TheoSea: Marching Theory to Flows and Light

Stalzer MA¹, Ju C²
¹Center for Data-Driven Discovery, California Institute of Technology.
²Minerva Schools at KGI.

*Corresponding Author: Stalzer MA, Center for Data-Driven Discovery, California Institute of Technology, California, USA.
E Mail: stalzer@caltech.edu

Received: 17 January 2021
Accepted: 23 January 2021
Published: 30 January 2021

Abstract
In the far-field of a radiating dipole antenna, sufficient data is required to rediscover the Maxwell Equations and the light wave equations, namely light speed c. TheoSea is about a program by Julia that does this in a few seconds, and the main insight is that the compactness of is theories drives the search. It has also rediscovered parts of the Navier-Stokes equations, giving some evidence of the generality of the discovery architecture and underlying algorithms. The program is a scientific method's computational embodiment: observation, consideration of candidate theories, and validation.

Keywords: Theosea, Julia program, Light.

INTRODUCTION

Significance. A computer can re-discover physical theories in multiple fields given the appropriate representation language and experimental data. It may only take a few seconds.

MOTIVATION AND BACKGROUND
This paper flows from a comment in the concluding remarks of a recent review (2016) of work in data-driven scientific discovery [1]. Specifically,

... It may be within existing computing and algorithmic technology to infer the Maxwell Equations directly from data with vector calculus knowledge.

This paper reports on recent progress towards this objective. It is based on [2, 3], but additionally presents generalizations and
the application to the Navier-Stokes equations. The overarching objective is to develop methods that can infer data from compact theories. Techniques for data intensive analysis are based on machine learning or statistics. These are pretty good ones, it is helpful, but it does not lead to deep understanding or insight. The scientific method and creative Scientists were very good at testing (experimenting) and constructing human comprehensibility. The models (theory). We transform both of these principles on their heads in this program: can a machine be given Figuring out mathematically compact hypotheses in a suitable virtual experiment (VE)? The initial application was in electrodynamics (the Maxwell Equations) [4], but then we applied it to laminar fluid flow and it worked there as well, rediscovering parts of the Navier-Stokes equations [5]. In this paper, these are referred to as Applications I & II. Eventually, the techniques developed are supposed to be applicable to data sets from actual measurements in a wide range of physics, engineering, and economic fields.

**Past work.** Attempts to use computers to rediscover physical laws may have originated in 1979 with BACON.3 [6]. The program successfully discovered the perfect gas law, P V = nRT, from small Tables of Info. The ideal gas law has been rediscovered by one of us (Stalzer) and William Xu of Caltech Using the method of this paper [7], with Van der Waals powers. In 2009, researchers rediscovered the kinematic equation for the double pendulum essentially using optimization methods to fit constants to candidate equations [8].

What differentiates this work is two fold: the concept of search driven by compactness and completeness, and targeting much more difficult theories like electrodynamics and fluid flow. Indeed, The first unification (electric and magnetic fields) was electrodynamics, and the special relativity of Einstein is baked right into the equations once the great observation is made that in all inertial reference frames, c is the same. TheoSea also describes the light wave equations as a result of the rediscovered free space Maxwell Equations.

**Julia.** TheoSea is written in Julia [9], a relatively recent language (roughly 2012) that is both easy to use and has high performance. At a high expressive level, Julia can be programmed, and yet Given enough type data, efficient machine code is automatically generated. In terms of Julia sets, TheoSea is a Julia meta-program that writes candidate theories which are then validated against results. In the candidate theories, the set elements are Julia expressions compiled corresponding to terms. At the same time, it is difficult to provide both high levels of expression and good low-level output in other languages such as C and Fortran, and that is a key motivation in the development of Julia.

**Plan.** The next section presents some preliminaries that are used by TheoSea in general, and specifically for the two applications. Central to the approach is the section on rapid enumeration (Sec. 2.1). Secs. 3 and 4 then present results for the two applications, particularly the virtual experiments, rediscovery, and run-times. The paper ends with some concluding remarks (Sec. 5).

**PRELIMINARIES**

Both applications rest on two key algorithms: rapid enumeration of theories, and connecting the theories to virtual experiments (Sec. 2.2).
Rapid listing of theories of candidates through a language $L$

Given the alphabet $A$ of symbols such as operators and fields, if there is a Turing machine that enumerates all valid strings in the language, the language $L$ is recursively enumerable [10].

By the infinite monkey theorem [11] the solution can be found — if the constants are limited to rationals — just by enumeration and validation. The purpose of this section is to demonstrate a way to do this enumeration in a tractable way that finds the most compact theory as well.

Abstract enumeration. Abstractly, think of an alphabet $A = \{A,B,C,\ldots\}$ where any letter can appear once in a sentence and the length of the alphabet is $n$. This is a basic problem of combinatorial enumeration and the solution to the number of $m$ size sets (later $m$ will be relabeled $q$) taken from $A$ is $C(n,m)$. What if, though, the symbols in the alphabet, letters, have different weights? What if the alphabet is more like $A = \{A = 1, B = 1, C = 4, D = 4, E = 4, F = 4, G = 7, H = 7, I = 7, J = 7, K = 4, L = 19\}$. This will significantly reduce the size of the enumeration and Below, the underlying motivation is shown in the electrodynamics decoder ring.

The TheoSea enumeration algorithm Constructs sets of increasing complexity $q$, where the sum is $q$ of the alphabet letter weights in a given candidate theory. It can be thought of as a form of Depth-First Iterative Deepening (DFID) [12] first formalized by R. E. Korf in 1985. Optimality flows from a theorem by Korf:

Theorem 1 (Korf 4.2) Depth-first iterative-deepening is asymptotically optimal among brute-force tree searches in terms of time, space, and length of solution.

By length of solution, Korf is meant to be the depth of the search where a solution is found. For TheoSea compactness is the sum of the symbol weights along a potential solution branch in the search.

It is perhaps easiest to think of the algorithm inductively. There is a theos data structure that contains all $q$ length theorems (sets) and it is built up from $q = 1$. The basic cases are the singleton theories of a given complexity, so $\text{theos}[1] = \{A, B\}$ and $\text{theos}[4] = \{D, \ldots\}$ and so on are available for the alphabet $A$. So the base cases are all set, such as $q = 1$; and then we use a $q!$, squeeze $m$'s for $q > 1$. All theories that may be of length $q$ are considered at stage $q$, marching $1$ upwards from $1$ and $m$ downwards from $q - 1$. In Korf 4.2, the correctness is immediate and the reality is true. That $q = 1 + m$: theories that are too short are discarded ($< q$), and the elements set are special. More details can be found in [2].

Performance. We are running an experiment comparing the enumeration of $A$ (Fast) with the same set of letters but with unit weights (Slow) to measure the output with a cutoff at $q = 14$. The total times for the above weighted $A$ are $\text{Quick} = 0.006s$, and $\text{Slow} = 20.1s^1$. A graph is in Fig. 1: compactness matters

---

\(^1\)The machine was a MacBook Pro (Retina, 13-inch, Late 2013) running a 2.8 GHz Intel Core i7 single-threaded using Julia 0.5.
This is one of those cases when symbolics and numerics do not play well together.

Figure 1. Time to discovery (s): Fast versus Slow brute force. The Slow search cuts off at 12 due to maximum complexity.

**Decoder ring for electrodynamics** The underlying motivation was described above and here is the decoder ring; think of the electric and magnetic fields with the A assignments of A =E electric field, B = B magnetic field...

<table>
<thead>
<tr>
<th>Operator</th>
<th>Term Cost</th>
<th>Alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{F}$</td>
<td>1</td>
<td>A, B</td>
</tr>
<tr>
<td>$\nabla \cdot \mathcal{F}$</td>
<td>4</td>
<td>C, D</td>
</tr>
<tr>
<td>$\nabla \times \mathcal{F}$</td>
<td>7</td>
<td>G, H</td>
</tr>
<tr>
<td>$\nabla^2 \mathcal{F}$</td>
<td>7</td>
<td>I, J</td>
</tr>
<tr>
<td>$\frac{\partial \mathcal{F}}{\partial t}$</td>
<td>4</td>
<td>E, F</td>
</tr>
<tr>
<td>$\frac{\partial^2 \mathcal{F}}{\partial t^2}$</td>
<td>19</td>
<td>K, L</td>
</tr>
</tbody>
</table>

Table 1: Complexity of each operator term working on a field $F \in \{E, B\}$

The Maxwell Equations are expressed in linear combinations of these terms as will be described in Sec. 3.1. The complexity metric is just 1+ the number of space-time derivatives taken.

**Theory Validation: Fitting Constants**

The glue to the virtual experiments (Secs. 3.2 & 4.1) that binds the enumeration (Sec. 2.1) is Seeking constants in the hypotheses of candidates that match the results. The issue is seeking an equation of the type (remember that $A = n$ is the size)

$$c_1 A_1 + \ldots + c_n A_n = 0 \quad (1)$$

Where by the enumeration, many of the $c_i$ are implicitly zero. If such $c_i$ do not exist, which is almost always the case, the theory is invalid. If they do exist, however, then the theory is valid with high Probability. As mentioned below, this is a linear algebra problem.
Finding the constants is equal to finding a linear system's null space. (for example, the data matrix extracted from Tab. 2). If the null-space dimension is 0, then the theory is 0. Since a trivial zero vector is the only solution, it is not real. If the null-space dimension is non-zero, it can only be 1, leading to a unique solution. The explanation is that we remove all valid sub-theories from the candidate theory in our enumerative method before deciding its constants. If the dimension of the null space of the resulting structure had been greater than 1, it would have meant that any sub-theory would not be removed, contradicting the assumption. Next, to find the rank of the null space and the null space itself, we cannot simply use Julia’s built-in rank() or nullspace() functions because the dynamic ranges are large (> $10^{30}$).² The solution is to use the decomposition of the singular value, where the number of zero singular values (SVs) is equal to the null space dimension. The insight is that if we scale B by a factor of c, it will be on the same scale as E, and after each column of the matrix has been normalized such that each column is on the same scale as the other, the resulting singular values (if non-zero) should also be on the same scale. We use the built-in Julia svdvals() function after scaling and normalizing to obtain a list of SVs rated from the largest to the smallest.

The dimension of the null space can be either 1 or 0, as discussed previously, and we only need to equate the smallest SV with the largest one to see if the former is smaller than the latter in order of magnitude. If so, by calling Julia’s svd() function, we can consider that as a zero, and proceed to retrieve the null space vector from the last column of $OV^T$ (as in $A = UΣV^T$).

The null space vector elements are the constants that we search for. If not, it means that the null space dimension is zero and we infer that the principle is invalid.

**APPLICATION I: ELECTRODYNAMICS AND LIGHT**

The first application is rediscovering the Maxwell Equations. The following three sections introduce the equations, describe the virtual experiment, and show the results of the rediscovery.

**The Maxwell Equations**

The Maxwell Equations in free space with the transformation $B' = cB$ are [4]:

\[
\begin{align*}
\nabla \cdot E &= 0 \\
\nabla \cdot B' &= 0 \\
\nabla \times E + \frac{1}{c} \frac{∂B'}{∂t} &= 0 \\
\nc\nabla \times B' - \frac{∂E}{∂t} &= 0
\end{align*}
\]

where $c = 2.99792458 \times 10^8 m/s$ (MKS units). The spatial-temporal coupling of E and B is how we get electromagnetic waves. The utility of the B transformation for numerical stability is discussed in the previous section and that is why the Equations look in a slightly strange form in terms of constants.
Connecting back to language, the divergence equations 2 & 3 are of complexity 4, and the space-time couplings 4 & 5 are of complexity 11. TheoSea does not know that, but by validating its candidate theories (strings over L) against data, the code performs vector calculus.

**Observations and the Virtual Experiment**

As shown in the geometry, the validation data is from the far-field of a radiating antenna for E,B Figure. 2 and data Table. 2.

<table>
<thead>
<tr>
<th>$r \times 10^{17}$</th>
<th>$\phi$</th>
<th>$\theta$</th>
<th>$E_P(0) \times 10^{-16}$</th>
<th>$B_P(0) \times 10^{-16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.905</td>
<td>0.997</td>
<td>(2.713, 3.455, 6.793)</td>
<td>(-6.363, 4.996, 0)</td>
</tr>
<tr>
<td>1</td>
<td>2.767</td>
<td>2.908</td>
<td>(2.018, -0.794, -0.516)</td>
<td>(-0.816, -2.074, 0)</td>
</tr>
<tr>
<td>1</td>
<td>4.631</td>
<td>0.291</td>
<td>(-0.214, -2.639, -0.793)</td>
<td>(2.754, -0.224, 0)</td>
</tr>
<tr>
<td>1</td>
<td>5.597</td>
<td>3.051</td>
<td>(-0.666, 0.546, -0.078)</td>
<td>(0.548, 0.669, 0)</td>
</tr>
<tr>
<td>1</td>
<td>0.468</td>
<td>2.369</td>
<td>(-4.299, -2.170, -4.690)</td>
<td>(-3.029, 6.001, 0)</td>
</tr>
</tbody>
</table>

Table 2. Experiments with geometry parameters and the corresponding field observations at point P and time t; adjusted for scale.

The fields at a far point P are [13]:

$$E = -\frac{\mu_0 p_0 \omega^2}{4\pi} \left(\frac{\sin \theta}{r}\right) \cos[\omega(t - r/c)] \theta$$  \hspace{1cm} (6)

$$B = -\frac{\mu_0 p_0 \omega^2}{4\pi c} \left(\frac{\sin \theta}{r}\right) \cos[\omega(t - r/c)] \phi$$  \hspace{1cm} (7)

where $\mu_0 = 4\pi \times 10^{-7}$ is the permeability of free space, $p_0$ is the strength of the dipole, and $\omega$ is the frequency of the dipole oscillation.

Figure 2. Geometry of the far-field of a dipole with moment $p$ at the origin oscillating at angular frequency $\omega$ at a far point $P$. 
Figure 3. Average intensity of the Poynting vector, the vector is proportional to \((E \times B)(\omega)\).

With different parameters \(r, \phi, \theta\) with a fixed \(\omega\), five virtual experiments were performed. The observables are \(E(x,t)\) and \(B(x,t)\), where \(x\) is in the region of the point \(P\). The good thing about this VE is that different derivatives of space and time can be analytically computed. These experiments are shown in Tab. 2, with the fields given at a steady state \(t = 0\).

Electrodynamics Results
As seen in the screenshot, TheoSea has rediscovered the Maxwell Equations Figure 4 and, using the methods mentioned above, it took about 5 seconds. For a bit more intuition, the magnitude of the Poynting vector is shown in Figure 3: the code is essentially quickly enumerating space-time theories on the surface of the saucer-shaped object and finding the most compact ones.

![Screenshot of TheoSea’s output showing the Maxwell Equations and the wave equations of light including the speed of light \(c \approx 2.99 \times 10^8\).](image)

In addition, it rediscovered

\[
\frac{1}{c^2} \frac{\partial^2 E}{\partial t^2} - \nabla^2 E = 0 \tag{8}
\]

\[
\frac{1}{c^2} \frac{\partial^2 B}{\partial t^2} - \nabla^2 B = 0 \tag{9}
\]
that is a plane electromagnetic wave traveling in free space: Light.\(^3,4\)

**APPLICATION II: LAMINAR FLUID FLOWS**

The Navier-Stokes equations for incompressible flows are [5]:

\[
\frac{\partial \rho}{\partial t} + \nabla (\rho \mathbf{V}) = 0 \tag{10}
\]

\[
\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla) \mathbf{V} = -\frac{\nabla P}{\rho} + \frac{\mu}{\rho} \nabla^2 \mathbf{V} \tag{11}
\]

where \( \mathbf{V} \) is the velocity of the flow field, \( p \) is the pressure, \( \rho \) is the density of the field, and \( \mu \) is the dynamic viscosity. According to our rule of assigning complexity scores, the first equation has complexity 6, and the second equation has complexity 31.

This section follows a similar pattern: virtual experiment, expression language, and rediscovery results.

**Fluid Dynamics Virtual Experiment**

Traditionally, simulating fluid flow amounts to solving the Navier-Stokes equations using techniques such as the finite element or the finite volume method, the essence of which lies in discretizing space and time and solving the field equations macroscopically. In contrast, the Lattice Boltzmann Method (LBM) operates on the mesoscopic scale [14]. Instead of solving the field equations themselves, LBM models fluid flow from the principles of statistical mechanics, and the flow is driven by collision of particle densities governed by the conservation of momentum. The insight is that from particle densities we can deduce macroscopic variables of interest: \( p, \mathbf{V}, \rho \).

The reasons we chose LBM over the traditional method to model fluid flow are twofold. First, the output data (such as pressure and density) from LBM are less accurate than what would be obtained from a typical PDE solver. Having noisy data in some way mimics the situations in the real world, which we want our automated approach, TheoSea, to eventually tackle. Second, it is not at all obvious to a first-timer that the laws of hydrodynamics can be derived from the Boltzmann equation in statistical mechanics. Doing so would need a few nontrivial approximations of the original equation [5], and we want to show that TheoSea can do this within seconds.

The simulation is on 2D laminar flow past a cylinder. All physical parameters of the simulation are the same as in Dr. Jonas Latt’s 2D flow around a cylinder\(^5\). In lattice units, the board has width 180 and length 420. The Reynolds number is 10, and the dynamic viscosity is \( \mu = 0.08 \). To create spatial variation, values of density, pressure, and velocity are interpolated at 18 coordinate points. Temporal variation is achieved by recording the values of these three variables at time step 999, 1000, and

\(^3\)The derivation of these wave equations from the Maxwell Equations takes humans some non-trivial vector calculus, and yet the machine did it by “enlightened” search.

\(^4\)This machine was a MacBook Pro (Retina, 15-inch) running a 2.5 Ghz Intel Core i7 single-threaded.
1001. The following table lists the interpolated values of density, pressure, and velocity at 5 sample lattice points after 999 time steps.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>ρρ</th>
<th>Pρ</th>
<th>Vρ</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>50</td>
<td>1.00144</td>
<td>0.33381</td>
<td>(0.04022, -0.00748)</td>
</tr>
<tr>
<td>60</td>
<td>50</td>
<td>1.00168</td>
<td>0.33389</td>
<td>(0.04133, -0.00965)</td>
</tr>
<tr>
<td>70</td>
<td>50</td>
<td>1.00195</td>
<td>0.33398</td>
<td>(0.04346, -0.01214)</td>
</tr>
<tr>
<td>180</td>
<td>60</td>
<td>1.00231</td>
<td>0.33410</td>
<td>(0.03672, 0.00802)</td>
</tr>
<tr>
<td>180</td>
<td>70</td>
<td>1.00238</td>
<td>0.33413</td>
<td>(0.03374, 0.00722)</td>
</tr>
</tbody>
</table>

Table 3. Experiments with geometry parameters and the corresponding field observations at point P and time step t = 999; adjusted for scale.

The magnitude of the velocity field after 999 time steps is shown in Figure 5.

Language

Similar to the previous section on electrodynamics, we need to start with singleton theories and represent them using an alphabet. What’s different is that here we have 3 basic variables, p, ρ, V, instead of 2, E, B, for the electrodynamic application. The dimensional relationships between the three variables are not as simple as that between E and B. Therefore, we need to start with educated guesses for what the initial singleton theories should be. Table 4 summarizes potential scalar singleton theories and their complexities.

The continuity equation is then [A,F] (Eqn. 10). Similarly, we can form vector singleton theories. To test whether TheoSea will rediscover the Navier-Stokes equations and find the constant for dynamic viscosity, we simply used the singleton theories present in the Navier-Stokes equations.

Fluid Dynamics Results

TheoSea rediscovered the Navier-Stokes equations for 2D incompressible laminar flow, Eqns. 10 & 11, as shown in Fig. 6. In addition, TheoSea finds the vector identity \( \nabla (\rho V) = \rho \nabla \cdot V + V \cdot \nabla \rho \), as well as the dynamic viscosity constant \( \mu = 0.08 \). The runtime was less than a half second.

---

5jonas.latt@unige.ch, Universite de Geneve, Switzerland.
6It is okay to put every reasonable singleton theories into the initial set, but in this example the result will be the same despite the extra time cost, as there is no other equations of significance other than the Navier-Stokes equations in this case.
The author encourages the Julia developers to continue work on threads as the model is natural for multicore processors. For example, the main thread could enumerate candidate theories and then send them to several worker threads for validation. At any instant severa.

![Velocity magnitude field after 999 time steps.](image)

**Figure 5. Velocity magnitude field after 999 time steps.**

<table>
<thead>
<tr>
<th>Alphabet</th>
<th>Candidates</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(\partial \rho / \partial t)</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>(\frac{1}{\rho} \partial \rho / \partial t)</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>(\nabla \cdot \nabla \rho)</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>(\nabla \cdot \mathbf{v})</td>
<td>4</td>
</tr>
<tr>
<td>E</td>
<td>(\rho \nabla \cdot \mathbf{v})</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>(\nabla \cdot (\rho \mathbf{v}))</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>(\nabla^2 \rho)</td>
<td>7</td>
</tr>
</tbody>
</table>

**Table 4. Scalar candidate singleton theories for Navier-Stokes.**

**CONCLUDING REMARKS AND FUTURE WORK**

There are many avenues for future development as briefly listed below.

- Expand the enumeration language to allow more expressive theories. Right now TheoSea is limited to theories of the form \(c \mathbf{A} + c \mathbf{A} + \ldots\) where the \(A\)s are operators over fields like \(E, B\). The Xu ideal gas law code works with scalar fields and exponents. Somehow these should be merged.
The theorem is of the form: $-1.0\nabla\cdot(p\nabla) - 0.9944882820239741ap/\partial t = 0$

The theorem is of the form: $-0.95568929560711p\nabla\cdot V - 0.9546251462820973ap/\partial t - 1.0\nabla\cdot p = 0$

The theorem is of the form: $0.99891045510067826p\nabla\cdot V - 0.9986714510019403p\cdot (p\nabla) + 1.0\nabla\cdot V = 0$

elapsed time: 0.118373131 seconds

The theorem is of the form: $-0.9966568274682825(\nabla\cdot V) V - 0.9987771504939688ap/\partial t - 1.0p\nabla/p$

$+0.07917488763140895v/\partial t = 0$

elapsed time: 0.291487127 seconds

**Figure 6. Screenshot of TheoSea’s output showing the Navier-Stokes equations and a vector identity. The dynamic viscosity constant is also found to be $\mu \approx 0.08$.**

- Bigger data and parallelism. The data set used was very small but semantically very rich. Other data sets would be much larger, such as for macroeconomics. Here, the on-the-fly compilation of Julia (of candidate theories) and support for parallel processing will be very beneficial, and this is one of the reasons why the language was chosen.

- Develop methods to deal with more noisy data. The current approach works with numerical noise but not measurement noise (Eqn. 1).

- The fully general Maxwell Equations can be rediscovered with the addition of a current $J$ and source region $\rho$. The changes to the virtual experiment and language $L$ are straightforward.

- Field discovery. The fields $E,B$ are treated as observables. It would be nice if TheoSea could discover the fields from the forces, e.g. $F = qE$. One step is to use a relativistic moving charge $q$ with velocity $u$, where the magnetic field can be written in terms of the electric field [15]: $B = (1/c^2) u \times E$. Then the field discovery problem reduces to finding the electric field and then the magnetic field will fall out from the search [16].

There may also be applications in the social sciences, such as macro-economics. Work is progressing in these areas, and focusing on the applicable representation language and executable semantics are the keys for new domains. The ultimate goal, however, is to generalize TheoSea: if a language and validator are supplied, is there a theory? The ultimate goal, however, is to generalize TheoSea: if a language and validator are supplied, is there a theory? Hamming stated [17] that:

*Einstein* knew in advance what the theory should look like, and he explored the theories with mathematical tools, not actual experiments.

The authors would like to propose an alternative: you will find the truth if you know the representation language of a theory and conduct a validated quest. It may be quicker than you thought.

**MATERIALS**

The Julia code is attached. The code is distributed under a Creative Commons Attribution 4.0 International Public License. It reproduces the screen results and timings of Figs. 4 and 6. The codes are simple to run, and contain the necessary instructions.
ACKNOWLEDGEMENTS
This research is funded by the Gordon and Betty Moore Foundation through Grant GBMF4915 to the Caltech Center for Data-Driven Discovery. Discussions with Mr. William Xu of Caltech Math/Computer Science were very helpful. The authors are grateful to Prof. S.G. Djorgovski of Caltech Astronomy and Prof. V. Chandler of KGI Natural Sciences for their support.

REFERENCES