Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

PyMaxEnt: A Python software for maximum entropy moment reconstruction

Tony Saad[1,*], Giovanna Ruai

Department of Chemical Engineering, University of Utah Salt Lake City, UT 84102, United States of America

A R T I C L E I N F O

Article history:
Received 16 July 2019
Received in revised form 21 October 2019
Accepted 21 October 2019

Keywords:
Maximum entropy reconstruction
Inverse moment problem
Particle size distribution

A B S T R A C T

PyMaxEnt is a software that implements the principle of maximum entropy to reconstruct functional distributions given a finite number of known moments. The software supports both continuous and discrete reconstructions, and is very easy to use through a single function call. In this article, we set out to verify and validate the software against several tests ranging from the reconstruction of discrete probability distributions for biased dice all the way to multimodal Gaussian and beta distributions. Written in Python, PyMaxEnt provides a robust and easy-to-use implementation for the community.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_241
Code Ocean compute capsule	NA
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	Python 3.x
Compilation requirements, operating environments & dependencies	Standard Python 3 installation with numpy and scipy
If available Link to developer documentation/manual	http://github.com/saadgroup/pymaxent/
Support email for questions	tony.saad@chemeng.utah.edu

1. Motivation and significance

Data analysis from experimental and numerical observations often requires post-processing to reconstruct primitive physical quantities [1]. For example, neutron flux distributions in nuclear reactors are not readily obtainable but have to be reconstructed from neutron moments measured numerically or experimentally [2,3]. This reconstruction of primitive quantities is usually hampered by noisy results and insufficient information rendering the reconstruction problem under-determined [1]. A classic example of this situation is the inverse-moment problem [4,5] stated as follows: *Given a finite number of known moments – e.g. averaged properties – for a given observation, find a unique distribution that is responsible for generating these moments*. For instance, in the simulation of particulate systems, one often solves for the average size and higher order moments of a particle density distribution [6]. Given these moments, the goal is then to reconstruct the unknown particle density distribution.

The inverse-moment problem is, by definition, under-determined since there is an infinite number of real-valued functions that could produce the desired moments. To obtain a unique distribution, one needs a set of constraints to close the system of equations. One such set of constraints is obtained by invoking the principle of maximum entropy [5,7].

* Corresponding author.
E-mail address: tony.saad@chemeng.utah.edu (T. Saad).

1 Assistant Professor.

1.1. Maximum entropy

The maximum entropy method is based on the concept that *the distribution that maximizes the information entropy is the one that is statistically most likely to occur* [5]. Information Entropy is defined as the average rate at which information is produced by a system. Information Entropy can alternatively be defined as a measure for how much of a system is unknown. The higher the information entropy is, the less we know about a process. When all known information has been taken into consideration, a system with maximum information entropy is the most probable state because it is the system in which the least amount of information has been defined. Mathematically, the information entropy S, of a distribution $p(x)$, is given by the integral

$$S = -\int_{\Omega} p(x)\ln p(x)\mathrm{d}x, \tag{1}$$

where Ω is the support of the distribution.

Given a finite number of moments for $p(x)$, one looks for a distribution $p(x)$ that maximizes S subject to these known moments. This can be formulated as a variational problem using Lagrangian multipliers. Our purpose is to find $p(x)$ that maximizes the information entropy S given in (1) subject to

$$\int_{\Omega} x^k p(x)\mathrm{d}x = \mu_k; \quad k = 0, 1, \ldots, N, \tag{2}$$

where $(N+1)$ is the number of known moments. Note that μ_k is known for $0 \le k \le N$. Then, by introducing Lagrangian multipliers λ_k, we define the entropy functional

$$H \equiv S + \sum \lambda_k \left(\int_{\Omega} x^k p(x)\mathrm{d}x - \mu_k\right), \tag{3}$$

or, more concisely,

$$H \equiv \int_{\Omega}\left[-p(x)\ln p(x) + \sum_{k=0}^{N}\lambda_k x^k p(x)\right]\mathrm{d}x - \sum_{k=0}^{N}\lambda_k\mu_k. \tag{4}$$

This functional is a maximum when the functional derivatives of H with respect to $p(x)$ and λ_k are zero: $\frac{\delta H}{\delta \lambda_k} = 0$ and $\frac{\delta H}{\delta p(x)} = 0$. The first of these derivatives does not provide us with any additional information since it returns the constraints defined in (2). The second of these derivatives, however, evaluates to

$$\ln p(x) = -1 + \sum_{k=0}^{N}\lambda_k x^k. \tag{5}$$

The general solution of (5) is

$$p(x) = e^{-1+\sum_{k=0}^{N}\lambda_k x^k} = e^{\sum_{k=0}^{N}\lambda_k x^k}, \tag{6}$$

where we have set $(\lambda_0 - 1) \to \lambda_0$.

1.2. Solution

To find the maximum entropy solution, we have to solve for the Lagrangian multipliers λ_k. For this, we must solve the following nonlinear system of equations

$$\begin{aligned} &\textstyle\int_{\Omega} e^{\lambda_0+\lambda_1 x+\cdots+\lambda_k x^k}\mathrm{d}x = \mu_0,\\ &\textstyle\int_{\Omega} x\, e^{\lambda_0+\lambda_1 x+\cdots+\lambda_k x^k}\mathrm{d}x = \mu_1,\\ &\qquad\vdots\\ &\textstyle\int_{\Omega} x^k e^{\lambda_0+\lambda_1 x+\cdots+\lambda_k x^k}\mathrm{d}x = \mu_k. \end{aligned} \tag{7}$$

A globally convergent Newton solver may be used to calculate λ_k.

The Jacobian can also be easily calculated. If we denote the moments based on the maximum entropy solution as

$$\widetilde{\mu}_k \equiv \int_{\Omega} x^k e^{\lambda_0+\lambda_1 x+\cdots+\lambda_k x^k}\mathrm{d}x, \tag{8}$$

then, the Jacobian is given by

$$\mathbf{J} \equiv \begin{bmatrix} \frac{\partial\widetilde{\mu}_0}{\partial\lambda_0} & \frac{\partial\widetilde{\mu}_0}{\partial\lambda_1} & \ldots & \frac{\partial\widetilde{\mu}_0}{\partial\lambda_k}\\ \frac{\partial\widetilde{\mu}_1}{\partial\lambda_0} & \frac{\partial\widetilde{\mu}_1}{\partial\lambda_1} & \ldots & \frac{\partial\widetilde{\mu}_1}{\partial\lambda_k}\\ \frac{\partial\widetilde{\mu}_2}{\partial\lambda_0} & \frac{\partial\widetilde{\mu}_2}{\partial\lambda_1} & \ldots & \frac{\partial\widetilde{\mu}_2}{\partial\lambda_k}\\ \vdots & \vdots & \vdots & \vdots\\ \frac{\partial\widetilde{\mu}_k}{\partial\lambda_0} & \frac{\partial\widetilde{\mu}_k}{\partial\lambda_1} & \ldots & \frac{\partial\widetilde{\mu}_k}{\partial\lambda_k} \end{bmatrix} = \begin{bmatrix} \widetilde{\mu}_0 & \widetilde{\mu}_1 & \cdots & \widetilde{\mu}_k\\ \widetilde{\mu}_1 & \widetilde{\mu}_2 & \cdots & \widetilde{\mu}_{k+1}\\ \widetilde{\mu}_2 & \widetilde{\mu}_3 & \cdots & \widetilde{\mu}_{k+2}\\ \vdots & \vdots & \vdots & \vdots\\ \widetilde{\mu}_k & \widetilde{\mu}_{k+1} & \cdots & \widetilde{\mu}_{k+k} \end{bmatrix}. \tag{9}$$

Note that $\tilde{\mu}_k$ is the kth moment based on the reconstructed distribution. The Newton solver will be based on finding λ_k such that $||\tilde{\mu}_k - \mu_k||$ is below a certain specified tolerance.

1.3. Initial guesses

The Newton method can be sensitive to initial guesses. It was found that convergence can be achieved in all of our experiments if the following initial guesses are used

$$\lambda_i^{\text{initial}} = \begin{cases} -\ln\sqrt{2\pi} & i = 0,\\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

These guesses are based on a Gaussian distribution with zero mean and unit variance ($\mu = 0$, $\sigma = 1$).

1.4. The discrete case

A similar analysis can be applied in the discrete case. For a set of discrete data points x_i We first define the discrete information entropy as

$$S = -\sum_i p_i \ln p_i, \tag{11}$$

with the kth moment given by

$$\mu_k = \sum_i x_i^k p_i. \tag{12}$$

Similar to the continuous case, we define the entropy functional as

$$\begin{aligned} H &\equiv S + \sum_k \lambda_k \left(\sum_i x_i^k p_i - \mu_k\right)\\ &= -\sum_i p_i \ln p_i + \sum_{k=0}^{N}\lambda_k\left(\sum_i x_i^k p_i - \mu_k\right), \end{aligned} \tag{13}$$

or, more concisely,

$$H \equiv \sum_i\left(\sum_{k=0}^{N}\lambda_k x_i^k p_i - p_i\ln p_i\right) - \sum_{k=0}^{N}\lambda_k\mu_k. \tag{14}$$

The functional is an extremum (can be shown to be a minimum) if the functional derivatives with respect to p_i and λ_k are zero: $\frac{\delta H}{\delta \lambda_k} = 0$ and $\frac{\delta H}{\delta p_i} = 0$ The first of these equations returns the moment constraint given by (12). The second functional derivative returns

$$\frac{\delta H}{\delta p_i} = \sum_{k=0}^{N}\lambda_k x_i^k - 1 - \ln p_i = 0, \tag{15}$$

whose solution for p_i is straightforward

$$p_i = e^{\sum_{k=0}^{N}\lambda_k x_i^k - 1} = e^{\sum_{k=0}^{N}\lambda_k x_i^k}, \tag{16}$$

where we set $\lambda_0 - 1 \to \lambda_0$. Similar to the continuous case, a globally convergent Newton solver can be used to solve for the Lagrangian multipliers λ_k.

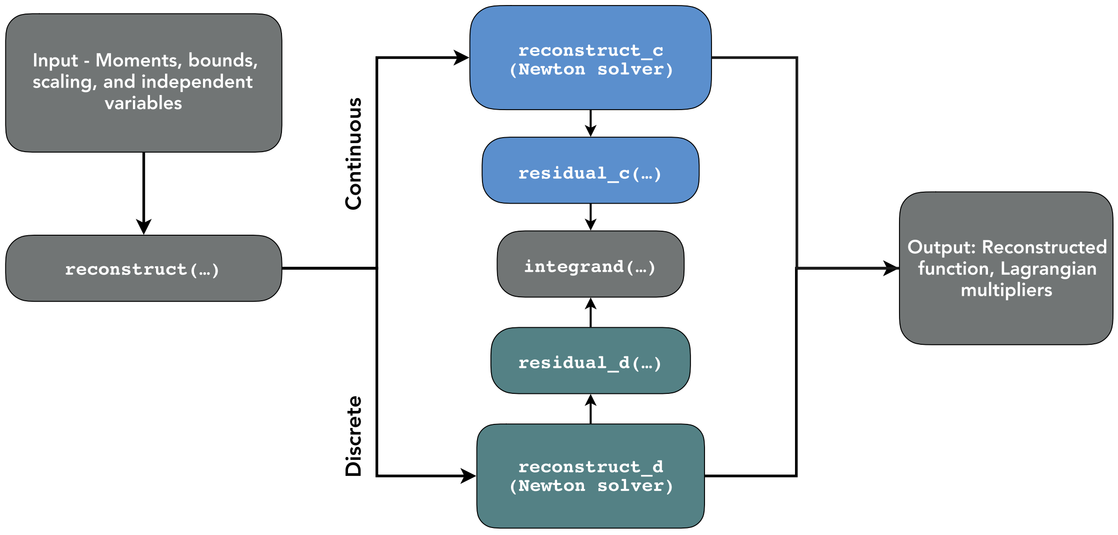
Input - Moments, bounds, scaling, and independent variables
reconstruct(…)
Continuous
Discrete
reconstruct_c (Newton solver)
residual_c(…)
integrand(…)
residual_d(…)
reconstruct_d (Newton solver)
Output: Reconstructed function, Lagrangian multipliers

Fig. 1. Overview of `PyMaxEnt` functionality.

2. Software description

The software is written in Python due to its popularity and ease of use in addition to the availability of a robust multidimensional nonlinear solver through `SciPy`. The `PyMaxEnt` software offers a single interface to both the continuous and discrete maximum entropy reconstructions along with a few useful functions described below.

2.1. Software architecture

The implementation of `PyMaxEnt` is very simple and takes form in a single Python file called `pymaxent.py`. For proper operation, three separate helper routines are implemented for the continuous and for the discrete case. These are a numerical integrator, a residual error calculator, and a Newton solver as shown in Fig. 1. The Newton solver is based on `SciPy`'s multidimensional root finding routine, `fsolve`. From the user's perspective, a single function call is made to the main routine, `reconstruct` (discussed below).

2.2. Software functionalities

To use `PyMaxEnt`, a single call to the function `reconstruct` is made. The function signature is

```
sol, lambdas = reconstruct(moments, rndvar, bnds)
```

Here, `moments` is a required list or array of known moments, `rndvar` is an optional argument containing discrete values of the random variable, and `bnds` is a tuple `[a,b]` containing the expected bounds of the resulting distribution. When `rndvar` is provided, the reconstruction assumes a discrete distribution. The code returns two quantities: (1) the functional form of the solution, i.e. (6), which can be plotted and, (2) a `numpy` array containing the Lagrangian multipliers. In the discrete case, the functional solution is simply a `numpy` array of values containing the function values corresponding to (16). Finally, `PyMaxEnt` provides a helper routine named `moments` that calculates the first k moments of a function and is useful for verification purposes.

2.3. Sample code snippets analysis

Below are some examples of using the `PyMaxEnt` software. Starting with an example to reconstruct a basic discrete distribution with two moments and six possible values for a random variable, x,

```python
from pymaxent import *
mu = [1,3.5]
x = [1,2,3,4,5,6]
sol, lambdas = reconstruct(mu,rndvar=x)
```

Similarly, for a continuous distribution, one passes a list of input moments. In this case, however, one must specify the bounds (`bnds`) to indicate that this is a continuous reconstruction. Here's an example for a Gaussian distribution

```python
from pymaxent import *
mu = [1,0,0.04]
sol, lambdas = reconstruct(mu,bnds=[-1,1])
# plot the reconstructed solution
x = np.linspace(-1,1)
plot(x,sol(x))
```

3. Illustrative examples

To test and validate `PyMaxEnt`, we set out to examine the reconstruction of a few probability distributions, spanning discrete to continuous complex distributions with infinite support.

3.1. Verfication

3.1.1. Balanced die

For a balanced die with face values ranging from 1 to 6, the probability of landing one side is 1/6. The expected value is therefore $\mu_1 = \sum(p_i x_i) = \frac{1}{6}\sum(x_i) = 3.5$. To run this test in `PyMaxEnt`, we simply set the following

```python
from pymaxent import reconstruct
mu = [1,3.5] # set the moments
# specify all possible values for the random variable
x = [1,2,3,4,5,6]
sol, lambdas = reconstruct(mu=mu,rndvar=x)
print(sol)
```

The value of sol returned by the software is

```
[0.166667 0.166667 0.166667 0.166667 0.166667 0.166667]
```

which is the expected uniform probability distribution of a balanced die.

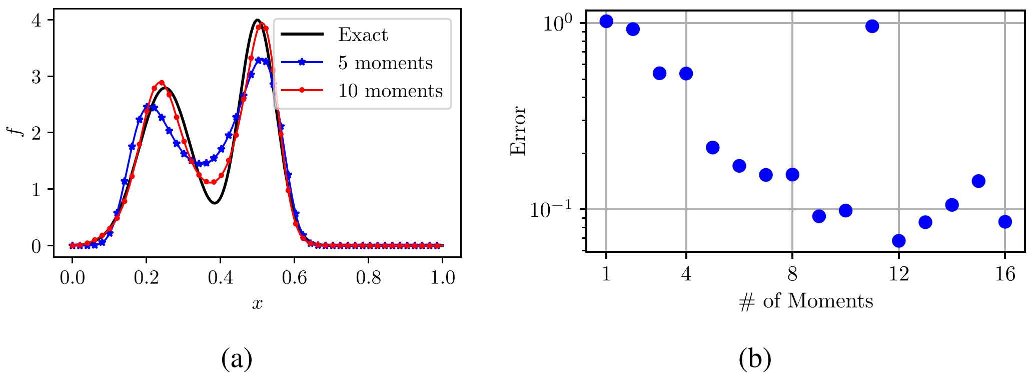

Fig. 2. Maximum entropy reconstruction of a bimodal Gaussian distribution with $\mu_0 = 0.25$, $\sigma_0 = 1/14$, $\mu_1 = 0.5$, and $\sigma_1 = 1/20$. (a) example reconstruction and (b) area error in reconstruction as a function of the number of input moments.

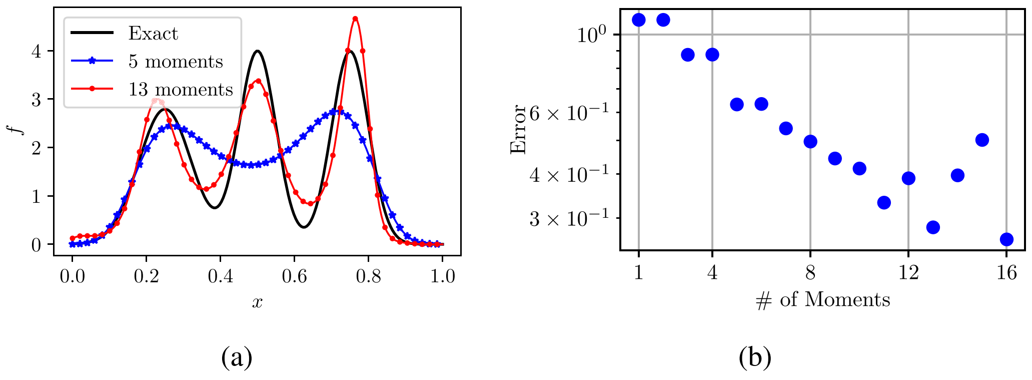

Fig. 3. Maximum entropy reconstruction for a trimodal Gaussian distribution $\mu_0 = 1/4$, $\mu_1 = 2/4$, $\mu_2 = 3/4$, $\sigma_0 = 1/14$, and $\sigma_1 = \sigma_2 = 1/20$. (a) example reconstruction and (b) area error in reconstruction as a function of the number of input moments.

3.1.2. Biased die

Consider a biased die with the following probabilities: $p(1) = p(2) = p(3) = 1/12$, $p(4) = 2/12$, $p(5) = 3/12$ and $p(6) = 4/12$. The expected value is $\mu_1 = 4.417$. To run this test we use

```
from pymaxent import reconstruct
mu = [1,4.417] # set the moments
# specify all possible values for the random variable
x = [1,2,3,4,5,6]
sol, lambdas = reconstruct(mu=mu,rndvar=x)
print(sol)
```

The value of `sol` returned by the software is

```
[0.0614560 0.0859580 0.120229 0.168163 0.235209 0.328985]
```

which corresponds to a non-uniform probability distribution for a biased die, with higher likelihood for the die to land on smaller values. In other words, the probability of landing on the number 1 is given by the reconstructed solution $p(1) = 0.0614560$, while the probability of landing on 2 is given by $p(2) = 0.0859580$, etc...

Of course there is an error in the computed probabilities as expected since the nonlinear solver acts on the moments of the distribution rather than the distribution itself. For example, in this case, the relative error for the moments is $\frac{\mu_{0,\text{exact}}-\mu_{0,\text{computed}}}{\mu_{0,\text{exact}}} \approx 10^{-13}$ and $\frac{\mu_{1,\text{exact}}-\mu_{1,\text{computed}}}{\mu_{1,\text{exact}}} \approx 10^{-7}$.

3.2. Predictivity and errors

To test the predictivity of `PyMaxEnt`, we picked a few non-trivial distributions, generated moments for those distributions, and then looked at the error in the *shape* of the reconstructed distribution versus the chosen distribution as we varied the number of moments input into the software. The error in the shape between the reconstructed distribution and the exact one is computed based on the area of the difference between the two curves as discussed in [8]. This difference can then be normalized by the area of the input curve (which is identically 1 in all cases here for probability density functions). Note that for all cases, the error in the computed moments versus the exact moments is at most 1.49×10^{-8} which is the default tolerance for the Newton solver.

3.2.1. Multimodal Gaussian distributions

We consider the bimodal Gaussian distribution given by

$$f(x) = \frac{1}{2\sigma_0\sqrt{2\pi}}\mathrm{Exp}\left[-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right] + \frac{1}{2\sigma_1\sqrt{2\pi}}\mathrm{Exp}\left[-\frac{(x-\mu_1)^2}{2{\sigma_1}^2}\right]. \tag{17}$$

Fig. 2a shows a comparison between a bimodal Gaussian with peaks of different heights and the maximum entropy reconstruction using 5 and 10 moments, respectively. Evidently, the larger the number of known moments, the more precise the reconstruction. In Fig. 2b, the area error committed in the reconstruction, versus an increasing number of moments is shown. A clear decrease in the error is visible up to 10 moments where strange behavior in the error is visible. This is due to the fact that at such large number of moments, roundoff errors dominate in the numerical integration and nonlinear solver. The nature of the reconstruction contains an exponential of a polynomial which in turn leads to an expected increase in roundoff errors.

For a trimodal Gaussian distribution, we use

$$f(x) = \frac{1}{3\sigma_0\sqrt{2\pi}}e^{-\frac{(x-\mu_0)^2}{2\sigma_0^2}} + \frac{1}{3\sigma_1\sqrt{2\pi}}e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + \frac{1}{3\sigma_2\sqrt{2\pi}}e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}. \tag{18}$$

The reconstructed distribution is shown in Fig. 3a where 5 and 13 moments are used to reconstruct the exact distribution. In this

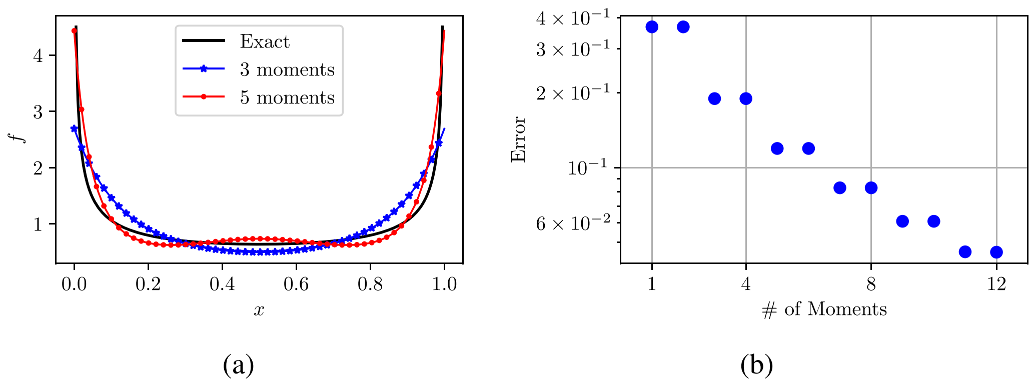

Fig. 4. Maximum entropy reconstruction for a beta distribution with $\alpha = 3$ and $\beta = 9$. (a) example reconstruction and (b) area error in reconstruction as a function of the number of input moments.

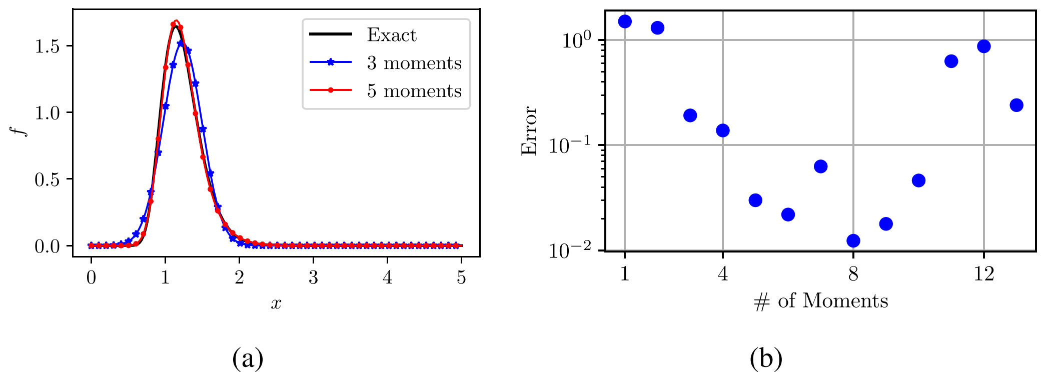

Fig. 5. Maximum entropy reconstruction for a log-normal distribution with $\mu = 0.2$ and $\sigma = 0.25$. (a) example reconstruction and (b) area error in reconstruction as a function of the number of input moments.

case, one needs higher order moments to capture the changes in convexity of this distribution.

Fig. 3b shows the error in the reconstruction as we increase the number of moments. Again, we see roundoff errors dominate at the 12th moment.

3.2.2. *Beta distribution*

The beta distribution can be a challenging distribution to reconstruct. It is given by

$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \tag{19}$$

where Γ is the generalized Gamma function and α and β are control parameters that dictate the shape of the beta distribution. Results for this case are shown in Fig. 4. The maximum entropy reconstruction is able to capture the key features of this distribution especially for the case with singularities at the domain boundaries.

3.3. *Distributions with semi-infinite support*

One could run into trouble with the numerical integrator when integrating over an infinite or a semi-infinite support. This behavior is observed especially with distributions that slowly decay towards infinity such as a log-normal distribution. However, if integration is carried out on a finite support, one is able to still recover a reasonable approximation to the distribution. As an example of this, a log-normal distribution was used with numerical integration performed over the interval [0, 10]. The resulting reconstruction is shown in Fig. 5.

4. Impact

The implementation presented in this paper has been used in reconstructing neutron flux distributions in nuclear reactors [2,3]. In addition, it was used in unpublished work by colleagues and collaborators at Brigham Young University, and at the University of Utah for the reconstruction of particle size distributions in carbon dioxide mineralization reactors [6] and in coal combustion simulations.

New research questions can be pursued in the context of high-performance computing for reacting particulate transport such as coal-combustion, gasification, fluidized beds, and nuclear reactors. Transporting individual particles (or clouds of particles) is a very costly proposition and one often resorts to methods that solve for the moments of these particle distributions. The ability to reconstruct the parent particle size distributions enables detailed insight into these systems such as understanding how coal particles move in coal combustors based on their size and composition.

This code will also be used in a new project that our group is working on which involves the modeling of hazardous particulate matter (PM2.5) in the atmosphere and how to accurately measure such particles using drones.

To the best of our knowledge, we are unaware of publicly available software that is similar to `PyMaxEnt`. We believe that making `PyMaxEnt` available to the community is a step in the right direction.

5. Conclusions

In this article, we set out to demonstrate the use of a Python software for reconstructing density distributions from given moments using the principle of maximum entropy. The software, `PyMaxEnt`, was shown to reconstruct discrete and continuous distributions with finite, semi-infinite, and infinite support. Care must be taken when a large number of moments is used due to the presence of round-off errors. The Newton solver was shown to be robust across all tests without changing the initial guess for the root finding algorithm. If necessary, users can very easily change the initial guesses of the nonlinear solver. Future work will include support for higher precision floating point numbers to reduce the amount of round-off errors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors greatly acknowledge support from the Undergraduate Research Office Program (UROP) and the department of Chemical Engineering at the University of Utah.

References

[1] von der Linden W. Maximum-entropy data analysis. Appl Phys A 1995;60(2):155–65.
[2] Crawford Douglas S, Saad Tony, Ring Terry A. Verification and validation of the maximum entropy method for reconstructing neutron flux, with MCNP5, Attila-7.1.0 and the GODIVA experiment. Ann Nucl Energy 2013;53:188–91.
[3] Crawford Douglas Stewart, Ring Terry Arthur. Verification of analytic energy moments for the one-dimensional energy dependent neutron diffusion equation with MCNP5 and Attila-7.1.0. 2012.
[4] Shohat James Alexander, Tamarkin Jacob David. The problem of moments, Vol. 1. American Mathematical Soc.; 1943.
[5] Mead Lawrence R, Papanicolaou Nikos. Maximum entropy in the problem of moments. J Math Phys 1984;25(8):2404–17.
[6] Abboud Alex W, Schroeder Ben B, Saad Tony, Smith Sean T, Harris Derek D, Lignell David O. A numerical comparison of precipitating turbulent flows between large-eddy simulation and one-dimensional turbulence. AIChE J 2015;61(10):3185–97.
[7] Bandyopadhyay K, Bhattacharya Arun K, Biswas Parthapratim, Drabold DA. Maximum entropy and the problem of moments: A stable algorithm. Phys Rev E 2005;71(5):057701.
[8] Jekel Charles F, Venter Gerhard, Venter Martin P, Stander Nielen, Haftka Raphael T. Similarity measures for identifying material parameters from hysteresis loops using inverse analysis. Int J Mater Form 2019;12(3):355–78.