Supplementary Materials: Learning the Effective Spin Hamiltonian of a Quantum Magnet

Yu *et al.*

A. Automatic Parameter Searching Algorithms

Here we briefly describe the five algorithms employed in our Hamiltonian searching, including the Bayesian (Algorithm 1), auto-gradient (Algorithm 2), Nelder-Mead simplex (Algorithm 3), simulated annealing (Algorithm 4), and the random grid search methods (Algorithm 5). These searching schemes can be combined with various many-body solvers in a rather flexible manner, while have different efficiency and accuracy, to determine the spin Hamiltonian [cf. Fig. 3(c) of the main text].

Algorithm 1: Bayesian Optimization

Initialize a statistical model;
for $i = 1$ **to** n **do**
  Select the next point $\mathbf{x}_{i+1}$ to evaluate by maximizing the Acquisition function $\mathbf{x}_{i+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_i)$;
  Evaluate objective function y_{i+1} at $\mathbf{x}_{i+1}$;
  Augment data $\mathcal{D}_{i+1} = \{\mathcal{D}_i, (\mathbf{x}_{i+1}), y_{i+1}\}$;
  Update statistical model with $\mathcal{D}_{i+1}$.

Algorithm 2: Multi-Restart Auto-Gradient

1 **for** $i = 1$ **to** n **do**
2 Randomly choose a starting point $\mathbf{x_i}$;
3 **for** $j = 1$ **to** m **do**
4 $x_{i,j+1} = x_{i,j} + \lambda B^{-1} \nabla f(x_{i,j})$, where B is an approximate Hessian and λ the learning rate;
5 **if** converged **then**
6 Go to line 2.

Algorithm 3: Nelder-Mead Simplex Algoritm

Initialize a list of point $(\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_{n+1})$ such that $f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_{n+1})$ are in an increasing order, where n is the dimension of $\mathbf{x}$;
while not converged **do**
  Generate the reflected point $\mathbf{r} = 2\mathbf{m} - \mathbf{x}_{n+1}$ where $\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$;
  switch $f(\mathbf{r})$ **do**
    case $f(\mathbf{r}) < f(\mathbf{x}_1)$ **do**
      Generate the expansion point $\mathbf{s} = \mathbf{m} + 2(\mathbf{m} - \mathbf{x}_{n+1})$;
      if $f(\mathbf{s}) < f(\mathbf{r})$ **then**
        Replace $\mathbf{x}_{n+1}$ with $\mathbf{s}$, sort $\{\mathbf{x}_i\}$ and **continue**;
        else
          Replace $\mathbf{x}_{n+1}$ with $\mathbf{r}$, sort $\{\mathbf{x}_i\}$ and **continue**;
    case $f(\mathbf{x}_1) \le f(\mathbf{r}) < f(\mathbf{x}_n)$ **do**
      Replace $\mathbf{x}_{n+1}$ with $\mathbf{r}$, sort $\{\mathbf{x}_i\}$ and **continue**;
    case $f(\mathbf{x}_n) \le f(\mathbf{r}) < f(\mathbf{x}_{n+1})$ **do**
      Generate $\mathbf{c} = \frac{1}{2}(\mathbf{r} + \mathbf{m})$;
      if $f(\mathbf{c}) < f(\mathbf{r})$ **then**
        Replace $\mathbf{x}_{n+1}$ with $\mathbf{c}$, sort $\{\mathbf{x}_i\}$ and **continue**;
    case $f(\mathbf{x}_{n+1}) \le f(\mathbf{r})$ **do**
      Generate $\mathbf{cc} = \frac{1}{2}(\mathbf{x}_{n+1} + \mathbf{m})$;
      if $f(\mathbf{cc}) < f(\mathbf{x}_{n+1})$ **then**
        Replace $\mathbf{x}_{n+1}$ with $\mathbf{cc}$, sort $\{\mathbf{x}_i\}$ and **continue**;
  Replace $\mathbf{x}_i$ with $\frac{1}{2}(\mathbf{x}_i + \mathbf{x}_1)$ for all $i = 2, 3..., n+1$, sort $\{\mathbf{x}_i\}$.

Algorithm 4: Simulated Annealing Algorithm

Initialize temperature $T_0 = 100$ and evaluate the loss function at point $\mathbf{x}$;
for $i = 1$ **to** n **do**
  Generates a random trial point $\mathbf{p}$;
  if $f(\mathbf{p}) < f(\mathbf{x})$ **then**
    Accept point $\mathbf{p}$;
    else
      Accept point $\mathbf{p}$ according to probability $1/(1 + \exp(\frac{\Delta}{T}))$, where $\Delta = f(\mathbf{p}) - f(\mathbf{x})$ and T is the current temperature;
  Cooling down $T = T_0 \times 0.95^i$.

Algorithm 5: Random Grid Searching

Discretize the parameter space into a uniform grid with n nodes $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$;
for $i = 1$ **to** n **do**
  Random select one of the unevaluated nodes $\mathbf{x}_i$ and calculate the $\mathcal{L}(\mathbf{x}_i)$;
  Mark $\mathbf{x}_i$ as evaluated.

B. Automatic Differentiation

In this section, we provide more details of automatic differentiation used in our auto-gradient scheme. Automatic differentiation is a well-developed technique in neural networks and deep learning [52]. A central ingredient of automatic differentiation is the so-called computational graph (see Fig. S1 for a typical computational graph for many-body calculations). To generate such a computational graph, one starts with the input parameters, goes through a number of intermediate computation nodes, and ends up with the final loss function.

To be specific, for the quantum many-body problems afore-mentioned in main text, starting with several model parameters, e.g., $\mathbf{x} \equiv \{J, \Delta, g, \cdots\}$, one defines the many-body model Hamiltonian $H(\mathbf{x})$. Given it either ED or thermal tensor network calculations, the partition function Z and thereafter thermodynamic observables $\{O_\alpha\}$, can be obtained. Basing on the calculated observables $\{O_\alpha\}$, a loss function can be properly designed [cf. Eq. (1)]. The above procedure constitutes a forward evaluation of the loss function, and henceforth a computational graph $\mathbf{x} \to H \to Z \to O_\alpha \to \mathcal{L}$ is generated (cf. the right-directed lines in Fig. S1).

On the fly of the forward process, the derivatives between adjacent computation nodes, i.e. $\{\frac{\partial H}{\partial \mathbf{x}}, \frac{\partial Z}{\partial H}, \frac{\partial O_\alpha}{\partial Z}, \frac{\partial \mathcal{L}}{\partial O_\alpha}\}$, are stored. Thus the derivatives of loss function with respect to the input parameters can be evaluated automatically via a chain rule,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial \mathcal{L}}{\partial O_\alpha} \frac{\partial O_\alpha}{\partial Z} \frac{\partial Z}{\partial H} \frac{\partial H}{\partial \mathbf{x}}. \tag{S1}$$

In our cases, since the number of input parameters [components in $\mathbf{x}$, typically a few to $O(10)$] is larger than the output (just a single value of loss $\mathcal{L}$), it is therefore more efficient to evaluate Eq. (S1) following the reverse-mode automatic differentiation (i.e., from left to right on the right-hand side of the equation). In this work, we have implemented a differentiable ED calculation with Pytorch [75], and the generalization to tensor networks is also feasible [42, 43].

C. Bayesian Optimization with Gaussian Process: Kernel Function, ARD, and Acquisition Function

As shown in Algorithm 1 and Fig. S2, in the Bayesian optimization we need to iteratively update a statistical model that can be used to estimate the overall landscape of $\mathcal{L}$, based on history queries. In this work, we choose a commonly used statistical model called Gaussian process, which fits well our problem and is denoted as

$$\mathcal{GP} : \mathcal{X}, \mathcal{D} \longrightarrow \mu, \sigma \tag{S2}$$

where $\mathcal{X}$ is the parameter space spanned by the parameter vectors $\mathbf{x}$, which could include, in practice, components J, g, Δ, etc. The set of total n history queries is noted as $\mathcal{D}_n = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n))$, with y_i being the evaluated function value at parameter $\mathbf{x_i}$, i.e., $\mathcal{L}(\mathbf{x_i})$. Then by assuming a joint multivariate Gaussian distribution over $(y_1, y_2, ..., y_n; y_{n+1})$, with the

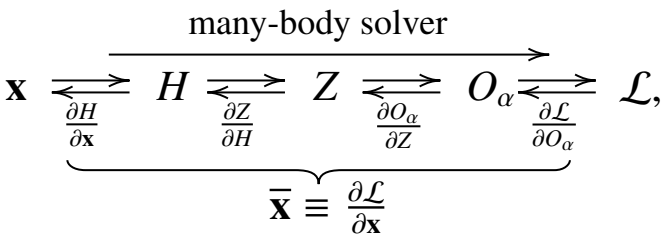

FIG. S1. A typical computational graph of the quantum many-body calculations, with the forward process indicated by all the right-directed lines, and the backward process by the left-directed lines.

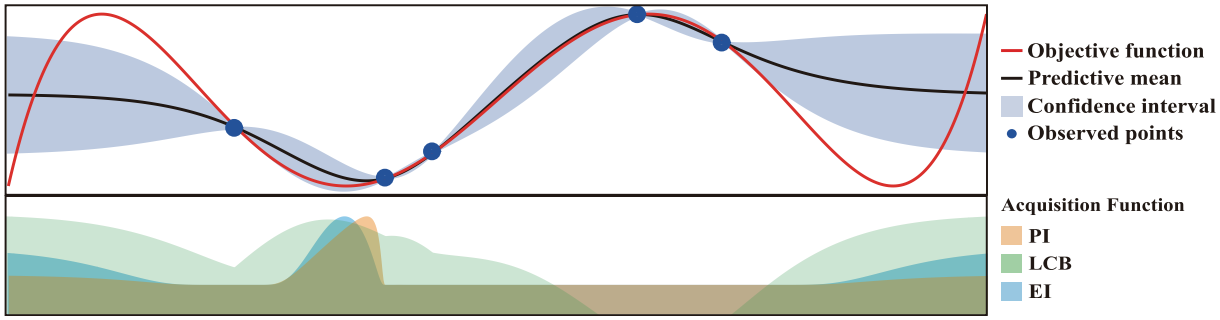

FIG. S2. Red line in the upper panel represent the unknown objective function, with blue dots the evaluated points. The black line and shade represent the predicted mean and confidence interval, respectively. Various shades in the lower panel corespondent to different acquisition functions.

covariances characterized by a kernel function $k(\mathbf{x_i}, \mathbf{x_j})$, and y_{n+1} to be estimated at $\mathbf{x}_{n+1}$, we can compute a *posterior* distribution of $y_{n+1} \sim \mathcal{N}(\mu_n, \sigma_n^2)$ by

$$\mu_n(\mathbf{x}_{n+1}) = \mathbf{k}(\mathbf{x}_{n+1})^\mathsf{T}\mathbf{K}^{-1}\mathbf{y}, \tag{S3}$$

$$\sigma_n^2(\mathbf{x}_{n+1}) = k(\mathbf{x}_{n+1}, \mathbf{x}_{n+1}) - \mathbf{k}(\mathbf{x}_{n+1})^\mathsf{T}\mathbf{K}^{-1}\mathbf{k}(\mathbf{x}_{n+1}), \tag{S4}$$

where a constant zero prior mean is assumed in the space $\mathcal{X}$. $\mathbf{y} = (y_1, y_2, ..., y_n)^\mathsf{T}$ is the vector of evaluated function values, $\mathbf{k}(\mathbf{x})^\mathsf{T} = (k(\mathbf{x}, \mathbf{x_1}), k(\mathbf{x}, \mathbf{x_2}), ...)$ and $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ are respectively the covariance vector and matrix , where $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}$ represent the calculated parameter points in the history queries. The quality of GP regression to fit the real landscape is determined by the choice of kernel function $k(\mathbf{x}, \mathbf{x}')$. In practice, we chose a Matérn−5 kernel, i.e.,

$$k_{Matérn5}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\sqrt{5}r)(1 + \sqrt{5}r + \frac{5}{3}r^2), \tag{S5}$$

where in the kernel function $r^2 = (\mathbf{x} - \mathbf{x}')^\mathsf{T}\mathbf{\Lambda}(\mathbf{x} - \mathbf{x}')$ and $\mathbf{\Lambda}$ is a diagonal matrix with length scale θ_i^2. Then we are left with hyperparameters θ_i to be determined, which describe the scale of the kernel function for each parameter. Fortunately, the GP model provide us a nice analytical expression of for the marginal likelihood with the following expression,

$$\log p(\mathbf{y}|\mathbf{x}, \theta) = -\frac{1}{2}\mathbf{y}^\mathsf{T}(\mathbf{K}^\theta)^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}^\theta| - \frac{n}{2}\log(2\pi). \tag{S6}$$

Note here θ represents a set of all the hyperparameters, and we can easily compute θ^* that maximize the marginal likelihood, as long as the kernel is differentiable with respect to θ. By denoting $\theta^* = \theta_{\mathrm{ML}}$, we take it as a point estimator for our hyperparameters. Besides, one can also use a maximum a posteriori estimation θ_{MAP} as the kernel parameters. This technique is often referred to as automatic relevance determination (ARD) kernels.

With the estimated mean μ_n and variance σ_n, we can estimate the landscape $\mathcal{L}(\mathbf{x})$, and choose the next point $\mathbf{x}_{n+1}$ by maximizing an acquisition function $\alpha(\mathbf{x})$, i.e., $\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x})$. A careful design of acquisition function is needed to balance the efficiency and exploration of the parameter space. Here we introduce three very popular acquisition functions that are commonly adopted: probability of improvement (PI), expected improvement (EI) and lower confidence bound (LCB). To be clear of the notations, the term "improvement" in the context of minimization means the diminution of the minimum. The three acquisition

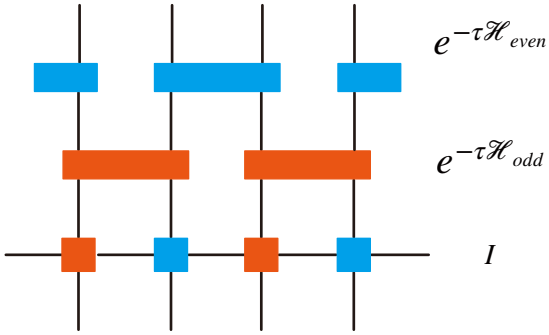

FIG. S3. Tensor network representation of the density matrix $\rho(\tau)$ [cf. Eq. (S15)]. The bottom line is the infinite-temperature density operator $\rho(0) = I$. The above blue/red blocks indicate the Trotter gates on even/odd bonds.

functions are

$$\alpha_{\mathrm{PI}}(\mathbf{x}; \mathcal{D}_n) = \mathbb{P}[\mathcal{L}(\mathbf{x}) \leq \tau] = \Phi\left(\frac{\tau - \mu_n(\mathbf{x})}{\sigma_n(\mathbf{x})}\right), \tag{S7}$$

$$\alpha_{\mathrm{EI}}(\mathbf{x}; \mathcal{D}_n) = \mathbb{E}[\tau - \mathcal{L}(\mathbf{x})] = (\tau - \mu_n(\mathbf{x}))\Phi\left(\frac{\tau - \mu_n(\mathbf{x})}{\sigma_n(\mathbf{x})}\right) + \sigma_n(\mathbf{x})\phi\left(\frac{\tau - \mu_n(\mathbf{x})}{\sigma_n(\mathbf{x})}\right), \tag{S8}$$

$$\alpha_{\mathrm{LCB}}(\mathbf{x}; \mathcal{D}_n) = \kappa\sigma_n(\mathbf{x}) - \mu_n(\mathbf{x}), \tag{S9}$$

where where ϕ and Φ denote the PDF and CDF of normal distribution, and $\tau = \mathcal{L}_{\min} - \xi$ with ξ an adjustable empirical parameter, and so is κ in LCB. It has been shown in previous works that $\xi = 0.01\sigma_f$, with σ_f being the standard deviation of $\mathbf{y}$, constitutes a setting that has an overall very good performance [55], which is adopted in this work. A visualization of Gaussian process and acquisition functions is in Fig. S2. An open-source python package was used in this work for numerical experiments [76]. Moreover, one could also choose information-based acquisition function or a portfolio of acquisition strategies to balance the efficiency and over-all exploration.

D. Thermodynamics Quantum Many-body Solvers

In this section, we introduce the basic idea of some quantum many-body calculation methods, including exact diagonalization (ED), linearized tensor renormalization group algorithm (LTRG) [36, 37], and exponential tensor renormalization group (XTRG) [39, 40].

ED. To calculate the thermodynamic properties of a quantum many-body systems, one needs to obtain the partition function $\mathcal{Z} = \mathrm{tr}(\hat{\rho}) = \mathrm{tr}(e^{-\beta\mathcal{H}})$ with high precision. For quantum lattice models with d-dimension local Hilbert space ($d = 2$ for spin-1/2 systems), the $\mathcal{N}$-site many-body bases constitute a $d^{\mathcal{N}}$-dimension space, and the Hamiltonian $\mathcal{H}$ is a $d^{\mathcal{N}} \times d^{\mathcal{N}}$ matrix. Limited by the numerical resources, currently one can only store and diagonalize a spin-1/2 Hamiltonian with size of $\mathcal{N} \lesssim 20$ sites. For those small systems, we diagonalize $\mathcal{H}$ by an invertible matrix $\mathcal{U}$ as

$$\mathcal{H} = \mathcal{U}\mathcal{D}\mathcal{U}^{-1} \tag{S10}$$

with $\mathcal{D}$ diagonalized. Thereafter, we obtain the density matrix of the system at the inverse temperature β

$$\rho = e^{-\beta\mathcal{H}} = \mathcal{U}\, e^{-\beta\mathcal{D}}\, \mathcal{U}^{-1}, \tag{S11}$$

the partition function

$$\mathcal{Z} = \mathrm{tr}(e^{-\beta\mathcal{H}}) = \mathrm{tr}(e^{-\beta\mathcal{D}}), \tag{S12}$$

and thus other thermodynamic quantities.

LTRG and XTRG. For larger system sizes, we resort to thermal tensor network methods LTRG (1D) and XTRG (2D) in this work. The basic idea of LTRG is to, firstly slice the lower-temperature density matrix $\rho(\beta) = e^{-\beta\mathcal{H}}$ into N small slots $\tau = \beta/N$, i.e.

$$\rho(\beta) = e^{-\beta\mathcal{H}} = (e^{-\tau\mathcal{H}})^N. \tag{S13}$$

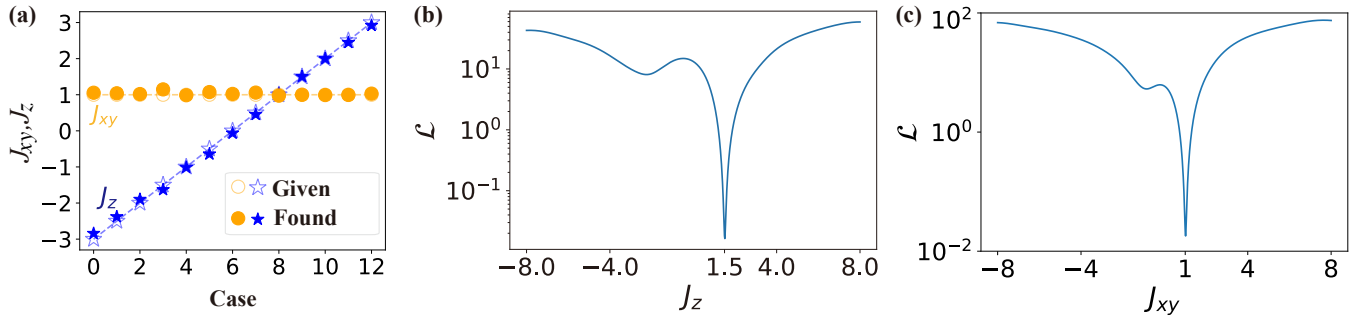

FIG. S4. (a) Unfilled asterisks and circles indicate given parameters J_z and J_{xy}, and filled asterisks and circles mark the found parameter by Bayesian optimization after 150 iterations. (b) A cross cut of the 2D landscape $\mathcal{L}$ in Fig. (2) of the main text, $\mathcal{L}(1.025, J_z)$ vs. J_z, with $J_{xy} = 1.025$ fixed at the predicted optimal value. (c) shows the cross cut of the 2D landscape $\mathcal{L}$ in Fig. (2), $\mathcal{L}(J_{xy}, 1.49)$ vs. J_{xy}, with a fixed $J_z = 1.49$ at its predicted optimal value. A sharp deep near the optimal parameter point can be clearly observed in both (b) and (c) panels.

For a one-dimensional system that contains only the nearest-neighboring interactions, the Hamiltonian can be divided into odd and even parts such that

$$\mathcal{H} = \mathcal{H}_{odd} + \mathcal{H}_{even}. \tag{S14}$$

Generally, these two parts are non-commutative, so we need to use Trotter-Suzuki decomposition to separate the two terms as

$$\rho(\tau) = e^{-\tau(\mathcal{H}_{odd}+\mathcal{H}_{even})} = e^{-\tau\mathcal{H}_{odd}} e^{-\tau\mathcal{H}_{even}} + O(\tau^2). \tag{S15}$$

Now we arrive at

$$\rho(\beta) = [\rho(\tau)]^N \simeq (e^{-\tau\mathcal{H}_{even}} e^{-\tau\mathcal{H}_{odd}})^N \tag{S16}$$

with discretization error $O(\tau^2)$. The tensor network representation of $\rho(\tau)$ is shown in Fig. S3, with an infinite-temperature density matrix (identity matrix) $\rho(0) = I$ explicitly shown. Therefore, Eq. (S16) can be viewed as a cooling process following a linear temperature gird, i.e. $0 \to \tau \to 2\tau \to 3\tau \to \cdots \to N\tau \equiv \beta$.

For a 2D or a quasi-1D systems with long range interactions, XTRG is a very powerful cutting-edge thermodynamics solver. Inspired by the logarithmic scaling of entanglement entropy $S_E \sim \ln\beta$ in critical systems, XTRG evolves the thermal state by squaring the density matrix with itself iteratively, yielding a logarithmic β grid when cooling the systems. In XTRG, we can start with an very high-temperature thermal state $\hat{\rho}(\tau)$, via series expansion construction

$$\hat{\rho}(\tau) = \sum_n \frac{(-\tau H)^n}{n!}.$$

Subsequently, by squaring the density matrix itself repeatedly, i.e.,

$$\hat{\rho}(2^n\tau) \cdot \hat{\rho}(2^n\tau) \to \hat{\rho}(2^{n+1}\tau),$$

we reach the low-temperature regime rapidly along a logarithmic inverse-temperature grid $\tau \to 2\tau \to 2^2\tau \to \ldots \to 2^n\tau$. As the exponential acceleration of cooling process has much smaller truncation steps, the precisions of XTRG calculations are significantly improved compared to the traditional linear cooling scheme, and thus it constitutes a very promising tool to perform low-temperature simulations.

The partition function $\mathcal{Z} = \mathrm{tr}[\rho(\beta)]$ can thus be obtained by fully contracting the tensor network, and the relevant thermodynamic quantities can be obtained directly from the partition function as,

$$f = -\frac{1}{\beta}\ln\mathcal{Z} \tag{S17a}$$

$$C = \beta^2 \frac{\partial^2 \ln\mathcal{Z}}{\partial\beta^2} \tag{S17b}$$

$$M = -\frac{\partial f}{\partial h} \tag{S17c}$$

$$\chi = \frac{M}{h} \tag{S17d}$$

where f is the free energy, C is the heat capacity, h is the magnetic field strength, M is the magnetization, and χ is the magnetic susceptibility.

E. Loss Function Adaptation

In the course of automatic parameter searching, we stick to the loss in Eq. (1) of the main text, with necessary adaptation for a better performance. Firstly, we show in Fig. S5(a) that the landscape of the native loss in Eq. (1) turns out to be not very ideal, in a sense that the global optimal parameter point (with minimal loss) resides within a rather flat region and is difficult to precisely locate it (see the performance in Fig. S6).

In practice, we can adapt the loss function to mitigate this problem, by transforming the native loss $\mathcal{L}$ into a logarithmic function [Fig. S5(b)]

$$\mathcal{L}' = \log(\mathcal{L}),$$

or the logistic-like functions [Fig. S5(c,d)]

$$\mathcal{L}_{\text{logistic}} = [1 + \exp(-\mathcal{L}/\tau)]^{-1} - 1/2.$$

In the latter case $\mathcal{L}_{\text{logistic}}$, different choices of τ values result in different landscapes. As shown in Fig. S5(c), $\tau = 300$ [$\sim \max(\mathcal{L})/10$] can offer some improvement to the region near the optimal parameter point and indeed lends a better performance (see Fig. S6). However, too small τ, e.g., $\tau = 10$ in Fig. S5(d), while making the optimal region even more steep, could introduce a "barren plateau" in the landscape that is problematic for the Bayesian optimization, preventing an efficient parameter searching (Fig. S6). In practice, the choice of parameter τ requires some prior knowledge of the fitting loss, e.g., its order of magnitude, which makes the logistic-like loss design less useful in practice.

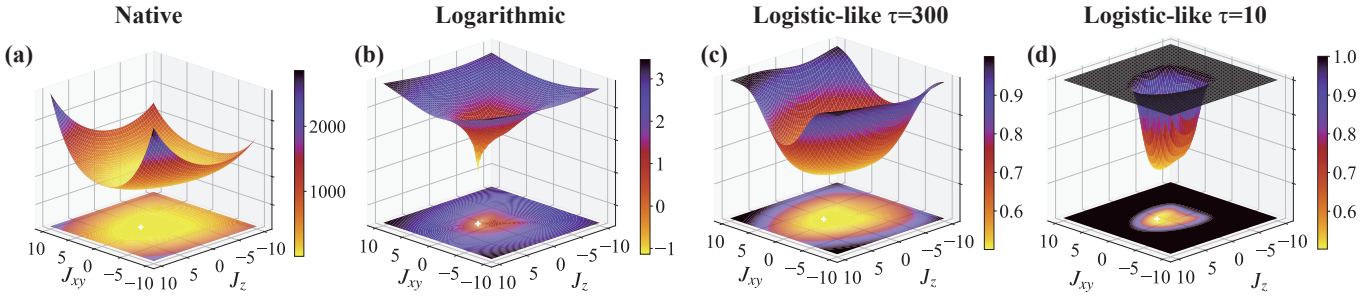

FIG. S5. The landscapes of (a) native fitting loss function, (b) logarithmic loss, and the logistic-like loss with (c) $\tau = 300$ and (d) $\tau = 10$. The same 'experimental" thermal data are generated from the XXZ chain with $J_{xy} = 1$ and $J_z = 1.5$, the same as that used in Figs.2 and 3 of the main text.

On the other hand, the logarithmic function $\mathcal{L}'$ in Fig. S5(c), which we find a very accurate and robust scheme in practical calculations, has clearly the best performance as compared to other loss function design. This can be understood from Fig. S6, where $\mathcal{L}'$ has a curved and sharpened regime around the optimal point while still preserving the overall shape of the landscape that is favorable for the intelligent optimizers.

F. More Results on the XXZ Heisenberg Spin Chain

With the training data from the given XXZ spin-chain model, here we show more cases to further validate the robustness of our method. For clarity, the ED calculation of 10 sites XXZ spin chain is used as a rudimentary many-body solver, although in practice we find ED calculations with 8-12 sites lead to virtually the same performance. In Fig. S4, we choose different $J_z \in [-3, 3]$ and a fixed $J_{xy} = 1$, and find that the Bayesian optimization can always retrieve the correct parameters in all cases and thus constitutes a robust approach.

Then we show the landscapes obtained at different T_{cut} temperatures, fitting jointly the specific heat C_m and susceptibility χ_z, and observed various landscapes in Fig. S7. In the definition of $\mathcal{L}$, c.f. Eq. (1) of the main text, when only C_m and χ_z are included, the optimizers can find two optimal parameters $J_{xy} = \pm 1$ and $J_z = 1.5$, which is understandable as indeed the two parameter points correspond to exactly the same energy spectra for $J_{xy} = \pm 1$, and our approach can automatically "learn" this fact from the thermal data. Such a two-fold degeneracy in the landscape can be removed once χ_{xy} is introduced to $\mathcal{L}$, where we can lift this degeneracy, as shown in Fig. S8.

Notably, we find that for a rather high T_{cut}, an oval ring with $J_x^2 + J_y^2 + J_z^2 = 2J_{xy}^2 + J_z^2 = const.$ indicated by the dashed circle in Fig. S7 (a-c). This can be understood, as the high temperature expansion of C_m only depends on the squared sum of spin XXZ

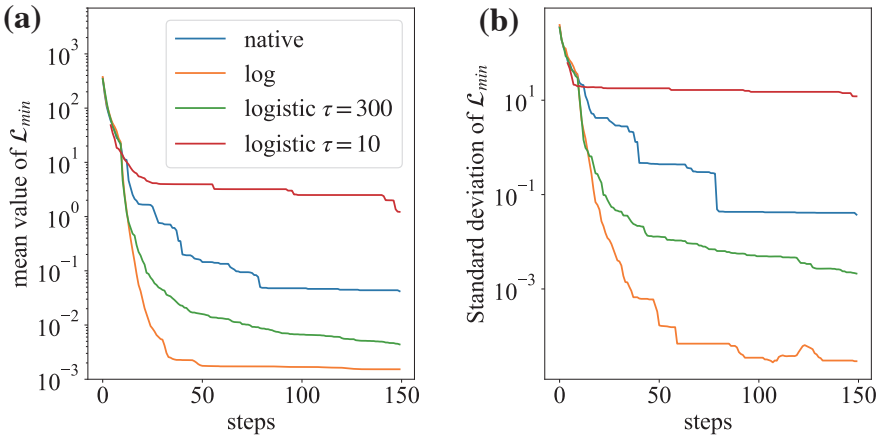

FIG. S6. (a) and (b) show respectively the mean value and standard deviation of the minimal loss $\mathcal{L}$ over 100 independent randomly initialized runs. The calculations are performed on XXZ chain with $J_{xy} = 1, J_z = 1.5$. For the logistic-like loss, we have tried two representative τ parameters, $\tau = 300$ and 10.

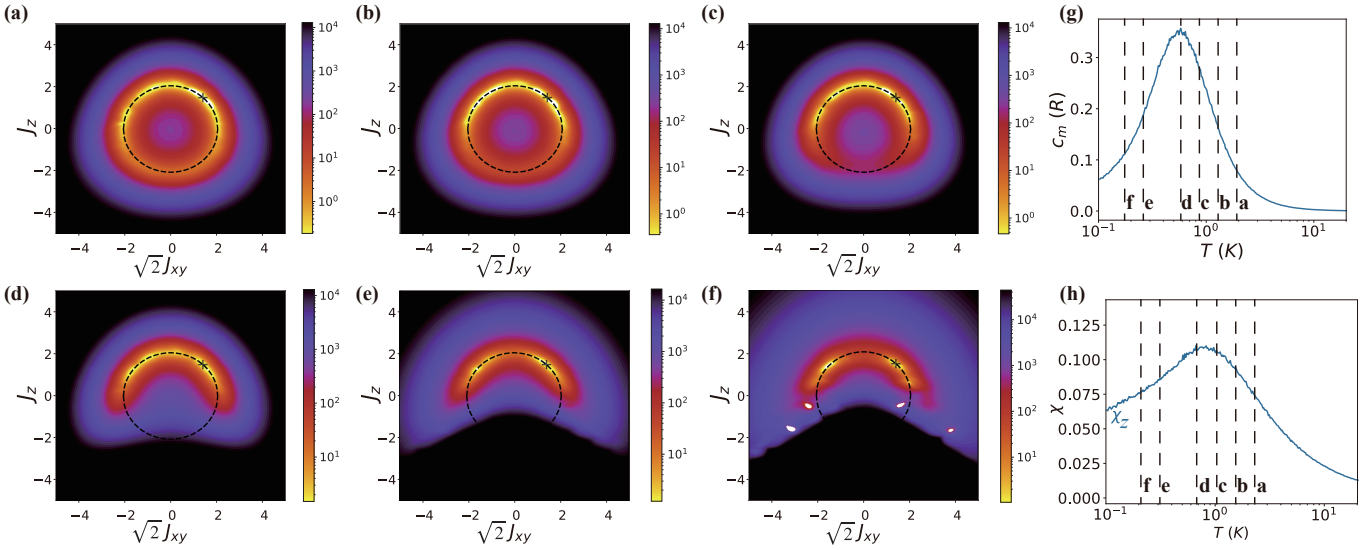

FIG. S7. (a-f) Landscape $\mathcal{L}$ interpolated by a grid search of a 30×30 grid with varying cut-off temperature T_{cut}, as shown in (g, h). Inconsistent points in (f) are due to interpolation errors. (g, h) indicate the fitted thermodynamic quantities and various T_{cut} values, above which the thermodynamics data are used for fitting. The dashed circle represent a ring with $2J_{xy}^2 + J_z^2 = const$.

interactions, i.e.,

$$C_m \simeq \frac{1}{\mathbb{Z}}\frac{\mathrm{Tr}(H^2)}{T^2} \simeq \sum_{\langle i,j\rangle}\sum_{\alpha=x,y,z}\frac{J_\alpha^2}{T^2}\frac{\mathrm{Tr}[(S_i^\alpha S_j^\alpha)^2]}{\mathrm{Tr}(\mathbb{I}_i\,\mathbb{I}_j)} = \frac{N}{16}\sum_\alpha \frac{J_\alpha^2}{T^2},$$

with $\mathbb{Z}$ the partition function and N the total site number. Due to the above relation, our automatic search can learn the magnitude of spin couplings. As T_{cut} further moves to lower temperatures, the oval ring gradually breaks and eventually converges to two [Fig. S7(d-f)] or one [Fig. S8(d-f)] bright points, depending on whether χ_{xy} data are included or not. From these panels, we also see that the fittings, although using only small-size ED results, when the T_{cut} is low enough for the systematic error to appear — be noted that there are still a lot more data points at much higher temperatures which are also incorporated into the loss function — the optimal parameters and the estimated landscape will not suddenly "crash". This means, although the choice of T_{cut} is "crucial" in principle as we do not want any systematic error in the theoretical calculation, in practice there is always a substantial interval to place the cutoff temperature T_{cut} for which the optimal parameters as well as the landscape are quite robust.

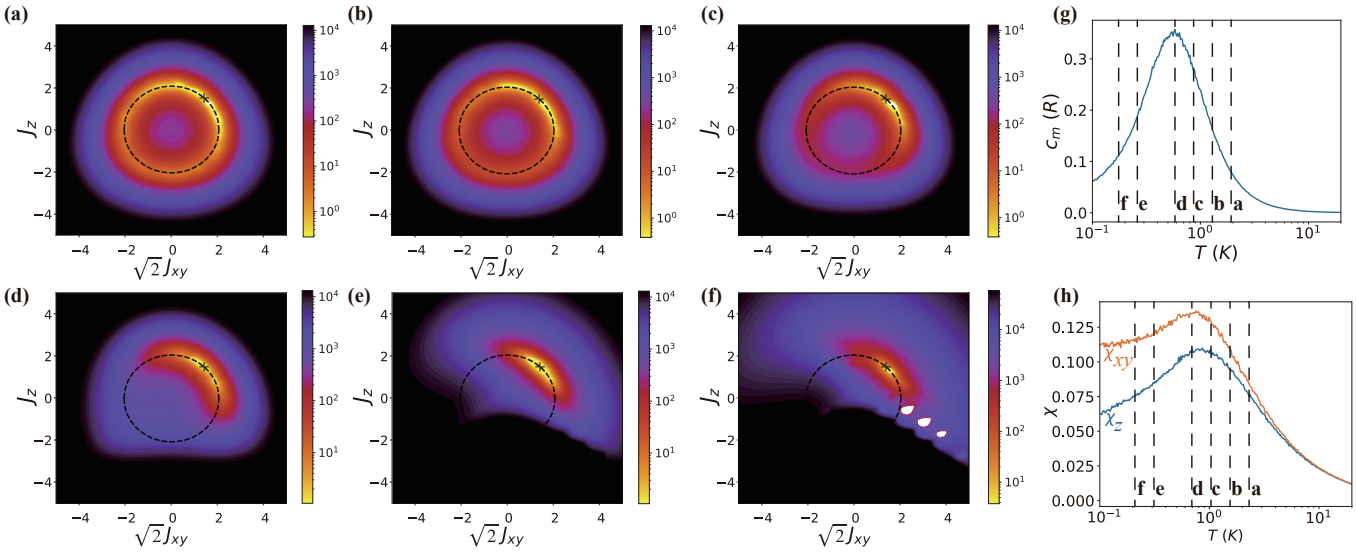

FIG. S8. Same layout as Fig. S7, with χ_{xy} included in the automatic parameter searching.

G. More Results on the Triangular-lattice Quantum Ising Magnet $TmMgGaO_4$

As a triangular-lattice quantum Ising magnet with emergent U(1) symmetry and Kosterlitz-Thouless (KT) phase transition, the rare-earth compound $TmMgGaO_4$ has raised great research recently (Refs. [61–66]). The accurate determination of its spin Hamiltonian has played an indispensable role in uncovering the KT transition and topological physics therein (Refs. [61]).

Now we employ the automatic parameter searching approach, with ED and XTRG as the high- and low-T solvers, respectively, to determine the spin Hamiltonian of $TmMgGaO_4$. In the XTRG calculations, $T_{\rm cut} = 1$ K is significantly lower than $T_{\rm cut} \simeq 4$ K for the ED solver (cf. Fig. S9), rendering greatly improved resolution in the loss landscapes in Fig. 5 of the main text. With the improved optimal parameter set obtained with XTRG solver, we see in Fig. S9 that the thermodynamics quantities including specific heat $C_m(T)$ and susceptibility $\chi_{\parallel}(T)$ indeed fit the experimental data much better, and the agreement is good till rather low temperature as compared to those from the ED solver.

The 9-site PBC ED and a 54-site cylinder XTRG calculations are employed in the thermodynamics fittings [c.f. Fig. S9](c). The T_{cut} values are set to be about 4 K (the peak of specific heat C_m) for the high-T solver ED and 1 K for the large-scale solver XTRG, respectively. Both C_m and $\chi_{\parallel}$ are used for fitting, and the results with the fitted optimal parameter are shown in Fig. S9(a,b).

H. Robustness of the Automatic Parameter Searching

Here we test the robustness of our automatic Hamiltonian searching against increasing noises and reducing the number of data points in thermodynamics measurements.

As experimental data inevitably have intrinsic noises, it is important that our automatic parameter searching is robust against those noises. Matter of fact, in the generated data in Fig. 3 of the main text, Gaussian noise has been introduced to mimic the "experimental" data, where our approach works well. On the other hand, in the measured thermal data of Copper Nitrate $[Cu(NO_3)_2 \cdot 2.5H_2O]$ in Fig. 4 and quantum Ising magnet $TmMgGaO_4$ in Fig. 5 of the main text, we deal with real experimental data with intrinsic noises. The excellent fitting results there also show that our method are indeed robust against noises.

Nevertheless, we still feel very necessary to perform a systematic test on thermal data with gradually increased noises. As shown in Fig. S10 and Fig. S11 below, there are more ($\sim$ 80) data points in Fig. S10 while much fewer ($\sim$ 20) points in Fig. S11. Weak and strong random noises are introduced to the data, following the Gaussian form $\mathcal{N}(\mu = 0, \sigma = 0.08 \times E_i)$ and $\mathcal{N}(\mu = 0, \sigma = 0.16 \times E_i)$, with E_i the value of data point i. In Figs. S10, we find the Gaussian noises have little effects on the overall shape of the loss landscapes (both the scanned and the Bayesian estimated ones). Moreover, as shown in Fig. S11, when reducing the data points and at the same time enhancing noises we still find very precise Bayesian estimate as compared to the true landscapes.

In condensed-matter experiments, typically there are abundant thermal data available from measurements. Usually one have various kinds of thermal data including the specific heat (under various fields), transverse and longitudinal magnetic susceptibilities, and the magnetization curves, etc. The measured curves usually contain many (a few tens to hundreds) data points at

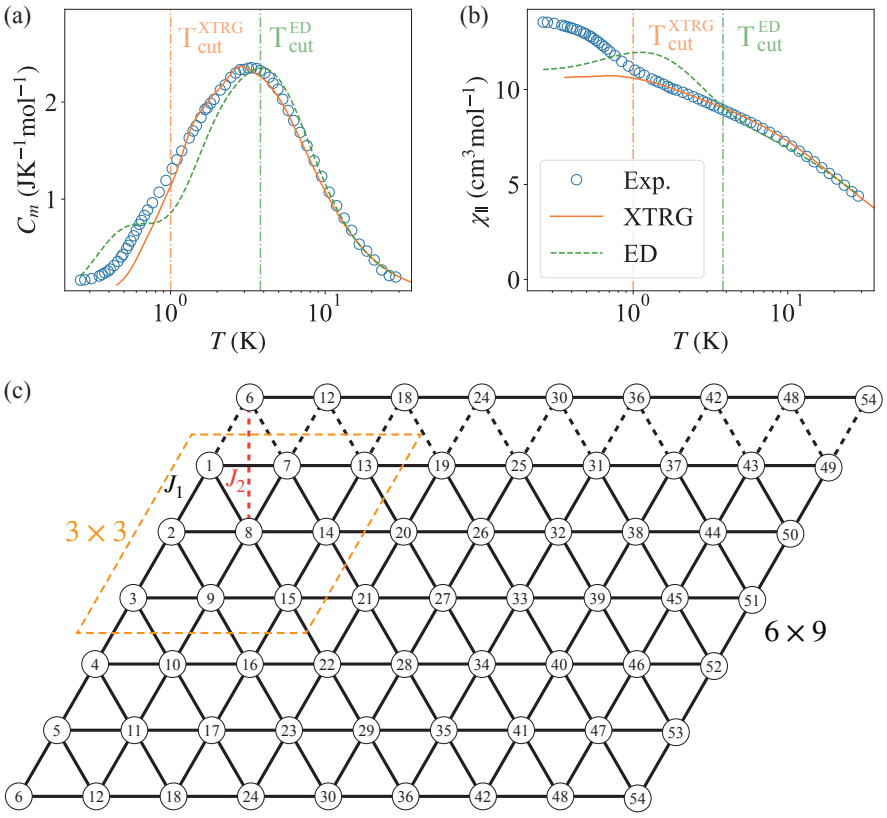

FIG. S9. The experimental data of $TmMgGaO_4$ [62], including (a) the magnetic specific heat C_m and (b) susceptibility $\chi_{\parallel}$ (blue shallow circles). The simulated results are computed by ED (green dashed line, with T_{cut} = 4 K) and XTRG (orange solid line, with T_{cut} = 1 K) with the fitted optimal model parameters (hollow asterisk for ED and the solid one for XTRG) marked in Fig. 5 of the main text. (c) depicts the geometries employed in the many-body calculations, i.e., a 3×3 PBC cluster for ED and the 6×9 cylinders for XTRG. The triangular lattice model has both NN (J_1) and NNN (J_2) interactions, depicted by the black solid and red dashed lines.

different temperatures. Relatively, the number of lattice model parameters is much smaller and such fittings are always overdetermined.

Despite of the above facts, we test in Fig. S12 that how our approach performs when the experimental data points are gradually reduced, e.g., from 69 in Fig. S12 (a), to 35 points in panel (b), and eventually to 18 points per curve in (c). In Fig. S12, we find the overall landscape does not change much, and the estimated optimal parameter through Bayesian optimization locates at almost the same place regardless of the numbers of data points involved. We ascribe this to the overdetermination nature of such thermodynamic fitting problems.

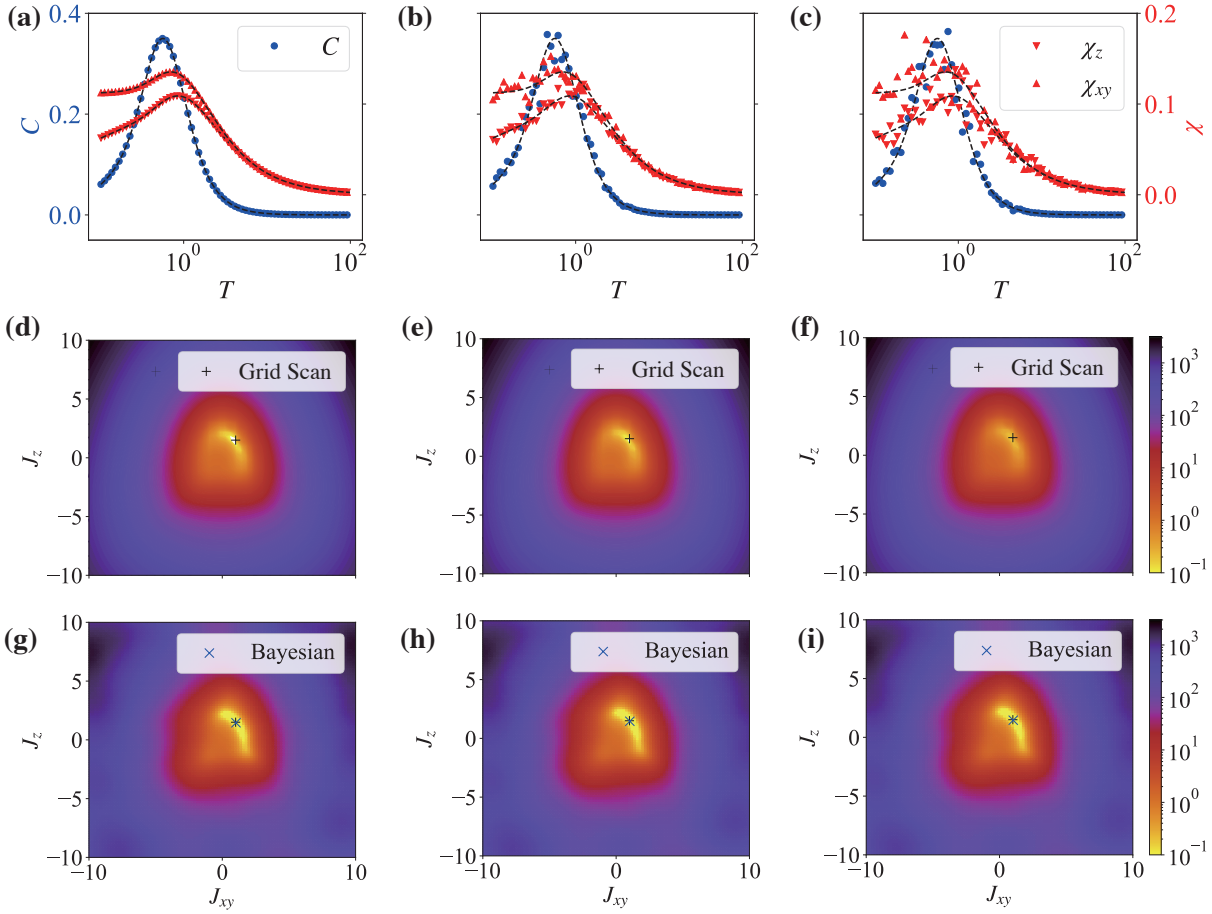

FIG. S10. (a-c) Thermal data including the specific heat C (blue circles ●) and magnetic susceptibility in both directions χ_{xy},χ_z (red upper ▲ and lower ▼ triangles) with gradually increasing noises. The landscapes scanned with grid search are shown in the upper panels (d-f) and the learnt landscapes are in the lower panels (g-i). The laters are obtained by the Bayesian optimization through 150 iterations. The plus (+) and cross (×) signs are the actual and Bayesian estimated optimal parameter points, respectively. In the fittings of three quantities, we set T_{cut} as the peak temperature of C, χ_{xy}, χ_z curves, respectively.

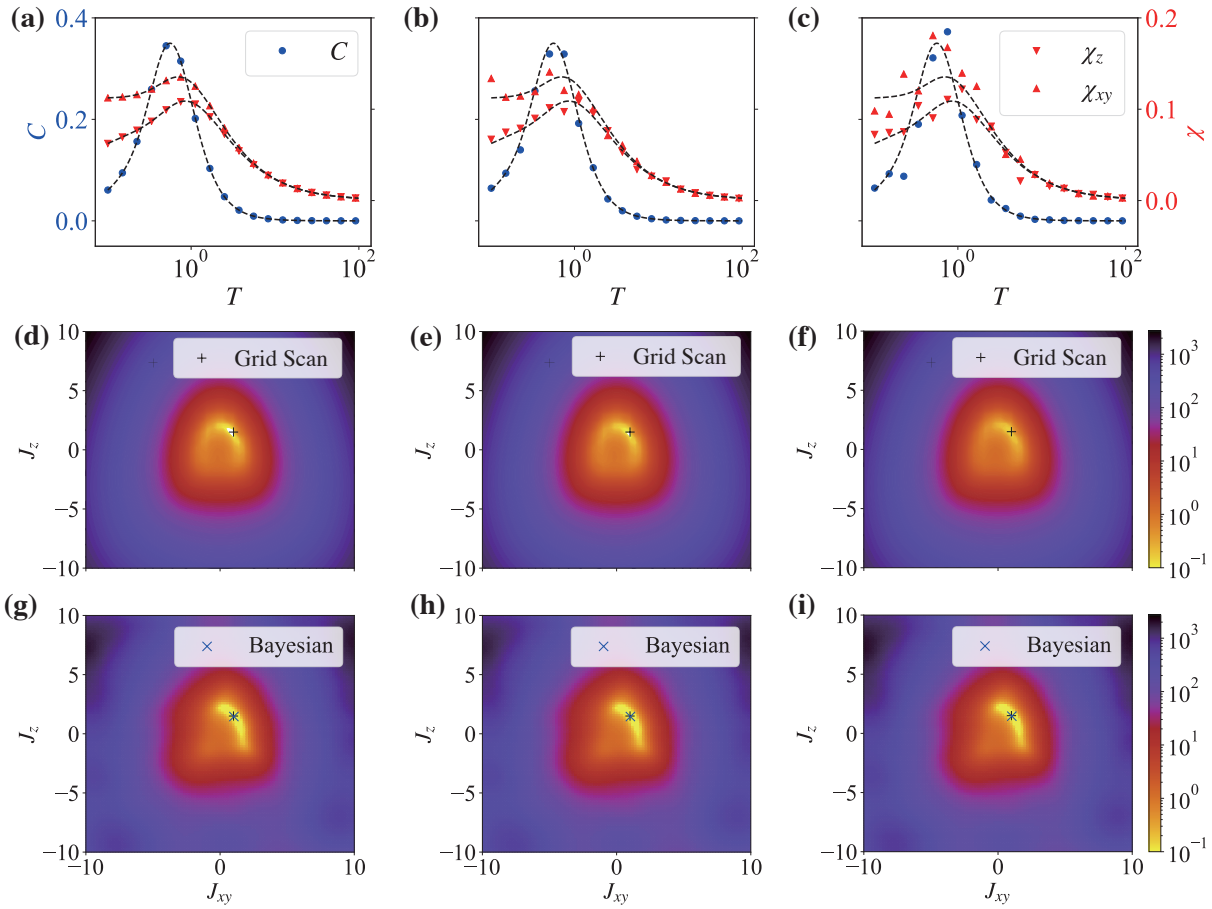

FIG. S11. (a-c) Same layout as in Fig. S10 while with sparse “experimental" data. The scanned and estimated landscapes are shown in (d-f) and (g-i), respetively.

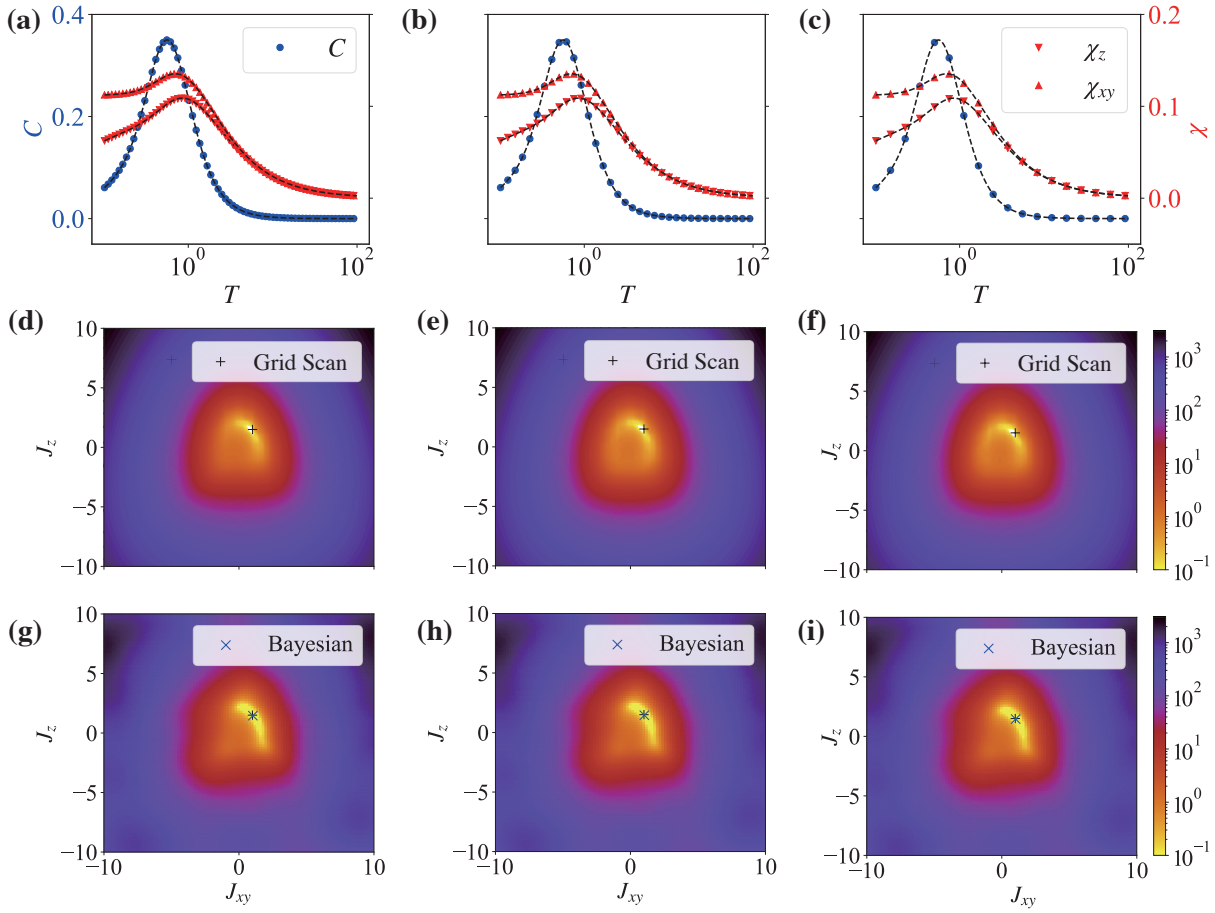

FIG. S12. (a-c) Thermodynamic data with gradually decreased number of data points. (d-f) The landscapes scanned with grid search are shown in the upper panels and the estimated landscapes (through 150 iterations of Bayesian optimization) in the lower panels (g-i).