**Learning hard quantum distributions with generative neural networks**

**Abstract**

Quantum computing has the potential to revolutionize numerous aspects of science and engineering in the future. However, building robust quantum computers involves enormous technological challenges. Researchers have developed a number of classical simulation algorithms for problems in many body physics. The most notable are machine learning algorithms. In recent years, there have been attempts on tasks to exhibit quantum advantage over classical computers. There are also attempts to close the gaps, various research has been devoted to simulating states of many-body quantum systems using classical computers, especially by employing machine learning algorithms. In this paper, we contribute to the development of these toolkits by studying the power of VAE and normalizing flows can represent the distributions generated by quantum computing. we follow and expand this approach by employing deep generative neural networks, including variational autoencoders and normalizing flows, to study different types of hard quantum distributions. Our results show that these modern generative networks are only capable of capturing partial features of the hard quantum distributions, which might be due to the inherent limitation for classical algorithms when encountering the complex quantum states with exponential growing Hilbert space. The results also show that variational autoencoders perform slight better than normalizing flows for many cases, and complimentary strength between the expressibility of variational autoencoders and normalizing flows are observed and worth further exploration.

# **Introduction**

Add ~2 paragraphs of intro and context and move last 3 para of this section here.

We have seen rapid developments of quantum computing since Richard Feynman proposed the idea in the early 1980s [1]. As a groundbreaking interdisciplinary research field spanning mathematics, physics, computer science and engineering, quantum computing is poised to shape our technological future.

Potentializing to transform almost everything in our life before long, quantum computing is on the brink to revolutionize numerous domains, including cryptography and cybersecurity, pharmaceutical research, material sciences, artificial intelligence, and many more. Tech giants, including IBM, Google, Alibaba, and many others, are racing to break through and dominate this space [add 4-6 references]

However, building quantum computers involves immense technological challenges: quantum states are intrinsically more fragile and can lose their coherence easily. Over the past couple of decades, experimentalists have constructed small scale quantum computers of various types, including superconductors or trapped ions. Even though tremendous efforts have been invested to develop the much-desired quantum computing technology, the ultimate “holy grail” of scalable quantum computer available for regular users is still far from practically feasible. Therefore, a natural question arises, how well can we represent and simulate quantum states using classical computers?

The fundamental challenge to describe quantum many-body problems arises from the feature that the Hilbert space dimension grows exponentially with the size of quantum system, making it prohibitively hard even for the most powerful classical computer to fully represents the states of the system. However, recently researchers have explored using modern cutting-edge machine learning techniques, particularly artificial neural networks (ANN) to encode quantum systems. For instance, Carleo and Troyer introduced a variational presentation of quantum states based on ANN with variable number of hidden neurons [2]. Gao and Duan demonstrated the advantage of deep over shallow neural network on representing most physical states including the ground state of many-body Hamiltonians [3]. Rocchetto, Grant, Strelchuk, Carleo and Severini have employed variational autoencoders (VAE), a type of generative AI model to efficiently learn states that are easy to simulate classically and can also compress and reduce the representation dimensions for states that are very hard for classical computers [4].

In this paper, we will employ generative AI models, specifically, VAE and normalizing flows (NF) to learn certain many-body distributions which are generated from quantum devices that aim to demonstrate quantum advantage. Two types of hard quantum distributions will be studied: (add these 2 distributions with reference); furthermore, we analyzed the performance of VAE and NF learning.

1. Technical background of quantum computing

The elementary building block of quantum computing is the quantum bit (qubit), the quantum version of a classical bit.

Unlike classical bits that can only be in a binary physical state of 0 or 1, qubits can be in a superposition state of |0⟩ and |1⟩ [5], denoted by Eq. (1).

(1)

For a qubit in superposition, its state is not fully known until it is measured; upon measurement, the wave function of the qubit collapses into one of two states based on probability. Entanglement of multi qubits in superstition states makes their computational power exponentially greater than their classical counterparts.

The states of a qubit can be illustrated on a Bloch sphere, named after physicist Felix Bloch [6], as shown in Figure 1.

A white circle in the sky

Description automatically generated

Figure 1. Bloch sphere depicting the states and positions of a qubit.

The evolution of qubits can be mathematically described as unitary operation in Hilbert space, which can be illustrated as the rotation around the x, y, and z axes by means of quantum gates or matrix operations. A sample operation of rotating a single qubit around x axis is shown as the unitary matrix in Eq. (2).

(2)

The benefit of exponentially growing power makes quantum computing one of the most active science research fields currently. A famous example demonstrating the advantage of quantum over classical computer is the task of factoring a large integer with *n* digits. The time complexity of any best classical algorithm is *O(exp(n))*; whereas Shor has demonstrated that this problem can be solved by quantum computer in polynomial time [7]. The term, quantum supremacy, refers to this type of advantage and dominance of quantum over classical computing.

# **Hard Quantum Distributions**

We will study two types of quantum distributions in this paper.

* 1. Distribution generated by quantum Fourier sampling by Fefferman and Umans

The first is a class of hard states introduced by Fefferman and Umans [8], produced by a certain quantum computational process. As pointed out in section 4 of ref. [8], a straightforward implementation of sampling from this distribution is to apply quantum Fourier transformation to a uniform superposition encoded by a function *h* related to the matrix permanent, and then perform the standard basis measurement. On the other side, there also exist a classical implementation of the sampling as described in ref. [4]. It is shown that such classical sampling algorithms are hard because it contains the matrix permanent problem which is not possible to compute in polynomial time.

The distribution that we will study originated from ref [8] and [4], For the purpose of exposition, we will restate the construction below.

Start with *m* monomials and an encoding function , an efficiently computable polynomial is defined as below:

, (3)

Where means taking the *i-th* bit. In this research, *h* is chosen to map the *i-th* permutation out of *n!* choices, which is related to the matrix permanent as pointed out in ref. [4]. Specifically, in this paper, .

Given a fixed integer *L* and a set vector . Construct another vector such that equals to the complex *L-ary* root of 1 raised to power of . Finally, the hard quantum distribution is defined as below.

, (4)

Plots depicting the empirical probability distribution generated by quantum devices with 3 and 8 qubits are illustrated in Figure 2 and Figure 3 respectively. (Explain why there are 8 or 256 possible outcomes.)

A graph with numbers and lines

Description automatically generated(add x & y)

Figure 2. Distribution generated by the quantum Fourier sampling on 3-qubits quantum computer proposed by Fefferman and Umans [add reference]

A barcode with numbers and text

Description automatically generated(add x & y)

Figure 3.

Distribution generated by the quantum Fourier sampling on 8-qubits quantum computer proposed by Fefferman and Umans [add reference]

* 1. Hard distribution generated by qsim quantum circuits

Another hard quantum distribution is sampled from a grid of quantum circuits assembled with certain randomness, as described ref. [9]. The quantum circuit is illustrated in Figure 4, which is taken from ref. [10].

A diagram of a diagram

Description automatically generated

Figure 4. Quantum circuit assembly taken from ref. [10]

Sampling of this distribution is implemented using pseudo-quantum simulator *qsim* [11] via *PennyLane-Cirq* [12] plugin in Python. (explain what is qsim, what is the circuit)

Quantum gates involved in this circuits include *RX(π/2)* and *RY(π/2),* and thegate, where *.* Other 2-qubits gates used in this circuits are *SISWAP* and *CPhase:*

(5)

(6)

Sample plots of this quantum distribution with 8 and 12 qubits is illustrated in Figure 5 and Figure 6 respectively.

A graph with blue lines

Description automatically generated(add x & y)

Figure 5. A Quantum distribution generated by measuring 8 qubits from execution on qsim circuits [ref]

A graph with blue lines

Description automatically generated(add x & y)

Figure 6. A Quantum distribution generated by measuring 12 qubits from execution on qsim circuits [ref]

# **Generative neural networks**

In this research we explore two types of generative artificial intelligence models, variational autoencoder and normalizing flows, to learn hard quantum distributions.

* 1. Variational autoencoder

Variational autoencoder (VAE) is a generative artificial intelligence method initially proposed by Kingma and Welling [13]. VAE is based upon autoencoder, which is a type of unsupervised model which performs efficient dimension reduction. The autoencoder architecture is illustrated in Figure 7. The encoder network compresses input data to a much smaller latent space which captures the essential data structure; then a decoder network reconstructs the output through a reverse process.

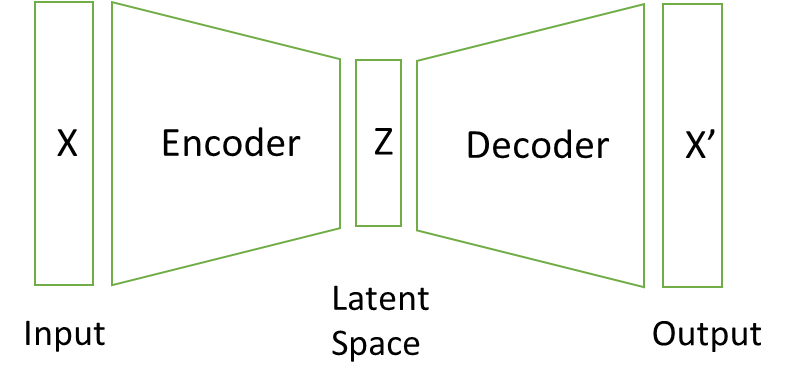


Figure 7. Illustration of an autoencoder

VAE takes autoencoder to the next level; it is capable of generating new contents by introducing a random sampler process in the latent space; such sampling is usually set to Gaussian. Therefore, VAE provides a statistical description of each latent space value. Such probabilistic nature allows VAE to generate many creative and diverse outputs. The architecture of VAE is illustrated in Figure 8.

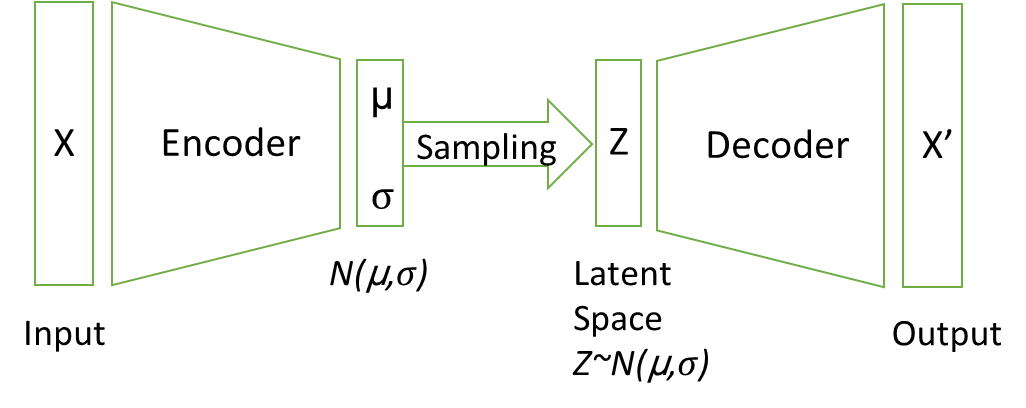


Figure 8. Illustration of a VAE

In order for the decoder to generate meaningful outputs, it is critical for latent space to be regularized. In practice, such regularization is achieved by adding a Kullback-Leibler divergence term to the loss function. So, the total VAE loss function is expressed as Eq. (7).

(7)

Ever since its inception a decade ago, VAE has found enormous success in a variety of applications across data and image creation, anomaly detection by understanding the underlying distribution, etc.

* 1. Normalizing flows

A normalizing flow (NF) [14] is a flow-based probabilistic generative model. NF explicitly models a probabilistic distribution by a series of change-of-variable operations. The essence of NF is to gradually transform a simple density through a series of reversible operations to generate a target complex distribution. The initial idea went back to Whitening transformations by Johnson [15] in 1966. The modern term “*Normalizing Flow”* was introduced by Tabak and Turner [16], where they formalized NF as a sequence of invertible simple maps.

Similar to the decoder of a VAE, the series of transformation functions in NF reconstruct the complex target distribution *P(x)* from a simple distribution *P(z)*, which is often set to Gaussian. A key feature of these functions is to maintain unit total probability, i.e., . Therefore, it is necessary for all these functions to be bijective, i.e. both surjective (onto) and injective (1 to 1). A natural consequence of using only bijectors is that the latent space of NF must be the same size as the inputs and outputs.

Mathematically, let the bijector be , and the inverse to be . Probability of x can be expressed as Eq. (8):

(8)

Where the last term is the determinant of the Jacobian of . The Jacobian is a matrix of derivatives of with respect to , as expressed in Eq. (9):

(9)

It can be seen that invertibility and differentiability are the must-have features for all NF bijectors, in order to ensure the applicability of the change-of-variable theorem and computation of the Jacobian.

What makes NF really powerful is to chain together multiple bijectors, similar to multiple layers of a deep neural network. Suppose , , …, ; and the inverse function be , …, , . Then Eq. (8) becomes Eq. (10) as below.

(10)

Trainable parameters of a NF are these bijectors. The loss function of NF is defined as the negative log-likelihood as Eq. (11) in a very straightforward way:

(11)

NF has become an extremely powerful generative tool with wide applications across density estimation, data, image, audio and video generation, anomaly detection, etc.

Comparing NF to other generative models like VAE and generative adversarial network (GAN), NF provides the advantage of directly modeling the probability density and likelihood which is very hard for VAE and GAN. On the other side, the latent space of NF must keep the same dimension, therefore NF does not allow for data compression which can be done by VAE easily.

# **Results**

We have generated these hard quantum distributions using F&U and qsim circuits. We will now study the difficulty of learning these distributions using very sophisticated machine learning tools VAE and NF described in previous section.

* 1. VAE study of hard quantum distributions

We code the VAE network in Python using the Keras/Tensorflow framework [add reference] and following the steps described in section 3.1. We chose the latent space dimension of our VAE to be 8. (describe more about Tensorflow tool) A summary diagram of the network is illustrated in Figure 9. The encoder network is set as two dense layers with *LeakyReLU* activation functions. The decoder network is exactly reverse of encoder. Total number of trainable parameters of this VAE is 5,014,152, making it trainable on the CPU of a PC.

A diagram of a computer program

Description automatically generated

Figure 9. Architecture of our VAE network. Details inside encoder and decoder are omitted. (add 4-8 sentences to describe this network)

We train VAE with the hard quantum distributions as described in Section 2.1 and 2.2 respectively. After the training, Gaussian samples are drawn from the latent space, and then passed through the decoder to reconstruct the desired distributions.

Figure 10 and Figure 11 show the VAE reconstructed 3 and 8 qubits distributions for the hard quantum distribution by Fefferman and Umans as described in Section 2.1. Comparing them with Figure 2 and Figure 3, it can be easily observed that VAE performs well on learning the simplerdistribution generated by quantum computation on 3 qubits, but struggles to learn the essential pattern of the more complex 8-qubits data.

A graph with blue lines

Description automatically generated(add x & y)

Figure 10. VAE reconstruction of quantum Fourier sampling on 3-qubits quantum computer proposed by Fefferman and Umans [ref]

A graph with blue lines

Description automatically generated(add x & y)

Figure 11. VAE reconstruction of quantum Fourier sampling on 8-qubits quantum computer proposed by Fefferman and Umans [ref]

VAE results for the states generated by the qsim circuits as in Section 2.2 are shown in Figure 12 and Figure 13. Comparing to Figure 5 and Figure 6, VAE performed reasonably fine on learning the distributions; however substantial discrepancies are notable.

A blue graph with white text

Description automatically generated

Figure 12. VAE reconstruction of quantum distributions generated by execution from qsim circuits with 8 qubits [ref]

A graph with blue lines

Description automatically generated

Figure 13. VAE reconstruction of quantum distributions generated by execution from qsim circuits with 12 qubits [ref]

* 1. NF study of hard quantum distributions

We build the NF network using the Tensorflow\_Probability (TSP) framework and following the steps described in Section 3.2. (add 1-2 reference on TSP) The network is coded in Python by referencing the tutorial from White [17]. As described in Section 3.2, the key components of NF are invertible and differentiable bijectors. To chain up these bijectors, we employ a masked autoregressive network which is ready to use in TSP. An autoregressive network satisfies the property that, for each dimension *i* of the data, distribution of *xi* is only dependent on *xj* for . The benefit of autoregressive network is so that the Jacobian matrix is always lower-diagonal, ensuring super-fast computation of the determinant. Therefore, autoregressive network allows computation-efficient forward transformation, making it a popular choice of NF training. To set up the NF network used in this paper, we chain up five layers of masked autoregressive networks. The loss function is set as the negative log probability as in Eq. (11) to train this NF. The layers of this NF network consist of a series of AutoregressiveNetworks interconnected by TFOpLambda operators. Total number of trainable parameters of this VAE is 98,640, again making it trainable on the CPU of a PC.The architecture diagram generated by Keras.utils.plot\_model() is too large to fit in one page; but it can be found on github [ref] (add a reference with github weblink).

This NF network is trained with the hard quantum distributions as described in Section 2.1 and 2.2 respectively. Afterwards, the reconstruct distributions are drawn from the trained NF.

Figure 14 and Figure 15 show the NF reconstructed 3 and 8 qubits distribution for the hard quantum distribution by Fefferman and Umans as described in Section 2.1. A close-up comparison with Figure 2 and Figure 3 shows that NF learned the overall distribution of both 3 and 8 qubits distributions; but it misses certain granular state features. Notably, Figure 14 overstates the probability of position 1 and 2 on the left side of Figure 2; and Figure 15 misses the probability annihilation for certain discrete numbers on right side of Figure 3.

A graph with numbers and lines

Description automatically generated(add x & y) (header)

Figure 14. NF reconstruction of quantum Fourier sampling on 3-qubits quantum computer proposed by Fefferman and Umans [ref]

A blue graph with white text

Description automatically generated

Figure 15. NF

reconstruction of quantum Fourier sampling on 8-qubits quantum computer proposed by Fefferman and Umans [ref]

NF results for the states generated by the qsim circuits as in Section 2.2 are shown in Figure 16 and Figure 17 . Comparing to Figure 5 and Figure 6, substantial discrepancies are observed. Even though NF is capable of learning the overall shape of the distributions, it lacks the capability to capture the granularities. (add a few sentences to describe what it does well; what it does not do well)

A graph of a number of data

Description automatically generated(x&y) revise header

Figure 16. NF reconstruction of quantum distributions generated by execution from qsim circuits with 8 qubits [ref]

A blue graph with white text

Description automatically generated

Figure 17. NF reconstruction of quantum distributions generated by execution from qsim circuits with 12 qubits [ref]

# **Conclusions and Discussions**

* 1. Performance analysis of VAE and NF
  2. (Recap: we generated the distribution; we learned via VAE and NF, etc.)

It is interesting to compare the performance of VAE and NF. VAE, as a more mature model extensively explored for image generation, only requires a much smaller size of latent space. Therefore, training of VAE is usually more stable and uses less computer resource. On the other hand, NF, as a newer flowed base model, keeps the latent space the same size as inputs, therefore it is capable of obtaining an exact estimate of likelihood of the samples. However, training of NF is typically less stable and more computing resource heavy. In practice of this research, we indeed encountered NaN(Not-a-Number) error frequently during the NF training processes. (explain why does this happen?)

Looking at the VAE and NF results together, we observe that VAE performs reasonably well except for the 8-qubits quantum distribution related to the permanent as in Figure 11. Interestingly, NF demonstrates the better performance for this quantum distribution as in Figure 15. The complementary strength of VAE and NF shall be worth further exploration.

How to quantify and explain the performance of VAE and NF?

(side by side relative to each other and compare)

It is worth noting that the networks in this research have been set up to be sufficiently small so it can be trained on the CPU of a personal computer. A much larger classical neural network will apparently be better at learning. Actually, if the number of neurons grows at exponential speed as the number of qubits, it will not only be able to learn the input sample, but also overfit the distribution certainly. The question is, will a classical neural network be able to learn quantum distribution with polynomial instead of exponential growing number of neurons? If so, what type of quantum distributions can it learn, and what types of distribution it lacks the capability?

* 1. Why are the generative AI models, VAE and NF, only capable of capturing partial features of these quantum distributions? Will their performance be significantly enhanced by revising the architecture of these generative NN models, adjusting hyperparameters, and increasing the amount of training data? Or will the performance of these classical NNs be inherently limited due to lack of expressibility when encountering the complex quantum states with exponential growing Hilbert space? The answer is still far from certain. But our study seems to suggest the latter case. Other quantum distributions

The approach of this papercan certainly be applied to other complex quantum problems. Another good candidate is boson sampling, which evaluates the expected values of permanents of matrices by sampling of bosons scattered by a linear interferometer, proposed by Aaronson and Arkhipov [18]. The photonic implementation of boson sampling is regarded as the most promising approach toward building linear optical quantum computing devices. In brief, for a system of N photons, a typical simplified form for the probability of detecting photons at the *kth* mode is:

, (12)

Where *Perm* stands of the permanent of matrix . U is an unitary matrix defined by the interferometer; is transformed from *U* by keeping the first *M* columns and repeating times of the *j-th* row. This problem is *#P-hard* for classical computers given the existence of the matrix permanent problem. In fact, the complexity of the best classical algorithm for exact boson sampling is *O(N2N+MN2)* for a system with N photons and M outputs modes. Various approaches of realizing boson sampling have been proposed, including the most natural photonic scattering, or using the superconducting resonator device as the interferometer, etc. Learning boson sampling distribution using VAE, NFs and other modern generative NN models will be important both from theoretical and practical perspectives. If certain classical NNs with polynomial number of latent space variables are found capable of learning the essentials, significant progress shall happen to study of molecular vibronic spectra, a practical problem closely related to boson sample. Nevertheless, this is beyond the scope of this paper, so we will leave it for further research in the future.

# **Supplemental materials**

The computer code for building VAE and NF networks, generating quantum distributions and simulations results are available upon reasonable request. (Add github repo link)

# Bibliography

|  |  |
| --- | --- |
| [1] | R. Feynman, "Simulating Physics with Computers," *International Journal of Theoretical Physics,* vol. 21, no. 6/7, pp. 467-488, 1982. |
| [2] | G. Carleo and M. Troyer, "Networks, Solving the Quantum Many-Body Problem with Artificial Neural," *Science,* vol. 355, no. 6325, pp. 602-606, 2017. |
| [3] | X. Gao and L.-M. Duan, "Efficient representation of quantum many-body states with deep neural network," *Nature Communications,* vol. 8, p. 662, 2017. |
| [4] | A. Rocchetto, E. Grant, S. Strelchuk, G. Carleo and S. Severini, "Learning hard quantum distributions with variational," *Nature, NPJ Quantum Information,* vol. 4, no. 28, 2018. |
| [5] | Y. Kwak, W. J. Yun, S. Jung and J. Kim, Quantum Neural Networks: Concepts, Applications, and Challenges., 2021. |
| [6] | F. Bloch, "Nuclear induction," *Physical Review,* vol. 70, no. 7-8, pp. 46-474, 1946. |
| [7] | P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994. |
| [8] | B. Fefferman and C. Chris Umans, "The Power of Quantum Fourier Sampling," *arXiv:1507.05592,* 2015. |
| [9] | T. Isacsson, "Beyond classical computing with qsim," 2023. [Online]. Available: https://pennylane.ai/qml/demos/qsim\_beyond\_classical/. |
| [10] | F. Arute, K. Arya, R. Babbush and e. al., "Quantum supremacy using a programmable superconducting processo," *Nature,* vol. 574, pp. 505-510, 2019. |
| [11] | Quantum AI team and collaborators, "qsim," 2020. [Online]. Available: https://github.com/quantumlib/qsim. |
| [12] | Cirq Developers, "Cirq," [Online]. Available: https://github.com/quantumlib/Cirq/graphs/contributors, https://github.com/quantumlib/Cirq. |
| [13] | D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114,* 2013. |
| [14] | G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *Journal of Machine Learning Research,* vol. 22, pp. 1-64, 2021. |
| [15] | R. Johnson, "The minimal transformation to orthonormality," *Psychometrika,* vol. 31, pp. 61-66, 1966. |
| [16] | E. G. Tabak and C. V. Turner, "A family of nonparametric density estimation," *Communications on Pure and Applied Mathematics,* vol. 66, no. 2, pp. 145-164, 2013. |
| [17] | A. D. White, "Deep Learning for Molecules and Materials," *Living Journal of Computational Molecular Science,* vol. 3, no. 1, p. 1499, 2021. |
| [18] | S. Aaronson and A. Arkhipov, "The Computational Complexity of Linear Optics," *Theory of Computing,* vol. 9, pp. 143-252, 2013. |