Few runs needed; uncertainty for free	One chart of the physics with exact derivatives, like a mesh of any resolution with no meshing step

1.4 Why Bayesian matters

A prior is a statement of belief before looking at data: “voltage only falls as the cell ages”, “power cannot exceed the Betz limit”, “a robot's weight must stay over its feet”. In probability language a prior is a penalty that grows as a state strays from the law. **That is how physics becomes something a network can learn from.** Because the whole model is probabilistic, it also returns a likely range with every answer, and says when a new reading looks unfamiliar.

1.5 Why differential geometry matters

Think of a map. A flat map of a hill is useless without its contour lines, because contour lines tell you how fast the ground rises. A trained model has the same structure. Its decoder turns a few knobs into the full set of measurements, and the set of all outputs is a curved surface. Differential geometry is the maths of such surfaces. It tells you that the surface has a shape, that you can measure along it, and that a new reading can sit on it or off it. **You never need to draw the surface by hand.** It exists because the decoder is a smooth map, and that is enough.

> **WHY BOTH, TOGETHER**
>
> Bayesian statistics gives the model its physics and its honesty about uncertainty. Differential geometry gives it a shape and a ruler. Either one alone is a good tool. Together they replace much of the brute force that physics AI has relied on.

THE DEMOCRATISATION DIVIDEND

Picture a brilliant student in a university lab in Pune, Nairobi, Kraków or Lima. The idea is sound, the physics is right, and the H100s are not coming. **MW is the equaliser.** A battery forecast in 2.3 milliseconds on a laptop CPU. A humanoid student policy that runs on one CPU thread. A 1000-qubit circuit on the machine in front of you. These were the preserve of clusters; with the right mathematics they are open to any team with a laptop and curiosity.

That is the dividend: not only a smaller bill, but more people able to do the science.

1.6 How this book is laid out

Chapter	What you get
2 · The maths	One chapter: a battery-led walk from one equation to a heat-rod PDE, the mesh, then MW
3 · Physics AI on your own equations	A step-by-step recipe, worked on a wind turbine, plus the live battery platform
4 · Commercial: battery analytics, battery cyber, RF warfare	The live battery platform, plus battery cybersecurity and RF electronic warfare
5 · Robotics	A distilled humanoid student that runs on a CPU, and what you can do with it
6–7 · Quantum	What a laptop lets you study, starting with Dr Zickert's books, then litho, chemistry and spin chains
8 · The compute gain, join in, open core	The arithmetic with Kerr black holes, every headline number, a bounty board, and how the work is sustained

THE COMPUTE ANGLE

The cost model of the whole book: pay once to learn the surface, then every question is nearly free. The chapters ahead show where that trade is a big win.

PART II

The maths

One chapter, one thread: a battery. From a single equation, to a PDE on a mesh, to the MW surface. First-year level.

2 · From an equation to MW

PART II · CHAPTER 2

From an equation to MW

Two ways to understand a physical system: walk it step by step, or learn its shape once.

QUICK ANSWER

A PDE solver chops the problem into small cells and marches forward in time, one local step at a time. MW learns the whole set of physically valid states as a smooth surface, and measures that surface with a ruler instead of marching along it. We use a battery as the running example, and start from a metal rod because it is the easiest PDE to see.

Part A: how a PDE solver works

2.1 Start tiny: one number (an ODE)

A battery's capacity fades at a rate proportional to what is left: **dQ/dt = −k·Q**. One unknown, one number to step forward. If Q = 100 today and k = 0.01 per day, tomorrow it is about 100 − 0.01×100 = 99. Repeat. That is all an ODE solver does: **new value = old value + (time step) × (rate)**.

2.2 The heat rod: where the PDE comes from

Take a thin metal rod, one end in ice, the other end hot. The heat equation is two physical laws combined, and nothing else goes in.

- **Law 1, energy conservation.** A thin slice's heat content changes only if more heat flows in one side than out the other: $\rho \cdot c \cdot \partial T/\partial t = -\partial q/\partial x$. Here ρ is density, c is specific heat (the energy to warm 1 kg by 1 degree), q is heat flux (how much heat crosses a cross-section each second).
- **Law 2, Fourier's law.** Heat flows from hot to cold, faster when the temperature changes steeply: $q = -k \cdot \partial T/\partial x$. The minus sign means heat runs downhill, and k is the thermal conductivity.

Put Law 2 into Law 1 and, with constant k, you get the heat equation:

$$\partial T/\partial t = \alpha \cdot \partial^2 T/\partial x^2, \quad \alpha = k / (\rho \cdot c)$$

α is the thermal diffusivity, how fast temperature differences even out. Copper is about 1×10^{-4} m²/s and steel about 1×10^{-5}, so copper levels out roughly ten times faster.

Reading the symbols in plain words. ∂T/∂t is how fast one spot is warming right now. ∂T/∂x is the slope of the temperature profile. $\partial^2 T/\partial x^2$ is the curvature, the change of the slope. Read it as *“my neighbours' average minus me”*: if you sit in a dip you warm up, if you sit on a peak you cool, and a straight-line profile does not change at all. So the whole PDE says: **every point drifts toward the average of its neighbours, at speed α**.

2.3 What a solver does: the mesh

T(x,t) has a value at infinitely many points, and a computer holds finitely many numbers. As written, the PDE is a statement about a continuous curve, not a procedure. The solver turns the statement into arithmetic in four moves.

Move 1: draw the shape. The rod is the domain. Fix the boundary conditions (both ends at 0 °C, touching ice) and the starting state (the middle is hot).

Move 2: chop it into cells, the mesh. In the finite-element method the pieces are elements, in the finite-volume method they are control volumes or cells, and the points where they meet (or their centres) are nodes. We use 5 nodes 1 cm apart (h = 1 cm). Nodes 1 and 5 are the ice, nodes 2, 3, 4 are unknown.

Move 3: replace slopes with neighbour arithmetic. The curvature at node i becomes (T[i−1] − 2·T[i] + T[i+1]) / h², which is (2/h²) × (neighbours' average − me), the plain-words reading above. The one PDE is now three coupled ODEs, each like dQ/dt = −kQ but linked to its neighbours. This conversion is the *method of lines*.

Move 4: step forward in time. The simplest rule is explicit Euler. With α = 1 cm²/s, h = 1 cm and dt = 0.2 s, r = α·dt/h² = 0.2 and each node updates by T_new[i] = T[i] + r·(T[i−1] − 2·T[i] + T[i+1]). By hand, node 3: 100 + 0.2·(0 − 200 + 0) = 60, so the peak loses 40, and node 2 gains 20. In code it is a few lines:

```
import numpy as np
alpha, h, dt = 1.0, 1.0, 0.2          # cm^2/s, cm, s
r = alpha*dt/h**2                      # 0.2  (must be <= 0.5)
T = np.array([0., 0., 100., 0., 0.])   # nodes 1..5, ends held at 0 C
for step in range(3):
    T[1:-1] = T[1:-1] + r*(T[:-2] - 2*T[1:-1] + T[2:])
    print(f"t={0.2*(step+1):.1f}s", T)
```

Time	T1 (ice)	T2	T3	T4	T5 (ice)
0.0 s	0	0	100	0	0
0.2 s	0	20	60	20	0
0.4 s	0	24	44	24	0
0.6 s	0	23.2	36	23.2	0

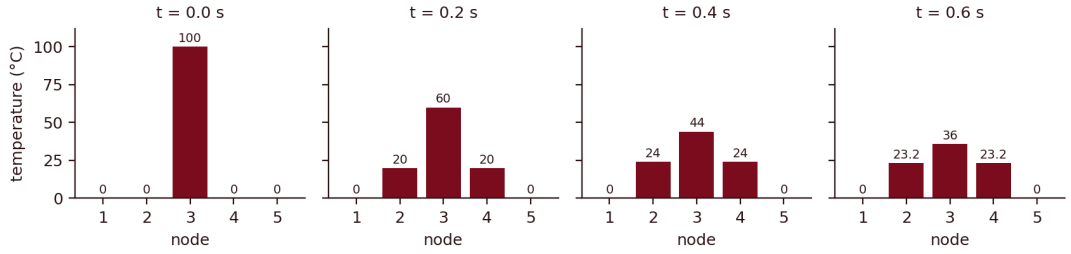


Figure 2. The hot spot spreads and flattens. The solver computes this node by node, step by step, and the heat slowly drains into the ice-cold ends, as physics demands.

2.4 The cost, and what it means for a battery

Stability. The simple rule works only while r = α·dt/h² ≤ 0.5. With dt = 0.6 s, r = 0.6 and after five steps the middle nodes read 48.9, −57.6, 48.9: the arithmetic has blown up. **Refining is expensive.** Halve h for better accuracy and dt must shrink 4×, so twice the nodes means about eight times the work. **Implicit methods** take big steps safely by solving a linear system for all nodes at once every step. The table of how each node's equation depends on each node's unknown is the *Jacobian*. For nonlinear problems it changes each step and the solver re-linearises with Newton's method, which is what “evaluating tangents” means in a solver.

The Doyle–Fuller–Newman (DFN) battery model uses the same template everywhere: **rate of change = (flow in − flow out) + source**.

Heat rod	Battery DFN
T (temperature)	c_s (lithium in the solid), c_e (lithium in the electrolyte), potentials φ_s, φ_e
$\alpha\ \partial^2T/\partial x^2$ (heat flows downhill)	$D\ \partial^2c/\partial x^2$ (lithium flows from high to low concentration: Fick's law), plus Ohm's law for charge
Ends held at 0 °C	Current drawn at the electrode surface
No source	Reaction source at each particle surface (Butler–Volmer)
1 unknown per node	Several unknowns per node
Mesh along x only	Mesh along x (across the cell) and along r (inside every particle): “pseudo-2D”

Lithium diffusing inside a particle is the heat equation with T replaced by concentration and α replaced by D. Degradation (SEI film growth, lithium plating) adds slow ODEs on top. The result is thousands of coupled unknowns, some of them algebraic constraints. Fast diffusion and slow ageing together make the system **stiff**: small steps for the fast physics while following thousands of cycles of the slow physics.

Where the GPU fits. Every mesh cell does the same arithmetic on its own data, so a GPU can update thousands of cells at the same instant. That speeds up each time step, and bigger meshes (a full 3D thermal model of a pack, say) benefit most. The time steps themselves still run one after another.

> **MAP ANALOGY, PART A**
> Old-style map making: a surveyor walks the ground, measures elevation and slope at one point, moves to the next, and stitches the pieces together. A finer map means more points and more walking, and a new area means starting the walk again. The mesh is the survey grid, the time-stepping is the walking, and the GPU is hiring many surveyors to measure grid points at once.

Part B: what MW does instead

2.5 A hidden knob and the things we measure

Every physical system has a hidden knob and things we can measure. A battery's knob is **z**, the usable lithium left (z = 1 is new, z = 0 is dead). We rarely see z directly. We see its effects: a voltage, a resistance. The question MW asks is: **if I turn the hidden knob a tiny bit, how much do my measurements move?** The answer is one number, the metric G. General Relativity uses the same mathematics (Riemann's geometry) to describe curved spacetime. MW borrows it for any physics.

Think of two spaces. The knob space has the coordinate z. The measurement space has coordinates (x_1, x_2) = (voltage, resistance), and a map x(z) says what the sensors read for each knob setting. In measurement space we already know how to measure length: square each change and add them up, with weights W so that volts and ohms compare fairly. The *pullback* carries that ruler back to knob space and answers: how long is a small step dz, as seen through the sensors?

$$G = J^{\mathsf{T}} \cdot W \cdot J,\quad J = \left(\frac{dx_1}{dz}\ ,\ \frac{dx_2}{dz} \right)^{\mathsf{T}}$$

J is the Jacobian, the column of slopes of each measurement with respect to the knob, and one row-times-column step is all the matrix work needed.

Worked example, a battery cell. Take open-circuit voltage $x_1 = U_0 - \alpha \cdot \ln z$ and SEI film resistance $x_2 = R_0 + \beta \cdot z^{-1/2}$. Three rules differentiate them ($d/dz \ln z = 1/z$, $d/dz\, z^n = n \cdot z^{n-1}$, constants vanish): $dx_1/dz = -\alpha/z$ and $dx_2/dz = -\beta/(2 \cdot z^{3/2})$. Giving volts and ohms their own weights (both 1 here), multiply the pairs and add:

$$G(z) = \alpha^2 / z^2 + \beta^2 / (4 \cdot z^3)$$

With $\alpha = 0.10$ and $\beta = 0.020$:

Cell state	z	G(z)
New	0.9	0.0125
Mid-life	0.6	0.0282
Near the knee	0.3	0.1148

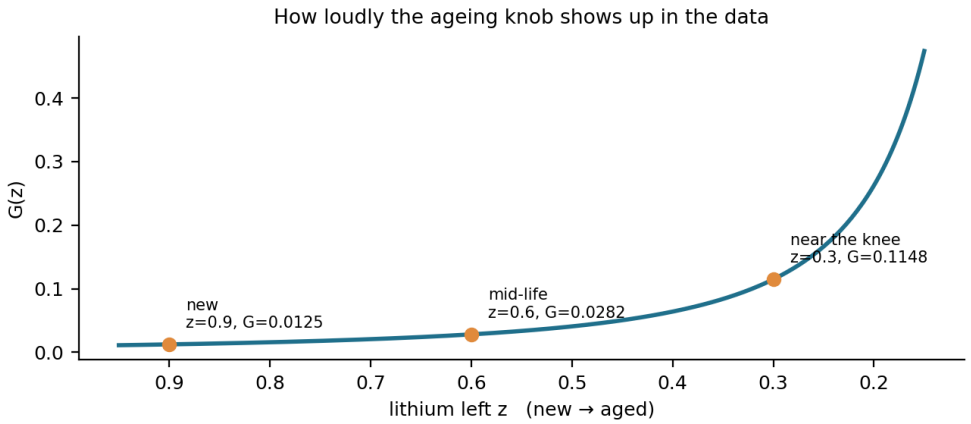


Figure 3. G(z) is calm for a new cell and climbs fast as the cell ages.

From new to the knee, G grows 9.2×: the same 0.01 of lithium lost moves the readings about 3× more (that is √G, the length scale). The nearer the end, the louder the ageing. The curve bends upward before the cell reaches its capacity cliff, and that bend is **lead time**: replace, re-rate or re-price the warranty before failure.

2.6 Why this makes the VAE Riemannian

In the example we wrote x(z) by hand. In MW the **VAE decoder** plays that role: a smooth function that takes the knobs z and produces the full set of measurements x(z). That gives three facts. **The map exists and is smooth**, so it has a Jacobian J everywhere, which the chain rule and automatic differentiation provide on demand. **So the pullback exists**: once J exists, $G = J^T W J$ exists, and a latent space equipped with a pullback metric is by definition a Riemannian manifold. **It need not be computed to be true**: the engine works with MW distance (decoded states compared with the weights W), and for tiny steps that distance is exactly G. The geometry is present in the model even when we never write out the matrix.

A PDE solver, by contrast, never builds a map from knobs to measurements. It has no ruler on the space of solutions. It marches along one path using local slopes for the next step.

2.7 What actually runs in MW

The real battery model reads 16 measurements with about 15 uncertain physics parameters: SEI growth, plating, swelling, cracking, temperature sensitivity. Ten values each gives 10^{15} combinations, and a solver would need one run for each of them. MW needs none of that:

- **Physics enters as priors.** Each parameter gets a likely range, not a fixed number. No PDE is solved inside MW.

- **A Bayesian VAE learns by Monte Carlo.** The priors cut away what physics forbids, so it explores only the physically possible region.
- **Physics is tied to the manifold.** The state of health the decoder produces must agree with the physics, so the learned surface x(z) obeys it.
- **Geometry is measured, not marched.** G = $J^T W J$ tells how the data moves as the knob turns.

The recipe, for any physics: pick what you measure and what physics allows (the priors); generate data from solver runs (your finite-element, CFD or battery-model solver) and field telemetry; train the Bayesian VAE; measure the MW distance. With more knobs, G becomes a small table (a matrix), and the idea is the same.

2.8 Evidence on public battery data

On 95 LFP cells from two open datasets, with held-out testing, the task was: from one measurement at 100 cycles, forecast the whole state-of-health path. Median error on held-out HUST cells, in percentage points (lower is better):

Method	Error (pp)
MW, monotone	2.44
MW, non-monotone	3.38
Physics alone	18.2

MW keeps your solver in the loop and uses it differently: a few hundred validated runs teach the surface once, then new readings are scored against it without re-solving. A new chemistry or operating range is the next milestone, taught the same way with its own runs.

Part C: side by side

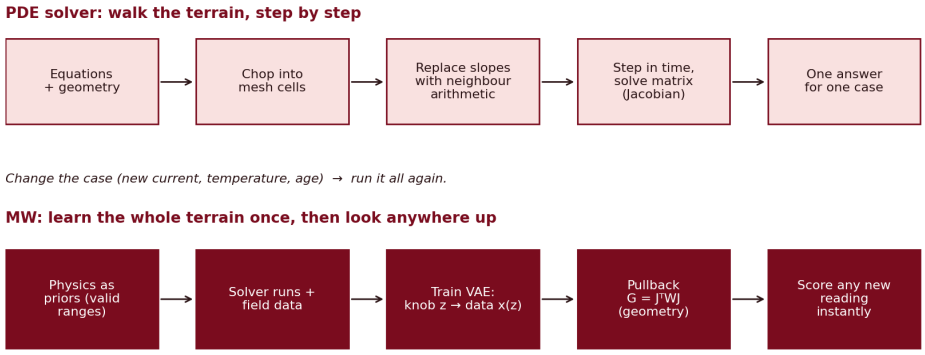


Figure 4. The two workflows. The solver route re-runs for every new case. The MW route pays once to learn the surface, then answers each question in a pass.

	PDE solver	MW
Starts from	Equations and a geometry	Equations as priors, plus solver runs and data
Builds	A mesh of cells	A smooth map x(z) from knobs to measurements
Core operation	Local slopes (Jacobian) to step forward	Pullback metric G = $J^T W J$ to measure the surface
Result	One trajectory for one case	The whole valid region, queried at any point
A new case	Solve again	Look up on the learned surface
Has a ruler on the solution space?	No	Yes: the pullback

	PDE solver	MW
Cost grows with	Finer mesh, smaller steps, stiffness	How well the priors and training data cover the region
Where hardware helps	GPU updates many mesh cells at once	Laptop CPU for queries; one-time training

MAP ANALOGY, COMPLETE

A PDE solver is the surveyor walking the ground point by point, measuring slope and elevation, then stitching the pieces. MW is the satellite photograph combined with software: the whole terrain is captured as one object, and distances and slopes can be read off it anywhere. The photograph is as good as what the satellite has seen, so the more regimes it has seen, the further it reaches.

2.9 Where to go deeper

The theory repository **MW_Geometric_Intelligence** holds the formal framework, primers and worked examples, including the battery tutorial. Next, Chapter 3 shows how to apply the same recipe to your own equations, with a wind turbine as the worked case.

THE COMPUTE ANGLE

A solver pays again for every new question. The learned surface is paid for once. If you ask a few questions, use the solver. If you ask thousands on the same physics, the surface starts to win.

PART III

Physics AI

Take any equation you solve today, and get the MW surface for it. A wind turbine as the worked case, and the commercial offerings: battery analytics (live), battery cybersecurity and RF electronic warfare.

3 · Use MW on your own physics
4 · Commercial: battery analytics, battery cyber and RF warfare

PART III · CHAPTER 3

Use MW on your own physics

A five-step recipe, worked end to end on a wind turbine, then the same recipe live in production on batteries.

IF YOU ARE SOLVING ANY PDE-DRIVEN CLUSTER WORK, TRY MW.

Heat, flow, stress, diffusion, electrochemistry, fields, orbits: if your results come from a solver that runs again and again on a cluster, the MW recipe is built for you. Keep your solver. Use a few hundred of its runs to teach MW the surface once, then ask your questions in milliseconds, on a laptop.

3.1 The recipe

Step	What you do	Wind-turbine example
1. Pick measurements	Choose the 4 to 20 numbers you care about	Wind speed, rotor speed, blade pitch, rotor power
2. Write the laws as priors	What must stay positive, balance, or stay below a bound	Power can never exceed the Betz bound, 16/27 of the wind power through the rotor; power is never negative
3. Sample your solver	A few hundred to a few thousand runs across the inputs that matter	Blade-element or CFD runs across wind speed, tip-speed ratio and pitch
4. Train a Bayesian VAE	Pyro, on a laptop, with the laws as extra terms	3 knobs, 4 readings, one Betz term
5. Query and measure	Ask what-ifs in milliseconds; check against solver runs it has not seen	Held-out power error, and how many random states break the Betz bound

3.2 Worked example: a 2 MW-class wind turbine rotor

A wind turbine turns moving air into rotor power. The standard result is $\mathbf{P = ½\ \rho\ A\ C_p\ v^3}$, where ρ is air density, A the swept area, v the wind speed and C_p the power coefficient. C_p depends on the tip-speed ratio $\lambda = \omega R/v$ (how fast the blade tip moves compared with the wind) and on blade pitch β. The flow around the blades is governed by PDEs (Navier–Stokes, or the blade-element-momentum model that simplifies them), which is why turbine design runs on clusters.

The physics gives a hard law: **no rotor can extract more than 16/27 ≈ 59.3% of the wind power** (the Betz limit). That one sentence is a prior. Everything the learned surface produces must sit below the Betz curve.

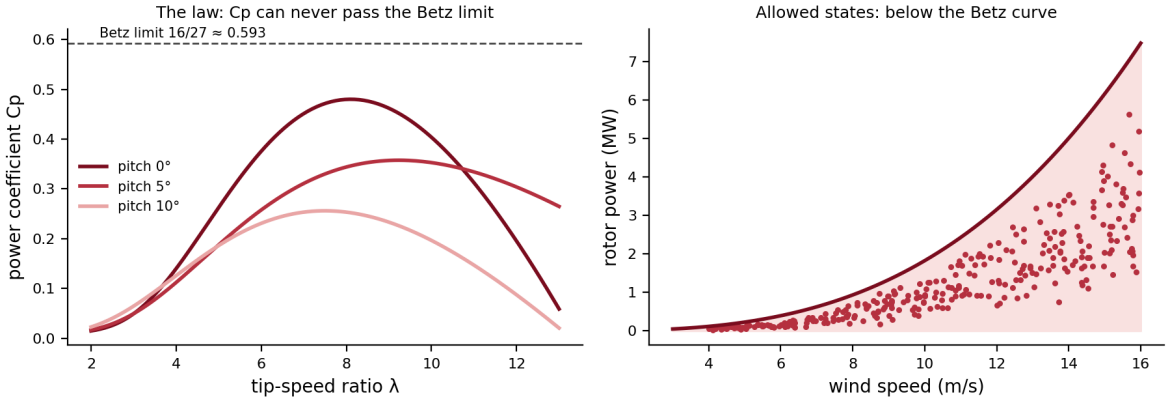


Figure 5. Left: the power coefficient for three pitch angles, always under the Betz limit. Right: the allowed region of the (wind speed, power) plane; every operating state lives below the curve.

In the demo, the solver stand-in is the standard closed-form $C_p(\lambda, \beta)$ model so that anyone can run it. In your own project you would swap in your blade-element or CFD runs. The measurements are four numbers (wind speed, rotor speed, pitch, power), and the three free inputs are the knobs. The key lines of the Pyro model add the law as one more term:

```
def model(x):
    pyro.module("dec", dec)
    with pyro.plate("d", x.shape[0]):
        z  = pyro.sample("z", dist.Normal(torch.zeros(3), 1.).to_event(1))   # 3 knobs
        xh = dec(z)                                                          # decoded readings
        pyro.sample("x", dist.Normal(xh, 0.05).to_event(1), obs=x)           # fit the solver r
uns
        # the law, as one more term: penalise power above the Betz bound
        pyro.factor("betz", -W * relu(power(xh) - betz_bound(xh))**2)
```

Results from the demo script (6,000 training runs, 1,500 held-out, CPU only):

	Without the Betz term	With the Betz term
Training time (2,000 steps)	44 s	27 s
Held-out power error (RMSE)	0.024 MW	0.026 MW
Random knob settings decoding above the Betz bound	0.36%	0.00%
Cost to decode one state	1.10 µs	1.05 µs

Read the table the way you would read an engineering report: the surface reproduces held-out solver runs, the law keeps the surface honest, and a state costs microseconds to decode. Swap in your own solver and your own bound (rated power, cut-in speed, a load limit) and you have the same experiment on your turbine.

3.3 What to do next with a turbine

- **Wake and farm layout:** add the wind speed at the downstream rotor as a knob, with a momentum-deficit prior.
- **Fatigue loads:** add blade-root bending as a measurement and a bound from the design envelope.
- **A digital twin from SCADA data:** place live readings on the surface and read the distance from normal; large distances flag yaw misalignment or sensor drift.
- **Design sweeps:** thousands of pitch and rotor-speed schedules, each a millisecond instead of a solver run.

TRY IT YOURSELF

Run the wind-turbine demo yourself (about 5 minutes on a laptop CPU):

```
pip install torch pyro-ppl numpy
git clone https://github.com/RahulModak74/MW_Geometric_Intelligence
cd MW_Geometric_Intelligence/WIND_TURBINE_EXAMPLE
python3 turbine_mw.py     # prints the table above
# file: MW_Geometric_Intelligence/WIND_TURBINE_EXAMPLE/turbine_mw.py
```

3.4 The same recipe, other equations

If you solve…	Measurements	A law to write as a prior
Heat exchangers	Inlet and outlet temperatures, flow rates, duty	Energy balance; outlet cannot beat the counter-flow limit
Structures (FEM)	Loads, displacements, stresses	Equilibrium; stress below yield; symmetry
CFD parametric sweeps	Drag, lift, pressure drop for geometry parameters	Mass conservation; positivity; known limits
Electromagnetics	Field amplitudes at ports	Reciprocity; passivity; energy conservation
Reservoir flow	Pressures, rates, saturations	Mass balance; saturation between 0 and 1
Battery degradation	Voltage, resistance, temperature, throughput	Capacity only falls; resistance only rises

3.5 The same recipe, live in production

The next chapter shows this recipe running in production on batteries: a platform that turns an ordinary BMS log into health, remaining life and cell diagnostics in seconds, on a laptop CPU.

THE COMPUTE ANGLE

A turbine rotor or a battery cell costs a solver run for every question. After the surface is learned, a question costs microseconds to milliseconds. If your work is a loop of solver runs, this is the first thing to try.

PART III · CHAPTER 4

Commercial: battery analytics, battery cyber and RF warfare

Live app: batteryanalytics.bayesiancybersecurity.com · business guide: “Xtreme Analytics”, Edition 4

QUICK ANSWER

This chapter covers the three commercial offerings: battery analytics (live, with clients), battery cybersecurity and RF electronic warfare (open for first pilots). Battery analytics first. The battery platform is the MW framework in production: it reads an ordinary 12–24 hour BMS log and returns health, remaining life, cell problems and data-quality checks in seconds, with no sensors to add. For LFP fleets it is in production, with forecasts validated on public Sandia and HUST data: about 2.4 points typical error on held-out cells, where the physics formula alone gives 18.

Why batteries are a good test

A battery is a system whose important state, how much usable capacity is left, is hidden, and whose ageing is the slow result of coupled physics (film growth, lithium loss, plating, cracking, heat). The physics-model route is the DFN solver of Chapter 2: expensive, parameter-hungry, and awkward to run across thousands of packs. The MW route is the one in Part IV: priors for the ageing physics, a learned surface of valid ageing states, and a log that places a pack on it. Batteries are also where real users, real fleets and real money meet the method, so it is where a claim can be checked in the field.

What the platform does

You upload a 12–24 hour BMS or CAN log in almost any column layout, choose the chemistry, and choose *Quick*. In seconds you get one page written for the person who pays for the battery. Tap *See detailed analytics* for ten engineering tabs on the same log. The business guide maps each business question to a screen:

The business question	Who usually asks	Quick page	Detailed tab
Is my battery healthy? The BMS says 100%.	Owner, fleet manager, buyer	Health tile	Pack overview
How long will it last?	Asset manager, CFO, lender	Life tile, outlook	Life forecast
What is this battery worth?	Buyer, refurbisher, insurer, lender	(detailed only)	Battery value · Life forecast · Cell internals
What warranty can I offer?	OEM, pack maker, insurer	Life + use tiles	Life forecast · Physics Lab
Which cell is the problem?	Maintenance lead	Cell grid, What to do	Pack overview · Cell internals
Weak cell or balancing job?	Operations, technician	Energy tile, balance note	Thermal & balance · Telemetry replay
Is it running hot? Where?	Maintenance, safety officer	Temperature line	Thermal & balance
Which habit should I change?	Operations, fleet manager	—	Capacity & usage · Virtual Lab

The business question	Who usually asks	Quick page	Detailed tab
Is the data trustworthy?	Security, auditor, insurer	Checks flagged	Pack overview · MW manifold & physics
Can I quote this number?	Anyone quoting a number	Ranges on tiles	MW manifold & physics · Data & conversion
How do I do this for 100 packs?	Fleet owner, OEM service head	—	Data & conversion · Fleet

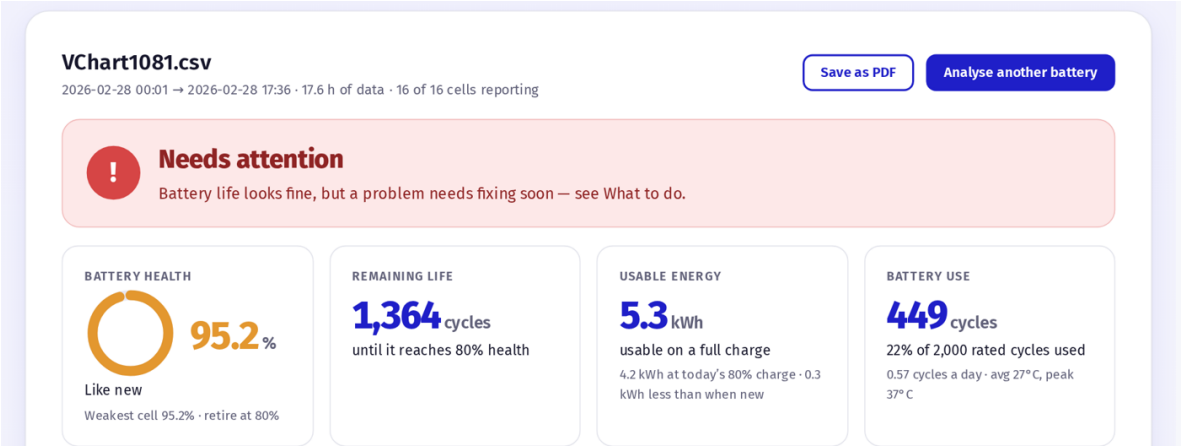


Figure 6. The quick page for a sample log: a health tile (here 95.2%), remaining cycles to 80% health, usable energy and battery age, with a “what to do” note. Screenshot from the live platform's business guide.

The MW ideas, as you see them on the screen

- **A log is a position fix.** Think of a doctor: today's blood pressure does not tell your whole future, but combined with what the doctor knows about people like you, it places you on a familiar path. The platform already knows how cells of your chemistry age (physics built into a compact statistical model), and your log places this pack on that path. *Chapter 2's knob z is that path.*
- **Each forecast carries a likely range,** because the model is Bayesian. *Chapter 2's uncertainty.*
- **One profile per chemistry:** voltage window, voltage curve, cell-spread alert levels and its own trained model. *The recipe of Chapter 2, repeated per chemistry.*
- **Ampere-hour adjustment:** for NMC and Na-ion, a pack of another size is mapped to the size the model was trained at, adjusting current, capacity and resistance together; the pages always show your pack's real values.

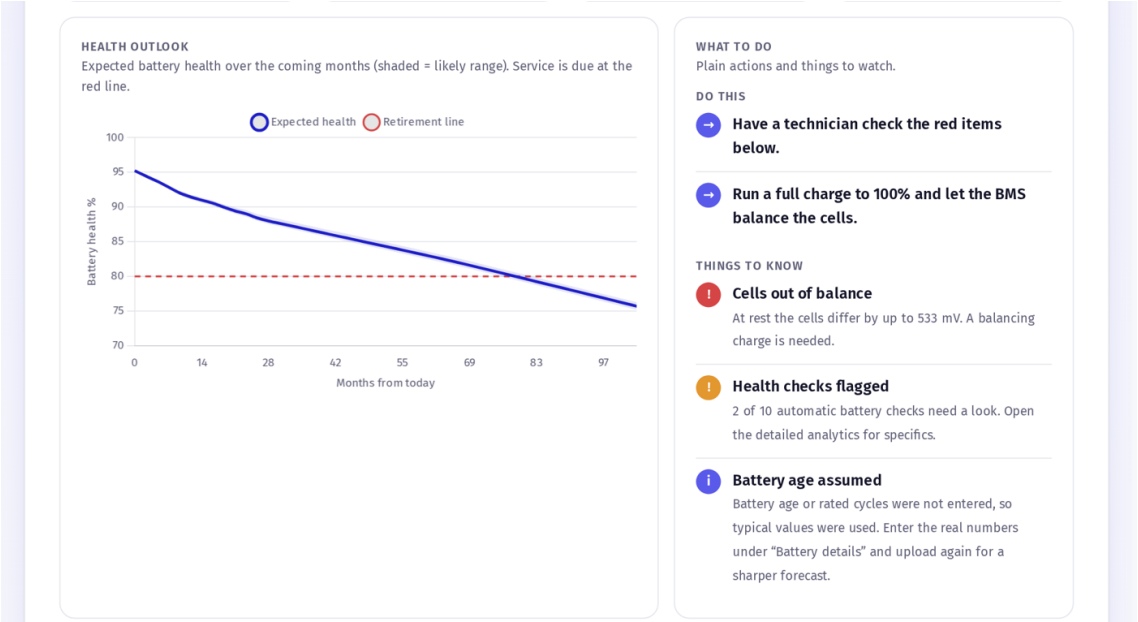


Figure 7. The life forecast with an expected-health path and a likely range, and a "what to do" panel. Screenshot from the business guide.



Figure 8. A forecast walk through the cycles (left) and the posterior on the end-of-life date (right): the range is part of the answer. Screenshot from the business guide.

The Virtual Lab, and distance from the surface

The Virtual Lab lets a user turn the levers a battery owner controls (temperature, depth of discharge, charge rate, state of charge, calendar age) and see what the learned surface implies. That is Chapter 2's "query a learned surface instead of re-solving", with a human-friendly screen. Another panel shows **how far a pack's state sits from the manifold**, by feature: this is the MW distance of the theory repository, used as a validity check on the data itself.

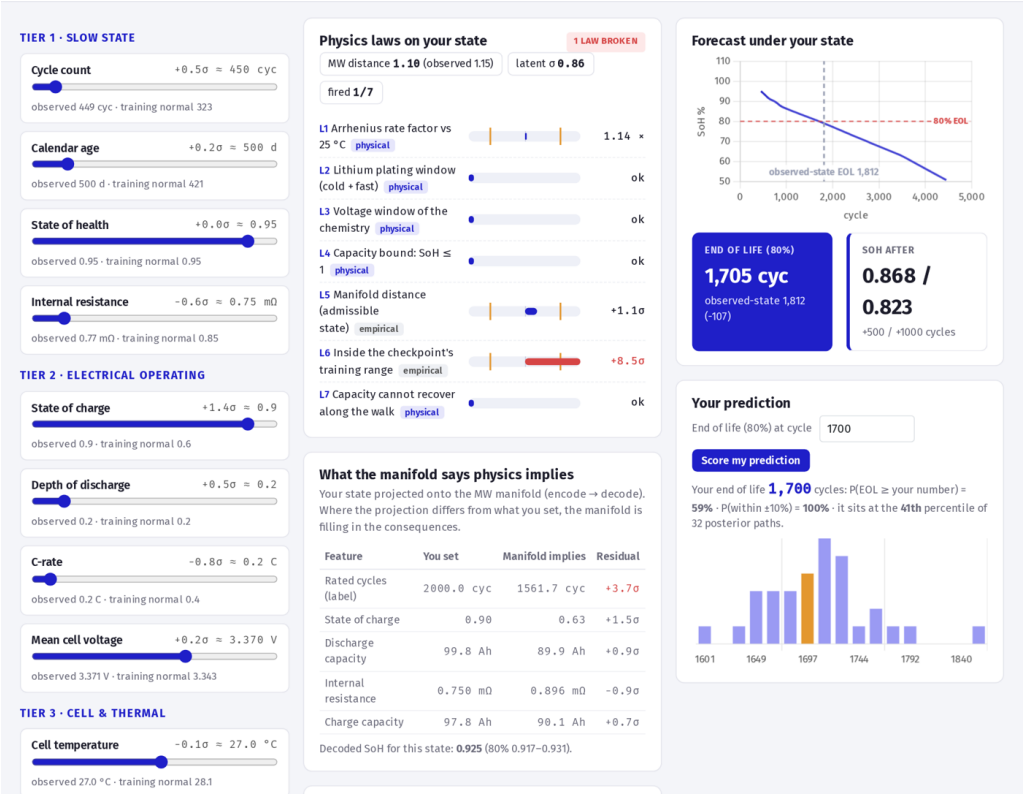


Figure 9. The Virtual Lab: levers on the left, the forecast under your settings and what the manifold implies for the physics on the right. Screenshot from the business guide.

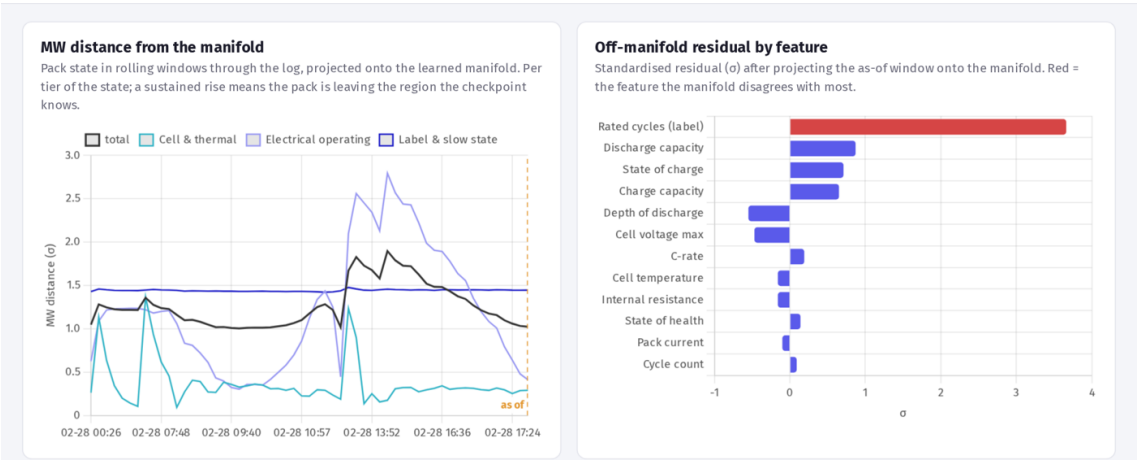


Figure 10. Off-manifold residual by feature: which readings disagree with the learned surface. Screenshot from the business guide.

Where it stands, per chemistry

Chemistry	Status	What that means
LFP	In production, fleet and public-data validated	Calibrated on production e-rickshaw fleet packs in India and tested on 95 public cells from Sandia and HUST, plus a sample problem from peer-reviewed research.
NMC	Validated on research data; pack-level rollout next	Trained and tested on the Sandia research dataset; an ampere-hour adjustment for cell size; under validation at pack level.
Na-ion	Validated on research data; pack-level rollout next	Trained and tested on a public research dataset of 1 Ah cells; an ampere-hour adjustment for larger cells; under validation at pack level.
NCA	Planned	The next chemistry on the list.

The evidence: high-fidelity forecasts on Sandia and HUST

LFP, on public research data. 95 A123 cells (18 from Sandia, 77 from HUST). From one measurement at 100 full cycles, the platform forecasts the whole health path and is scored on 50 held-out cells that never touched calibration or training:

Result	Number
Platform against the physics formula alone, typical error	2.4–3.4 against 18–21 points
On unfamiliar cells (Sandia): health-only-falls model against the flexible model	3.0 against 20.4 points
Before the knee (health above 85%, HUST)	1.4 against 3.4 points; the health-only-falls model leads
After fade speeds up (below 85%, HUST)	2.5 against 3.3 points; the flexible model leads
Sandia cells whose end of life is reached in the forecast	15 of 18 (flexible) against 5 of 18 (health-only-falls)

NMC and Na-ion. In the platform's own tests on held-out cells, a chemistry needs its own model, and calibrating one is a calibration job rather than a research project:

Median error, state-of-health points	MW (own chemistry)	Physics formula alone	LFP model on the other chemistry
NMC (11 held-out Sandia cells, near room temperature)	**2.4**	11.7	5.5–7.3
Na-ion (32 held-out cells)	**1.4–2.0**	2.8	3.3–4.1

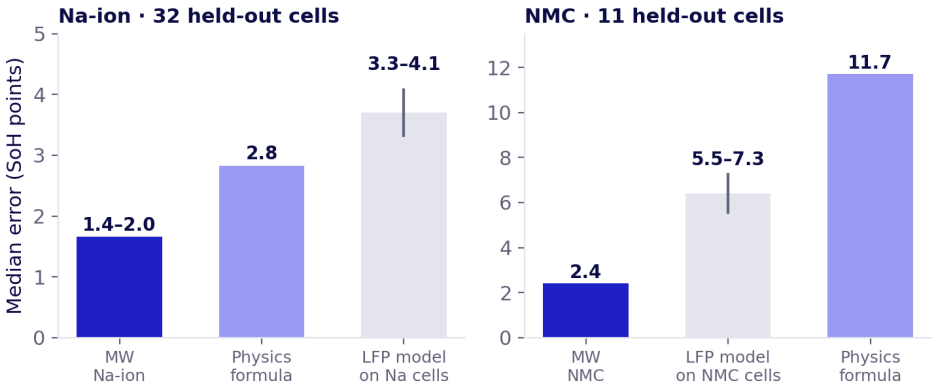


Figure 11. Median forecast error on held-out cells, Na-ion (left) and NMC (right). Screenshot from the business guide.

The compute story

Item	Measured	Source
One full forecast (120 steps, 32 posterior samples, 10 cells)	2.3 ms median (~437 per second) on a laptop CPU, 8 threads	edge benchmark, i5-12450H
One single-state evaluation	0.22 ms	same
Model size	27,856 parameters, 137 kB; loads in 7.8 ms	same
A 1,000-scenario test campaign	about 3 minutes	pipeline documentation
A forecast sweep through the web app (400 steps, 20 samples)	283.6 s end to end, including app overhead	older checkpoint

Item	Measured	Source
Training the deployed model	about half an hour to an hour on one laptop, once	training log
Na-ion: calibration to scored models	about 25 minutes on a laptop	the business guide

Using the model takes milliseconds to seconds. Training the deployed model takes about half an hour to an hour on one laptop. No cluster is involved at any step.

Try it

The live platform is at **https://batteryanalytics.bayesiancybersecurity.com**. The business guide (*Xtreme Analytics: the battery analytics platform*, Edition 4) walks through every screen in plain language. The open-source framework behind it is in the theory repository.

THE COMPUTE ANGLE

This is the live example of Part VI's trade: a one-off training cost of under an hour on one laptop, then forecasts in milliseconds, in a product that fleets use.

NEXT FOR THE PLATFORM

The road ahead is exciting and well defined: a **head-to-head against standard physics-model solvers** (SPM, SPMe, DFN) on the same cells, a **timed CPU-only training run**, **tighter forecast ranges** as long-term fleet data arrives, **NCA** as the next chemistry, and pack-level rollout for NMC and Na-ion. Each one is a chance to show the same recipe winning on more ground.

Battery cybersecurity: same physics, a security lens

Cyber-physical attacks end in heat and current, not in a stolen file. The MW surface that scores a battery's health also scores whether it is behaving.

Offering. Bayesian Cybersecurity builds digital twins of battery energy storage stations (BESS), runs red-team reconstructions on isolated infrastructure, and trains operators to recognise the early stages of a fault or an attack. It sits next to the battery analytics platform above: the same cell physics, applied to security.

The twin is a three-tier range on Podman: an energy management system (EMS), a power conversion system (PCS) and a battery management system (BMS) that speaks the real command language lithium packs use. The battery is deterministic physics. Command a discharge and the modelled pack drains until its own safety limits trip, about a hundred seconds in the demonstration. The physics decides what happens, so the result is a faithful rehearsal, not an animation.

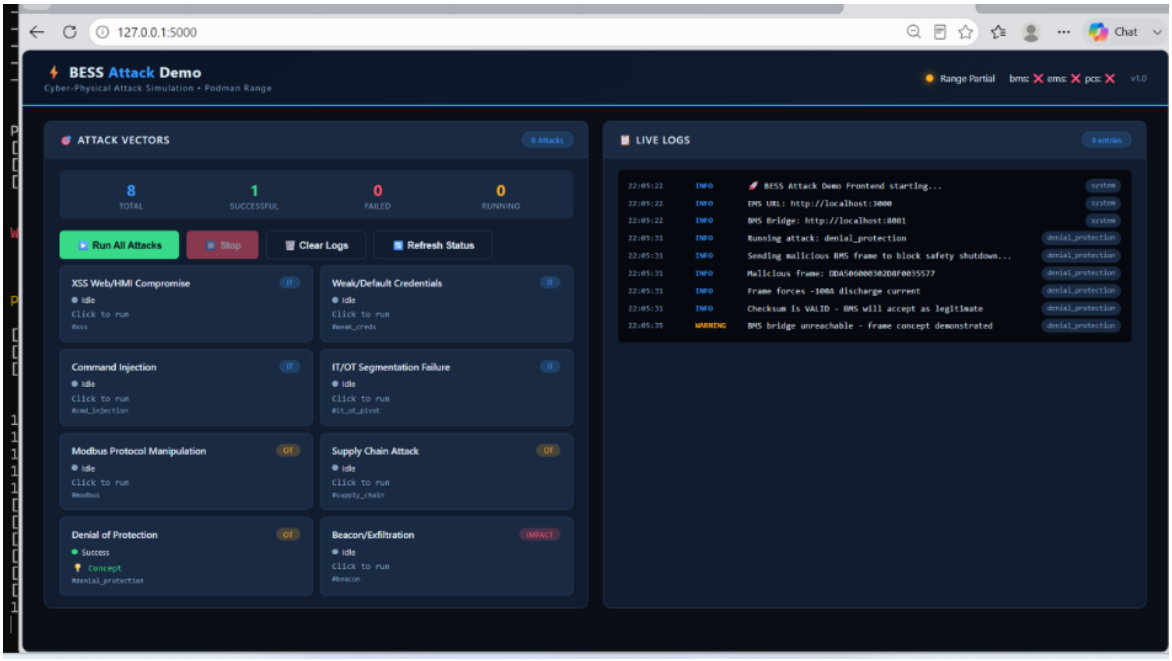


Figure 12. The BESS attack range console: eight attack vectors across IT, OT and physical-impact tiers, with a live log. The frame in the log carries a valid checksum, so the BMS treats it as a legitimate command.

Why this matters. The most dangerous command is a valid one. A frame with a correct format and checksum passes every signature check, so only a view of the physics can tell that the pack is being driven somewhere its healthy behaviour never goes. That is exactly what the MW surface measures: distance from the shape of valid battery states.

WHAT YOU GET

Digital twins of your station for safe reconstruction. **Physics-consistency monitoring** that scores telemetry against the learned surface. **Forensic readiness**: which access logs, bus records and BMS/PCS event histories to keep, so any future incident can be explained. **Operator training** on early fault and attack signatures. If you run grid storage, EV charging or any battery-heavy asset, this is a conversation we would love to have.

RF electronic warfare: a physics checker for world models

Score any spectrum state, real or imagined, on one physics scale.

Offering. The MW RF Sentinel learns what valid radio-frequency states look like as a shape, from four physics domains (navigation and aviation, cellular, drone and UAV, maritime and tactical). Any spectrum state then gets a distance to that shape. Far from the shape means abnormal: a real threat, or a hallucination by a learned model. Our RF electronic warfare submission is in for ADITI 4.0, the Ministry of Defence's iDEX innovation challenge, under Q5.

World models such as Dreamer learn to predict forward. MW is not a competing world model; it is the checking layer on top. We forked the open R2-Dreamer (Morihira et al., ICLR 2026) and put MW on it, so every imagined rollout is graded against what actually happened, on the same physics scale.

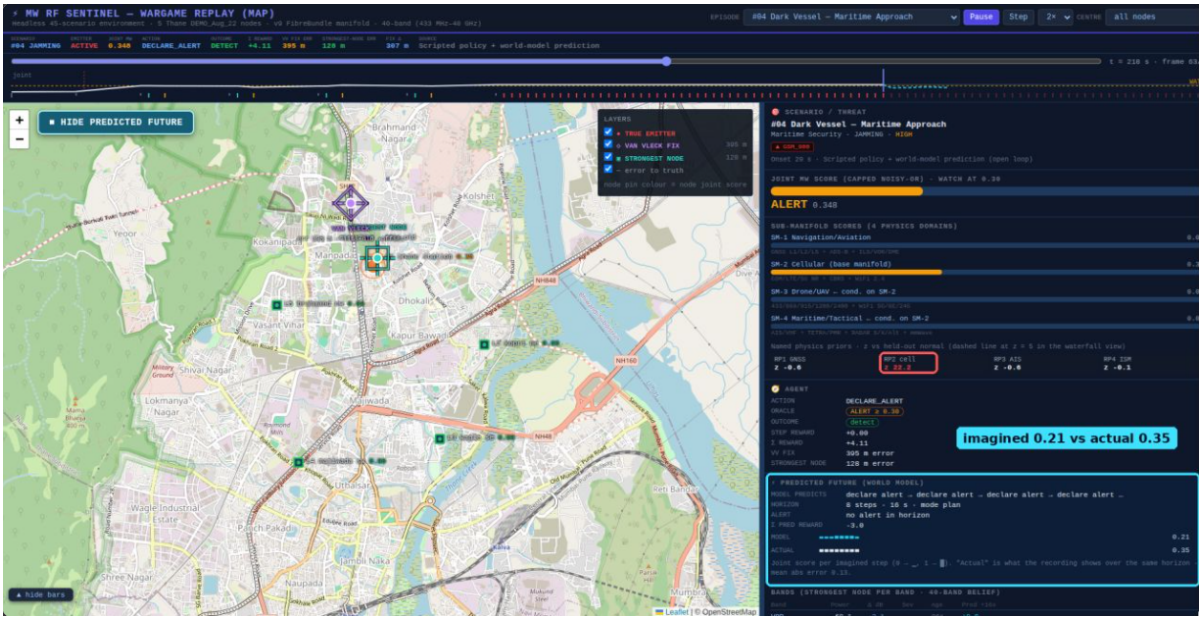


Figure 13. The RF Sentinel wargame replay. Simulated jamming near Thane creek, five sensors, 40 bands. MW raises an ALERT (joint score 0.35) and names the failing physics prior (the cellular prior, 22.2 sigma from normal); the world model imagines 0.21 against an actual 0.35.

Actionable alerts. A black-box alert says “trust me”. An MW alert says “the cellular prior sits 22 sigma from normal”, so the operator knows which physics assumption failed. The gap between imagined and actual score is the honest measure of how far to trust the model today.

What	Measured in the repository
Scoring speed	about 7 ms per spectrum state (median 7.1 ms) on 2 CPU cores, no GPU
Scenario library	45 scripted RF scenarios, 40 bands, 5 sensors
Sentinel threshold rule	detects 97% of scenarios with 0 false alarms on 30 clean episodes (10 scenarios, 3 seeds)
Differentiable physics score	tracks the engine's own score at Spearman 0.999 on 51 golden inputs
Where it runs	browser replay and a CPU backend, simulation first

PILOTS WELCOME

Battery analytics already serves clients. **Battery cybersecurity and RF electronic warfare are open for first pilots**: utilities, storage operators, defence labs and integrators who want a physics layer under their detection. Same framework, different priors.

PART IV

Robotics

A distilled humanoid student that runs on a CPU: what it is, what you can do with it, and how to reproduce it.

5 · A humanoid on a laptop

PART IV · CHAPTER 5

MW_Robotics: a humanoid on a laptop

github.com/RahulModak74/MW_Robotics · Apache-2.0

QUICK ANSWER

The result: a small CPU policy (8 MB) plus a physics "flight computer" completes 58 of 58 simulated humanoid tests, matching NVIDIA's teacher, which was trained on GPU clusters from 142,000 motions. The student alone needs one CPU thread and 2.3 ms per step. **The frontier:** the flight computer takes about 0.16 s to approve a move and the robot needs 20 ms. We publish the exact bottlenecks and invite the robotics community to help cross the real-time threshold.

WHAT YOU CAN DO ON A CPU

Run the distilled student. An 8 MB policy, 2.3 ms per step on one CPU thread against a 20 ms budget. **Simulate it.** MuJoCo and the ProtoMotions evaluator run the 58-test suite on a laptop CPU. **Re-tally the result.** One script prints TOTAL 58/58 from the recorded run. **Make your own student.** Generate motions, record the teacher, train, and report your own count. Only the teacher, trained once by NVIDIA, needed GPUs.

The idea

Today's best humanoid controllers are **data policies**: very large networks trained on huge motion libraries in GPU data centres. MW_Robotics tries **physics policies**: keep the policy small enough for any CPU, write down the physics the robot must obey (balance, capture point, torso upright), and let that physics do the safety work at run time. Both the student and its flight computer run with no GPU.

Step one: from a large policy to a small student

A robot policy looks at the robot's situation and answers "what do I do now?", 50 times a second. For NVIDIA's Unitree G1 tracker:

- **It sees (the observation, 349 numbers):** the 29 joint angles and their speeds, how the body is turning, which way gravity points from the torso, where the reference motion wants the body 1, 2, 4 and 8 steps from now, and the last command.
- **It does (the action, 29 numbers):** a target angle for every joint; the motors' own controllers pull the joints there.

The teacher drives the robot in the MuJoCo simulator on a few hundred motions (the 4 originals plus 324 generated from text prompts such as "walk in a circle" or "do a squat" with NVIDIA's open Kimodo model). Every step records (what it saw, what it did). Collecting and training take hours on a gaming laptop, not weeks on a cluster.

The student then learns a **manifold of normal states**: it compresses the 349 numbers into 16 and rebuilds them (a Bayesian VAE in Pyro). The set of situations it can rebuild well is a 16-dimensional surface inside the 349-dimensional space. Two things come out for free: a **position on the map** (which kind of moment is this: mid-step, arms up, crouching) and a **distance to the map** (how unfamiliar is this situation). A last-layer Laplace approximation gives an exact Bayesian posterior over the action, so the student knows when it does not know. The result is one small network: 1.65 million parameters, 7.7 MB, about 3.3 million multiply-adds per

step.

In one line: the teacher shows the way; the manifold remembers where the robot can be; the Bayesian head knows when it does not know.

Step two: the flight computer

Like an aircraft's fly-by-wire system, the student is the pilot and states intent. The flight computer knows the airframe exactly: it predicts the pilot's plan 0.4 s ahead with an exact physics model, scores the predicted states against an **envelope** of physical laws, and passes the command through unchanged unless the predicted state leaves the envelope. If it does, it sends the smallest correction that keeps the robot inside it (a search over 63 candidate corrections).

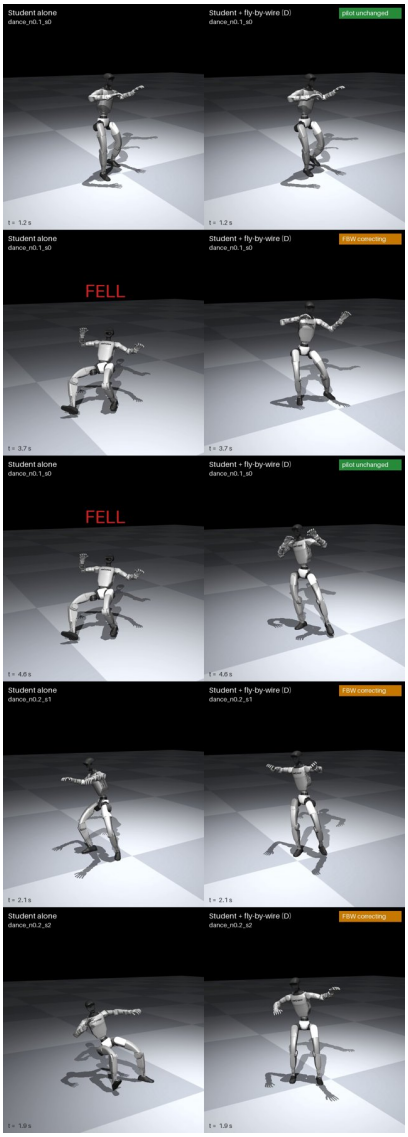


Figure 14. A dance the student never saw, with action noise. Left: the student alone falls. Right: the same student with the fly-by-wire finishes. A badge shows, per step, whether the flight computer passed the command through or corrected it. Image from the repository.

Designing the envelope

The envelope is a set of physical laws on the *predicted* states, each with a tolerance and a weight:

Law	Setting	Why
Predicted fall (pelvis below 0.45 m or torso flatter than 60°)	On; cost 1000 × steps remaining	The hard boundary; earlier falls cost more

Law	Setting	Why
Capture point within 5 cm of the support	Weight 50	Where the robot would have to step to stop. Measured against the segment between grounded feet, so normal walking does not trigger it
Torso upright within 50°	Weight 50	Stops the trunk tipping during fast upper-body motion
Pelvis height, spin	Off	Added nothing here

It was chosen by **ablation** on the dance clip, same student and seeds each time. This is Chapter 2's “subtract a law and see what breaks”, done on a robot:

Config	Laws	Dance clip
A	Predicted fall only	5/7
B	Fall + upright	4/7
D	**Fall + upright + capture point**	**7/7**

The capture point is the law that matters. Every correction D made on the dance runs was triggered by the capture-point law. A only reacts once a fall is predicted; D keeps the robot inside its balance limits before a fall is predicted. D was then held to a no-regression gate on 51 motions it had never been tuned on: 51/51, with zero regressions.

What was measured

Test: a Unitree G1 in MuJoCo, using NVIDIA ProtoMotions' own evaluator. 58 tests: one unseen dance clip run 7 times (clean, plus action noise at two levels with 3 seeds each), 21 unseen familiar motions, and 30 unseen generated dances.

Policy	Dance (7 runs)	21 familiar	30 dances	Fall warning (AUROC)
MW student (100 motions) + flight computer	**7/7**	**21/21**	**30/30**	no falls left to warn about
MW student (100 motions), alone	3/7	17/21	20/30	0.994
MW level 2 (324 motions), alone	7/7	20/21	24/30	0.996
MW level 1: Bayesian head only	5/7	19/21	27/30	0.992
Plain MLP (324 motions)	6/7	18/21	26/30	none
NVIDIA teacher (142,000 motions)	7/7	21/21	30/30	–

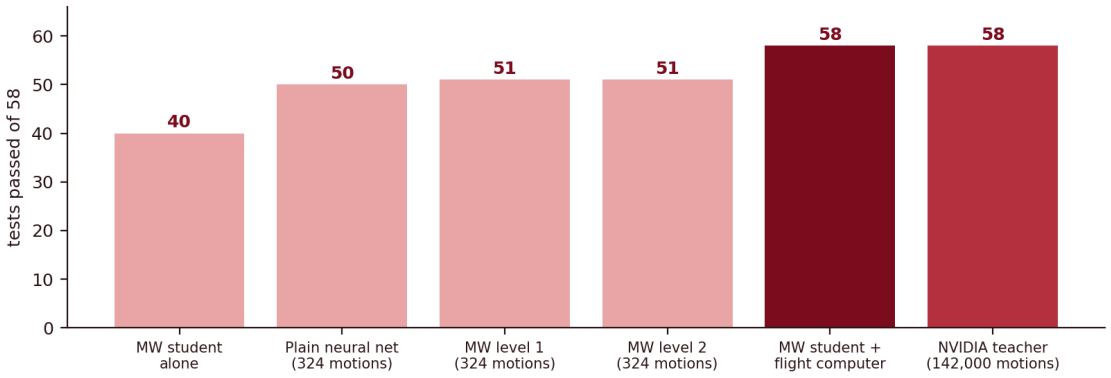


Figure 15. Tests passed out of 58 (sums of the three columns in the table). The flight computer lifts the 8 MB student from 40 to 58, the teacher's score.

- **58/58 with zero regressions:** the flight computer lifted the student on all 18 tests it missed alone and kept every test it already passed. The student was not retrained. The flight computer's model matched the simulator to at most 1×10^{-7} per step.
- **Warning:** every one of 26 difficult cases across three students was flagged 0.16–0.96 s ahead.
- **Smooth by design:** joint tracking on the 40 tests that pass both ways changes by a median of 3.8%, and the flight computer steers on about 20% of steps, trading a touch of tracking for balance.
- **Size and speed:** the level-2 student runs in **2.3 ms per control step on one thread of a laptop CPU** (Intel i5-12450H). The robot needs a decision every 20 ms.

Example: a student that beats its teacher (the pendulum)

A baby learns to walk by copying its parents, but gravity teaches too: every wobble the parents never showed is still judged by physics. A plain distilled policy has only the parent. An MW policy also gets **physics priors**, laws checked on the *consequences* of each action.

The demo balances an inverted pendulum. The teacher is a good controller. The data are the teacher's own trajectories from small leans, so lean and speed always move together and many functions fit equally well. The plain student fits the actions. The prior student fits the same actions *plus one law*: after the action, the capture point (lean + speed / natural frequency) must shrink. The law is checked on extra unlabelled states, bigger pushes and faster spins, where no teacher is needed, only physics.

Pushes from upright at speeds the data never had (5 seeds, 35 tests)	Fell
Plain student	**18 of 35**
Prior student	**0 of 35**
Teacher	0 of 7

Both students fit the teacher's data equally well (mean squared error 0.0000 to 0.0012). **The data cannot tell them apart; only the prior decides what happens outside it.** With a slow, soft teacher, the prior student stayed within 2° on far more pushes (8 exceeded 2° against 25 for the teacher and the plain student), because the law contains knowledge the teacher's data does not.

TRY IT YOURSELF

The pendulum demo runs in minutes on any CPU:

```
git clone https://github.com/RahulModak74/MW_Robotics.git
cd MW_Robotics && pip install -r requirements.txt
python examples/policy_priors_demo.py 0,1,2,3,4
```

The next frontier: real time (a bounty for robotics labs)

The robot needs a decision every 20 ms. The flight computer is the frontier. Here is what has been measured on the same 58 tests and the same student:

Version	Score	Time per decision
Flight computer on every step	58/58	~0.16 s to approve, ~3 s to correct
The student's own uncertainty decides when the flight computer acts	58/58	median 2.6 ms; 78.7% of steps within 20 ms; the rest take seconds
Same, with a cheaper search	50/58	82% of steps within 20 ms; the rest up to ~0.3 s in Python

Version	Score	Time per decision
Search runs in the background; its answer arrives late	40/58	every step under 20 ms

The look-ahead also runs in C++ with the student's weights inside, about 4.5× faster (a full correction in ~0.26 s on a laptop, still 13 control steps). The repository maps the timing budget: the correction has to land within a few control steps, and the skill seems to live in searching again at every step rather than in a map from state to action. That is the target for the bounty.

What “GPU-free” means here, exactly

At run time everything is CPU: the student, the flight computer and the tests. Preparing the student (generating motions, recording the teacher, training) is a gaming-laptop job; the repository mentions consumer GPUs many teams already own. The teacher itself was trained by NVIDIA on GPUs, and removing the teacher, designing policies from scratch with Bayesian priors about the hardware, is Step 2 of the roadmap.

Reproduce it

Step	What	Time
1	Generate motions with Kimodo, collect teacher rollouts, make the evaluation libraries	hours
2	Train the 100-motion student (`r7c_mwv3ctl_s100`)	hours
3	Run the 58 tests (Python flight computer, or the C++ filter)	~6 h on 6 threads, or ~3 h on 4
4	Count the result: `score58.py` prints TOTAL 58/58	seconds

The shortest check needs nothing installed: `python step1_extraction/cpp_student/results/CPP_FULL_STUDENT/score58.py` re-tallies the recorded run to TOTAL 58/58. No weights are shipped (they derive from NVIDIA's model), so you generate your own student and report your own count, whether it is 58 or not.

THE ROBOTICS BOUNTY BOARD

The repository lists ten ideas it has not had time to try, each with a cheap first test (for example a 2 ms look-ahead, search in fewer dimensions, spending the warning time, or a learned envelope checked by the exact physics). Chapter 8 collects them.

THE COMPUTE ANGLE

Measured: 2.3 ms per step on one laptop CPU thread for the student. The 58/58 result needed a laptop and hours of preparation, and it did not need a cluster.

PART V

Quantum

What a laptop lets you study, and where to start: Dr Zickert's books first, then litho, chemistry and spin chains.

6 · Quantum on a laptop: the idea
7 · Use cases: from the books to litho

PART V · CHAPTER 6

Quantum on a laptop: the idea

Qubits as spheres, a chain of small tensors, and what that lets you study without a queue.

QUICK ANSWER

A quantum computer of n qubits is described by 2^n numbers, which is why laptops give up near 30 qubits. But real states are usually not tangled everywhere. MW keeps each qubit as a small sphere and chains them together, so memory grows with the number of qubits, not with 2^n, for as long as the entanglement stays low.

The idea in plain words

A single qubit is a point on a sphere (the Bloch sphere): two numbers, an angle θ and an angle φ. Fifty independent qubits are fifty spheres, 100 numbers in all. Fifty *entangled* qubits, in the worst case, need 2^{50} numbers, about 18 petabytes. Most states that matter in physics sit between those extremes. They are entangled, but only locally, between neighbours, and a chain of small tensors captures that cheaply.

MW pictures it exactly this way: each qubit is a sphere, and each sphere is conditioned on the ones before it. That chain has a standard name in physics, the **matrix product state (MPS)**. The size of the link between neighbours (the **bond dimension** χ) is the dial: χ = 1 is independent spheres, and larger χ carries more entanglement.

MW language	Standard language	In mwqc
n independent Bloch spheres (θ, φ per qubit)	product state = MPS with bond dimension χ = 1	`BlochState`
fibre bundle: qubit k conditioned on the qubits before it	MPS bond carrying χ-dimensional context	`MPSState(chi_max=χ)`
“physically realisable states live on a small manifold”	low-entanglement (area-law) states have small χ	`entanglement_entropy()`, `bond_dims()`
“Bayesian uncertainty, reducible with effort”	truncation error: discarded weight, reducible by raising χ	`truncation_error`, `fidelity_estimate`

THE ONE-LINE PRINCIPLE

Memory is linear in qubit count as long as entanglement stays bounded. For product states (χ = 1) and low-entanglement states (small χ), 1000 qubits fit in kilobytes or megabytes on a laptop. When entanglement grows, χ must grow, up to $2^{n/2}$ in the worst case. No representation avoids that worst case.

The repository is explicit: this is well-established physics (Vidal 2003; DMRG, White 1992). **What mwqc adds is** a small, transparent, CPU-first stack that keeps the MW framing as a diagnosable backend, reports an error bound with every expectation value, and couples the simulator to Bayesian physics surrogates whose stated uncertainty is checked on held-out data.

Example: a 1000-qubit Fourier transform on a laptop

The quantum Fourier transform (QFT) on n qubits is a staple of quantum algorithms. Written as n(n−1)/2 gates between *adjacent* qubits, a 1000-qubit QFT has 500,500 gates. Gate by gate on the MPS engine:

Part	Input	Result (n = 1000, 500,500 gates)
A	a basis state |j⟩, j a random 1000-bit integer	χ = 1 throughout; truncation error 0; max angle error vs the closed form **2.7×10^{-15} rad**; state **32 kB**; 123 s
B	a random point on every Bloch sphere (the output is *entangled*)	max bond **7**; fidelity estimate 1.000000; state **1.44 MB**; entropies 0.06–0.53 bits per cut; 177 s
C	validation of B's method at n = 20 against exact	χ = 1: true F 0.230 (est. 0.238); χ = 2: 0.9600 (0.9613); χ = 4: 0.9999937 (0.9999937); χ ≥ 8: 1.0

Reading it: (A) confirms the textbook closed form gate by gate. (B) is the result worth quoting: a general 1000-qubit QFT is cheap because **the QFT creates little entanglement**, not because the qubits are independent. (C) shows the tracked error estimate following the true fidelity as χ grows (0.23 at χ = 1, 0.96 at χ = 2, essentially 1 by χ = 4). Peak process memory was 557 MB, almost all of it the stored circuit, not the quantum state.

How the method scales: entanglement against depth

Random brick-wall circuits on 40 qubits with χ ≤ 64 show how the approach scales. Entanglement grows linearly with depth, χ grows exponentially, and the fidelity estimate tells you exactly when to raise χ:

Depth	2	4	6	8	10	12	16	20
Max bond	2	4	8	16	32	64	64	64
Fidelity estimate	1	1	1	1	1	1.000	**0.869**	**0.384**
Mid-chain entropy (bits)	0.36	1.06	2.13	2.94	3.44	3.66	5.19	5.20

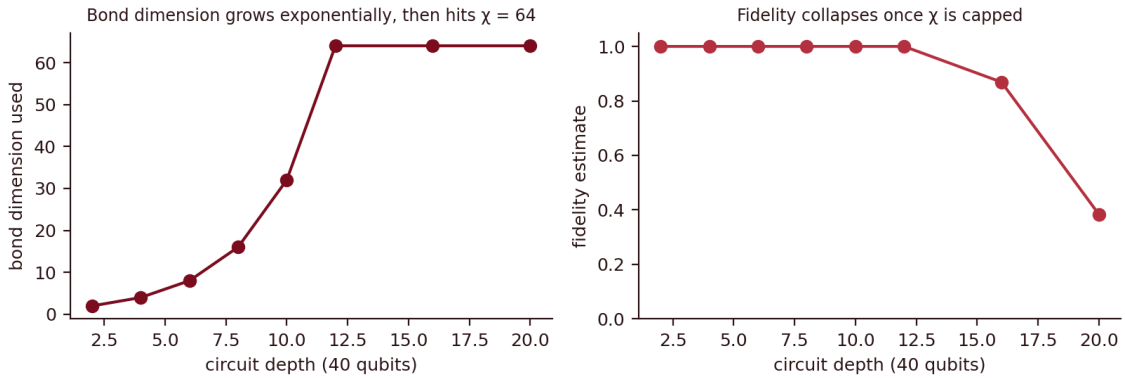


Figure 16. Measured scaling on 40 qubits: bond dimension and fidelity against circuit depth (values from the repository).

By contrast, a GHZ state on 1000 qubits, “maximally global”, needs only χ = 2 (0.13 MB, 0.8 s for 1000 shots). What matters is entanglement, not how global the state looks. The method is built for **sparse, low-entanglement states**.

> **THE COMPUTE ANGLE**
> Everything in this chapter ran on a laptop CPU, with an error bar on each result. The dial that sets the price is entanglement, and you can read it off the bond dimension.

PART V · CHAPTER 7

Use cases: from the books to litho

Start with Dr Zickert's PyQML books, then chemistry, litho and spin chains, all on a CPU.

QUICK ANSWER

Start by running Dr Zickert's open-source PyQML books on your laptop, with one import line and no sign-up. When you want more, the same code carries past 30 qubits, and the same toolkit runs chemistry surrogates, an EUV litho model and 100- to 200-spin quantum magnets.

Where it helps: no queue, no account, no allocation

Real quantum hardware is reached through a cloud service. That means an account, a queue, a limited number of runs, and thousands of shots per job. Running a circuit can take seconds to hours, and every question needs a new run. A *simulator* runs where you are, but a dense one hits a memory wall. mwqc is built for the case a student, a teacher or a lab meets first: **run the experiment on the machine in front of you, with nothing to sign up for**, see the answer with an error bar, and know when it stops being trustworthy.

Dr Frank Zickert's PyQML books: run the book's code, no sign-up

Dr Frank Zickert's *Hands-On Quantum Machine Learning With Python* (PyQML; Volume 1: *Get Started*, 2021; Volume 2: *Combinatorial Optimization*, 2023) teaches quantum computing the way a programmer learns: every idea is a short Python program you run and look at. The books' code is open source on GitHub. The code uses the classic Qiskit API (`from qiskit import Aer, execute`), and mwqc keeps that API alive so the books run today.

The mwqc repository solves that, with **one import line changed**, done for you by a script:

```
# original book code             ->   on mwqc
from qiskit import QuantumCircuit, Aer, execute
                                       from mwqc.qiskit_compat import QuantumCircuit, Aer, exec
ute
# everything below this line is the book's own code, unchanged
```

- **No IBM account, no API token, no sign-up.** Every number printed in both books comes from Aer, a classical simulator running on the reader's own computer, not from a quantum chip. mwqc does that same job locally.
- **Volume 1: all chapters run.** `python3 pyqml_port/port_book.py --book book_vol1 --out pyqml_port/ported_vol1` rewrites the imports; open the notebooks in Jupyter and run them top to bottom while reading the book.
- **Volume 2: the hand-built chapters run on mwqc** (observables, Hamiltonians as sums of Pauli terms, VQE by hand, Max-Cut as an Ising problem, QAOA by hand). Four notebooks call Qiskit's ready-made algorithm library, so they run *unchanged* in a pinned, free, local Qiskit 0.41 environment that the repository sets up.
- **Hardware cells:** cells in one Volume 2 chapter that load an IBM Quantum account are for real-hardware access; the simulator and noise-model cells in the same chapter run locally.
- **Checked, not assumed.** The same source runs on Qiskit with Aer and on mwqc, and outputs are compared call by call (statevector fidelity, unitary difference, total-variation distance for

counts). Book-style programs agree to 10^{-14} on exact outputs, and sampled counts agree within shot noise (TVD ≤ 0.02).

How something this small does the job

The books use a narrow slice of Qiskit: 25 circuit methods, 3 backends and 3 result calls, with circuits of at most 12 qubits. Implementing that slice exactly is a small, checkable piece of code (about 2,500 lines of Python in mwqc). That is why the books run on mwqc with one import line.

Nothing from the books is copied into the mwqc repository: you clone the books' code yourself (one command) and the porter changes the imports on your copy. The notebooks are the listings, and the explanations live in the books, so buy them and read along.

A BOOK WE RECOMMEND

Dr Frank Zickert's PyQML books are very useful, and we recommend you buy them if you are serious about pursuing quantum machine learning. The code in this chapter runs on your laptop, and the books supply the explanations, the intuition and the exercises that make it stick. Read along, run each listing, and let mwqc carry you past the book's qubit counts when you are ready.

TRY IT YOURSELF

Volume 1 on mwqc, from a fresh clone:

```
git clone https://github.com/RahulModak74/Quantum_Compute_MW.git && cd Quantum_Compute_MW
pip install -r requirements.txt matplotlib jupyterlab
git clone https://github.com/quantum-machine-learning/Hands-On-Quantum-Machine-Learning-Wi
th-Python-Vol-1 book_vol1
python3 pyqml_port/port_book.py --book book_vol1 --out pyqml_port/ported_vol1
jupyter lab pyqml_port/ported_vol1
```

When the student reaches the 30-qubit wall: the same code keeps going

A dense simulator stores all 2^n amplitudes. At 16 bytes each, that is 4.3 GB at 28 qubits, 17 GB at 30 qubits, 275 GB at 34 and about 18 petabytes at 50. A laptop reaches its wall at roughly 28 to 30 qubits. This is where most students stop and where most tutorials end. In mwqc the wall is a *step*, not a cliff, because the circuit API stays the same and only the engine underneath changes:

Stage	Qubits	Engine in mwqc	What you get	Evidence
1. The books	up to ~12	exact statevector	exact results, as in the books	book-style programs agree with Qiskit + Aer
2. Bigger exact experiments	up to 28	exact statevector ($16 \cdot 2^n$ bytes)	the reference for checking everything else	MPS vs statevector fidelity $> 1 - 10^{-10}$ on random circuits
3. Past the dense wall	30 to 1000	MPS (`MWEstimator(method="mps")`); same circuit API	an answer with a tracked error bar	1000-qubit QFT in 32 kB to 1.44 MB; 100- and 200-spin dynamics and ground states vs exact
4. Track entanglement	any	measure χ and the fidelity estimate	a clear signal when entanglement outruns the method	40 qubits, depth 20 exceeds $\chi = 64$ (fidelity 0.384)
5. Optional next level	—	`to_qiskit()` converts a circuit to Qiskit	use the full Qiskit stack or real hardware when it is needed	IBM account needed only here

A student's journey is therefore continuous: the same circuit-writing habits carry from a 2-qubit Bell state in the book to a 100-qubit quantum magnet, with an error bar that says when the answer can be trusted. The 100- and 200-spin examples are in the next sections. Be clear about the boundary, which the repository states: this works for low-entanglement states, and a generic deep random circuit is out of reach for *any* classical method, mwqc included.

Use cases

Chemistry, checked. PySCF → Jordan–Wigner → UCCSD-VQE, as a *validation reference*:

System	Qubits	Pauli terms	E_{FCI} (Ha)	VQE − FCI (mHa)	HF − FCI (mHa)
H_2 0.735 Å	4	15	−1.137306	0.000	20.3
H_2 1.50 Å (stretched)	4	15	−0.998149	0.000	87.3
LiH 1.595 Å	12	631	−7.882402	0.010	20.4
H_2O	14	1086	−75.012578	0.097	49.6

VQE lands within chemical accuracy of the exact answer in 8 of 8 runs. The cost is visible: qubits = 2 × orbitals and the Pauli-term count grows as orbitals^4, so a periodic battery-cathode cell is far outside dense emulation. The realistic path is a surrogate.

Kohn–Sham surrogate. Learning six valence eigenvalues of an Fe–O cluster (a proxy motivated by $LiFePO_4$) from geometry alone, on 200 real PySCF calculations (about 2.2 CPU-hours to make):

Model	MAE (meV)
Predict the mean	210.4 ± 7.7
Ridge regression on distances	86.0 ± 6.1
MWSurrogate (5-member ensemble)	**36.8 ± 4.4**

That is below chemical accuracy (43.6 meV) against the PySCF reference. The ±2σ coverage on held-out data is 66% raw and 93% after calibration (target 95%). Inference takes about 0.2 ms against about 40 s per PySCF solve, which is about 10^5× per query but *amortised*: the 2.2 CPU-hours of training data must be paid first and the geometries are sampled within ±0.08 Å.

EUV lithography. An aerial-image and critical-dimension (CD) surrogate with a hard physics constraint: any image through a pupil of radius NA/λ has its spectrum confined to $|f| \le 2NA/\lambda$, and a layer that projects every prediction onto that set makes a physically impossible image *unrepresentable*.

Metric (test set)	plain CNN	band CNN (physics layer)
Aerial-image RMSE	0.0045	**0.0039**
Out-of-band spectral power	5.2×10^{-5}	**2.3×10^{-15}** (zero by construction)
CD MAE (nm)	0.18	**0.17**
±2σ CD coverage, raw → calibrated (target 95%)	50% → 94%	42% → 93%
OOD detection AUROC (unseen pattern families)	1.000	1.000
Time per mask: forward model vs surrogate	20.7 ms vs 2.2 ms (9×)	same

Time per mask drops from 20.7 ms to 2.2 ms (9×) on this model.

100- and 200-spin quantum magnets. Quench dynamics and ground states of interacting spin chains, checked against the exact free-fermion solution at any N:

Run	Result
Integrable quench, N = 100, χ = 8 / 16 / 32	max error over all 100 sites and all times 1.0×10^{-4} (the time-step error), ~5 s each
Chaotic Ising chain, N = 100 (no exact answer)	χ = 16 vs χ = 32: max difference 2.3×10^{-7}
Critical Ising ground state, N = 100	E = −126.96187618 vs exact −126.96187674: error **5.6×10^{-7}**, 3 min
Critical Ising ground state, N = 200	error 9.3×10^{-6} (relative 4×10^{-8}), 11 min
Heisenberg chain, N = 100, χ = 32	energy per bond at the centre −0.44317 vs the Bethe-ansatz value 1/4 − ln 2 = −0.44315

Horizon thermality. The vacuum of a quantum field seen from one side of a boundary is a thermal state with local temperature T(d) = v/(2πd) (the Unruh effect, the flat-space cousin of Hawking radiation). Input: only the Hamiltonian of the critical Ising chain. Nothing thermal goes in. The local inverse temperature agrees with the Unruh value to within 0.4% of the finite-size theory, and the central charge from the entanglement entropy is 0.5135 against the exact ½.

What mwqc covers of the circuit-writing world

The repository measures coverage on the parts of the Qiskit-style workflow that apply to MW: building circuits, running them, counts and statevectors, expectation values, and the code in the PyQML books. “Covered” means implemented *and* tested against an exact answer. The percentages are the repository's own judgement calls on feature surface, stated so they can be checked.

Summary figure	Value
The Qiskit surface that applies to MW (unweighted mean of the eight rows)	**≈ 50%**
What an existing textbook QML codebase actually calls (the PyQML books)	**≈ 85%**

In plain words: for the things a student, a teacher or a research group typically does first (build circuits, run them, get counts and statevectors, read expectation values, run the books' code), mwqc covers most of what is used. When you want more, `to_qiskit()` hands any circuit onward in one call.

What this enables, for any lab or classroom

- **A classroom that cannot wait for a queue.** Thirty students can each run the book's programs on their own laptop, at the same time, with nothing to sign up for.
- **A lab with no cloud budget.** Verified 100- to 200-spin simulations and a 1000-qubit circuit on one CPU.
- **A path, not a wall.** From the book's 2-qubit programs to bigger experiments with an error bar, using the same circuit API, and a conversion to Qiskit when full hardware access is wanted.

The state of quantum hardware, as sourced by the repository

The same family of method as mwqc's MPS, tensor networks and Pauli propagation, reproduced IBM's 127-qubit “utility” experiment on classical computers within weeks (Kim et al., Nature 618, 500, 2023; Tindall et al., PRX Quantum 2024; Begušić and Chan, Science Advances 2024), which shows how far CPU methods reach on chemistry and materials today. Fault-tolerant machines are on vendor roadmaps for about 2029 and later, and hardware has run circuits on 50 to 150 physical qubits. A CPU-first toolkit gives you verified results now, with hardware welcome when it arrives.

PART VI

The compute gain, and joining in

Speed and cost with Kerr black holes as the open-code test, every headline number, a bounty board, and the open-core model.

8 · The compute gain, joining in, and the open-core model

PART VI · CHAPTER 8

The compute gain, joining in, and the open-core model

The arithmetic, the truth about the numbers, how to join, and how the work is sustained.

QUICK ANSWER

The cost of a hard problem moves from the query to a one-time learning step that fits on a laptop. This chapter shows the arithmetic with Kerr black holes as the open-code test, lists every headline number with the file that holds it, and then opens the door: a weekend plan, a bounty board, and the open-core model that sustains the work.

The cost of a hard problem

Numerical relativity on a cluster, against the MW pipeline on a laptop CPU.

QUICK ANSWER

A typical binary-black-hole simulation costs about 1.3×10^{5} core-hours on a cluster. Learning the whole Kerr family took about 7 core-hours on a laptop, and each template after that costs 0.16 s. As compute bills, that is a ratio of 10^{6} to 10^{9}, depending on how many templates you make. It is a comparison of bills, not of identical physics.

The open-code test: Kerr black holes

A rotating black hole has a known exact geometry, the Kerr metric. That makes it the ideal control: if a learned surface carries Kerr's geometry well enough to give the right orbits and gravitational waves, we know exactly what learning costs, because we can check against the exact answer. The KERR_SPACETIME folder in QUANTUM_GRAVITY_WITH_MW does this on a laptop CPU (Intel i5-12450H, no GPU), with no calibration factor.

One surface for every spin from 0 to 0.95. Orbital frequencies agree with the exact Bardeen–Press–Teukolsky values to at most 0.18%, the innermost stable circular orbit to at most 0.12% over 39 spins, and the gravitational-wave match under Advanced-LIGO noise is at least 0.99995 at six spins the model never saw. Template banks accept a mismatch of 0.03; the worst here is 5×10^{-5}.

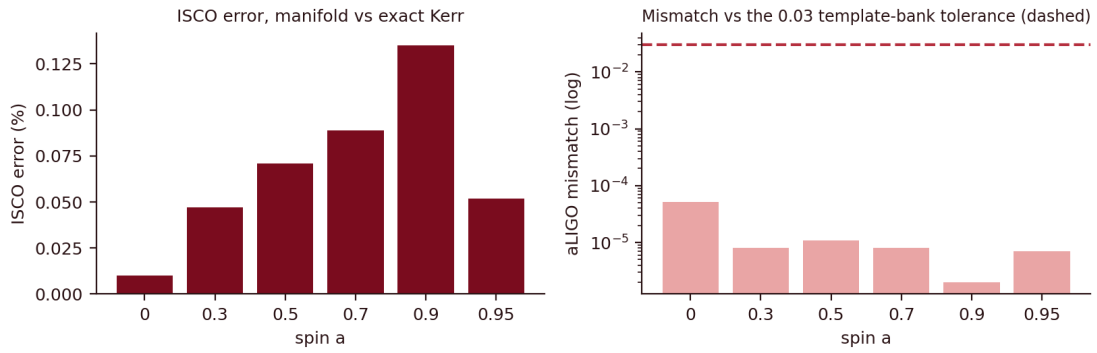


Figure 17. ISCO error and aLIGO mismatch at six spins not in the training set. The worst mismatch, at spin 0, is about 600 times inside the template-bank tolerance.

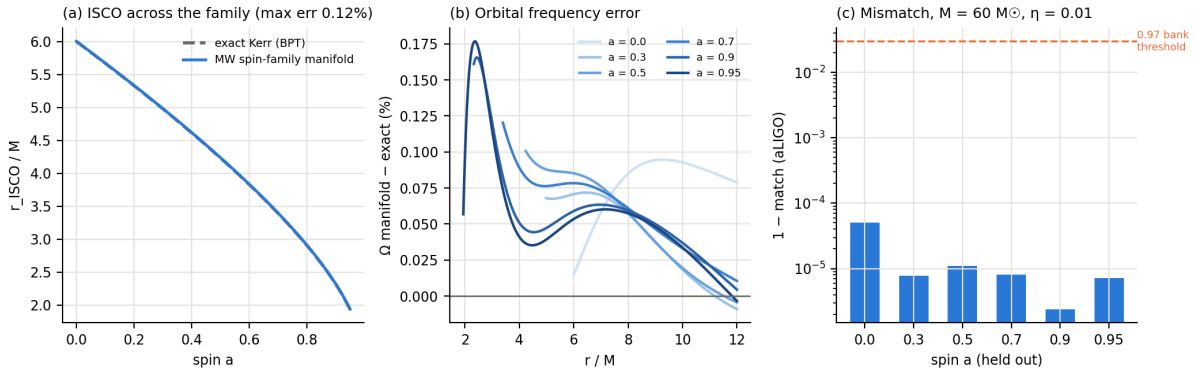


Figure 18. The spin-family figure from the repository: exact and learned ISCO curves lie on top of each other across 39 spins.

The two machines

Supercomputer route. Numerical relativity (NR) solves Einstein's equations for two merging black holes on a mesh. The SXS Collaboration's third catalogue (arXiv:2505.13378) holds 3,756 simulations with an estimated 480,000,000 core-hours in total and a median of 22 orbits per run. That is on average 480,000,000 ÷ 3,756 ≈ **1.28×10^5 core-hours** (4.6×10^8 core-seconds) per simulation, on clusters.

MW route. From the open code in the Kerr folder, on one laptop (Intel i5-12450H, no GPU):

Stage	Cost	How often
Exact geodesic data (6,000 orbits, 1.31 M points)	220 s	once
Train one manifold for spins 0 → 0.95	71.5 min × 6 threads ≈ 7.2 core-hours	once
Read Ω(r), E(r) and the ISCO for a new spin	0.30 s	per spin
One adiabatic inspiral template	0.16 s median (29 to 4,545 GW cycles)	per template

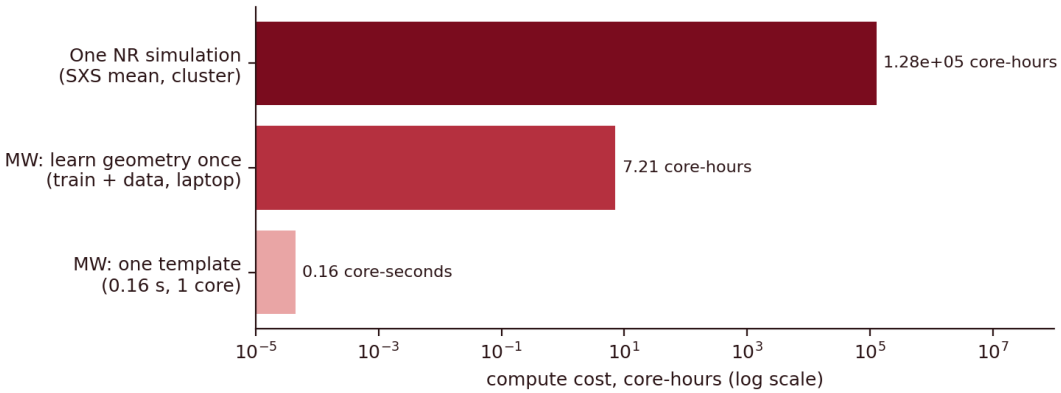


Figure 19. Compute cost on a log scale. Each step to the right is a factor of ten.

The 10^6–10^9 figure, and exactly what it is

Divide the cost of one NR simulation by the cost of producing a Kerr template through the manifold. The answer depends on three choices, shown rather than hidden. Let NR = 1.28×10^5 core-hours, one-time MW cost = 7.2 core-hours, template = 0.16 s:

Way of counting	Calculation	Ratio
Marginal: one template, charged as 1 core	4.6×10^8 core-s ÷ 0.16 s	2.9×10^9
Marginal: charge all 12 logical CPUs	4.6×10^8 core-s ÷ (0.16 s × 12)	2.4×10^8

Way of counting	Calculation	Ratio
All-in, a bank of 100 templates (12-core charge)	100 × NR ÷ (7.2 core-h + 100 templates)	1.8×10^{6}
All-in, a bank of 1,000 templates (1 core)	same, including one-time training	1.8×10^{7}
All-in, a bank of 10,000 templates (1 core)	same	1.7×10^{8}
All-in, a bank of 100,000 templates (1 core)	same	1.1×10^{9}
One NR simulation against the whole one-time MW pipeline	1.28×10^{5} core-h ÷ 7.2 core-h	1.8×10^{4}

So the range **10^{6} to 10^{9}** in core-hours is real arithmetic. It runs from a small bank counted conservatively (about 10^{6}) to the marginal cost of one more template (about 10^{9}). The one-time training cost is included in every row except the two marginal ones.

WHAT THE RATIO COMPARES

The ratio compares the compute bill for obtaining a Kerr template through the manifold with the bill for one numerical-relativity run. The real prize is **acquiring the geometry**: once the manifold is learned, any new spin costs 0.3 s.

Where the real gain is

The prize is **obtaining the geometry**. A new spacetime with no closed-form answer needs one hour of laptop training plus a few hundred validated runs, and then 0.3 s per parameter value. The alternative is a cluster-scale simulation campaign. Kerr is the control experiment where the exact answer is known, so the accuracy lost by the learned manifold can be measured: orbital frequency error at most 0.18%; ISCO error at most 0.12% over 39 spins; aLIGO match at least 0.99995 at all six test spins; no calibration factor.

The next experiment in the roadmap infers the metric from orbits alone, which is exactly the case where a cluster-scale alternative exists and a laptop-scale answer would matter.

Everything measured, in one place

Speed, size and training cost across the repositories and the live platform.

QUICK ANSWER

Querying a trained MW model costs milliseconds on a CPU in every repository. Training costs minutes to an hour on a laptop.

Domain	Query or run	One-time cost
Robot student (G1)	2.3 ms per control step, one CPU thread (budget 20 ms)	hours on a gaming laptop (data + training); the teacher was trained by NVIDIA on GPUs
Battery platform	2.3 ms per forecast; 0.22 ms per state; 137 kB model	about half an hour to an hour on one laptop
Kerr spacetime	0.30 s per new spin; 0.16 s per template	220 s data + 71.5 min training, ~7 core-hours
Gravity + quantum	exact geometry at 240 states, ~15 min CPU	~12 min per training run on a laptop RTX 3050
Kohn–Sham surrogate	0.2 ms per query (vs ~40 s per PySCF solve)	2.2 CPU-hours of PySCF data

Domain	Query or run	One-time cost
EUV lithography surrogate	2.2 ms per mask (vs 20.7 ms for the optics model)	training on 2,000 masks
1000-qubit QFT (mwqc)	~2–3 min, 32 kB to 1.44 MB of state	—
100/200-spin magnets (mwqc)	minutes on a CPU, checked vs exact	—

A pattern runs through the table: **the cost moves from the query to the one-time learning**, and that learning fits on a laptop in every case measured.

THE COMPUTE ANGLE

If you make a handful of queries, a solver or a library is probably the right tool. If you make thousands to millions of queries on the same physics, or you must combine laws from different fields, the one-time cost of learning the surface starts to pay.

The truth about the numbers

Radical transparency: what we measured, and the next milestone for each.

QUICK ANSWER

Every headline number points to a file you can open and a command you can run, and every one has a next milestone you can help reach.

> **“Pay once to learn the surface. Ask a million questions.”**

What we measured	Where to check it	Our next milestone
One typical binary-black-hole simulation costs about 1.3×10^5 core-hours; learning the Kerr family took about 7 core-hours on a laptop CPU, a compute-bill ratio of 10^6–10^9 depending on bank size.	SXS catalogue (arXiv:2505.13378); KERR_SPACETIME benchmark.json	Discover the metric from orbits alone
Kerr geometry to ≤0.18% in Ω, aLIGO match ≥ 0.99995, spins 0 to 0.95, no calibration factor.	KERR_SPACETIME v3 results	Deformed Kerr, eccentric and inclined orbits, comparison with NR waveforms
58 of 58 simulated humanoid tests; the student runs at 2.3 ms per step on one CPU thread.	MW_Robotics score58.py	A flight computer inside the 20 ms control step, then hardware
Gravity and quantum content in one learned object, with shuffle and no-prior controls (Hawking R^2 0.83 against 0.42).	QUANTUM_GRAVITY_WITH_MW / Nov-2026_tests	New regimes and richer data
A battery forecast in 2.3 ms on a CPU from a 137 kB model; trained in under an hour on one laptop; held-out error about 2.4 points against 18 for the physics formula alone.	edge benchmark; Sandia and HUST results (Chapter 4)	solver head-to-head, CPU-only training, tighter ranges, NCA
1000-qubit low-entanglement circuits and 100–200-spin dynamics on a laptop CPU, with an error bar, checked against exact answers; about 50% of the applicable Qiskit surface covered.	Quantum_Compute_MW README and results/	Bigger circuits and richer spin systems
Dr Zickert's PyQML code runs on mwqc by changing one import line, with no account or sign-up.	pyqml_port/	Every Volume 2 notebook on mwqc itself

What we measured	Where to check it	Our next milestone
Small models with an MW verifier inside GRPO, for structured verticals such as tabular ML.	Commercial work (Part VIII)	More verticals

Find the file, run the command, and you can say any of these with full confidence. Reach the next milestone and you can add a row of your own.

Join in: the bounty board

You do not need a cluster, a robot or a licence to work on any of this.

QUICK ANSWER

Reproduce a result, break a claim, try one of the open problems, or point the recipe at your own physics. Everything is on GitHub.

> **“Reproduce it. Break it. Build on it.”**

Your first weekend

When	What to do	What you will have
Friday night	Clone Quantum_Compute_MW and run the 20-second mwqc quickstart	Your first CPU quantum run, with an error bar
Saturday	Port Volume 1 of Dr Zickert's PyQML books with one import line	A whole quantum-ML book running on your laptop
Sunday	Run the Kerr benchmark (kerr_v2_gw.py) and post your CPU model and timing as a repository issue titled “Laptop Leaderboard”	Black-hole geometry on your machine, and your name on the board

The bounty board

Bounty	Why it matters	Reach out via
A flight computer inside the 20 ms step	Takes the humanoid result from simulation toward hardware	MW_Robotics issues
Discover the Kerr metric from orbits alone	The case where a cluster-scale alternative exists and a laptop answer changes the game	QUANTUM_GRAVITY_WITH_MW issues
Push the gravity-quantum model into new regimes	Shows how far one learned object can carry joint physics	QUANTUM_GRAVITY_WITH_MW issues
Bigger quantum experiments on mwqc	Share your qubit counts, depths, timings and the physics you studied	Quantum_Compute_MW issues
solver head-to-head and a CPU-only training run	A direct speed comparison on the same cells	Battery analytics, via the theory repository

Solved bounties are credited by name in the next edition of this book.

Who we would like to hear from

- **University and research labs** with a physics model you query many times, and no appetite for a cluster bill.

- **Robotics groups** with a robot and an interest in a physics flight computer, or who can make the real-time and hardware steps.
- **Numerical relativists and gravitational-wave groups** who can test the Kerr work against NR waveforms and push the metric-from-orbits experiment.
- **Quantum-computing teachers and students** who can use the mwqc and PyQML path and report where it breaks.
- **Battery labs and fleet operators** who can supply long-term data, other chemistries and sizes, or run the solver head-to-head.
- **Mathematicians** who can sharpen or correct the geometry. Corrections are welcome.

Open problems, by repository

MW_Robotics publishes its next big problem (58 of 58 inside the 20 ms control step) with ten ideas and a cheap first test for each. Run one and report the result either way:

#	Idea	Cheap first test
1	Full search with 2 ms look-ahead physics	the repository's single-command run; the cheapest next run
2	Search sizes between the cheap and the full search	find the smallest search that keeps 58 and time it
3	A correction that stays valid when it arrives late (a small feedback law instead of a fixed offset)	the four dance-rescue runs with asynchronous search
4	Search in fewer dimensions (the winning corrections may lie in a few directions)	log the winners, find their main directions
5	Spend the warning time: spread one search across several control steps	the alarm already leads the fall by tens of steps
6	A learned envelope with an exact check: the network proposes three or four candidates, the physics decides	does the true best appear among the network's top four, offline?
7	A simpler body model inside the search, the full model for the finalists	centroidal or pendulum model for most candidates
8	Batched physics on a consumer GPU (a comparison point)	63 rollouts in parallel
9	More authority, less search (a stepping rule or angular-momentum task)	hard rescues may need a step or an arm swing
10	Give the student memory (recurrent, or a short history)	does it call the flight computer less often?

QUANTUM_GRAVITY_WITH_MW / KERR_SPACETIME: infer the metric from orbits alone with no analytic metric supplied (stationarity, axisymmetry, asymptotic flatness and the Hamiltonian constraint as priors) and recover Ω(r), E(r) and the ISCO blind; then a spacetime with no closed form (deformed Kerr), checked against direct numerical solution; eccentric and inclined orbits; Teukolsky fluxes. **Nov-2026_tests**: add new regimes and richer data to the joint model.

Quantum_Compute_MW: run your own circuits and spin chains, share your timings and bond dimensions, and extend the toolkit with the solvers and lattices you care about.

Battery analytics: the solver head-to-head (SPM, SPMe, DFN) on the same held-out cells; a timed CPU-only training run; seed variance and held-out cell-type splits; calibration of the forecast ranges; long-term fleet data; other chemistries and cell sizes.

MW_Geometric_Intelligence: sharper proofs, a cleaner statement of the universality claim and its boundary, and an update of its README tables to the November results.

How to contribute

- **Reproduce first.** Pick the shortest check in a repository and run it. Report your numbers, whether or not they match.

- **Use GitHub issues and pull requests** in the relevant repository. Each repository has its own guidance (the robotics repository has a CONTRIBUTING.md).
- **Report results in the same form** as the repository does: the same tests, the same frozen model where one is given, the score, the hardware (CPU model, threads, governor) and how many runs. Do not tune on the held-out tests.
- **Post your timings.** Add your CPU model, thread count and times to the “Laptop Leaderboard” issue so everyone can see what a laptop can do.
- **Use it freely.** The repositories are open source (Apache-2.0 for MW_Robotics and Quantum_Compute_MW, MIT for the Kerr folder), so you can build on them in research, teaching and products.

The ladder, for a new reader

If you have…	Start with…	You will learn…
An afternoon	The pendulum demo (robotics) or the 20-second mwqc quickstart	The recipe in miniature, on any CPU
A weekend	Kerr: read the folder README, run kerr_v2_gw.py; or port Volume 1 of the PyQML books	How a learned manifold is checked against an exact answer; or the quantum path from book to 100 qubits
A month	Reproduce the 58-test robotics run with your own student; or the Nov-2026 tests	How controlled tests are built; what a physics flight computer does
A research programme	Point the recipe at your own physics, with a solver you already own to make the training data	Whether a learned surface pays for your problem, and where it shines

THE SHORT VERSION

Use whatever little maths we know to democratise, not to build moats. Compute is a good tool, and it should not be the only door. If part of what a cluster does can be done by a better representation of the problem, then more people can do science and build robots, and we would like to find out how much.

The open-core model, and MW for language models

Open mathematics and tools; turnkey services that sustain the work.

QUICK ANSWER

MW shines wherever there are laws to write down. English has no governing equations, so for language models MW lives in one high-value place: the **verifier that scores a model's work in GRPO**, in structured verticals such as tabular machine learning. That is where small models plus MW can punch far above their size.

Where MW shines: systems with laws

Everything in this book rests on one move: write the laws a system obeys as priors, so that the allowed states form a surface worth learning (Chapter 2). A battery obeys electrochemistry. A robot obeys mechanics. A black hole obeys Einstein's equations. A quantum chain obeys the Schrödinger equation. Each gives a prior, and each prior cuts the space of possibilities down.

English has no governing equations. There is no conservation law for a sentence, no PDE for a paragraph, no energy that a paragraph must balance. The set of “valid” English texts is not carved out by physics, so there is nothing for a physics prior to cut. That is why the geometry in this book will not help a chatbot write better prose, answer a general question, or reason in free text.

Where it shines for language models: the verifier in GRPO

Modern language-model training increasingly uses **reinforcement learning with verifiable rewards**. In one popular form, GRPO (group-relative policy optimisation), a model produces several attempts at a task, each attempt is scored by a **verifier**, and the model is nudged toward the attempts that scored better than their group. The model's text is not what is being checked. The *work the text produces* is: code that runs, a pipeline that trains, a prediction that scores.

That is where MW fits. If the work has checkable structure (a data table with a target, a validation split, a leakage rule, a known physical or statistical constraint), then priors and geometric checks can act as the verifier, and can supply a reward that is more reliable than a bare score. The model does the judgement (which features to build, which decisions to make, what to try next), and the harness and the verifier do the dependable part.

Which verticals

In scope: tasks with exploitable structure, where a narrow harness, operator libraries and MW verifiers do the reliable work and the model makes the judgement calls. Structured tabular machine learning, such as the tabular competitions in MLE-bench, is the first such vertical.

The working rules for the project are the same as everywhere in this book: models small enough to run on a single consumer GPU (up to about 8 billion parameters at 4-bit), every result reported with its denominator, and controls always shown, namely the harness with no language model at all and the bare model with no harness. A harness that does most of the work must be credited for it, and the controls are how that is measured.

The open-core model

The mathematics and the tools are free to change the world. The four repositories in this book are open, their numbers are open, and anyone can read, run and extend them. **The turnkey services are how the team sustains the work:** the battery analytics platform of Chapter 4, the dashboard services, and the language-model benchmark harness with its MW verifier for GRPO. That last one is private, commercial work, delivered to users as a plugin for the DeepSeek Harness (DSH), the same route the battery and dashboard services take to a chat interface.

It is the model that open-source communities know and respect: open foundations that everyone builds on, and sustainable services on top. Every open result stays open, and the services pay for the next result.

What to expect

- **Expect** MW to keep growing wherever a model's work can be checked against structure or physics, in training (as a reward) and in deployment (as a guard).
- **Expect** every result we publish to come with its denominator and its controls, so you can trust it and reproduce it.
- **Expect** useful MW wherever a model's work can be checked against structure or physics, in training (as a reward) and in deployment (as a guard).
- **Do expect** any result that is published to come with its denominator and its controls.

THE SHORT VERSION

MW is a physics method. It works where there are laws to write down. For language models that means the part of training where a model's *work* is verified against structure, in verticals such as structured tabular ML. The commercial work in that area is private. Everything else in this book is open.

THE COMPUTE ANGLE

The cost logic is the same as elsewhere in the book: a small model, a harness and a verifier, run on a single consumer GPU, instead of a frontier model on a cluster. For this vertical, that is the commercial proposition.

BACK MATTER

Glossary

Terms used in this book, in plain words.

Term	Meaning
Compute gate	The barrier that keeps teams without a cluster or a GPU budget out of frontier physics and AI. The thing this book sets out to lower.
Democratisation dividend	More people able to do the science, not only a smaller bill.
Known frontier	A marked next milestone that anyone can help reach, with a test you can run.
Open core	Open mathematics and tools for everyone, with turnkey services that sustain the work.
Allowed set / manifold	The set of states that obey the laws. A curve or surface that looks flat up close.
Bayesian prior	A statement of what you believe before looking at data, such as “voltage only falls as the cell ages”.
Bond dimension (χ)	The size of the link between neighbouring qubits in a matrix product state. It must grow with entanglement.
Capture point	Where a robot would have to step to come to a stop. A balance law used in the flight computer.
Chart	A flat map of part of a manifold.
Control volume / element / cell	The small pieces a solver chops the domain into (finite-volume or finite-element methods).
Decoder	The part of a VAE that turns latent knobs into measurements. A smooth map, so it has a Jacobian.
DFN model	Doyle–Fuller–Newman: coupled PDEs for lithium diffusion, charge conservation and reaction kinetics inside a battery cell.
Envelope	The set of physical laws a flight computer keeps the robot inside.
Explicit / implicit step	Explicit: new value from old values only (simple, can blow up). Implicit: solve for new values together (safe for stiff problems).
Flight computer	A physics model that predicts the policy's plan and corrects it only if it leaves the envelope.
Geodesic	The shortest path between two points on a curved surface.
ISCO	Innermost stable circular orbit around a black hole.
Jacobian	The matrix of slopes of every output with respect to every input.
Knob	The hidden quantity that moves you along the allowed set, such as the usable lithium left (z).
Match (waveforms)	A normalised overlap between two waveforms. 1 is identical; template banks use 0.97 as a minimum.
Mesh	The chopped-up domain a solver works on, made of nodes and cells.
Method of lines	Turning a PDE into many coupled ODEs, one per node.
MPS (matrix product state)	A chain of small tensors that stores a low-entanglement quantum state with memory linear in the qubit count.

Term	Meaning
MW distance	How far a state sits from the learned manifold, weighted by physics-derived importances.
Newton's method	Repeatedly linearising a nonlinear problem and solving the linear version.
Numerical relativity (NR)	Solving Einstein's equations for merging black holes on a mesh, on supercomputers.
ODE / PDE	Ordinary or partial differential equation: how a number, or a whole profile, changes.
Pullback metric	The ruler $G = J^T W J$ carried from measurement space back to knob space through the map x(z).
Riemannian manifold	A manifold with a ruler (a metric) that gives lengths, angles and curvature.
Stiffness	When one part of a system changes much faster than you want to step, forcing tiny steps.
Surrogate	A trained model that stands in for an expensive calculation.
VAE	Variational autoencoder: an encoder and decoder trained to compress and rebuild data through a small latent space.
Van Vleck determinant	A factor in the geometry of the world function that measures how neighbouring geodesics spread.

BACK MATTER

Sources and links

Where every number in this book comes from.

Repository or work	Link
MW_Geometric_Intelligence (theory)	github.com/RahulModak74/MW_Geometric_Intelligence
MW_Robotics (Apache-2.0)	github.com/RahulModak74/MW_Robotics
QUANTUM_GRAVITY_WITH_MW (Nov-2026_tests, KERR_SPACETIME)	github.com/RahulModak74/QUANTUM_GRAVITY_WITH_MW
Quantum_Compute_MW (mwqc)	github.com/RahulModak74/Quantum_Compute_MW
Battery analytics, live app	batteryanalytics.bayesiancybersecurity.com
Student-t(ν = 0.8) prior preprint (Zenodo)	doi.org/10.5281/zenodo.19304813
World Models with an Irreversible Time Axis (Zenodo)	doi.org/10.5281/zenodo.22954118
Kerr software citation (Zenodo)	doi.org/10.5281/zenodo.19305273
SXS Collaboration, third catalogue of binary black hole simulations	arXiv:2505.13378 (3,756 simulations; 480,000,000 core-hours)
Bardeen, Press, Teukolsky, Rotating black holes	ApJ 178, 347 (1972)
Johannsen and Psaltis, deformed-Kerr metric	Phys. Rev. D 83, 124015 (2011)
LALSuite 7.7.1 (LIGO reference models and noise curve)	LIGO Scientific Collaboration
Kim et al., IBM 127-qubit utility experiment	Nature 618, 500 (2023)
Tindall et al.; Begušić and Chan (classical reproductions)	PRX Quantum 2024; Science Advances 2024

Repository or work	Link
Dr Frank Zickert, Hands-On Quantum Machine Learning With Python, Vols. 1 and 2	pyqml.com

The preprints are not peer reviewed. This book is an explanation of the repositories as they stood in November 2026, and the repositories are the authority on their own numbers.