# FastPivot: An Algorithm for Inverse Problems

**Yuling Guan**[1] and **Ang Li**[2] and **Sven Koenig**[2] and **Stephan Haas**[1] and **T. K. Satish Kumar**[1,2]
[1]*Department of Physics and Astronomy*
[2]*Department of Computer Science*
University of Southern California
{yulinggu, ali355, skoenig, shaas}@usc.edu, tkskwork@gmail.com

***Abstract*— The laws of physics are usually stated using mathematical equations, allowing us to accurately map a given physical system to its response. However, when building systems, we are often faced with the inverse problem: How should we design a physical system that produces a target response? In this paper, we present a novel algorithm, called FastPivot, for solving such inverse problems. FastPivot starts with a system state and invokes alternating forward and backward passes through the system variables. In a forward pass, it leads the current state of the system to its response. In the subsequent backward pass, a small amount of information is allowed to percolate from the target response back to the system variables. FastPivot produces good quality solutions efficiently. We demonstrate the promise of FastPivot on the inverse problem of placing atoms in a bounded region using a scanning tunneling microscope to achieve target responses in the density of states. We also compare FastPivot to Monte Carlo methods and analyze various empirical observations.**

## I. Introduction

The laws of physics are usually stated using mathematical equations at different spatiotemporal scales. These equations allow us to accurately map a given physical system to its response. For example, the mathematical equations that describe gravity allow us to predict the trajectories of planets in distant galaxies; and the mathematical equations that describe Faraday's law of electromagnetic induction allow us to predict the amount of energy generated by wind turbines. Similarly, at a microscopic scale, the Boltzmann equation of statistical mechanics allows us to analyze hysteresis in magnetic materials [1], [2]; and modern theories allow us to calculate the density of states (DoS) for a given placement of atoms of a specific kind in a bounded region on a substrate.

Despite the usefulness of mathematics in "forward" reasoning, i.e., from the system state to its response, we often face the inverse question while building systems: How should we design a physical system that produces a target response? Such inverse problems arise at both macroscopic and microscopic scales. At macroscopic scales, inverse problems can involve the design of circuits [3], Carnot engines [4], radio-frequency (RF) antennas [5], or aircraft with aerodynamic optimizations [6], as illustrated in Figure 1. At microscopic scales, inverse problems can involve the placement of atoms in a bounded region using a scanning tunneling microscope (STM) to achieve target responses in the DoS or in the spectrum of plasmonic excitations [7]. Unfortunately, inverse problems are combinatorial in nature and cannot always be solved using standard tools from mathematics.

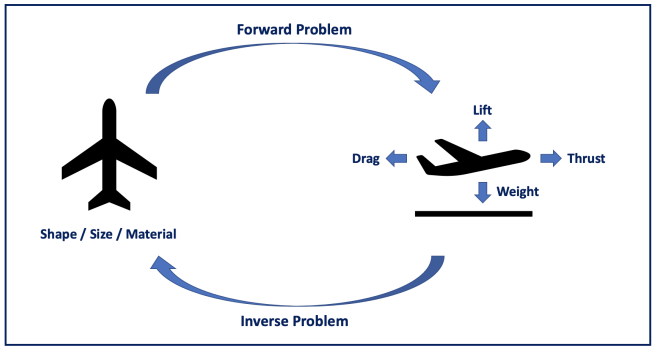


Fig. 1. Illustrates an inverse problem in the domain of aerodynamic optimization. Given various parameters such as the shape, size, and the material of the aircraft, the "forward" problem is to compute the four forces acting on it, i.e., the lift, weight, thrust, and the drag. The inverse problem is to compute the optimal values of the design parameters to achieve desired values of the four forces.

Generally speaking, inverse problems in computational physics present the following challenges: (a) The inverse map from the response to the system is only implicitly defined, i.e., we cannot input the target response into a mathematical equation; (b) the inverse map is not unique; (c) the search space is very large; and (d) there is a substantial involvement of complex numbers, especially at microscopic scales. Inverse problems are particularly hard to solve when the map from the system to its response involves many cascading steps. Each step may be hard to reverse or may introduce branching possibilities if its inverse is not unique. For these reasons, inverse problems are fundamentally combinatorial in nature. Hence, we can benefit from techniques developed in computer science, artificial intelligence (AI), machine learning (ML), and operations research (OR).

Some inverse algorithms have been designed for specific inverse problems using data-driven methods like neural networks [8], [9], [10]. However, such methods are mostly tailored to specific problems and are not broadly applicable. In fact, it is even difficult to train a neural network to learn the inverse mapping of a step that computes the eigenvalues of a matrix satisfying a certain property $P$. This is so because it is hard to identify a unique matrix that satisfies $P$ given only its eigenvalues. So far, very few general principles have been developed for the design of inverse algorithms. The Monte Carlo method [11] is one such general principle.

In this paper, we present a novel algorithm, called Fast-

Pivot, for solving inverse problems. FastPivot introduces a general principle that can be described as follows. We start with a system state and invoke alternating forward and backward passes through the system variables. In a forward pass, we calculate the system's response from its current state. In the subsequent backward pass, we percolate a small amount of information from the target response back to the system variables. The inner loop implements several alternating forward and backward passes in the hope of convergence. The outer loop keeps track of the best solution found so far and triggers algorithm termination based on the availability of computational resources.

While the Monte Carlo method works on a discrete space, i.e., on a lattice, FastPivot works on a continuous space and produces good quality solutions efficiently. We demonstrate its success on the DoS design problem, i.e., the inverse problem of placing atoms in a bounded region using an STM to achieve target responses in the DoS. We also compare FastPivot to the Monte Carlo method and analyze various empirical observations. In general, we observe that FastPivot outperforms the Monte Carlo method in higher dimensions and is better suited for continuous spaces.

## II. Background

The STM is a type of microscope that can be used for imaging surfaces at the atomic level. It does this by applying an electrical voltage to a very sharp metal wire tip that scans the surface (substrate) very closely. It can also be used to manipulate individual atoms. In fact, the STM can be used to place an individual atom at any desired location in a bounded region on a substrate. IBM's nanophysicists have even created a minute-long film with 242 stop-motion frames, called "A Boy and His Atom", by moving individual carbon atoms on a copper substrate. Figure 2 shows the schematic and an actual STM.

Because of its ability to place individual atoms at desired locations on a substrate, the STM supports nanofabrication. In turn, nanofabrication is a cornerstone capability in nanotechnology for creating nanostructures. Because of their size, nanostructures often have special properties that macroscopic structures of the same material may not have. These special properties can be exploited for various applications, including drug delivery and quantum dots.

In this context of using an STM for nanofabrication, one of the fundamental inverse problems is the following: Given $N$ atoms of a certain kind and a bounded region on a substrate, where should we place these atoms to achieve a target response in the DoS? In order to understand this question, we first need to understand the "forward" version of this problem: For a given placement (configuration) of $N$ atoms, what is the DoS that it determines? This determination is done as described below.

First, we subscribe to the tight-binding model in solid-state physics. The tight-binding model [12], [13] is an effective approach for estimating the electronic band structure by using approximations of the atomic wave function. Even though the tight-binding model is a single-electron model, it provides good results for a wide variety of solids. We consider a non-periodical system, which has the following Hamiltonian:

$$\hat{H} = -\sum_{i,j} t_{i,j} \left( \hat{c}_i^\dagger \hat{c}_j + \hat{c}_i \hat{c}_j^\dagger \right), \tag{1}$$

where $\hat{c}_i^\dagger$ and $\hat{c}_i$ represent the electron creation and annihilation operators, respectively, at site $\mathbf{r}_i$, and $t_{i,j}$ is the spatial decay long-range hopping term given by the power law:

$$t_{i,j} = \frac{t}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^q}. \tag{2}$$

Here, $t$ is a constant, and $\mathbf{r}_i$ is the position vector of atom $i$, for $1 \leq i \leq N$. Moreover, $q$ is the power decay parameter that reflects an algebraic variation of the overlap integral with inter-atomic separation. It is specific to the material and is measurable via experiments [14].

Given the position vectors $\mathbf{r}_1, \mathbf{r}_2 \ldots \mathbf{r}_N$ of the $N$ atoms, the steps required to determine the DoS are as follows:

1) Determine the Hamiltonian $\mathbf{H}$ as the $N \times N$ matrix with $\mathbf{H}_{i,j} = t_{i,j}$.
2) Determine the eigenvalues $\mathbf{e}_1, \mathbf{e}_2 \ldots \mathbf{e}_N$ of $\mathbf{H}$.
3) Determine the DoS by placing uniform Lorentzian functions of appropriate height and width centered at each of the eigenvalues.

Essentially, the DoS describes the proportion of states occupied by the system at each energy level. It peaks at the eigenvalues of $\mathbf{H}$. Figure 3 shows the DoS for three different configurations of atoms.

The inverse problem, of interest in this paper, is to find the position vectors $\mathbf{r}_1, \mathbf{r}_2 \ldots \mathbf{r}_N$ in a bounded region given the target eigenvalues of $\mathbf{H}$, i.e., the positions of the peaks in the target DoS.

## III. Methodology

In this section, we provide a description of the Monte Carlo and FastPivot algorithms.

Let the target eigenvalues be $\lambda_1 \geq \lambda_2 \ldots \lambda_N$. All these eigenvalues are real since the Hamiltonian is always a symmetric square matrix. For position vectors $\mathbf{r}_1, \mathbf{r}_2 \ldots \mathbf{r}_N$ of the $N$ atoms in a $K$-dimensional space, let $\mathbf{R}$ denote the $K \times N$ position matrix with columns $\mathbf{r}_1, \mathbf{r}_2 \ldots \mathbf{r}_N$. $\mathbf{R}$ represents a configuration of the $N$ atoms in $K$-dimensional space, and $K$ is usually 2 or 3. Let $\mathbf{H}$ be the Hamiltonian corresponding to a position matrix $\mathbf{R}$; and let $\mathbf{e}_1 \geq \mathbf{e}_2 \ldots \mathbf{e}_N$ be its eigenvalues. $\mathbf{R}$ can be assigned a score to measure how close it is to achieving the target eigenvalues. One such score is the Root Mean Square Error (RMSE) given by $\texttt{score}(\mathbf{R}) = \sqrt{\frac{\sum_{i=1}^{N} (\mathbf{e}_i - \lambda_i)^2}{N}}$.

### A. The Monte Carlo Algorithm

The Monte Carlo algorithm works on a discretization of the continuous space and is inherently limited by the resulting discretized grid. It uses the Metropolis-Hastings importance sampling procedure [11] in its inner loop and generally works as follows.

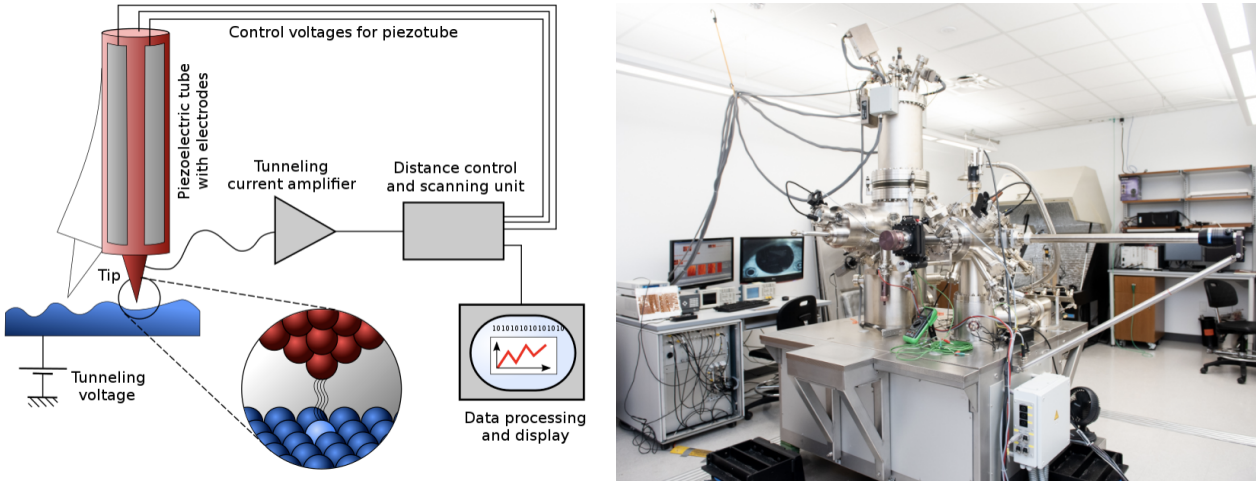


Fig. 2. Shows an STM. The left panel, borrowed from `Wikipedia`, shows the schematic of an STM. The right panel, borrowed from `https://tmi.utexas.edu/facilities/instrumentation/scanning-tunneling-microscopes`, shows an actual STM.

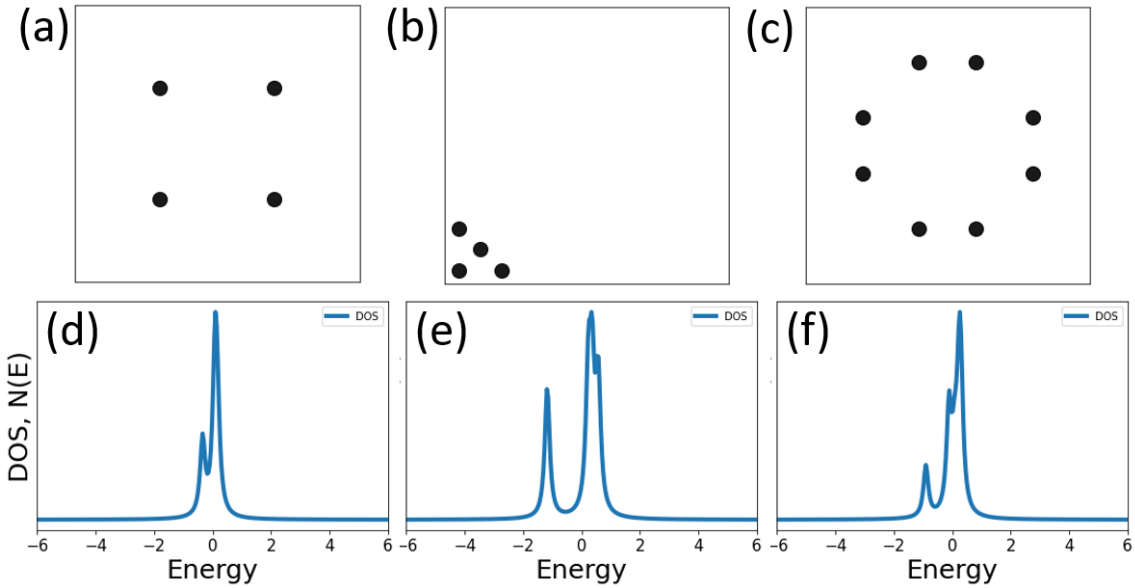


Fig. 3. Shows the DoS corresponding to configurations of atoms in a 2D unit square. (a), (b), and (c) show three different configurations with 4, 4, and 8 atoms, respectively. (d), (e), and (f) show the DoS corresponding to (a), (b), and (c), respectively.

In the outer loop, the algorithm runs a user-specified number of trials. In the inner loop, i.e., in each trial, the algorithm starts by generating a position matrix $\mathbf{R}^0$ by choosing the position vector of each atom uniformly at random in the bounded region. The corresponding Hamiltonian $\mathbf{H}^0$ is also computed. Subsequently, with each increment in the iteration number $i$, the algorithm updates the current position matrix $\mathbf{R}^i$ and computes the corresponding Hamiltonian $\mathbf{H}^i$. $\mathbf{R}^{i+1}$ is constructed from $\mathbf{R}^i$ after considering to move a randomly chosen atom from its current position to a neighboring empty position on the discretized grid to obtain $\mathbf{R}'$. If $\texttt{score}(\mathbf{R}') < \texttt{score}(\mathbf{R}^i)$, the move is accepted and $\mathbf{R}^{i+1}$ is set to $\mathbf{R}'$. Otherwise, the move is accepted with probability $e^{\frac{\texttt{score}(\mathbf{R}^i)-\texttt{score}(\mathbf{R}')}{k_B T}}$, where $k_B$ is the Boltzmann constant and $T$ is a "temperature" that starts high and decreases according to an annealing rule as the iteration number increases. The inner loop is repeated for a user-specified maximum number of iterations before the next trial of the outer loop is initiated. Upon termination, the algorithm returns the position matrix with the lowest recorded score.

### *B. The FastPivot Algorithm*

The FastPivot algorithm works on continuous spaces and does not have the inherent limitations of a discretization. The pseudocode for the FastPivot algorithm is presented in Algorithm 1. Algorithm 1 uses Algorithms 2, 3, and 4 as helper functions.

The input to Algorithm 1 consists of: (a) the number of atoms $N$, (b) the target eigenvalues $\lambda_1 \geq \lambda_2 \ldots \lambda_N$, (c) the number of dimensions $K$, (d) the bounded region of

**Algorithm 1:** `FastPivot`: An algorithm that embeds $N$ atoms in a $K$-dimensional space to achieve target eigenvalues of their interaction Hamiltonian.

**Input:** the number of atoms, $N$; the target eigenvalues, $\lambda_1 \geq \lambda_2 \ldots \lambda_N$; the number of dimensions, $K$; the bounding length on each dimension, $B_1, B_2 \ldots B_K$; the power used in the Hamiltonian terms, $q$; the seed position matrix, $\mathbf{R}^0$; the sampling range vector, $\vec{\sigma}$.

**Parameters:** maximum number of iterations of the inner loop, `MaxSteps`; maximum number of iterations of the outer loop, `MaxTrials`; cutoff improvement parameter in the inner loop, $\gamma$; threshold error parameter in the outer loop, $\varepsilon$.

**Output:** $K \times N$ predicted position matrix, $\mathbf{R}$.

Let $\Lambda$ be a diagonal matrix of the target eigenvalues $\lambda_1 \geq \lambda_2 \ldots \lambda_N$.
Let $\mathbf{R}$ be a $K \times N$ matrix with columns $\mathbf{r}_1, \mathbf{r}_2 \ldots \mathbf{r}_N$.
*bestScore* $\leftarrow \infty$.
*best*$\mathbf{R} \leftarrow$ *Null*.
**for** *trial* $= 1, 2 \ldots$ `MaxTrials` **do**
  **for** $i = 1, 2 \ldots N$ **do**
    Pick $\mathbf{r}_i$ to be a $K$-dimensional vector, whose $k$-th coordinate, for $1 \leq k \leq K$, is chosen uniformly at random from the interval $[\min(0, \mathbf{r}_i^0[k] - \vec{\sigma}[k]), \max(B_k, \mathbf{r}_i^0[k] + \vec{\sigma}[k])]$.
  $\mathbf{H} \leftarrow$ `ComputeHamiltonian`$(\mathbf{R}, q)$.
  *currentScore* $\leftarrow$ `Score`$(\mathbf{H}, \Lambda)$.
  *previousScore* $\leftarrow \infty$.
  *step* $\leftarrow 1$.
  **while** *(step* $\leq$ `MaxSteps` *or* *currentScore* $-$ *previousScore* $\leq -\gamma$*) and* *(currentScore* $> \varepsilon$*)* **do**
    *previousScore* $\leftarrow$ *currentScore*.
    $\mathbf{R} \leftarrow$ `EigenChange`$(\mathbf{R}, \mathbf{H}, \Lambda, K, q)$.
    $\mathbf{H} \leftarrow$ `ComputeHamiltonian`$(\mathbf{R}, q)$.
    *currentScore* $\leftarrow$ `Score`$(\mathbf{H}, \Lambda)$.
    *step* $\leftarrow$ *step* $+ 1$.
  **if** *currentScore* $<$ *bestScore* **then**
    *bestScore* $\leftarrow$ *currentScore*.
    *best*$\mathbf{R} \leftarrow \mathbf{R}$.
  **if** *bestScore* $\leq \varepsilon$ **then**
    Break.
**return** *best*$\mathbf{R}$.

space specified by $B_1, B_2 \ldots B_K$ and defined by the $K$-dimensional orthotope between the farthest corners $(0, 0 \ldots 0)$ and $(B_1, B_2 \ldots B_K)$, and (e) the power used in the Hamiltonian terms $q$. In addition to these inputs, the user can also choose to input: (f) an initial guess of the position matrix $\mathbf{R}^0$, and (g) an allowed perturbation on it using the sampling range vector $\vec{\sigma}$. The output of Algorithm 1 is a predicted position matrix $\mathbf{R}$ intended to achieve the target eigenvalues.

Algorithm 1 also uses several parameters. It uses

**Algorithm 2:** `EigenChange`: A gradient-descent algorithm that updates the position matrix based on the target eigenvalues.

**Input:** $K \times N$ position matrix, $\mathbf{R}$; the Hamiltonian, $\mathbf{H}$; the target diagonal eigenvalue matrix, $\Lambda$; the number of dimensions, $K$; the power used in the Hamiltonian terms, $q$.

**Parameters:** number of updates, $\tau$; learning rate, $\eta$.

**Output:** modified position matrix, $\mathbf{R}$.

Let $\mathbf{V} \times \mathbf{E} \times \mathbf{V}^{-1}$ be the eigendecomposition of $\mathbf{H}$, where $\mathbf{E}$ is a diagonal matrix of non-increasing eigenvalues, and the columns of $\mathbf{V}$ represent the corresponding eigenvectors.
$\mathbf{D} \leftarrow \mathbf{V} \times \Lambda \times \mathbf{V}^{-1}$.
**for** $update = 1, 2 \ldots \tau$ **do**
  $\mathbf{R}' \leftarrow \mathbf{R}$.
  **for** $i = 1, 2 \ldots N$ **do**
    **grad** $\leftarrow 2 \sum_{j \neq i} (\mathbf{D}_{i,j} + \frac{1}{\|\mathbf{r}'_j - \mathbf{r}'_i\|^q})(\frac{-q}{\|\mathbf{r}'_j - \mathbf{r}'_i\|^{q+2}})(\mathbf{r}'_i - \mathbf{r}'_j)$.
    $\mathbf{r}_i \leftarrow \mathbf{r}'_i - \eta$**grad**.
**return** $\mathbf{R}$.

**Algorithm 3:** `ComputeHamiltonian`: An algorithm that computes the Hamiltonian given the position matrix and the nature of the pairwise interactions.

**Input:** position matrix, $\mathbf{R}$; the power used in the Hamiltonian terms, $q$.

**Output:** the Hamiltonian, $\mathbf{H}$.

Construct an $N \times N$ matrix $\mathbf{H}$ with the $(i, j)$-th term defined as follows:
**if** $i = j$ **then**
  $\mathbf{H}_{i,j} \leftarrow 0$.
**else**
  $\mathbf{H}_{i,j} \leftarrow -\frac{1}{\|\mathbf{r}_i - \mathbf{r}_j\|^q}$.
**return** $\mathbf{H}$.

**Algorithm 4:** `Score`: A scoring function for measuring the difference between the current eigenvalues and target eigenvalues.

**Input:** the Hamiltonian, $\mathbf{H}$; the target diagonal eigenvalue matrix, $\Lambda$.

**Output:** a score, *RMSE*.

$\mathbf{E} \leftarrow$ a diagonal matrix of non-increasing eigenvalues of $\mathbf{H}$.
*RMSE* $\leftarrow \sqrt{\frac{\sum_{i=1}^{N} (\mathbf{E}_{i,i} - \Lambda_{i,i})^2}{N}}$.
**return** *RMSE*.

`MaxTrials` for the maximum number of iterations of the outer loop and `MaxSteps` for the maximum number of iterations of the inner loop. It also uses a cutoff improvement parameter $\gamma$ in the inner loop to recognize quiescence and a threshold error parameter $\varepsilon$ in the outer loop to recognize convergence to a high-quality configuration.

On Lines 1-4, Algorithm 1 performs initializations. It stores the target eigenvalues in a diagonal matrix $\Lambda$ with $\Lambda_{i,i}$ set to $\lambda_i$, for $1 \leq i \leq N$. It also initializes *best*$\mathbf{R}$ and *bestScore*, which are intended to record the best configuration encountered so far and its corresponding score, respectively. On Lines 5-22, the algorithm runs its outer loop for a maximum of `MaxTrials` iterations. On Lines 12-17, the algorithm runs its inner loop for a maximum of `MaxSteps` iterations. On Lines 6-11, the outer loop performs the initializations for the inner loop. On Lines 18-22, the outer loop processes the results of the inner loop. On Line 23, the algorithm returns the best position matrix recorded across all iterations of the outer loop.

On Lines 6-7, the outer loop initializes a position matrix by picking each of its columns, i.e., the position vectors of the $N$ atoms, at random. Each position vector is restricted to be in the $K$-dimensional orthotope between the farthest corners $(0,0\ldots0)$ and $(B_1,B_2\ldots B_K)$. This can be done by picking the $k$-th coordinate of each position vector $\mathbf{r}_i$ to be within the interval $[0,B_k]$, for $1 \leq k \leq K$. If the user specifies an area of interest, this interval can be narrowed down to an interval of length $2\vec{\sigma}[k]$ centered around the coordinate of interest in that dimension $\mathbf{r}_i^0[k]$. On Lines 8-9, the outer loop computes the Hamiltonian and its score for the initialized position matrix. On Lines 18-20, the outer loop checks whether the inner loop just encountered a position matrix that is better than the currently best one. If so, it updates *best*$\mathbf{R}$ and *bestScore*. On Lines 21-22, the outer loop checks whether the algorithm has converged to a high-quality configuration with a score no greater than the user-specified threshold error parameter $\varepsilon$. If so, no further iterations of the outer loop are deemed necessary.

On Lines 12-17, the inner loop continues only while "reasonable" progress is made. That is, if the improvement in score between consecutive iterations is no greater than a user-specified cutoff parameter $\gamma$, we declare quiescence and break out of the inner loop. The inner loop makes calls to Algorithms 2, 3, and 4. Algorithm 3 computes the Hamiltonian $\mathbf{H}$ for a given position matrix $\mathbf{R}$ according to the specified value of $q$. Algorithm 4 computes the score of $\mathbf{H}$, corresponding to a position matrix $\mathbf{R}$, by comparing its eigenvalues against the target eigenvalues. In particular, it returns the RMSE between the lists of eigenvalues sorted in non-increasing order.

On Lines 15-16, the inner loop makes a forward pass, i.e., it computes the score of a position matrix via computing the Hamiltonian. On Line 14, the inner loop makes a backward pass using Algorithm 2. The objective is to percolate information from the target eigenvalues back to the position matrix. While this "reversal" of steps is the essence of the inverse problem and is already known to be hard, the main idea is based on the observation that it becomes easy in the context of the most recently completed forward pass. On Line 1, Algorithm 2 computes an eigendecomposition of $\mathbf{H}$, where $\mathbf{E}$ is a diagonal matrix of non-increasing eigenvalues, and the columns of $\mathbf{V}$ represent the corresponding eigenvectors. On Line 2, it uses this eigendecomposition to reconstitute a new Hamiltonian $\mathbf{D}$ by merely replacing the diagonal matrix of eigenvalues $\mathbf{E}$ with the diagonal matrix of the target eigenvalues $\Lambda$. Such a reversal step from the target eigenvalues to a new Hamiltonian with the same eigenvalues is enabled by the most recently completed forward pass that yields $\mathbf{H}$ (and consequently its eigenvectors).

| Algorithm | $N=4$ | $N=8$ | $N=16$ | $N=32$ | $N=64$ |
|---|---|---|---|---|---|
| Monte Carlo | 0.1283 | 0.3126 | 0.5007 | 0.9842 | 0.7322 |
| FastPivot | 0.0596 | 0.2958 | 0.6120 | 1.9073 | 2.5399 |
| Monte Carlo | 0.1382 | 0.0949 | 0.1412 | 0.1701 | 0.2323 |
| FastPivot | 0.0540 | 0.0947 | 0.1324 | 0.1396 | 0.1840 |

TABLE I

A COMPARISON OF AVERAGE RMSE VALUES IN 2D (FIRST TWO ROWS) AND 3D (LAST TWO ROWS) WITH `MaxSteps` SET TO $10N$.

Ideally, Algorithm 2 also needs to execute a reversal step from the new Hamiltonian $\mathbf{D}$ to a new position matrix $\mathbf{R}$. Although doing this reversal directly is deemed to be hard, it becomes slightly easier in the context of the most recently completed forward pass since the current values of the position vectors $\mathbf{r}_1,\mathbf{r}_2\ldots\mathbf{r}_N$ are available. However, they do not make the reversal straightforward either. Therefore, the backward pass makes use of the observation that it suffices to make only small updates in the direction of the reversal, since doing so across many forward and backward passes achieves the same overall effect. On Lines 3-7, Algorithm 2 does this by using $\tau$ steps of gradient descent with respect to the position vectors aimed at minimizing the Frobenius norm of $\mathbf{D}-\mathbf{H}$. The updates use a user-specified learning rate $\eta$.

## IV. RESULTS AND DISCUSSION

In this section, we present our empirical results and analyses. Numerical experiments were conducted on a 3.6 GHz AMD Ryzen 5 3600 6-core CPU with 16 GB RAM. All algorithms were implemented in Python 3.6. We used the tight-binding model and placed atoms in either a 2D or 3D bounded region of space to achieve the target DoS eigenvalues. To facilitate a fair comparison of the Monte Carlo and FastPivot algorithms, we gave both algorithms the same number of inner loop and outer loop iterations. $\tau$ and $\eta$ were set to 50 and 0.1, respectively.

All our test instances were "reverse engineered". That is, they were generated as follows. We first picked a random configuration of the $N$ atoms in the bounded region of space. We computed the Hamiltonian using the tight-binding model and then its eigenvalues where the DoS peaks. We provided these eigenvalues as input to the Monte Carlo and FastPivot algorithms and challenged them to reconstruct the original

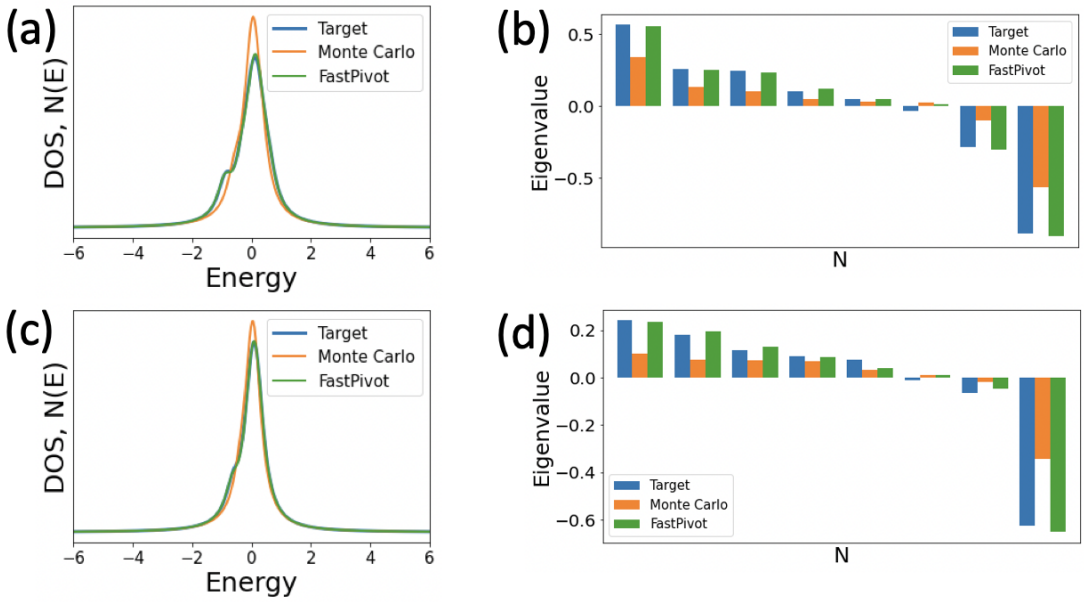


Fig. 4. Shows a comparison of the Monte Carlo and FastPivot algorithms. (a) and (b) show the DoS and eigenvalues, respectively, for the configurations returned by the Monte Carlo and FastPivot algorithms in comparison with the target DoS and eigenvalues on a test instance with 8 atoms in a 2D bounded region of space. (c) and (d) show the DoS and eigenvalues, respectively, for the configurations returned by the Monte Carlo and FastPivot algorithms in comparison with the target DoS and eigenvalues on a test instance with 8 atoms in a 3D bounded region of space. Both algorithms were given a single trial with 2496 steps in the 2D case and 4128 steps in the 3D case.

| Algorithm | $\varepsilon = 0.10$ | $\varepsilon = 0.08$ | $\varepsilon = 0.06$ | $\varepsilon = 0.04$ | $\varepsilon = 0.02$ |
|---|---|---|---|---|---|
| Monte Carlo | 7/10 | 7/10 | 5/10 | 1/10 | 1/10 |
| FastPivot | 10/10 | 8/10 | 10/10 | 10/10 | 9/10 |
| Avg #Steps | 72.75 | 103.84 | 356.97 | 647.38 | 2496.31 |
| Monte Carlo | 10/10 | 9/10 | 9/10 | 6/10 | 1/10 |
| FastPivot | 10/10 | 10/10 | 10/10 | 10/10 | 10/10 |
| Avg #Steps | 28.05 | 127.77 | 200.20 | 794.64 | 4128.07 |

TABLE II

A COMPARISON OF SUCCESS RATES IN 2D (FIRST THREE ROWS) AND 3D (LAST THREE ROWS) WITH $N = 8$ AND DIFFERENT VALUES OF $\varepsilon$.

configuration. Of course, the algorithms were also free to construct any other configuration with the same eigenvalues. The reverse engineering merely validates each test instance and assures us that the DoS is achievable.

Figure 4 shows the comparative performances of the Monte Carlo and FastPivot algorithms on two example test instances with 8 atoms each but one in 2D and one in 3D. The FastPivot algorithm with $\gamma = 0$ outperforms the Monte Carlo algorithm in these cases.

Table I shows the comparative performances of the Monte Carlo and FastPivot algorithms on a suite of test instances. Each entry in the table represents the RMSE value averaged over the same 20 test instances supplied to both algorithms. For each test instance, both algorithms were given 20 trials and $10N$ steps in each trial. The test instances are categorized according to the number of atoms $N$. The top two rows correspond to test instances that use a 2D bounded region of space; and the bottom two rows correspond to test instances that use a 3D bounded region of space. The FastPivot algorithm with $\gamma = -\infty$ marginally outperforms the Monte Carlo algorithm on (a) 3D test instances and (b) 2D test instances when $N$ is small.

Although the Monte Carlo algorithm marginally outperforms the FastPivot algorithm on 2D test instances when $N$ is large, this does not automatically imply that the Monte Carlo algorithm is preferred on such instances. This is so because, interestingly, the FastPivot algorithm marginally outperforms the Monte Carlo algorithm on 2D test instances when $N$ is small. This observation can be exploited to design a "hierarchical" version of the FastPivot algorithm for larger values of $N$. A comparison of the proposed hierarchical version of the FastPivot algorithm against the Monte Carlo algorithm is delegated to future work.

Table II shows the comparative performances of the Monte Carlo and FastPivot algorithms on another suite of test instances with $N = 8$ and different values of $\varepsilon$. In this scenario, we test FastPivot's ability to avoid local minima by giving it a single trial in the outer loop and an unlimited number of steps in the inner loop with $\gamma = 0$. Under these conditions, the FastPivot algorithm terminates only when

it either meets the targeted solution quality specified by $\varepsilon$ or reaches a local minimum. For a fair comparison, a single trial and the same number of steps utilized by the FastPivot algorithm before termination were also provided to the Monte Carlo algorithm. For each value of $\varepsilon$, the same set of 10 instances was supplied to both algorithms. The entries '-/10' in the table indicate the number of instances solved by them. The average number of steps required for the FastPivot algorithm to terminate on these instances is also reported. The FastPivot algorithm encounters local minima much less often than the Monte Carlo algorithm. This accounts for its superior performance.

## V. Conclusions and Future Work

In this paper, we presented FastPivot, an algorithm for solving inverse problems. FastPivot starts with a system state and invokes alternating forward and backward passes through the system variables. In a forward pass, it calculates the system's response from its current state. In the subsequent backward pass, it percolates a small amount of information from the target response back to the system variables. We demonstrated the success of FastPivot in the context of using an STM for nanofabrication. We addressed an important problem: Given $N$ atoms of a certain kind and a bounded region on a substrate, where should we place these atoms to achieve a target response in the DoS? We showed that FastPivot produces good quality solutions efficiently for this problem. We also compared FastPivot to Monte Carlo methods and observed two advantages of FastPivot: (1) It outperforms Monte Carlo methods in higher dimensions; and (2) it solves the problem on continuous spaces.

In future work, we will impart various algorithmic enhancements to FastPivot. We will also develop a hierarchical version of it. On the application side, we will address other important nanofabrication tasks, such as the design of collective response [15], [16], [17]. In particular, we will apply the FastPivot approach to identify nanostructures with specifically tailored plasmonic excitations, i.e., with customized plasmon frequencies, intensities, and spatial profiles. The benefit of such an endeavor is that controlling collective modes in nanostructures would enable true functionality on the quantum level, such as quantum sensing and switching [18].

## VI. Acknowledgements

This work at the University of Southern California is supported by NSF under grant number 2112533. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the sponsoring organizations, agencies, or the U.S. Government.

## References

[1] A. Li, Y. Guan, S. Koenig, S. Haas, and T. K. S. Kumar, "Generating the top $k$ solutions to weighted CSPs: A comparison of different approaches," in *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, 2020.

[2] Y. Guan, A. Li, S. Koenig, S. Haas, and T. K. S. Kumar, "Hysteresis in combinatorial optimization problems," in *Proceedings of the International FLAIRS Conference*, 2021.

[3] S. Richer and D. DiVincenzo, "Circuit design implementing longitudinal coupling: A scalable scheme for superconducting qubits," *Phys. Rev. B*, vol. 93, 2016.

[4] L. Chen, X. Zhu, F. Sun, and C. Wu, "Exergy-based ecological optimization of linear phenomenological heat-transfer law irreversible carnot-engines," *Applied Energy*, vol. 83, no. 6, 2006.

[5] L. M. Borges, N. Barroca, H. M. Saraiva, J. Tavares, P. T. Gouveia, F. J. Velez, C. Loss, R. Salvado, P. Pinho, R. Gonçalves, N. B. Carvalho, R. Chavéz-Santiago, and I. Balasingham, "Design and evaluation of multi-band RF energy harvesting circuits and antennas for WSNs," in *Proceedings of the International Conference on Telecommunications*, 2014.

[6] S. N. Skinner and H. Zare-Behtash, "State-of-the-art in aerodynamic shape optimisation methods," *Applied Soft Computing*, vol. 62, 2018.

[7] E. Le Moal, S. Marguet, D. Canneson, B. Rogez, E. Boer Duchemin, G. Dujardin, T. V. Teperik, D.-C. Marinica, and A. G. Borisov, "Engineering the emission of light from a scanning tunneling microscope using the plasmonic modes of a nanoparticle," *Phys. Rev. B*, vol. 93, 2016.

[8] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos, "Using deep neural networks for inverse problems in imaging: Beyond analytical methods," *IEEE Signal Processing Magazine*, vol. 35, no. 1, 2018.

[9] E. A. Voronin, V. N. Nosov, and A. S. Savin, "Neural network approach to solving the inverse problem of surface-waves generation," *Journal of Physics: Conference Series*, 2019.

[10] Y. Gao, H. Liu, X. Wang, and K. Zhang, "On an artificial neural network for inverse scattering problems," *Journal of Computational Physics*, vol. 448, 2022.

[11] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*. New York: Wiley Series in Probability and Statistics, John Wiley and Sons, 2011.

[12] A. N. Rudenko and M. I. Katsnelson, "Quasiparticle band structure and tight-binding model for single- and bilayer black phosphorus," *Phys. Rev. B*, vol. 89, 2014.

[13] M. Nakhaee, S. A. Ketabi, and F. M. Peeters, "Tight-binding model for borophene and borophane," *Phys. Rev. B*, vol. 97, 2018.

[14] G. V. Nazin, X. H. Qiu, and W. Ho, "Visualization and spectroscopy of a metal-molecule-metal bridge," *Science*, vol. 302, no. 5642, 2003.

[15] Y. Guan, Z. Jiang, and S. Haas, "Control of plasmons in topological insulators via local perturbations," *Phys. Rev. B*, vol. 104, 2021.

[16] H. Schlömer, Z. Jiang, and S. Haas, "Plasmons in two-dimensional topological insulators," *Phys. Rev. B*, vol. 103, 2021.

[17] Y. Guan, S. Haas, H. Schlomer, and Z. Jiang, "Plasmons in z2 topological insulators," 2022. [Online]. Available: https://arxiv.org/abs/2205.04062

[18] A. Levi and S. Haas, *Optimal Device Design*. Cambridge University Press, 2009.