I. INTRODUCTION

Over the past year, physicists have started to use deep learning techniques to study the field of condensed matter physics. Most of the tasks are done by supervised learning algorithms. In supervised learning, the algorithm is trained on data with labels, which are assigned to data points. After successful training, high precision predictions can be made for labels of previously unseen data. In addition to supervised learning, there are unsupervised learning algorithms that can find structure in unlabeled data. It has become possible to use unsupervised learning techniques to reproduce the Monte Carlo sampling state of the Ising model, and discovering phase transitions in an unsupervised manner has become mainstream. Restricted Boltzmann machines, variational self-coding, and generative adversarial networks have been widely used for the simulation of Ising models with good results. However, none of these models can take into account the molecule-molecule interactions and the influence of the edges between molecules in the physical model during training. Also, the microstructure of many physical models is irregular, and the generated models are poorly generalized and can generally only accommodate one size at one temperature. This requires a deep neural network that can consider unstructured data and can take into account node-to-node and node-to-edge relationships.

Fortunately, the research on graph neural networks in the field of deep learning has been growing in enthusiasm in recent years, and the excellent ability of graph neural networks to handle unstructured data makes them particularly useful for applications in physics, biology, and chemistry. Most physical models are essentially a graph structure. Compared with traditional convolutional networks, graph convolutional networks are better at encoding structural information of graphs and can learn better and deeper representation features. Graph variational self-coding is an unsupervised graph generation algorithm, which is the product of combining graph neural network and generative network, and it can effectively use the graph structure of data to simulate the distribution of training data.

In this paper, we integrate a graph convolutional network into a graph variational self-encoder framework to construct an Ising model simulator, which is used to simulate Ising model configurations at different temperatures. The trained simulator can effectively generate Ising model states with physical properties that are not different from those obtained from conventional Monte Carlo simulations. It can also be adapted to a wide range of different sizes and significantly reduces the simulation time. We will first introduce the general form of the Ising model, graph variational neural networks and graph convolutional networks. In the core of the paper, we construct a graph variational self-encoder combined with a graph convolutional network to simulate the 2D Ising model at different temperatures and verify the validity of our method by comparing the generated data with those from Monte Carlo simulations.

II. ISING MODEL AND GRAPH NEURAL NETWORKS

A. Ising model

In statistical physics, the Ising model is described as the set of binary spins with coupling interactions in some lattice. Considering that N spins s=s_i can take the value $\pm$ 1, where the index i marks the position of the spin s_i, the standard Hamiltonian of the Ising system includes only the nearest-neighbor interactions, and each spin direction may be "up" (+1) or "down" (-1), although the generalized model may include long-range interactions and more spin-selective directions. The standard Hamiltonian quantities are:

$$H = -J \sum_{neighbors} S_i * S_j \tag{1}$$

B. Graph neural network

A graph is a data structure that models a set of objects (nodes) and their relationships (edges). This year, research on analyzing graphs with machine learning methods has received increasing attention due to the powerful expressiveness of graph structures. Graph neural network is a general term for algorithms that use neural networks to learn graph structured data, extract and uncover features and patterns in graph structured data, and meet the needs of graph learning tasks such as clustering, classification, prediction, segmentation, and generation. One motivation for the development of graph neural networks originated from convolutional neural networks. The widespread use of convolutional neural networks has led to breakthroughs in machine learning and opened the era of deep learning. However, convolutional neural networks can only extract potential features in regular Euclidean space data, which cannot correspond well to the complex and variable graph data in reality, and how to apply convolutional neural networks to the non-Euclidean space of graph structure has become the key problem of graph neural network models. The descriptions and definitions of the relevant symbols in graph neural networks are given below

Definition 1 *Graph:*The graph is formed by the nodes and the edges connecting the nodes, usually noted as $G = (V, E)$. where $V = \{v_1, v_2, ..., v_n\}$ represents the set of nodes,$E = \{e_1, e_2, ..., e_n\}$ represents the set of edges, and edges can also be represented as (v_1, v_2). Usually nodes are also called vertices or intersections. Edges are also called links or arcs. A generic graph representation is a quintet: $G(V, E, A, X, D)$. Where A represents the

adjacency matrix of the graph, X represents the identity matrix of the nodes, and D represents the degree matrix.

Definition 2 *Adjacency matrix*: The adjacency matrix of a graph refers to the matrix used to represent the connectivity of the nodes in the graph. This matrix can be binary or weighted. For an undirected graph with N nodes, the adjacency matrix is an N*N real symmetric matrix.

Definition 3 *Degree matrix*: The degree of a node represents the number of edges connected to that node. The degree matrix of a graph is the matrix that describes the degree of each node in the graph. The degree matrix is a diagonal matrix, and for undirected graphs, only the incoming degree matrix or the outgoing degree matrix is generally used.

Definition 4 *Combinatorial Laplacian matrix*, also known as standard Laplacian matrix, is a combination of diagonal matrix and adjacency matrix:

$$L = D - A \tag{2}$$

The matrix has non-zero elements only at the central node and at the nodes connected to the first order, and zero everywhere else. Laplacian matrix is also a form of graph.

Definition 5 *Normalized Laplacian matrix*

$$L^{sym} = I - D^{-1/2}AD^{-1/2} \tag{3}$$

Its element value is :

$$L^{sym}_{(i,j)} = \begin{cases} 1, & \text{i=j,} deg(v_i) \neq 0 \\ \frac{-1}{\sqrt{deg(v_i)deg(v_j)}}, & i \neq j \text{ ,} v_i \text{ and } v_j \text{ are adjacent} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $deg(v)$ is denoted as the degree of node v

1. Graph Convolutional Network

A graph convolution network is derived from the traditional convolutional neural network. Graph convolution can be of two kinds, spectrum-based approach and space-based approach. The spectrum-based approach defines graph convolution from the perspective of graph signal processing by introducing filters, so the spectrum-based graph convolution can be understood as removing noise from the graph signal. The space-based graph convolution method constructs the graph convolution by pooling information from neighboring nodes.

Before understanding graph convolution, we need to have a basic concept of convolution operation. Convolutional networks are first applied to images, where we use the pixel points on the image as features to perform the convolution operation to extract features. The random shared convolution sum is used to obtain a weighted sum of pixel points to extract a particular feature, and then back propagation is used to optimize the convolution kernel parameters to automatically extract features, which is the cornerstone of convolutional neural network to extract features. The convolution operation is equivalent to aggregating the information of surrounding pixels to the central pixel of the convolution kernel and extracting the features of that part.

So we want to use convolution in the graph domain to use a general paradigm for graph feature extraction, similar to image, when we want to get the feature representation of a node, we can aggregate the information of neighboring nodes, which can be understood in this way, each node in the graph is changing its state all the time until the final balance because of the influence of the more distant points of the neighboring node kernel, the closer the neighbor The closer the neighbor, the greater the influence.

2. Graph Convolutional Generalization

For a graph convolutional network, we want to learn its feature representation for each node, and any graph convolution can be written as a nonlinear function such that:

$$H^{l+1} = f(H^l, A) \tag{5}$$

$H^0 = X$ is the input to the first layer, l represents the number of layers of the neural network, and A represents the adjacency matrix.

The traditional graph convolutional network approach is to add information from neighboring nodes to this node:

$$f(H^l, A) = \sigma(AH^lW^l) \tag{6}$$

W^l represents the parameter matrix of the lth layer, and σ represents the activation function. According to the matrix multiplication it can be seen that each node in the above equation combines the information of the neighboring nodes, but since only the adjacency matrix is used, the diagonal of the adjacency matrix is 0, which cannot reflect the information of the nodes themselves, so we use the Laplacian matrix instead of the adjacency matrix

$$f(H^l, A) = \sigma(LH^lW^l) \tag{7}$$

The Laplace matrix is introduced in the above equation(7), thus solving the problem of not considering the information self-transfer of its own nodes, but since it is not normalized, we regularize the Laplace matrix to obtain the graph convolution

$$f(H^l, A) = \sigma(L^{sym}H^lW^l) \tag{8}$$

The above description is in matrix form, we can also look at the formula from the perspective of a single node, for the node feature h_i^{l+1} in the l+1 layer, for his neighboring nodes $j \in N$, N is the set of neighboring nodes used by node i, so the update formula of the graph neural network can also be depicted as

$$h_i^{l+1} = f(h_i^l) = \sigma(h_i^l \bullet W_1^{l+1} + \sum_j h_j^l \bullet W_2^{l+1}) \tag{9}$$

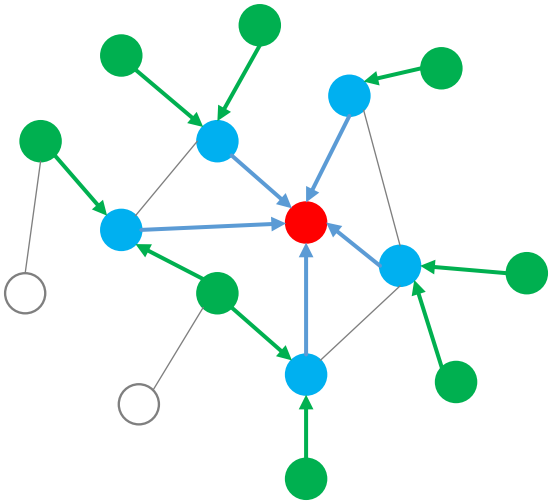


FIG. 1: This is figure 1

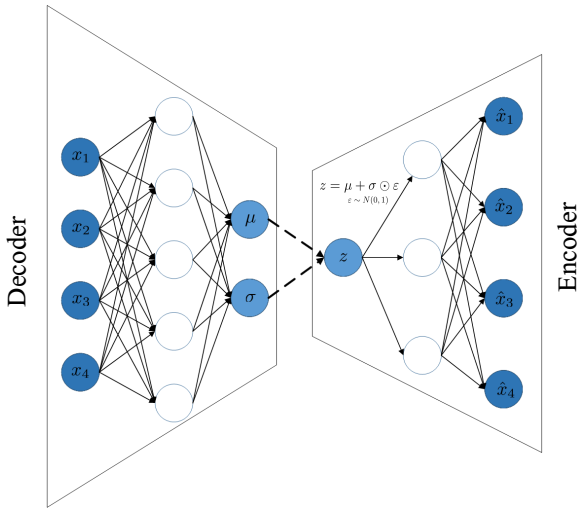


FIG. 2

3. *Variational auto-encoder*

A variational self-encoder is a generative model of variational inference, consisting of a network of encoders and decoders, which is a model with hidden variables. In a variational self-encoder, we base on the following assumptions: Our sample x is generated by some latent variable z through some mapping, and z is not a fixed value, but obeys a distribution: $z \sim P_\theta(z)$, then $x \sim p_\theta(x|z)$, where P is a family of distributions determined by the parameter θ, and θ is the true distribution that we need to find. So our fundamental goal is to determine the values of z and θ so that we can obtain the sample x. In order to infer $p_\theta(x|z)$, we need to maximize the marginal log likelihood $log(p_\theta(x))$ We can rewrite the equation:

$$\begin{aligned} log(p_\theta(x)) &= log \int p_\theta(x|z)dz = log \int p_\theta(x|z)p(z)dz \\ &= log \int p_\theta(x|z)p(z)\frac{q_\phi(z|x)}{q\phi(z|x)}dz \\ &\geq \mathbb{E}_{z\sim q_\phi(z|x)}[log(p_\theta(x|z)] - D_{KL}[q_\phi(z|x)||p(z)] \\ &:= \varepsilon(x,\theta,\phi) \end{aligned} \tag{10}$$

We apply Jensen's inequality in the fourth step. Integrating the potential variable z in the second step is usually tricky, so we introduce an approximate posterior distribution $q_\phi(z|x)$ with the parameter set ϕ and use the variational inference principle to obtain an easy-to-handle bound on the marginal log-likelihood, i.e., the evidence lower bound (ELBO). We use $p(z) = N(1,0)$ as the latent variable prior. ELBO is an easy-to-process lower bound on the log-likelihood, and thus maximizes the inference $p_\theta(x|z)$. $\mathbb{E}_{z\sim q_\phi(z|x)[log(p_\phi(x|z))]}$ can be interpreted as the reconstruction error, since maximizing it will make the output of the decoder similar to the input of the encoder, and $D_{KL}[q_\phi(z|x)||p(z)]$ is the KL scatter (see *** for details), a value that measures the similarity of two distributions and that ensures that the latent representation is Gaussian, such that data points with similar characteristics have similar Gaussian representations.

We have outlined the general idea behind latent variable models, but we still need to specify the approximate (variational) posterior $q_\phi(z|x)$ and the model likelihood $log(p_\theta(x|z))$. A common choice for approximating the posterior is to decompose the Gaussian distribution

$$q_\phi(z|x) = \prod_{j=1}^{d} N(z_j|\mu_j, \sigma_j^2) \tag{11}$$

where μ and σ denote the mean and standard deviation of the model.

In summary, VAE is an encoder network based on extracting the mean and variance of a potential Gaussian representation $q_\phi(z|x)$ of $p(z|x)$. The decoder in VAE, which uses a Gaussian $q_\phi(z|x)$ sample as input, generates new samples according to the distribution $p_\theta(x|z)$. We compute all parameters by maximizing ELBO using backpropagation, but the Gaussian distribution is random and non-integrable. Therefore a deterministic and differentiable mapping between the output of the encoder and the input of the decoder needs to be established. We need to represent the random variable z as a differentiable and invertible variation g of another auxiliary random variable ϵ (i.e., $z = g(\mu, \sigma, \epsilon)$). We employ a re-referencing trick such that $g(\mu, \sigma, \epsilon) = \mu + \sigma \odot \epsilon$ i.e.

$$z = \mu + \sigma \odot \epsilon \tag{12}$$

where $\epsilon \sim N(0,1)$ and $\odot$ is the product of elements.

III. SPIN VARIATIONAL GRAPH AUTO-ENCODER

We want to construct a graph variational self-encoder that applies the idea of VAE to graph structure data. We want our graph variational self-encoder to generate new Ising configurations of different sizes in steady state. However, we cannot use VAE directly because traditional convolution can only be applied to Ising configurations of the same size, and different Ising configurations are equivalent to different graph structures with different numbers of nodes resulting in the inability to use convolution directly. At this point, we can use the graph convolution mentioned above, which can effectively describe different graph structures, train different sized configurations in the same neural network, and get the generation effect we want, i.e., generate steady-state Ising configurations of different sizes.

A. Encoder

The first two GCN layers generate a high-dimensional feature matrix, while we consider the edge information in the graph structure, the edge features are involved in the convolutional aggregation, refer to the previous update formula of the graph convolution can get our new update formula.

$$\mathbf{x}_i' = \mathbf{W}_1\mathbf{x}_i + \mathbf{W}_2 \sum_{j\in\mathcal{N}(i)} e_{j,i}\cdot\mathbf{x}_j \tag{13}$$

$e_{j,i}$ denotes the edge weights from source node j to target node i. The result of the upper convolution in the convolution operation is used as the input to the lower convolution to perform the stacking of convolutions.The third convolutional layer we use two different convolutional networks to obtain $\mu, \log\sigma^2$ respectively

$$\mu = GCN_\mu(\hat{X}, A) \tag{14}$$

$$\log\sigma^2 = GCN_{\log\sigma^2}(\hat{X}, A) \tag{15}$$

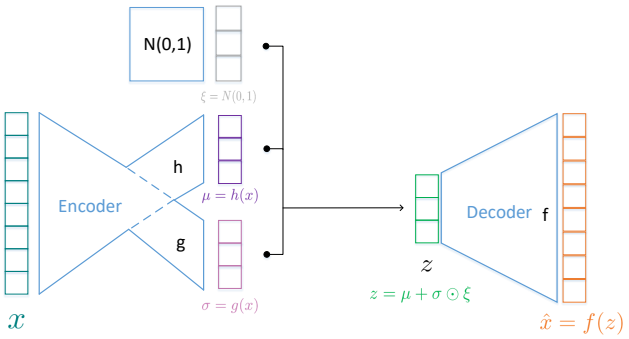


FIG. 3: A figure caption. The figure captions are automatically numbered.

Then we can sample Z by eqs. 12. from the distribution.

B. Decoder

Our sampling result z is a high-dimensional representation of the nodes. In the decoder part, since the connection between the nodes of the Ising model is fixed, we can ignore the edge properties of the Ising model in the decoder part and use a fully connected neural network for each node in the potential variable z to reduce its dimensionality to 1. At this point we can think of the reconstruction problem as classifying each node, Using the sigmod activation function, we then classify them by value. We obtain our generated reconstruction model

$$\overline{x}_i = MLP(z_i) \tag{16}$$

z_i represents the high-dimensional representation of the nodes, which are transformed into the reconstructed nodes we need by the fully-connected neural network. Since we perform the fully-connected neural network for each high-dimensional node and finally downscale to 1 dimension, our model can adapt to various sizes of configurations. We realize that one model can generate multiple configurations of various sizes, Meanwhile, because of the mixed training of multiple models, some noise is added to the training, which enhances the robustness of the model and the model generation is better.

After several iterations, we can consider the model training successful when the value of the loss function is less than the threshold, and then we can save the model and the trained μ and σ and use the decoder to generate the configuration we want. Our model can generate multiple sizes of configurations, just let the sizes you need to train together, and after the training is finished, we can use the configurations under the size to fix the model to the size we need, then we can generate the configurations under the size we want.

IV. SIMULATION RESULTS AND ANALYSIS

To verify the effectiveness of the network described above, we build an Ising simulator that produces Ising states in a 32 × 32 lattice near its critical temperature. The model we are simulating is descried by a Hamiltonian of the eqs.1

Insert an image using either the `graphics` or `graphix` packages, which define the `\includegraphics{#1}` command. (The two packages differ in respect of the optional arguments used to specify the orientation, scaling, and translation of the image.) To create an alignment, use the `tabular` environment.

The best place to locate the `figure` or `table` environment is immediately following its first reference in text; this sample document illustrates this practice for Fig. 3,

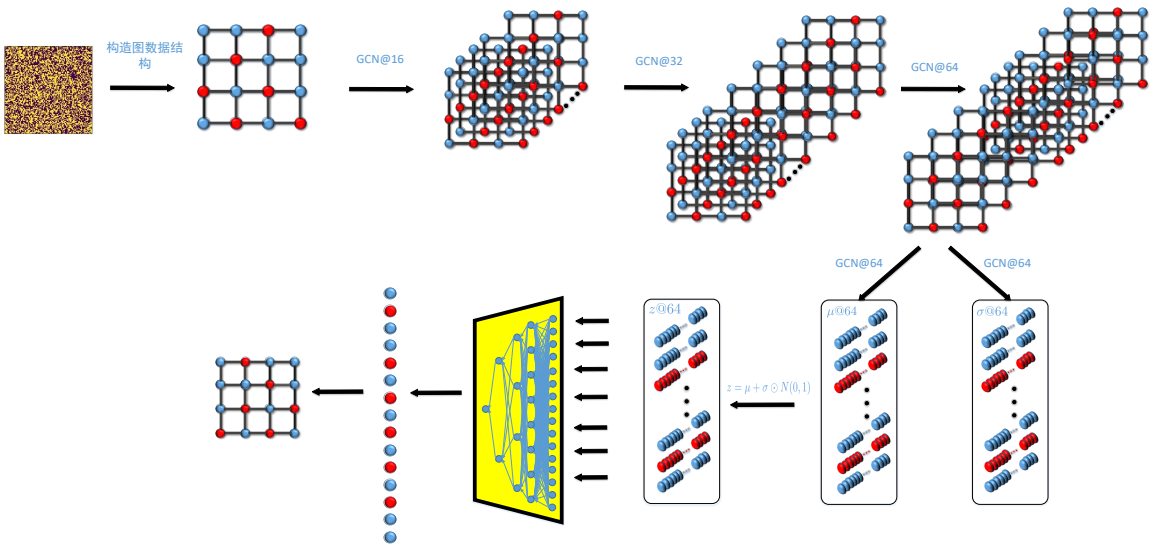


FIG. 4: Use the figure* environment to get a wide figure that spans the page in `twocolumn` formatting.

TABLE I: This is a wide table that spans the full page width in a two-column layout. It is formatted using the `table*` environment. It also demonstates the use of `\multicolumn` in rows with entries that span more than one column.

	D^1_{4h}		D^5_{4h}	
Ion	1st alternative	2nd alternative	lst alternative	2nd alternative
K	$(2e)+(2f)$	$(4i)$	$(2c)+(2d)$	$(4f)$
Mn	$(2g)$[a]	$(a)+(b)+(c)+(d)$	$(4e)$	$(2a)+(2b)$
Cl	$(a)+(b)+(c)+(d)$	$(2g)$[a]	$(4e)$[a]	
He	$(8r)$[a]	$(4j)$[a]	$(4g)$[a]	
Ag		$(4k)$[a]		$(4h)$[a]

[a] The z parameter of these positions is $z \sim \frac{1}{4}$.

which shows a figure that is small enough to fit in a single column.

In exceptional cases, you will need to move the float earlier in the document, as was done with Table I: LaTeX's float placement algorithms need to know about a full-page-width float earlier.

Fig. 4 has content that is too wide for a single column, so the `figure*` environment has been used.

TABLE II: Numbers in columns Three–Five are aligned with the "d" column specifier (requires the `dcolumn` package). Non-numeric entries (those entries without a ".") in a "d" column are aligned on the decimal point. Use the "D" specifier for more complex layouts.

One	Two	Three	Four	Five
one	two	three	four	five
He	2	2.77234	45672.	0.69
C[a]	C[b]	12537.64	37.66345	86.37

[a] Some tables require footnotes.
[b] Some tables need more than one footnote.

The content of a table is typically a `tabular` environment, giving rows of type in aligned columns. Column entries separated by `&`'s, and each row ends with `\\`. The required argument for the `tabular` environment specifies how data are aligned in the columns. For instance, entries may be centered, left-justified, right-justified, aligned on a decimal point. Extra column-spacing may be be specified as well, although REVTeX 4 sets this spacing so that the columns fill the width of the table. Horizontal rules are typeset using the `\hline` command. The doubled (or Scotch) rules that appear at the top and bottom of a table can be achieved enclosing the `tabular` environment within a `ruledtabular` environment. Rows whose columns span multiple columns can be typeset using the `\multicolumn{#1}{#2}{#3}` command (for example, see the first row of Table I).

Tables **??**, I, II, and III show various effects. A table that fits in a single column employs the `table` environment. Table I is a wide table, set with the `table*` environment. Long tables may need to break across pages. The most straightforward way to accomplish this is to specify the `[H]` float placement on the `table` or `table*`

environment. However, the LaTeX 2_ε package `longtable` allows headers and footers to be specified for each page of the table. A simple example of the use of `longtable` can be found in the file `summary.tex` that is included with the REVTeX 4 distribution.

There are two methods for setting footnotes within a table (these footnotes will be displayed directly below the table rather than at the bottom of the page or in the bibliography). The easiest and preferred method is just to use the `\footnote{#1}` command. This will automatically enumerate the footnotes with lowercase roman letters. However, it is sometimes necessary to have multiple entries in the table share the same footnote. In this case, there is no choice but to manually create the footnotes using `\footnotemark[#1]` and `\footnotetext[#1]{#2}`. `#1` is a numeric value. Each time the same value for `#1` is used, the same mark is produced in the table. The `\footnotetext[#1]{#2}` commands are placed after the `tabular` environment. Examine the LaTeX source and output for Tables **??** and III for examples.

Video **??** illustrates several features new with REVTeX4.2, starting with the `video` environment, which is in the same category with `figure` and `table`.

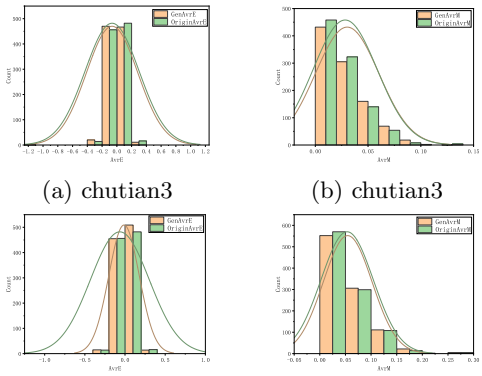
The `\setfloatlink` command causes the title of the video to be a hyperlink to the indicated URL; it may be used with any environment that takes the `\caption` command. The `\href` command has the same significance as it does in the context of the `hyperref` package: the second argument is a piece of text to be typeset in your document; the first is its hyperlink, a URL.

Physical Review style requires that the initial citation of figures or tables be in numerical order in text, so don't cite Fig. 4 until Fig. 3 has been cited.

TABLE III: A table with numerous columns that still fits into a single column. Here, several entries share the same footnote. Inspect the LaTeX input for this table to see exactly how it is done.

	r_c (Å)	r_0 (Å)	κr_0		r_c (Å)	r_0 (Å)	κr_0
Cu	0.800	14.10	2.550	Sn[a]	0.680	1.870	3.700
Ag	0.990	15.90	2.710	Pb[b]	0.450	1.930	3.760
Au	1.150	15.90	2.710	Ca[c]	0.750	2.170	3.560
Mg	0.490	17.60	3.200	Sr[d]	0.900	2.370	3.720
Zn	0.300	15.20	2.970	Li[b]	0.380	1.730	2.830
Cd	0.530	17.10	3.160	Na[e]	0.760	2.110	3.120
Hg	0.550	17.80	3.220	K[e]	1.120	2.620	3.480
Al	0.230	15.80	3.240	Rb[c]	1.330	2.800	3.590
Ga	0.310	16.70	3.330	Cs[d]	1.420	3.030	3.740
In	0.460	18.40	3.500	Ba[e]	0.960	2.460	3.780
Tl	0.480	18.90	3.550				

[a] Here's the first, from Ref. 10.
[b] Here's the second.
[c] Here's the third.
[d] Here's the fourth.
[e] And etc.



(a) chutian3 (b) chutian3

(c) chutian3 (d) chutian3

FIG. 5: mother of chutian

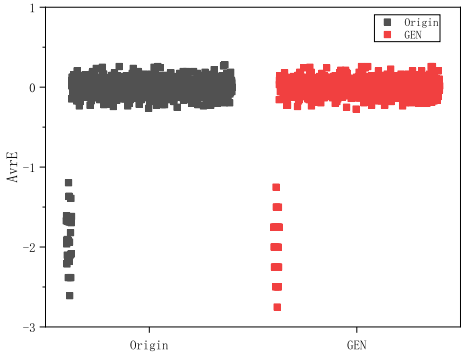


FIG. 6: Comparison of energy scatter diagrams generated by the hybrid model of 16 and 32 sizes for 32 sizes at T=3 temperature with those generated by the MCMC method

ACKNOWLEDGMENTS

We wish to acknowledge the support of the author community in using REVTeX, offering suggestions and encouragement, testing new versions, ...

Appendix A: Appendixes

To start the appendixes, use the `\appendix` command. This signals that all following section commands refer to appendixes instead of regular sections. Therefore, the `\appendix` command should be used only once—to setup the section commands to act as appendixes. Thereafter normal section commands are used. The heading for a section can be left empty. For example,

```
\appendix
\section{}
```

will produce an appendix heading that says "APPENDIX A" and

```
\appendix
\section{Background}
```

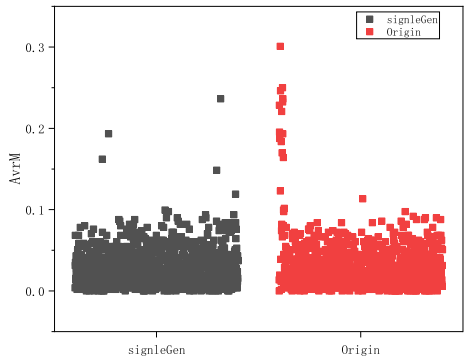


FIG. 7: Comparison of Magnetization scatter diagrams generated by a separate 32-size model at T=3 temperature with those generated by conventional methods

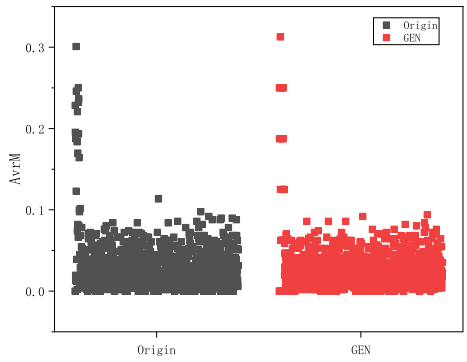


FIG. 8: Comparison of Magnetization scatter diagrams generated by the hybrid model of 16 and 32 sizes for 32 sizes at T=3 temperature with those generated by the MCMC method

will produce an appendix heading that says "APPENDIX A: BACKGROUND" (note that the colon is set automatically).

If there is only one appendix, then the letter "A" should not appear. This is suppressed by using the star version of the appendix command (`\appendix*` in the place of `\appendix`).

Appendix B: A little more on appendixes

Observe that this appendix was started by using

```
\section{A little more on appendixes}
```

Note the equation number in an appendix:

$$E = mc^2. \tag{B1}$$

1. A subsection in an appendix

You can use a subsection or subsubsection in an appendix. Note the numbering: we are now in Appendix B 1.

Note the equation numbers in this appendix, produced with the subequations environment:

$$E = mc, \tag{B2a}$$
$$E = mc^2, \tag{B2b}$$
$$E \gtrsim mc^3. \tag{B2c}$$

They turn out to be Eqs. (B2a), (B2b), and (B2c).

[1] A. G. Agarwal. Proceedings of the Fifth Low Temperature Conference, Madison, WI, 1999. *Semiconductors*, 66:1238, 2001.

[2] R. Ballagh and C.M. Savage. Bose-einstein condensation: from atomic physics to quantum fluids. In C.M. Savage and M. Das, editors, *Proceedings of the 13th Physics Summer School*. World Scientific, Singapore, 2000.

[3] R. Ballagh and C.M. Savage. *Bose-Einstein condensation: from atomic physics to quantum fluids, Proceedings of the 13th Physics Summer School*. World Scientific, Singapore, 2000.

[4] E. Beutler. volume 2, chapter 7, pages 654–662. McGraw-Hill, New York, 5 edition, 1994.

[5] E. Beutler. In E. Beutler, M. A. Lichtman, B. W. Coller, and T. S. Kipps, editors, *Williams Hematology*, volume 2, chapter 7, pages 654–662. McGraw-Hill, New York, 5 edition, 1994.

[6] N. D. Birell and P. C. W. Davies. *Quantum Fields in Curved Space*. Cambridge University Press, 1982.

[7] Y. Burstyn. Proceedings of the 5th International Molecular Beam Epitaxy Conference, Santa Fe, NM. (unpublished), 5–8 October 2004.

[8] E. B. Davies and L. Parns. Trapped modes in acoustic waveguides. *Q. J. Mech. Appl. Math.*, 51:477–492, 1988.

[9] A. Einstein, Yu Podolsky, and N. Rosen. *Phys. Rev.*, 47:777, 1935.

[10] R. P. Feynman. *Phys. Rev.*, 94:262, 1954.

[11] W. K. Fields. ECE Report No. AL944, 2005. Required institution missing.

[12] Jr. G. P. Berman and Jr. F. M. Izrailev. Stability of nonlinear modes. *Physica D*, 88:445, 1983.

[13] M. P. Johnson, K. L. Miller, and K. Smith. personal communication, 1 May 2007.

[14] S. R. Kawa and S.-J. Lin. *J. Geophys. Res.*, 108(D6):4201, 2003. DOI:10.1029/2002JD002268.

[15] Donald E. Knuth. volume 1 of *The Art of Computer Programming*, section 1.2, pages 10–119. Addison-Wesley, Reading, Massachusetts, second edition, 10 January 1973. A full INBOOK entry.

[16] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, second edition, 10 January 1981. A full BOOK entry.

[17] Jill C. Knvth. The programming of computer art. Vernier

Art Center, Stanford, California, February 1988. A full BOOKLET entry.

[18] Daniel D. Lincoll. Semigroups of recurrences. In David J. Lipcoll, D. H. Lawrie, and A. H. Sameh, editors, *High Speed Computer and Algorithm Organization*, number 23 in Fast Computers, part 3, pages 179–183. Academic Press, New York, third edition, September 1977. A full INCOLLECTION entry.

[19] Larry Manmaker. *The Definitive Computer Manual*. Chips-R-Us, Silicon Valley, silver edition, April-May 1986. A full MANUAL entry.

[20] Édouard Masterly. Mastering thesis writing. Master's project, Stanford University, English Department, June-August 1988. A full MASTERSTHESIS entry.

[21] J. Nelson. U.S. Patent No. 5,693,000, 12 December 2005.

[22] J. Nelson. TWI Report 666/1999, January 1999. Required institution missing.

[23] J. K. Nelson. M.S. thesis, New York University, 1999.

[24] Automatically placing footnotes into the bibliography requires using BibTeX to compile the bibliography.

[25] Alfred V. Oaho, Jeffrey D. Ullman, and Mihalis Yannakakis. On notions of information transfer in VLSI circuits. In Wizard V. Oz and Mihalis Yannakakis, editors, *Proc. Fifteenth Annual ACM*, number 17 in All ACM Conferences, pages 133–139, New York, March 1983. ACM, Academic Press. A full INPROCEDINGS entry.

[26] W. Opechowski and R. Guccione. *Introduction to the Theory of Normal Metals*, volume IIa, page 105. Academic Press, New York, 1965.

[27] W. Opechowski and R. Guccione. Introduction to the theory of normal metals. In G. T. Rado and H. Suhl, editors, *Magnetism*, volume IIa, page 105. Academic Press, New York, 1965.

[28] W. Opechowski and R. Guccione. Introduction to the theory of normal metals. In G. T. Rado and H. Suhl, editors, *Magnetism*, volume IIa, page 105, New York, 1965. Academic Press.

[29] Wizard V. Oz and Mihalis Yannakakis, editors. *Proc. Fifteenth Annual*, number 17 in All ACM Conferences, Boston, March 1983. ACM, Academic Press. A full PROCEEDINGS entry.

[30] F. Phidias Phony-Baloney. *Fighting Fire with Fire: Festooning French Phrases*. PhD dissertation, Fanstord University, Department of French, June-August 1988. A full PHDTHESIS entry.

[31] B. Quinn, editor. *Proceedings of the 2003 Particle Accelerator Conference, Portland, OR, 12-16 May 2005*, New York, 2001. Wiley. Albeit the conference was held in 2005, it was the 2003 conference, and the proceedings were published in 2001; go figure.

[32] J. Smith. *Proc. SPIE*, 124:367, 2007. Required title is missing.

[33] J. Smith, editor. *AIP Conf. Proc.*, volume 841, 2007.

[34] J. M. Smith. In C. Brown, editor, *Molecular Dynamics*. Academic, New York, 1980.

[35] J. M. Smith. *Molecular Dynamics*. Academic, New York, 1980.

[36] J. S. Smith and G. W. Johnson. *Philos. Trans. R. Soc. London, Ser. B*, 777:1395, 2005.

[37] R. Smith. Hummingbirds are our friends. *J. Appl. Phys. (these proceedings)*, 2001. Abstract No. DA-01.

[38] S. M. Smith. Ph.D. thesis, Massachusetts Institute of Technology, 2003.

[39] V. K. Smith, K. Johnson, and M. O. Klein. Surface chemistry and preferential crystal orientation on a silicon surface. J. Appl. Phys. (submitted), 2010.

[40] W. J. Smith, T. J. Johnson, and B. G. Miller. Surface chemistry and preferential crystal orientation on a silicon surface. J. Appl. Phys. (unpublished), 2010.

[41] Tom Térrific. An $O(n \log n / \log \log n)$ sorting algorithm. Wishful Research Result 7, Fanstord University, Computer Science Department, Fanstord, California, October 1988. A full TECHREPORT entry.

[42] Ulrich Ünderwood, Ned Ñet, and Paul P̄ot. Lower bounds for wishful research results. Talk at Fanstord University (A full UNPUBLISHED entry), November, December 1988.

[43] Edward Witten, 2001.

[44] V. E. Zakharov and A. B. Shabat. Exact theory of two-dimensional self-focusing and one-dimensional self-modulation of waves in nonlinear media. *Zh. Eksp. Teor. Fiz.*, 61:118–134, 1971.

[45] Y. M. Zalkins. e-print arXiv:cond-mat/040426, 2008.