Acceleration of Iterative Image Restoration Algorithms

David S. C. Biggs and Mark Andrews *

A new technique for the acceleration of iterative image restoration algorithms is proposed. The method is based on the principles of vector extrapolation and does not require the minimization of a cost function. The algorithm is derived and its performance illustrated with Richardson–Lucy (R–L) and maximum entropy (ME) deconvolution algorithms and the Gerchberg–Saxton magnitude and phase retrieval algorithms. Considerable reduction in restoration times is achieved with little image distortion or computational overhead per iteration. The speedup achieved is shown to increase with the number of iterations performed and is easily adapted to suit different algorithms. An example R–L restoration achieves an average speedup of 40 times after 250 iterations and an ME method 20 times after only 50 iterations. An expression for estimating the acceleration factor is derived and confirmed experimentally. Comparisons with other acceleration techniques in the literature reveal significant improvements in speed and stability.

Key words: Image restoration, acceleration, deconvolution, iterative algorithms, vector extrapolation, Richardson–Lucy, maximum entropy, Gerchberg–Saxton.

1 Introduction

Iterative image restoration algorithms are popular methods for deconvolving images from the Hubble Space Telescope (HST), medical tomography, and satellite sensors. These methods are non–linear and offer more effective restoration than simple techniques, such as linear inverse filtering, which often fail [1]. Nonlinear techniques are useful when the data is noisy or incomplete, which is usually the case in practical applications.

Iterative techniques operate on the result of the previous iteration and are normally slow to converge toward the final result. They give better control and enhanced results compared with linear methods, but have considerable computational requirements. Iterative restoration is routinely used in the astronomical community, where the expense of computing resources and time is small when compared with that of collecting the data. This cost is restrictive in more down–to–earth applications where restoration of a single image could take several hundred iterations and many hours of processing to achieve.

This paper presents a new method for accelerating the convergence of iterative restoration algorithms, termed automatic acceleration. Acceleration of convergence means faster processing and allows iterative techniques to be used in applications where they would otherwise be deemed too slow to be practical.

The technique is derived by considering the general form of an iterative algorithm and then applying a type of vector extrapolation. Four iterative methods, the Richardson–Lucy (R–L) and maximum entropy (ME) deconvolution algorithms and the Gerchberg–Saxton (G–S) magnitude and phase retrieval algorithms, are used to illustrate the diverse range of iterative methods the acceleration technique can be applied to. There is a bias toward examples using the R–L algorithm because of the large amount of research already invested in accelerating this method.

*The authors are with the Department of Electrical and Electronic Engineering, University of Auckland, Private Bag 92019, Auckland, New Zealand.

Section 2 briefly describes the R–L algorithm and presents an overview of several acceleration techniques in the literature. The theory behind the acceleration algorithm is given and the method derived. An estimate of the acceleration factor at each iteration is proposed, and further enhancements to the technique are described, which use more information to increase acceleration. Section 3 illustrates the performance of the acceleration techniques when they are applied to the various algorithms. This section also discusses the results obtained when the techniques are applied in the presence of noise. Areas for future research are discussed before Section 4 concludes the paper. The appendixes provide the theory behind the proposed acceleration method.

2 Theory

2.1 The Richardson–Lucy Restoration Algorithm

The R–L algorithm [2, 3] is an iterative technique used heavily for the restoration of astronomical imagery in the presence of Poisson noise. It attempts to maximize the likelihood of the restored image by using the expectation–maximization algorithm [4]. The algorithm requires a good estimate of the process by which the image is degraded for accurate restoration. The degradation can be caused in many ways, such as subject movement, out–of–focus lenses, or atmospheric turbulence, and is described by the point spread function (PSF) of the system. The image is assumed to come from a Poisson process and, therefore, is corrupted by signal–dependent noise. There may also be electronic or quantization noise involved in obtaining the image.

The image degradation is modeled as

$$g = h \otimes f + n, \qquad (1)$$

where f is the original undistorted image, g is the distorted noisy image, h is the PSF of the system, $\otimes$ is the convolution operator, and n is the corrupting noise.

The iterative R–L algorithm may be succinctly expressed as

$$\hat{f}_{k+1} = \hat{f}_k \left(h * \frac{g}{h \otimes \hat{f}_k} \right) \triangleq \psi(\hat{f}_k), \qquad (2)$$

where $\hat{f}_k$ is the estimate of f after k iterations, $*$ is the correlation operator, and $\psi(\cdot)$ is called the R–L function. The image $h \otimes \hat{f}_k$ is referred to as the reblurred image.

An individual pixel normally changes most rapidly in the first few iterations and slower as the restoration continues. A pixel may show exponential convergence or more complicated behavior such as oscillations. Figure 1 shows the path of an individual pixel from a 10,000 iteration R–L restoration on an eight–bit image presented below [Fig. 4(d)]. The pixel in Fig. 1(a) illustrates the rapid changes in the first few iterations which slows as the iterations increase. The path of a pixel can also be seen to be continuous — noise and discontinuities are not visible. Figure 1b shows that the characteristic scale of the oscillations increases exponentially. These properties will be used below to develop the acceleration technique.

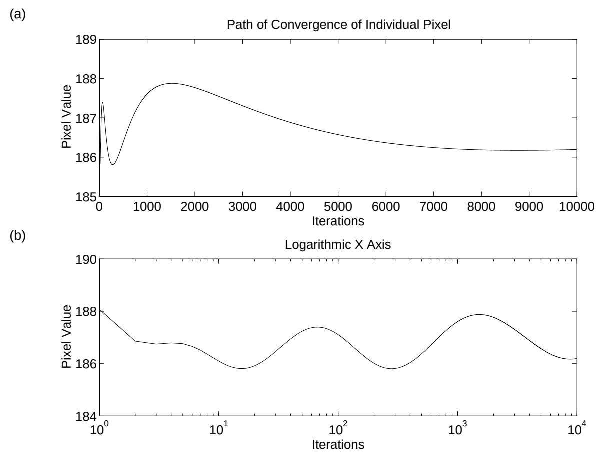

Figure 1: Path of an individual pixel over 10,000 R–L iterations, (a) linear, (b) logarithmic scales.

It is normally necessary to terminate the restoration process before convergence is reached to prevent the algorithm from causing noise amplification. The termination condition is often based on the statistics of residual noise. An alternative approach is to employ noise attenuation methods [5].

The R–L algorithm maintains two properties between the distorted and the restored image, conservation of energy and nonnegativity. Maintaining the total energy is important for applications such as astronomy and electron microscopy where the distortion does not alter the total number of photons or electrons detected. Nonnegativity is important, as the photon or electron count at any pixel cannot be negative.

2.2 Overview of Acceleration Methods

The aim of an acceleration algorithm is to reduce the time required to achieve a certain level of restoration without introducing unwanted artifacts. The acceleration factor is the ratio of the computational time required by the unaccelerated and accelerated methods to restore the image to an equivalent level. An error measure is often used to compare the level of restoration between images. If the acceleration method requires little additional processing, then the computational time is proportional to the number of iterations performed. To perform acceleration, it is necessary to determine which parameter is appropriate to modify — not in all cases will it be the image pixels. A number of techniques for the acceleration of the R–L algorithm will now be discussed [6, 7, 8, 9, 10, 11].

A simple way to increase the speed of convergence is to predict where each pixel in the image is going from the correction generated by each iteration. One method proposed by Meinel [6] alters the R–L algorithm by introducing an exponential correction factor:

$$\hat{f}_{k+1} = \hat{f}_k \left(h \star \frac{g}{h \otimes \hat{f}_k} \right)^{\kappa}, \qquad \text{where } \kappa > 1. \qquad (3)$$

This approach has the virtue of automatically preserving nonnegativity though a potential lack of stability at later stages of the iteration process was reported by Meinel.

An additive correction factor can be obtained by a line search technique:

$$\hat{f}_{k+\lambda} = \hat{f}_k + \lambda g_k, \qquad (4)$$

$$\text{where } g_k \triangleq \psi(\hat{f}_k) - \hat{f}_k. \qquad (5)$$

Here g_k is the difference vector generated by the iterative algorithm and is used as an estimate of the gradient direction. From Eq. (4), the estimated acceleration factor achieved by each iteration is equal to λ.

One method that uses the line search approach ([7]) adjusts λ to maximize the log-likelihood probability at each iteration and uses a Newton–Raphson iteration to find its new value. It offers speedups of 2 to 5 times but requires limits on λ to prevent instability. Another technique similar to this method, with a maximum acceleration factor of 7 after 10,000 iterations, was described by Holmes and Liu [8] . Maximizing a function in the direction of the gradient vector is termed steepest ascent while minimising a function (in the negative gradient direction) is called steepest descent. All discussions relating to ascent methods apply equally to descent methods. Kaufman [9] has also shown how the expectation–maximization step of R–L is related to steepest ascent.

Problems can occur with steepest ascent methods if the acceleration step is too large or inaccurate. Errors are introduced, which, if amplified by the acceleration process, can result in instability. Thus a steepest ascent method requires an acceleration iteration followed by a correction iteration with little acceleration. This approximately halves the gain made by each acceleration step. The method of acceleration described by Eq. 4 shall be referred to as technique I.

A gradient search method superior to steepest ascent is the conjugate gradient method [12]. This algorithm requires the gradient of the cost function and an accurate line search technique. This approach has been proposed by the authors [10] as well as by Kaufman [9] and by Lane [11]. One difficulty with line search techniques is the need to efficiently and accurately maximize a cost function. Several function evaluations may be required, which can involve significant computing time. It is also necessary to be able to determine a cost function and gradient measure for the iterative algorithm. This may not be trivial for algorithms such as the error–reduction or G–S type algorithms [13]. One of our aims in this paper is to propose an acceleration technique that requires as little information about the iterative algorithm as possible.

2.3 New Acceleration Method

A new technique is now proposed that attempts to overcome some of the problems with the line search methods described previously. The main modification is in the way the direction vector is calculated and applied. Line search methods base the direction vector on the difference generated by applying the iterative algorithm to the current prediction (Eq. 5). The new technique calculates the direction as the difference between the current iteration and the previous iteration. If x_k is the iterated point, y_k the predicted point, h_k the direction vector, and α_k the acceleration parameter then:

$$\begin{aligned} y_k &= x_k + \alpha_k h_k, & (6)\\ \text{where } h_k &\triangleq x_k - x_{k-1}, & (7)\\ x_{k+1} &= y_k + g_k, & (8)\\ g_k &\triangleq \psi(y_k) - y_k. & (9)\end{aligned}$$

This method shall be referred to as technique II and is graphically shown in Fig. 2. This method of acceleration is a form of vector extrapolation that predicts subsequent points based on previous points. Each iteration provides a correction step plus the information used in adjusting the step length.

Acceleration can be applied when the restoration changes slowly between each iteration and also if the algorithm is insensitive to the changes introduced. These are both properties of the R–L algorithm.

2.4 Determining Level of Acceleration

Determining the level of acceleration to apply after each iteration is important for maximizing the speedup and preventing instabilities. As mentioned previously, the R–L algorithm can be accelerated by use of the log likelihood as a cost function. However, this function is difficult to maximize analytically, and numerical approximations are normally used [7, 8].

Also, each iterative algorithm requires for its own individual cost function to be devised and evaluated. This may be mathematically or computationally difficult depending on the algorithm. Therefore there would be great advantages in an acceleration technique that offers large acceleration factors without maximizing a cost function.

This leads us to a technique we have termed automatic acceleration. In automatic acceleration only previous information is used for prediction, with no cost function to guide us. Technique II is well suited to this type of acceleration.

One method for determining the acceleration parameter α_k is derived in Appendix A. The operator centered dot operator is an element–wise multiplication, and $\sum$ sums all elements in the array:

$$\alpha_k = \frac{\sum g_{k-1} \cdot g_{k-2}}{\sum g_{k-2} \cdot g_{k-2}}, \quad 0 < \alpha_k < 1. \qquad (10)$$

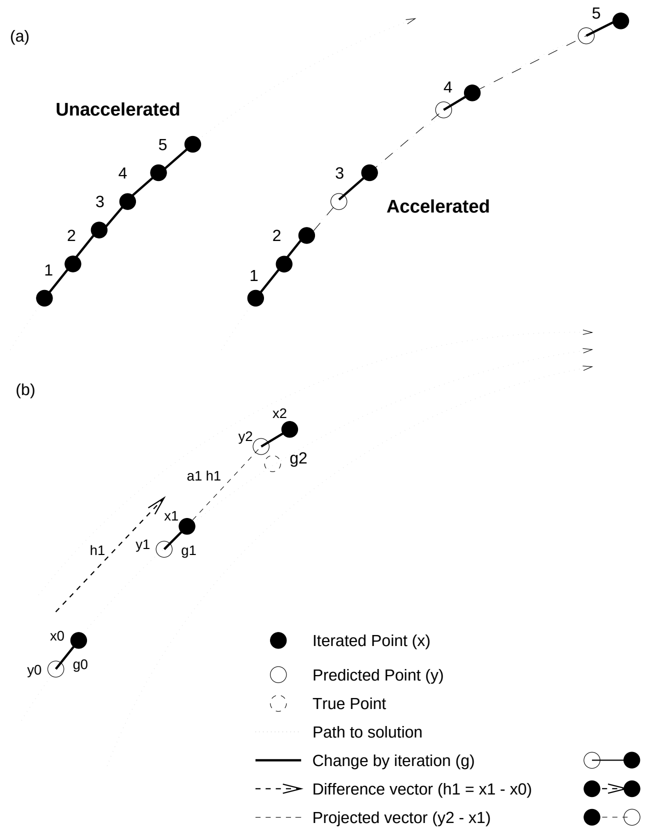

Figure 2: (a) Conceptual difference between the unaccelerated and new accelerated method for five iterations. (b) Vector diagram illustrating the acceleration method in geometric terms.

The value of α_k should be constrained between 0 and 1. The upper bound is not normally reached, as most algorithms have a decreasing rate of convergence, but enforcing the constraint prevents the algorithm from causing the prediction vector to grow exponentially. Should α_k become negative, then its value is set to zero, which causes the acceleration process to terminate and begin again. No acceleration is applied for the first iteration, as there is insufficient information to calculate α_k.

This formulation of α_k takes into account two factors — the geometric convergence of each iteration step and the similarity in direction of the gradient vectors. The maximum level of acceleration is achieved if each gradient vector is collinear. This can be illustrated by considering the one dimensional sequence 2^{-k}, which has linear convergence. Applying the new acceleration method transforms the sequence to $2^{-k(k+1)/2}$, which has superlinear convergence [14].

An interesting feature of this method of calculating α_k is that it will not accelerate vectors already accelerated by the steepest ascent approach, as each gradient vector is orthogonal to the previous one.

When calculating α_k it is useful also to determine a second parameter, β_k, which can be used later in estimating the acceleration factor:

$$\beta_k = \frac{\sum g_{k-1} \cdot g_{k-2}}{\sum g_{k-1} \cdot g_{k-1}} = \alpha_k \frac{\sum g_{k-2} \cdot g_{k-2}}{\sum g_{k-1} \cdot g_{k-1}}. \qquad (11)$$

2.5 Estimating the Acceleration Factor

The estimated acceleration achieved by each iteration by technique II is different from that achieved by line search methods (technique I). Even though the acceleration parameter is less than unity, significant acceleration is achievable because the acceleration is related recursively to all previous acceleration steps.

The estimated acceleration (η_k) at iteration k is

$$\eta_k = 1 + \sum_{j=1}^{k} \prod_{m=k-j+1}^{k} \gamma_m, \qquad (12)$$

where γ is the efficiency parameter. Appendix B defines efficiency and derives this approximation for the acceleration factor. If α_k is calculated from Eq. 10, then γ_k can be approximated by:

$$\gamma_k = (\alpha_k \beta_k)^2. \qquad (13)$$

This formulation has been used for all results in this paper and was found to be relatively accurate, though it may not be applicable for all images and algorithms. The total effective number of unaccelerated iterations is the sum of the acceleration factors at the current and at each previous step. The average acceleration factor is the effective number of iterations divided by the current iteration, k.

2.6 Increasing Acceleration

Technique II can be further enhanced by use of more information to increase the accuracy of the prediction process or to accelerate it further. The current acceleration process employs only the first two terms in the Taylor's Series expansion of Eq. 4. The full Taylor's Series expansion is

$$\begin{aligned} f_{k+\alpha} &= f_k + \alpha \Delta f_k + \frac{1}{2!}\alpha^2 \Delta^2 f_k + \frac{1}{3!}\alpha^3 \Delta^3 f_k + \\ &\quad \cdots + \frac{1}{n!}\alpha^n \Delta^n f_k + \cdots, \qquad (14) \end{aligned}$$

where $\Delta^n f_k$ is the nth finite difference at f_k. Using higher–order terms is not efficient when technique I is used because additional unaccelerated iterations are required for calculation of the higher derivatives. However, it is relatively simple when technique II is used, as the higher derivatives can be calculated from the previous iterations. The $\Delta^2 f_k$ term is calculated as

$$\begin{aligned} \Delta^2 f_k &= \Delta f_k - \Delta f_{k-1} \qquad (15) \\ &= (f_k - f_{k-1}) - (f_{k-1} - f_{k-2}) \qquad (16) \\ &= f_k - 2f_{k-1} + f_{k-2}, \qquad (17) \end{aligned}$$

thus no addition iterations are required, only the storage for an extra image. Only a few higher–order derivatives should be used, as their influence decreases, and they become less accurate, as the order increases. Using the second derivative allows the acceleration factor to be increased, and a modified acceleration parameter is

$$\alpha_k = \left(\frac{\sum g_{k-1} \cdot g_{k-1}}{\sum g_{k-2} \cdot g_{k-2}} \right)^{\frac{1}{2}}, \quad 0 < \alpha_k < 1. \quad (18)$$

This new formulation for α_k is based on the assumption that the improved direction vector only needs to take account of the geometric convergence of the restoration. It should be noted that the finite difference approximation for the second derivative does not take into account the different step size between each iteration. Despite these approximations, using a second–order term has been found to be remarkably effective in giving an extra boost to the acceleration process.

A far more ambitious goal is to use higher–order terms to predict more than one step ahead. This should be particularly effective during latter iterations when the restoration is changing very slowly. This is an area of continuing research.

While this enhancement to the acceleration process produces higher levels of acceleration, only results for the first–order implementation of technique II will be presented.

3 Results and Discussion

This section presents the results obtained by application of the proposed acceleration technique to a variety of iterative algorithms with simulated and real blurred imagery. One of the original unblurred eight–bit images is shown in Fig. 3, with the selected region measuring 128×128 pixels. The tail section is a detailed image, and the identification number 01568 will be used in illustrating the acceleration methods.

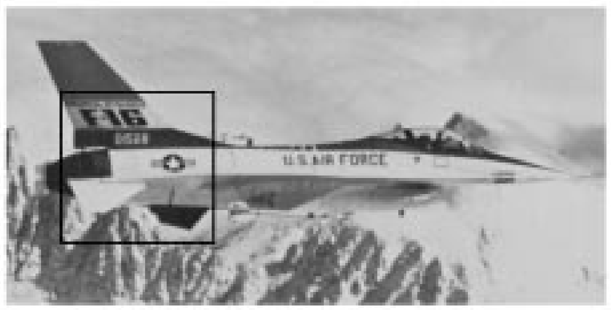

Figure 3: Original F16 image from which the selected region of the tail is used, measuring 128×128 pixels.

The error measure used to compare results is a normalized mean square error (MSE) between the restoration and original unblurred image. The MSE is normalized with respect to the initial MSE.

The majority of the tests involved the R–L algorithm. The following abbreviations are used for the unaccelerated and accelerated R–L restorations:

RL0(n) = Unaccelerated R–L for n iterations.
RL1(n) = Accelerated R–L for n iterations.

with the acceleration parameter calculated from Eq. 10. If the initial image presented to the R–L algorithm is flat, then the first iteration will produce $h \star g$. Therefore $h \star g$ will be used as the initial image for each restoration.

3.1 Noise–Free Images

A Gaussian PSF with the form $\exp(-r^2/5)$, where r is the radius in pixels, was used to blur the image to give the distorted image in Fig. 4(a). The first test added no noise to the image so as not to obscure any oscillations or artifacts introduced during restoration. Deconvolution was performed with the R–L algorithm, and restorations after 250 iterations are presented.

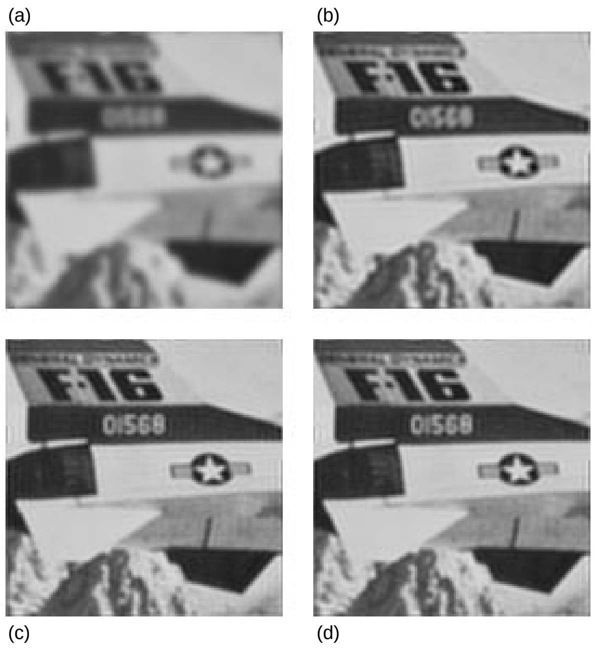

Figure 4: Blurred and restored images using R–L; (a) blurred, (b) unaccelerated RL0(250), (c) accelerated RL1(250), (d) unaccelerated RL0(10,000).

The RL0(250) restoration in Fig. 4(b) shows increased detail over the blurred image but the identification number is not readable. RL1(250) gives Figure 4(c), which is further restored and produces a readable identification number. An unaccelerated restoration was then performed until the MSE matched that of the RL1(250) result. This required approximately 10,000 iterations and is shown in Fig. 4(d), which also visually matches the accelerated restoration [Fig. 4(c)]. Therefore RL1(250) restoration has an overall acceleration factor of 40 times. Using Eq. 12 gives an estimated acceleration factor of 37.3 times, which correlates well with the experimental result. By contrast, acceleration using the steepest descent approach results in a speedup of only five times.

Figure 5(a) shows the acceleration parameter (α_k) produced by the RL1(250) restoration. The curve increases rapidly from zero and tends toward unity as the restoration proceeds. The instantaneous and average acceleration factor at each iteration, based on Eq. 12, are shown in Fig. 5(b). The acceleration factor has an approximate linear increase with each iteration.

One limiting factor in applying acceleration is the necessity to prevent unwanted artifacts in the restoration. One common artifact is oscillations or rippling in the restoration. The unaccelerated R–L algorithm does introduce slight oscillations near edges in the image — caused by the Gibbs effect. Applying excessive acceleration can increase the significance of these oscillations. Both restorations have a similar rippling artifact, which suggests the method for calculating the acceleration parameter is appropriate.

Figure 6(a) shows the MSE between the original and the restored images for RL0(250) and RL1(250) restorations. The curves illustrate how the acceleration in-

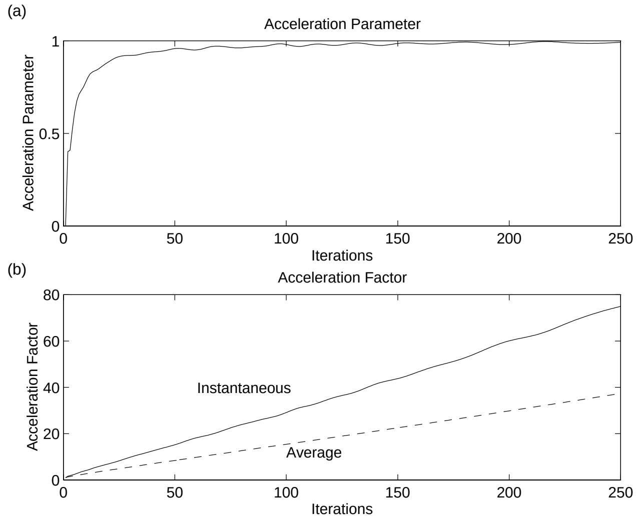

Figure 5: (a) Acceleration parameter (α_k) for RL1(250), (b) instantaneous and average acceleration factor for RL1(250).

creases the rate of convergence, especially during the initial iterations. Figure 6(b) shows the same data but with the x–axis altered to reflect the effective number of iterations performed. This causes the MSE curves to align almost exactly, which further confirms the accuracy of the estimated acceleration factor.

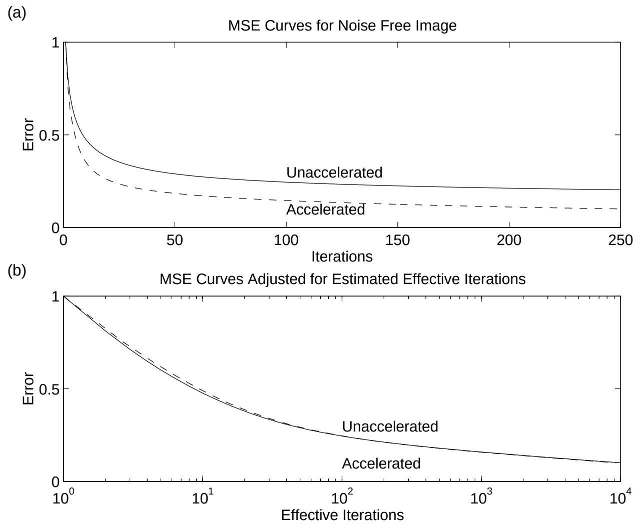

Figure 6: (a) MSE between original and restored image (no noise) and (b) plotted with effective iterations, which shows the curves match. Unaccelerated RL0(10,000), solid curve; accelerated RL1(250), dashed curve.

3.2 Performance with Poisson Noise

Unless modified to cope with noise, the R–L algorithm can cause unwanted artifacts during restoration by attempting to overfit the noisy data. It is normally necessary to halt restoration when artifacts begin to appear. To determine if noise has any influence on the acceleration method, the blurred image was corrupted with Poisson noise and then restored. Poisson noise was simulated by taking unit variance Gaussian noise and scaling it by the square root of the pixel intensity before adding it to the image. Because Poisson noise is dependent on the image intensity, two tests were performed, the first with the mean of the image as 1,000 and the second with mean 10,000. A lower image intensity results in a lower signal–to–noise ratio. Restorations were performed for 100 different Poisson noise processes.

The results for both experiments are summarized in Table 1, which indicates image mean, whether acceleration is applied, the iteration at which the minimum MSE is achieved, the acceleration factor, the estimated number of iterations at the minimum MSE and the figure showing resulting restoration. The results for the 1,000 mean and 10,000 mean images are shown in Figs. 7 and 9 respectively, with corresponding MSE curves in Figs. 8 and 10. The MSE graphs have a solid curve as the average MSE, and the dashed curves show the maximum and minimum MSE at each iteration.

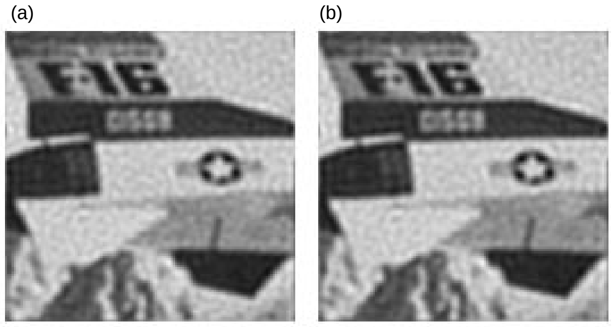

Figure 7: R–L restoration of image corrupted with Poisson noise (Image mean = 1,000); (a) Unaccelerated RL0(32), (b) Accelerated RL1(11).

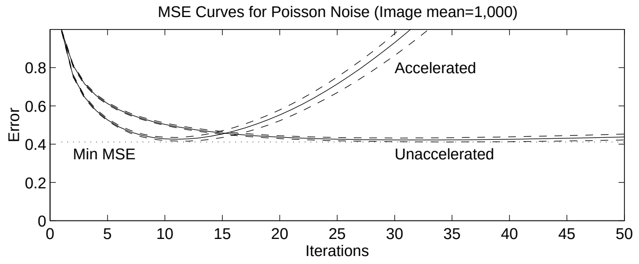

Figure 8: MSE curves of restoration for Fig. 7. Image mean, 1,000. Results are averaged over 100 different Poisson noise processes; average MSE (solid curve) with maximum and minimum MSE (dashed curves) at each iteration. Minimun unaccelerated MSE, dotted curve.

Owing to the high level of noise in the first experiment, relatively few iterations can be performed before noise amplification becomes significant. Despite this, the acceleration method was still successful in reducing the number of iterations by a factor of three. The image with mean 10,000, and therefore higher signal–to–noise ratio, allowed more iterations to be performed.

The accelerated algorithm works just as well in the presence of noise as the unaccelerated algorithm because a similar minimum MSE is achieved. The normal practice of halting the restoration after a certain number of iterations is important because the acceleration can cause the restoration to quickly diverge from the desired results if too many iterations are performed.

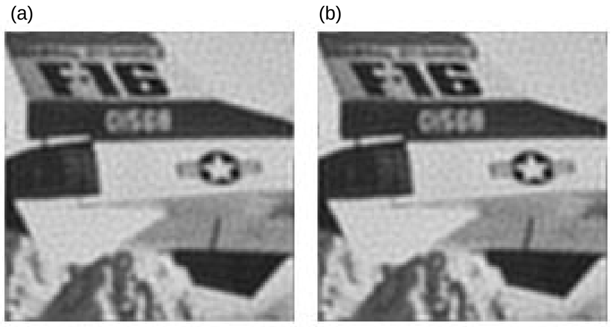

Figure 9: R–L restoration of image corrupted with Poisson noise (Image mean, 10,000); (a) Unaccelerated RL0(151), (b) accelerated RL1(26).

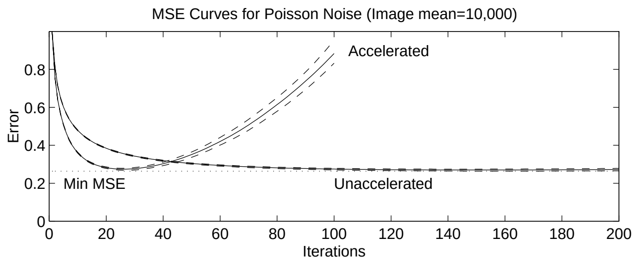

Figure 10: MSE curves of restoration for Fig. 9. Image mean, 10,000. Results averaged over 100 different Poisson noise processes. Curves have same definitions as in Fig. 8.

Table 1: Results of Restoration with Poisson Noise

Image Mean	Accel.	Min MSE at Iter.	Accel. Factor	Est. Iter.	Fig.
1,000	No	32	–	–	7(a)
1,000	Yes	11	3	32	7(b)
10,000	No	151 ± 9	–	–	9(a)
10,000	Yes	26 ± 1	5.8	138	9(b)

3.3 Acceleration of Maximum Entropy Algorithm

The new acceleration technique was applied to an ME algorithm as described by Bonavito *et al* [15]. The method uses Lagrange multipliers (λ_k) to determine the restored image (f_k) such that

$$f_k \;=\; c\exp(h \star \lambda_k), \tag{19}$$

where c is a scaling factor. An example restoration in [15] is performed on a Hubble Space Telescope (HST) image that suffers from spherical abberation and detector saturation [1]. The R Aquarii image (HST dataset x0c90101t_cvt.c1h) is shown in Fig. 11(a) with an over-exposed central region and Reseau marks visible. Those areas of invalid data are masked, and the resulting image is shown in Fig. 11(b). The HST image was obtained before installation of the corrective optics, and the PSF used for restoration was provided (HST dataset x0cj010bt_cvt.c1h).

Figure 11(c) shows the result of 1,000 unaccelerated iterations of the ME method. This is the same number

[1]The image of the binary stellar system R Aquarii was observed with the faint object camera and made available through the Science Assessment and Early Release Observations programme of the Space Telescope Science Institute.

as performed in [15] and a similar restoration is produced. An accelerated ME restoration was performed for 50 iterations and the result is shown in Fig. 11(d). Both restorations produce comparable results, indicating that the accelerated algorithm reduced the number of iterations by a factor of approximately 20. The large acceleration factor is confirmed experimentally when it is noted that the acceleration parameter was approximately unity throughout the restoration.

It is important to note that the Lagrangian multipliers (λ_k) are the controlling variable in the ME algorithm and, therefore, these should be accelerated rather than the pixel values of the restored image.

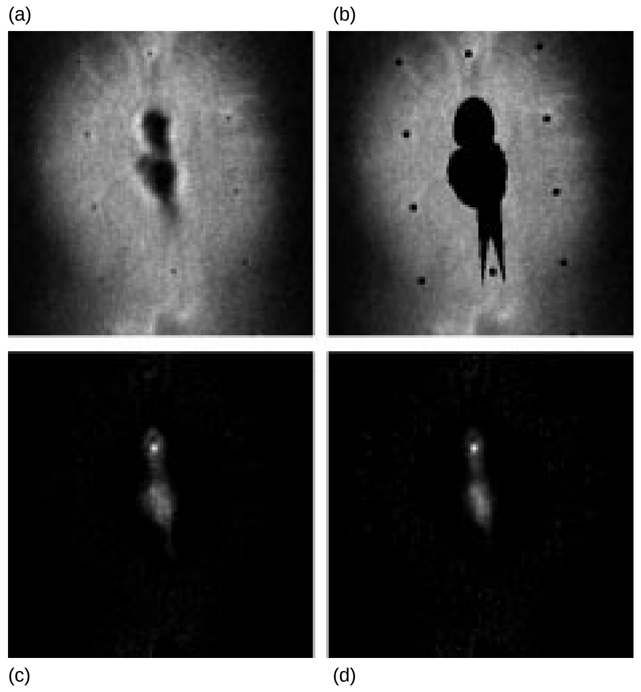

Figure 11: Restoration of R Aquarii using ME. (a) Raw HST data, (b) mask applied to invalid data, (c) unaccelerated (1,000 iterations), (d) accelerated (50 iterations).

3.4 Acceleration of the Gerchberg–Saxton Algorithm

The Gerchberg–Saxton (G–S) algorithm is one popular method for attempting Fourier magnitude or phase retrieval. This algorithm can be painfully slow to converge [13] and is a good candidate for applying acceleration. To test the G–S algorithm, the original F16 image was padded with zeros to 256×256 pixels, and then either the Fourier magnitude or phase was removed. Figure 12(a) shows the result of magnitude retrieval by the generic G–S algorithm after 1,000 iterations, and Fig. 12(b) shows the result after 70 accelerated iterations. The accelerated result contains fewer artifacts than 1,000 unaccelerated iterations, indicating a speedup of over 14 times. This acceleration factor is greater than that predicted analytically, possibly because the acceleration helps avoid situations in which the G–S algorithm stagnates.

Phase retrieval is more difficult to achieve, as the standard G–S algorithm often stagnates. Applying accelerated G–S phase retrieval to the test image caused the algorithm to stagnate faster, achieving a lower MSE in seven iterations than the unaccelerated G–S did in 100 iterations. One method of preventing stagnation is to use the hybrid input–output algorithm. There has been some preliminary success in achieving acceleration with this modification of the G–S algorithm.

It is important to note that acceleration was applied in the frequency domain for phase retrieval and in the spatial domain for magnitude retrieval.

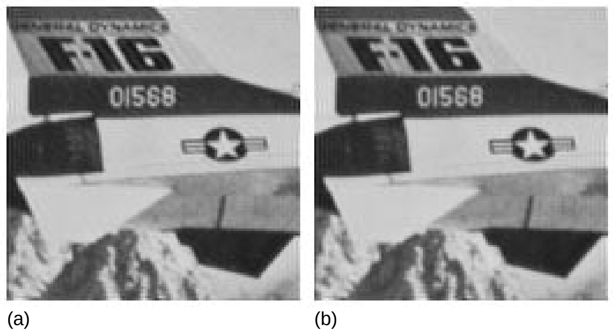

Figure 12: Magnitude retrieval using the Gerchberg–Saxton algorithm; (a) Unaccelerated (1,000 iterations), (b) accelerated (70 iterations)

3.5 Practical Considerations When Applying Acceleration

When applying the acceleration techniques described in this paper, there are several important considerations that should be made to ensure the best result is achieved. The first is whether the iterative algorithm can be accelerated. Several test restorations should be made first to observe whether the restoration converges smoothly. The path an individual pixel (or other variable) takes should be smooth and continuous with no discontinuities or noise. The algorithm should also be insensitive to small changes in the image between each iteration. Algorithms that require several hundred or several thousand iterations should achieve greater acceleration factors than those that require only a few iterations.

Special consideration should be made regarding which part of the iterative algorithm is to be accelerated. When the R–L algorithm is used, the estimated restoration after each iteration is accelerated, whereas the Lagrangian multipliers [Eq. 19] are accelerated for the ME algorithm.

Acceleration is normally begun immediately if a good estimate is used as the starting point for the restoration. However, if constant or random valued data is used as the first estimate then, to prevent excessive acceleration at the start, a few unaccelerated iterations should be performed before acceleration is applied.

Stability during restoration will be maintained with technique II if α_k is between 0 and 1, and the image retains non–negativity. A check at each iteration should be made to ensure an accelerated pixel does not become negative.

3.6 Future Research

There are several areas for further investigation into the new acceleration method (technique II). Applying the acceleration method to other iterative algorithms should help to uncover limitations of the method and will show the extent to which the acceleration method can be applied in practice. The use of technique II should allow the use of iterative image restoration algorithms in applications where they were previously considered too slow. As mentioned previously, there is considerable scope for achieving higher levels of acceleration when more information is used in the acceleration process.

The new acceleration method could also be used as a generic acceleration method when iterative algorithms are designed. This would allow the algorithm to be designed for producing the best result rather than having to compromise for speed considerations.

4 Conclusion

Traditional acceleration methods using a gradient search approach have been presented but require careful use to prevent instability during restoration. A new method, called automatic acceleration, has been proposed and shown to achieve significantly increased acceleration over previous methods with little computational overhead. An example R–L restoration achieves an average speedup of 40 times after 250 iterations, and an ME method by 20 times after only 50 iterations. Acceleration is achieved without the need for a cost–function to be determined or minimized.

The new method is stable, and an estimated acceleration factor has been derived and confirmed by experiment. The acceleration technique has been successfully applied to Richardson–Lucy, maximum entropy and Gerchberg–Saxton restoration algorithms and can be integrated with other iterative techniques. The technique shows considerable promise in accelerating a wide range of iterative image restoration algorithms, which should allow them to be used in applications where previously they would have been considered impractical.

5 Acknowledgments

The authors thank Russell Smith of The University of Auckland and Richard Lane of Canterbury University for their contributions to this paper. The reviewers are also to be acknowledged for their useful comments and suggestions. D.S.C. Biggs also wishes to acknowledge the support of The University of Auckland and the Royal Society of New Zealand through the R.H.T. Bates Scholarship.

References

[1] T. J. Cornwell. Where have we been, where are we now, where are we going? In R. J. Hanisch and R. L. White, editors, *The Restoration of HST Images and Spectra II.* Space Telescope Science Institute, Baltimore, Md. 1994.

[2] William H. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, Vol. 62(No. 1):55–59, January 1972.

[3] L. B. Lucy. An iterative technique for the rectification of observed images. *The Astronomical Journal*, Vol. 79(No. 6):745–754, June 1974.

[4] V. Vardi, L. A. Shepp, and L. Kaufman. A statistical model for positron emission tomography. *Journal of the American Statistical Association*, Vol. 80(No. 389):8–37, March 1985.

[5] Richard L. White. Image restoration using the damped Richardson–Lucy method. In R. J. Hanisch and R. L. White, editors, *The Restoration of HST Images and Spectra II.* Space Telescope Science Institute, Baltimore, Md. 1994.

[6] Edward S. Meinel. Origins of linear and non-linear recursive restoration algorithms. *Journal of the Optical Society of America A*, Vol. 3(No. 6):787–799, June 1986.

[7] H.M. Adorf, R.N. Hook, L.B. Lucy, and F.D. Murtagh. Accelerating the Richardson–Lucy restoration algorithm. In *4th ESO/ST-ECF Data Analysis Workshop*, pages 99–103. European Southern Observatory, Garching, Germany, 1992.

[8] Timothy J. Holmes and Yi-Hwa Liu. Acceleration of maximum-likelihood image restoration for fluorescence microscopy and other noncoherent imagery. *Journal of the Optical Society of America A*, Vol. 8(No. 6):893–907, June 1991.

[9] Linda Kaufman. Implementing and accelerating the EM algorithm for positron emission tomography. *IEEE Transactions on Medical Imaging*, Vol. 6(No. 1):37–51, March 1987.

[10] D. S. C. Biggs and M. Andrews. Conjugate gradient acceleration of maximum-likelihood image restoration. *Electronics Letters*, Vol. 31(No. 23):1985–1986, 9th Nov 1995.

[11] R. G. Lane. Methods for maximum likelihood deconvolution. *Journal of the Optical Society of America A*, Vol. 13:1992–1998, 1996.

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies in C.* Cambridge University Press, 2nd edition, 1992.

[13] J. R. Fienup. Phase retrieval algorithms: a comparison. *Applied Optics*, Vol. 21(No. 15):2758–2769, 1 August 1982.

[14] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization.* Academic, New York, 1981.

[15] N.L. Bonavito, J.E. Dorband, and T. Busse. Maximum entropy restoration of blurred and oversaturated Hubble Space Telescope imagery. *Applied Optics*, Vol. 32, No. 29:5768–5774, 10 October 1993.

Appendix A: Acceleration Parameter for Technique II

We consider the iterative algorithm $\psi(\cdot)$, which generates a new estimate (x_{k+1}) given the current vector (x_k):

$$\begin{aligned} x_{k+1} &= \psi(x_k) \\ &= x_k + g_k, \end{aligned}$$

and x_k slowly converges towards a solution at x_s. The change g_k introduced at iteration k can be considered a step in the gradient ascent direction toward the solution. The function to be maximized can be approximated locally by a multidimensional quadratic:

$$\begin{aligned} m(x) &= -(x - x_s)^\top A(x - x_s), \\ \nabla m(x) &= -2A(x - x_s), \end{aligned}$$

where A is a square, symmetric, and positive definite matrix. Taking a step in the gradient direction $\nabla m(x)$ for each iteration produces a path through the multidimensional space towards the solution. This path can be described by the solution to the difference equation in terms of the eigenvalues (Λ_A) and eigenvectors (S) of A.

$$\begin{aligned} g_k &= \epsilon \nabla m(x_k) \\ &= -2\epsilon A(x_k - x_s), \\ A &= S\Lambda_A S^{-1}, \\ B &= I - 2\epsilon A, \\ \Lambda_B &= I - 2\epsilon\Lambda_A, \\ x_k &= S\Lambda_B^k S^{-1}(x_0 - x_s) \ + \ x_s, \\ \epsilon &< \frac{1}{2\max(\Lambda_A)}, \end{aligned}$$

x_s is the vector x_k converges. Λ_A and Λ_B are diagonal matricies, and ϵ is a scalar used to prevent Λ_B from becoming negative. We have no knowledge of the form of the quadratic or its eigenvalues or eigenvectors.

The aim of acceleration is to move along the path (or close to it) faster than the iterative algorithm. If this is possible, then the accelerated method should result in the same solution given that the same path was taken. Using the current and the previous estimates and associated gradient information, the aim is to predict a new estimate further along the path.

$$\begin{aligned}
x_k &= S\Lambda_B^k S^{-1}(x_0 - x_s) \; + \; x_s, \\
g_{k-1} &= S\Lambda_B^k (I - \Lambda_B^{-1}) S^{-1}(x_0 - x_s), \\
x_{k-\delta} &= S\Lambda_B^{k-\delta} S^{-1}(x_0 - x_s) \; + \; x_s, \\
g_{k-\delta-1} &= S\Lambda_B^{k-\delta}(I - \Lambda_B^{-1}) S^{-1}(x_0 - x_s).
\end{aligned}$$

The difference between the current (x_k) and previous estimate $(x_{k-\delta})$ is

$$h_{k,-\delta} \; = \; x_k - x_{k-\delta} \;\; = \;\; S\Lambda_B^k(I - \Lambda_B^{-\delta})S^{-1}(x_0 - x_s).$$

Given estimates at x_k and $x_{k-\delta}$ the aim is to predict $x_{k+\delta}$:

$$\begin{aligned}
x_{k+\delta} &= S\Lambda_B^{k+\delta} S^{-1}(x_0 - x_s) \; + \; x_s, \\
g_{k+\delta-1} &= S\Lambda_B^{k+\delta}(I - \Lambda_B^{-1}) S^{-1}(x_0 - x_s), \\
h_{k,\delta} \; = \; x_{k+\delta} - x_k &= S\Lambda_B^{k+\delta}(I - \Lambda_B^{-\delta}) S^{-1}(x_0 - x_s).
\end{aligned}$$

Using a first–order extrapolation we predict $h_{k,\delta}$ by a least–squares projection of $\alpha_k h_{k,-\delta}$, where α_k is a scalar:

$$\begin{aligned}
\hat{h}_{k,\delta} &= \alpha_k h_{k,-\delta}, \\
0 &= (h_{k,\delta} - \alpha_k h_{k,-\delta})^\top h_{k,-\delta}, \\
\alpha_k &= \frac{{h_{k,\delta}}^\top h_{k,-\delta}}{{h_{k,-\delta}}^\top h_{k,-\delta}}.
\end{aligned}$$

We do not have knowledge of $h_{k,\delta}$ but we do have the gradient information:

$$\begin{aligned}
h_{k,\delta} = x_{k+\delta} - x_k &= S\Lambda_B^{k+\delta}(I - \Lambda_B^{-\delta}) S^{-1}(x_0 - x_s), \\
I - \Lambda_B^{-\delta} &\approx \delta(I - \Lambda_B^{-1}), \\
h_{k,\delta} &\approx \delta S\Lambda_B^k (I - \Lambda_B^{-1}) S^{-1}(x_0 - x_s) \\
&= \delta g_{k-1}, \\
h_{k,-\delta} &\approx \delta g_{k-\delta-1}, \\
\alpha_k &\approx \frac{{g_{k-1}}^\top g_{k-\delta-1}}{{g_{k-\delta-1}}^\top g_{k-\delta-1}}.
\end{aligned}$$

From this we can predict a new point (y_k) and apply the iterative algorithm to generate the next estimate (x_{k+1}) and gradient (g_k):

$$\begin{aligned}
y_k &= x_k + \alpha_k(x_k - x_{k-1}), \\
x_{k+1} &= \psi(y_k) \\
&= y_k + g_k, \\
\alpha_{k+1} &= \frac{{g_k}^\top g_{k-1}}{{g_{k-1}}^\top g_{k-1}}, \\
y_{k+1} &= x_{k+1} + \alpha_{k+1}(x_{k+1} - x_k).
\end{aligned}$$

This formulation of the acceleration parameter requires no knowledge of the quadratic A or its eigenvalues and eigenvectors and does not require that they be constant over the entire error surface.

Appendix B: Estimating the Acceleration Factor for Technique II

Being able to estimate the acceleration at each step is useful in comparing the technique with unaccelerated restoration and determining how effective the acceleration process is.

Continuing from Appendix A, it can be seen that the acceleration achieved at iteration k is related to how effectively $h_{k,\delta}$ is estimated. If γ_k is the efficiency parameter, then the step actually achieved is $h_{k,\gamma_k\delta_k}$:

$$\begin{aligned}
\hat{h}_{k,\delta} &= \alpha_k h_{k,-\delta_k}, \\
h_{k,\gamma_k\delta_k} &= \alpha_k h_{k,-\delta_k},
\end{aligned}$$

and therefore the effective number of unaccelerated iterations achieved by acceleration is $\gamma_k\delta_k$. The value of δ_k can be estimated from all previous values of γ_k:

$$\delta_k \;\; = \;\; 1 + \sum_{j=1}^{k-1} \prod_{m=k-j}^{k-1} \gamma_m.$$

The acceleration factor, η_k, is given by $\gamma_k\delta_k$ plus the single step of the iterative algorithm:

$$\eta_k \;\; = \;\; 1 + \sum_{j=1}^{k} \prod_{m=k-j+1}^{k} \gamma_m.$$

The value of γ_k is difficult to determine exactly, though it is proposed that it can be estimated by

$$\hat{\gamma}_k \; = \; \alpha_k \frac{\|g_{k-2}\|}{\|g_{k-1}\|} \left(\frac{{g_{k-1}}^\top g_{k-2}}{\|g_{k-1}\| \, \|g_{k-2}\|} \right)^M .$$

The estimates of acceleration achieved in this paper use $M = 3$ [Eq. 13] which correlate well with experimental results.