

Acta Crystallographica Section A

**Foundations of
Crystallography**

ISSN 0108-7673

A fast Newton method for entropy maximization in statistical phase estimation

Zhijun Wu, George N. Phillips Jr, Richard Tapia and Yin Zhang

Copyright © International Union of Crystallography

Author(s) of this paper may load this reprint on their own web site provided that this cover page is retained. Republication of this article or its storage in electronic databases or the like is not permitted without prior permission in writing from the IUCr.

A fast Newton method for entropy maximization in statistical phase estimation

Zhijun Wu,^{a*} George N. Phillips Jr,^b Richard Tapia^c and Yin Zhang^c

^aDepartment of Mathematics, Iowa State University, Ames, IA 50010, USA, ^bDepartment of Biochemistry, University of Wisconsin–Madison, Madison, WI 53706, USA, and ^cDepartment of Computational and Applied Mathematics, Rice University, Houston, TX, 77005, USA. Correspondence e-mail: zhijun@iastate.edu

A fast Newton method is presented for solving the entropy maximization problem in the Bayesian statistical approach to phase estimation. The method requires only $O(n \log n)$ instead of standard $O(n^3)$ floating point operations per iteration, while converging in the same rate as the standard Newton method. The method is described and related computational issues are discussed. Numerical results on simple test cases are also presented to demonstrate the behavior of the method.

© 2001 International Union of Crystallography
Printed in Great Britain – all rights reserved

1. Introduction

The phase problem in X-ray crystallography is often approached using statistical methods. In the Bayesian approach (Bricogne, 1984, 1988, 1991, 1993, 1997) to the phase problem, the joint probability distribution of the structure factors is required for every basis set of the structure factors. It is computed by maximizing the entropy of the crystal system subject to certain constraints on the structure factors (Jaynes, 1957, 1968; Bricogne, 1984). The entropy-maximization problem is an infinite-dimensional convex programming problem but can be solved in a finite dual space by using a standard Newton method (Alhassid *et al.*, 1978; Dennis & Schnabel, 1983; Fletcher, 1987).

The Newton method converges fast but is costly; in general, requiring $O(n^3)$ floating point operations per iteration, where n is the number of variables. For entropy maximization, n can be as large as several tens of thousands. Furthermore, the problem needs to be solved repeatedly. Therefore, the standard Newton method is too expensive to be used in practice.

Alternative approaches have been proposed (Bricogne, 1984, 1993; Prince, 1989; Bricogne & Gilmore, 1990). Although cheaper per iteration, they lost the fast convergence property of the Newton method. In other words, the methods may need many more iterations to converge.

In this paper, we present a fast Newton method for solving the entropy-maximization problem. The method is equivalent to the standard Newton method in the sense that it generates the same iterates and hence has the same convergence rate as the standard Newton. On the other hand, the cost per iteration is reduced from $O(n^3)$ to $O(n \log n)$ floating point operations.

More specifically, we consider the entropy maximization problem in its dual form as first suggested by Alhassid *et al.* (1978). We apply a so-called Sherman–Morrison–Woodbury formula to the inverse of the Hessian of the objective function.

The inverse can then be computed in terms of the inverse of a Karle–Hauptman matrix (Karle & Hauptman, 1952). The inverse of the Karle–Hauptman matrix can be computed by using a fast Fourier transform, which requires only $O(n \log n)$ floating point operations (Van Loan, 1992; Tolimieri *et al.*, 1997). Then, the total cost in computing a Newton step can be reduced to $O(n \log n)$ floating point operations.

The paper is organized as follows. We describe the entropy-maximization problem and its dual form in §2. We discuss previous approaches to the problem in §3, and present our method and related convergence and complexity results in §4. We present preliminary computational results in §5, and conclude the paper in §6.

2. Entropy maximization

Consider a crystal system with a unit-cell space \mathbf{V} . Let ρ be a density distribution function that describes the electron-density distribution of the system. Let m be the uniform distribution function. We consider the entropy of ρ relative to m ,

$$S_m(\rho) = - \int_{\mathbf{V}} \rho(r) \log[\rho(r)/m(r)] dr.$$

Let $\{H_j : j = 1, \dots, n\}$ be a set of integer vectors in the reciprocal space of the crystal. We assume that the H_j are not equal to zero. We are interested in an electron-density distribution ρ such that the entropy of ρ relative to m is maximized under the condition that the structure factors F_{H_j} of ρ must be equal to a given set of values $F_{H_j}^*$, $j = 1, \dots, n$. This problem can be formulated as a constrained maximization problem,

$$\begin{aligned} \max_{\rho} \quad & \mathcal{S}_m(\rho) \\ \text{s.t.} \quad & F_{H_j} = \int_{\mathbf{v}} \rho(r) C_{H_j}(r) dr = F_{H_j}^*, \quad j = 1, \dots, n \\ & \int_{\mathbf{v}} \rho(r) dr = 1, \end{aligned}$$

where

$$C_{H_j}(r) = \exp(2\pi i H_j^T r), \quad j = 1, \dots, n.$$

Note that the objective function of the problem is concave and the constraint functions are linear. The entropy-maximization problem therefore belongs to a special class of optimization problems called convex programming problems. A convex programming problem has nice properties. For example, a local solution to a convex programming problem is also global, and it can be obtained by solving an equivalent dual problem.

By following a routine procedure, we can obtain a dual problem for the entropy maximization problem in the following form:

$$\min_{\lambda_{H_1}, \dots, \lambda_{H_n}} \mathcal{D}(\lambda_{H_1}, \dots, \lambda_{H_n}) = \log Z(\lambda_{H_1}, \dots, \lambda_{H_n}) - \sum_{j=1}^n F_{H_j}^* \lambda_{H_j},$$

where $\lambda_{H_1}, \dots, \lambda_{H_n}$ are complex variables called dual variables and

$$Z(\lambda_{H_1}, \dots, \lambda_{H_n}) = \int_{\mathbf{v}} m(r) \exp \left[\sum_{j=1}^n \lambda_{H_j} C_{H_j}(r) \right] dr.$$

We call the original entropy-maximization problem the primal problem and the variable ρ the primal variable. The dual problem is clearly simpler than the primal problem: It is an unconstrained optimization problem. The objective function is also defined in finite-dimensional space, while the primal problem is an infinite-dimensional problem since its variable ρ is actually a function.

The primal and dual variables have the following relation:

$$\rho(r) = \frac{m(r)}{Z(\lambda_{H_1}, \dots, \lambda_{H_n})} \exp \left[\sum_{j=1}^n \lambda_{H_j} C_{H_j}(r) \right].$$

Therefore, given any $\lambda_{H_1}, \dots, \lambda_{H_n}$, a corresponding density distribution function ρ can immediately be defined, and the entropy \mathcal{S}_m and the structure factors F_{H_j} of ρ can also be computed.

For any $\lambda_{H_1}, \dots, \lambda_{H_n}$, the gradient and the Hessian of the objective function \mathcal{D} can be computed by using the following formulas:

$$\nabla \mathcal{D} = F - F^*, \quad \nabla^2 \mathcal{D} = K - FF^H,$$

where $F = (F_{H_1}, \dots, F_{H_n})^T$, $F^* = (F_{H_1}^*, \dots, F_{H_n}^*)^T$ and K is a matrix, $K_{jk} = F_{H_j - H_k}$. We can show that $\nabla^2 \mathcal{D}$ is positive definite, therefore \mathcal{D} must be a strictly convex function. For detailed mathematical proofs, readers are referred to Wu *et al.* (2001).

3. Previous approaches

We now consider the solution of the dual minimization problem. The objective function of the problem is \mathcal{D} . The variables are $\lambda_{H_1}, \dots, \lambda_{H_n}$. We want to find a set of $\lambda_{H_1}, \dots, \lambda_{H_n}$ so that $\mathcal{D}(\lambda_{H_1}, \dots, \lambda_{H_n})$ is minimized. Since \mathcal{D} is strictly convex, it follows that the dual problem can be solved by using a standard Newton method. In this method, we first choose an initial point $\lambda^{(0)} = (\lambda_{H_1}^{(0)}, \dots, \lambda_{H_n}^{(0)})^T$. We then compute a sequence of points by the following iteration:

$$\lambda^{(l+1)} = \lambda^{(l)} - [\nabla^2 \mathcal{D}(\lambda^{(l)})]^{-1} \nabla \mathcal{D}(\lambda^{(l)}),$$

where $\lambda^{(l)} = (\lambda_{H_1}^{(l)}, \dots, \lambda_{H_n}^{(l)})^T$, $l \geq 0$. The sequence will converge to a point where \mathcal{D} is minimized.

The Newton method converges to the solution quickly, typically in a few iterations. However, it is very costly since in each iteration it usually requires $\mathcal{O}(n^3)$ floating point operations to compute the inverse of the Hessian multiplied by the gradient of the objective function.

In X-ray crystallography applications, the entropy-maximization problem can be very large: When transformed to the dual problem, n , the number of variables, can be as large as several tens of thousands. Therefore, a straightforward implementation of Newton's method will not be practical, especially when the problem needs to be solved many times, as required in the Bayesian statistical approach to phase determination. In order to reduce the cost of Newton's method, Bricogne (1984) suggested an approximation of the Hessian of \mathcal{D} so that the inverse of the approximated Hessian can be computed in a cheaper way. As we have shown before, the Hessian of \mathcal{D} is equal to a Karle–Hauptman matrix K minus a matrix FF^H for some vector F . Therefore, Bricogne (1984) suggested using this K matrix as an approximation to the Hessian of \mathcal{D} , since the inverse of K can be computed by using a fast Fourier transform, which costs only $\mathcal{O}(n \log n)$ floating point operations. However, after the approximation, the minimization method is no longer formally the Newton method, and fast convergence of the method is no longer guaranteed. Bricogne (1993) showed reasonably efficient solutions obtained by the approximation method on some test problems but, in general, the method may take more iterations to converge and the total cost can still be quite large. Some other approaches have also been proposed such as using the BFGS method (Prince, 1989). The BFGS method does not compute the inverse of the Hessian explicitly but the inverse actually is required in the Bayesian analysis following entropy maximization. Therefore, it is still an issue how the inverse of the Hessian can be computed with less than $\mathcal{O}(n^3)$ floating point operations.

4. A fast Newton method

We now present a fast Newton method for solving the entropy maximization problem. The method requires only $\mathcal{O}(n \log n)$ floating point operations for each of its iterates, yet has the same convergence rate as the standard Newton method.

1. Input initial $\lambda^{(0)}$. Set $l = 0$.

2. Repeat

(a) Compute

$$Z(\lambda^{(l)}) = \int_{\mathbf{V}} m(r) \exp \left[\sum_{j=1}^n \lambda_{H_j}^{(l)} C_{H_j}(r) \right] dr$$

$$\rho^{(l)}(r) = \frac{m(r)}{Z(\lambda^{(l)})} \exp \left[\sum_{j=1}^n \lambda_{H_j}^{(l)} C_{H_j}(r) \right]$$

(b) Compute, for $j = 1, \dots, n$,

$$F_{H_j}^{(l)} = \int_{\mathbf{V}} \rho^{(l)}(r) \exp(2\pi i H_j^T r) dr$$

(c) Set

$$\Delta^{(l)} = F^* - F^{(l)}$$

$$V^{(l)} = [K^{(l)}]^{-1} \Delta^{(l)}$$

$$t^{(l)} = [K^{(l)}]^{-1} F^{(l)}$$

(d) Compute

$$\Delta \lambda^{(l)} = V^{(l)} + \frac{[F^{(l)}]^H V^{(l)}}{1 - [F^{(l)}]^H t^{(l)}} t^{(l)}$$

$$\lambda^{(l+1)} = \lambda^{(l)} + \Delta \lambda^{(l)}$$

$$l = l + 1$$

(e) If the optimality condition is satisfied, go to 3.

3. Set $\lambda^* = \lambda^{(l)}$, $\rho^* = \rho^{(l)}$. Stop.

Figure 1

Outline of the fast Newton method.

First, consider the iteration of Newton's method we have discussed in the previous section. Note that the major computation cost in the iteration is the computation of the so-called Newton step,

$$-(\nabla^2 \mathcal{D})^{-1} \nabla \mathcal{D}.$$

Based on previous discussion, the gradient and the Hessian of \mathcal{D} can be computed in the following form:

$$\nabla \mathcal{D} = F - F^*, \quad \nabla^2 \mathcal{D} = K - FF^H.$$

Because of the special structure of $\nabla^2 \mathcal{D}$, it follows that the inverse of $\nabla^2 \mathcal{D}$ can be computed in terms of the inverse of K by using a so-called Sherman–Morrison–Woodbury formula.

The Sherman–Morrison–Woodbury formula usually applies to nonsingular matrices (see Sherman, 1978; Dennis & Schnabel, 1983; Fletcher, 1987). For our purpose, we give a more specific version of the formula for positive definite matrices in the following.

Let T and S be two Hermite matrices, U a vector and suppose that $T = S - UU^H$. Let S be nonsingular and $\sigma = U^H S^{-1} U$. Then, if S is positive definite and $\sigma < 1$, T is also positive definite and

$$T^{-1} = S^{-1} + S^{-1} U U^H S^{-1} / (1 - \sigma).$$

Table 1

Comparison with a gradient algorithm (number of iterations).

No. of factors	8	16	32	64	128	256	512
Gradient	65	68	68	67	65	65	64
Fast Newton	6	6	6	6	6	6	6

Table 2

Comparison with an approximation algorithm (number of iterations).

No. of factors	8	16	32	64	128	256	512
Approximation	18	16	16	14	14	14	14
Fast Newton	6	6	6	6	6	6	6

It is not difficult to show that K is positive definite and $F^H K^{-1} F < 1$. Therefore, we can apply the Sherman–Morrison–Woodbury formula to $\nabla^2 \mathcal{D}$ to obtain

$$(\nabla^2 \mathcal{D})^{-1} = K^{-1} + \frac{K^{-1} F F^H K^{-1}}{1 - F^H K^{-1} F}.$$

It follows that

$$-(\nabla^2 \mathcal{D})^{-1} \nabla \mathcal{D} = -(K - FF^H)^{-1} (F - F^*) = V + \frac{F^H V}{1 - F^H U} U,$$

where

$$V = K^{-1}(F^* - F), \quad U = K^{-1}F. \quad (1)$$

The inverse of K and its multiplication with a vector can all be computed by using a fast Fourier transform. Therefore, the total cost for computing the Newton step can now be reduced to $\mathcal{O}(n \log n)$ floating point operations. For a more formal discussion on these issues, readers are referred to Wu *et al.* (2001).

Fig. 1 contains an outline of the fast Newton method constructed using the above formulas. In the first step, an initial guess $\lambda^{(0)}$ is given and l is set to zero. Then step 2 is repeated. First, in step 2(a), given $\lambda_{H_1}^{(l)}, \dots, \lambda_{H_n}^{(l)}$, the corresponding $\rho^{(l)}$ and $Z(\lambda^{(l)})$ are computed; they can be computed together through an inverse Fourier transform, which requires only $\mathcal{O}(n \log n)$ floating point operations. In step 2(b), another Fourier transform is applied to $\rho^{(l)}$ to obtain the structure factors $F_{H_j}^{(l)}$ and the cost is again of the order of $n \log n$.

In step 2(c), two Fourier transforms are required for two matrix–vector products, $[K^{(l)}]^{-1} \Delta^{(l)}$ and $[K^{(l)}]^{-1} F^{(l)}$. Finally, in step 2(d), the Newton step is formed with the Sherman–Morrison–Woodbury formula and a new iterate $\lambda^{(l+1)}$ is obtained. The whole step 2(d) requires only vector–vector operations and costs $\mathcal{O}(n)$ floating point operations. In the end of the iteration, if the new iterate is optimal, the whole procedure stops; otherwise, it continues until the iteration converges. In any case, each iteration requires only $\mathcal{O}(n \log n)$ floating point operations.

5. Preliminary computational results

We have implemented the fast Newton method in *Matlab* and compared it with several other methods including a gradient method, a standard Newton method and a method with Hessian approximation. By Hessian approximation, we mean

that we use K instead of $K - FF^H$ as an approximation to the Hessian of the function \mathcal{D} .

We have tested the methods with a set of model problems generated in the following procedure. We first constructed a one-dimensional density distribution function. We then generated seven sets of structure factors from the function. The first set has 8 structure factors, the second 16, the third 32, the fourth 64, the fifth 128, the sixth 256 and the seventh 512. From each set of structure factors, we define an entropy-maximization problem with the corresponding structure factors as the constraints. We then obtain seven entropy-maximization problems.

We applied the methods to the model problems and recorded the number of iterations required for the method to converge to the solution to the problem. We also recorded the total number of floating point operations for each run.

Note that we used the *Matlab* routine for the fast Fourier transform required in the methods. We also used the *Matlab* routine for the linear system solution in the standard Newton method. In each iteration, the gradient method needs to compute the function and the gradient. Given $\lambda_{H_1}, \dots, \lambda_{H_n}$, the function and the gradient of \mathcal{D} can both be obtained by fast Fourier transform. Therefore, the cost of each iteration in the gradient method is $\mathcal{O}(n \log n)$. However, the gradient method converges only linearly and may take too many iterations to reach a solution.

When the Hessian is approximated by K , the inverse of the Hessian can be computed in $\mathcal{O}(n \log n)$. So the approximation method should take the same order of floating point operations in each iteration as the fast Newton method. However, because of the approximation, the method is no longer a Newton method and, therefore, the fast convergence rate of Newton's method will be lost.

Tables 1, 2 and 3 show the performance of the methods in terms of the number of iterations or the total number of floating point operations. Table 1 shows the numbers of iterations required for the gradient method. Clearly, it took many more iterations to converge than the fast Newton method, which for all the test problems took only 6 iterations.

Table 2 shows the number of iterations required by the approximation method. For all the problem instances, the method took two to three times more iterations than the fast Newton method. Since the test problems we have constructed are relatively simple and easy, we expect the difference between the two methods to be bigger in practice when the problem is larger and more complicated. In any case, there is no theory to guarantee that the approximation method will converge fast.

Table 3 shows the total numbers of floating point operations required by the standard and fast Newton methods. As we can see from the table, the standard Newton method required many more floating point operations than the fast Newton method. For example, for the test problem with 128 structure factors, the fast Newton method required about 200 000 floating point operations, while the standard Newton method

Table 3

Comparison with a standard Newton algorithm (number of floating point operations).

No. of factors	8	16	32	64	128	256	512
Standard	12873	57869	308769	1912061	13192377	97403093	747295473
Fast Newton	8596	22347	48307	104211	224403	481971	1031891

needed more than 13×10^6 floating point operations. The fast Newton method was about 60 times faster. When the problem size becomes larger, we observed even bigger differences between the two methods.

6. Concluding remarks

In this paper, we report our studies on the entropy maximization problem in the Bayesian statistical approach to the phase problem in protein X-ray crystallography. Since the solution to the problem is required in every step of the Bayesian method, an efficient method for solving the problem is important especially for large-scale applications. Previous approaches used standard Newton or approximation methods. They were either costly, requiring $\mathcal{O}(n^3)$ computation time, or not able to guarantee fast convergence, where n is the number of structure factors of interest. We derived a formula to compute the inverse of the Hessian in $\mathcal{O}(n \log n)$ computation time, thereby reducing the time complexity of the Newton method. As a result, we should now be able to apply Newton's method to large-scale problems with both low computational cost and fast convergence rate.

We have described our computational experiments with the fast Newton method. The results from using the method for a set of simple test problems were compared with some other methods and show that the fast Newton method converged in fewer iterations than a typical gradient method and a method with Hessian approximation, although they all required the same order of floating point operations in each iteration. On the other hand, the fast Newton method required much less computation than the standard Newton method, although they both converged in the same rate. The results imply that the fast Newton method can be used to reduce the high cost of the standard Newton method, while converging as fast as the standard Newton and certainly faster than the gradient and Hessian-approximation methods. This makes it possible to solve large-scale entropy maximization problems in practice and to develop more efficient and reliable phase estimation procedures for structure determination.

Work supported in part by the Keck Center for Computational Biology and DOE/LANL Contract 03891-99-23 (ZW); in part by Welch Foundation Grant C-1142 and NSF RTG Grant BIR-94-13229 (GNP); in part by DOE DEFG05-86ER25017 and DOE/LANL Contract 03891-99-23 (RT); and in part by DOE Grant DE-FG03-97ER25331 and DOE/LANL Contract 03891-99-23 (YZ).

References

- Alhassid, Y., Agmon, N. & Levin, R. D. (1978). *Chem. Phys. Lett.* **53**, 22–26.
- Bricogne, G. (1984). *Acta Cryst.* **A40**, 410–445.
- Bricogne, G. (1988). *Acta Cryst.* **A44**, 517–545.
- Bricogne, G. (1991). *Acta Cryst.* **A47**, 803–829.
- Bricogne, G. (1993). *Acta Cryst.* **D49**, 37–60.
- Bricogne, G. (1997). *Methods Enzymol.* **276**, 361–423.
- Bricogne, G. & Gilmore, C. J. (1990). *Acta Cryst.* **A46**, 284–297.
- Dennis, J. E. Jr & Schnabel, R. B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hal.
- Fletcher, R. (1987). *Practical Methods of Optimization*. New York: Wiley.
- Jaynes, E. T. (1957). *Phys. Rev.* **106**, 620–630.
- Jaynes, E. T. (1968). *IEEE Trans.* **SSC-4**, 227–241.
- Karle, J. & Hauptman, H. (1952). *Acta Cryst.* **3**, 181–187.
- Prince, E. (1989). *Acta Cryst.* **A45**, 200–203.
- Sherman, A. H. (1978). *SIAM J. Num. Anal.* **15**, 755–771.
- Tolimieri, R., An, M. & Lu, C. (1997). *Algorithms for Discrete Fourier Transform and Convolution*. New York: Springer.
- Van Loan, C. (1992). *Computational Frameworks for the Fast Fourier Transform*. Philadelphia: SIAM.
- Wu, Z., Phillips, G. N. Jr, Tapia, R. & Yin, Z. (2001). *SIAM Rev.* In the press.