

MINIMISATION METHODS IN GRAVITATIONAL LENSING

A. Adeyemo (4108556)

&

P. Smith (4117971)

June 03, 2014

ABSTRACT

Minimisation techniques have been used to identify the shape and position of lensing masses in our universe for the past three decades. In this paper the minimisation problem inherent in solving extended source strong gravitational lensing is approached with various minimising algorithms to identify the most robust and efficient methods. The success of each technique shows where it can best be applied. The minimising techniques compared were the grid search, downhill-simplex, Markov chain Monte Carlo, genetic, simulated annealing, hybrid genetic and downhill-simplex, nested Markov chain Monte Carlo sampling, multi-nested Markov chain Monte Carlo sampling, and multi-nested genetic Markov chain Monte Carlo sampling algorithms. This report presents our findings showing the most robust and most efficient approaches to the problem.

CONTENTS

Abstract	1
1. Introduction	3
2. Theory	5
2.1 Gravitational lensing.....	5
2.2 Minimisation algorithms.....	6
2.2.1 Grid search	6
2.2.2 The downhill-simplex algorithm.....	6
2.2.3 The Markov chain Monte Carlo algorithm	7
2.2.4 Genetic algorithms	7
2.2.5 Simulated annealing.....	7
2.2.6 Nested Sampling	8
3. Methodology	8
3.1 Model setup	8
3.2.1 Grid search	10
3.2.2 The downhill-simplex algorithm.....	10
3.2.3 The Markov chain Monte Carlo algorithm	10
3.2.4 Genetic algorithm	10
3.2.5 Simulated annealing.....	11
3.2.6 Hybrid genetic and downhill-simplex algorithm.....	11
3.2.7 Nested Markov chain Monty Carlo sampling	11
3.2.8 Multi-nested Markov chain Monty Carlo sampling	12
3.2.9 Multi-nested genetic Markov chain Monty Carlo sampling	13
4. Results	13
4.1 Grid search.....	13
4.2 Downhill-simplex, MCMC, genetic and simulated annealing algorithms.....	14
4.3 Genetic algorithm population comparison.....	16
4.4 Nested Markov chain Monty Carlo Sampling algorithms	17
5. Discussion	19
6. Summery and conclusion	21

1. INTRODUCTION

The gravitational lensing of distant galaxies, particularly those with large photon deflection angles from extended sources, provide a rich source of information on the structure and distribution of matter (possibly including dark matter) in the lensing galaxies/galaxy clusters. The matter distributions are used, amongst other things, to find indirect evidence of dark matter, and help to create physically accurate N-body simulations. This information however is obscured behind non-linear problems that cannot be solved analytically. Markov Chain Monte Carlo and Simplex algorithms have been used for decades to solve these problems. In this report we present the findings of our exploration into various minimisation algorithms; those previously used, and those not used before. It should be said that the supposed dark matter distributions found via these methods are indirectly detected. Until dark matter is directly detected and proven to exist in the same quantities detected indirectly, the dark matter distributions found by these algorithms, and in other indirect detection experiments (such as velocity distribution analyses), may not be used to draw any conclusions about matter distribution. They may not be matter distributions at all but something else entirely, such as an effect described by an as yet undiscovered emendation to our theory of gravity. However we will not be going into this and will talk about the problem as though dark matter exists.

Einstein's theory of general relativity tells us that photons, due to their lack of mass, travel along null geodesics, or straight lines in spacetime. However around large bodies of mass spacetime curves, and as such photons turn corners whilst going in straight lines. The angles that these photons are deflected by tell us the shape of the spacetime they travel through. Since the shape of spacetime is governed by the mass present, it is possible to determine the position and density of any mass contributing to the curvature of the spacetime the light travels through. However, in the cases of strong gravitationally lensed galaxies, the distances between galaxies are colossal and the still astronomically huge depth of the lensing galaxies/galaxy clusters are by comparison negligible so the calculations required to model the system are simplified. What is required is the minimisation of a function in parameter space. This function is a mathematical description of the un-lensed source and the lens itself.

Einstein published his general theory of relativity in 1916 [1], and just 3 years later Arthur Eddington and Frank Watson Dyson observed that the stars close to the sun were slightly out of position, [2] proving that gravitational fields have an effect on light. Not until 1937 did anyone consider galaxies as lenses [3], and another four decades before the first gravitational lens was discovered by Dennis Walsh, Bob Carswell, and Ray Weymann [4]. In 1964 Refsdal used results he found for a star acting as a lens to show the Hubble constant and mass of a lensing galaxy could be derived for a simple spherical galaxy lensing a supernova [5] [6]. However it is difficult to constrain a mass profile from strong lensing alone, only being able to comment on the average surface mass density in an annulus bounded by lensed images. So in 2005 Rusin and Kockanek studied these annuli on 22 galactic lenses, where each was at a different radius due to differing distances between source and lens. Assuming the mass profiles of the 22 galaxies to be self-similar and found that the slope of the density profile was isothermal within error [7]. However strong lensing can only probe a narrow range of radii, by combining it with analysis of the weak lensing of point sources, which is observed at much larger radii, mass models for the quasar 0957+561 were improved [8].

In 1987 Soucail, Fort, Mellier, and Picat observed strong lensing of galaxy clusters A370 and C12244 [9]. Amongst other explanations it was proposed that the extended objects were warped background galaxies [10], which was confirmed a year later as the redshift of the arc in A370 was different to that of the rest of the cluster [11]. In the early to mid 1990's the arcs in various galaxy clusters were studied. It was immediately obvious that the mass distributions of the clusters could not be axially symmetric as there were no counter-arcs. Within two months of each other two groups asserted that a large amount of the clusters' mass could not be attached to the galaxies as the arcs' radii were too large [12] [13]. Straight arcs found in two clusters implied a very high mass to light ratio [14] [15] [16], and finally it was shown the density profiles of the clusters were steep, as shallower profiles would cause thicker arcs [13]. Before the arcs produced by these extended sources were studied the image studied would be formed by a point source, like a quasar. At first similar approaches to those used on point sources were used for extended sources. One such method proposed by Kayser and Schramm in 1988 was to find regions in the image with the same surface brightness, and therefore must

have the same point in the source, then minimise the dispersion of the source points by varying the mass profile [17]. This approach was successfully applied a year later to the radio Einstein ring MG1131+0456 by Kochanek et al [18].

This method is flawed, however. It does not take into account the PSF (point-spread function) of the image. The diffraction of light in either the atmosphere or the detection equipment itself causes the light from a point to spread forming a Gaussian intensity distribution. This means that single points in the source do not correspond to single points in the image, but overlapping Gaussians. To correctly find the source surface brightness profile and the mass distribution of the lens one must take into account the PSF. This is done by simulating the lensing of a model source and lens (including the PSF), creating a modelled image to be compared to the actual image. Different models are trialled minimising the difference between the actual and modelled images. A χ^2 test is used to find the difference between the actual and modelled images, calculated using;

$$\chi^2 = \sum_i \frac{(a_i - m_i)^2}{\sigma} \quad (1)$$

Which is a sum over all pixels where a_i is the count in each actual image pixel, i , and m_i is the count in each modelled image pixel. σ is the standard deviation of the noise in the actual image.

The way that we have tested algorithms, and the way many have approached solving real problems is to parameterise the source profile and lens mass distribution. Parameterisation of the source and mass forces solutions to be smooth. However if the source is complex the number of parameters required to describe it could be very high. This was taken to an extreme by Tyson, Kochanski and dell'Antonio, who used 232 parameters to describe the source light profile lensed by C1 0024 + 1654 in 1998 [19]. Without having prior knowledge of the source it can be very difficult to select appropriate and adequate parameters. The other approach is to pixelate the source, with each pixel becoming a parameter. This means there is no requirement to find a good parameterisation, and removes bias from the choice of parameterisation. However the resolved light source can be noisy, which is overcome by adding a term to the merit function to "regularise" it. The solution is found by balancing the minimisation of the source light profile and the minimisation of the regularising term. This should mean the resolved light source

is smooth. This method was used by Wallington, Kockanek, and Narayan in 1996, solving for the source light profile and mass distribution of the radio Einstein Ring MG 1654 + 134 [20].

The approach that Wallington et al used in the paper mentioned above was to minimise the negative of the entropy (negentropy). The counts of each source pixel, i , are labelled by s_i , and the negentropy is $\sum_i s_i \ln(s_i)$. The merit function is;

$$G = \chi^2 + \lambda \sum_i s_i \ln(s_i) \quad (2)$$

Where χ^2 is calculated as in equation 1 and the λ coefficient is a generic weighting term. The algorithm they used was built so that λ decreased incrementally outside the χ^2 minimisation. For more efficient running speed they also nested the loops in which the mass profile parameters and source light pixel parameters were minimised, so the pixels χ^2 was minimised for a fixed mass profile then the mass parameters χ^2 minimised for a fixed source, meanwhile the λ was decreased outside (as the higher the λ higher the χ^2) until a sufficiently low χ^2 was found to ensure an accurate result. In 2006 Suyu et al showed how Bayesian statistics could be used to find the best λ , aiding in source regularisation [21].


In 2003 Warren and Dye found a more efficient simplified method, replacing the negentropy term with a linear regularisation term. They showed that for a fixed mass profile the source could be found through a matrix inversion of the arc on the lens. This makes the minimisation routine a lot more efficient and therefore can be carried out for more iterations to find a better fit. They also found that thanks to the magnification and multiple imaging, enough constraints could be applied to the system to eliminate the λ loop entirely. This meant that the merit function was just χ^2 . This means that thanks to the linear inversion this 'semi-linear' process a minimum is found for the inversion of the light source on a fixed mass, and then the minimum of these minima is found for the mass profile parameters. This is much faster allowing the algorithm to be much more accurate and precise, and also unbiased as there is no attempt to smooth the result [22].

The following year Treu et al combined the gravitational lensing and dynamic analysis of the velocity dispersion in the lensing galaxy or cluster to constrain a mass profile of early type galaxies and clusters with the aim of finding the amount and distribution of dark matter in said galaxies and clusters [23]. In 2007 Barnabé et al used a hybrid Markov Chain Monte Carlo and Simplex method with stellar dynamics and

Then in 2009 Suyu et al pixelated both the source light profile and the mass distribution, solving for both simultaneously by making perturbative corrections to the starting model combined with the aforementioned Bayesian statistical analysis [25]. This was applied to B1608 + 656 observed by the Hubble space telescope, and successfully reconstructed its lens mass distribution and source light profile. This method is of particular use in obtaining the Hubble constant from time-delay lenses.

2.1 GRAVITATIONAL LENSING

2.1 GRAVITATIONAL LENSING

$$\alpha = \frac{4Gm}{c^2 h} \quad (3)$$


A diagram showing a particle of mass m (represented by a black dot) at a vertical distance b from a curved surface. A blue arrow points from the particle to the surface, and another blue arrow points from the surface downwards. The surface is a smooth curve that starts flat on the left and curves upwards to the right. A small 'x' marks the point on the surface directly below the particle.

Whilst in our case, as mentioned in the introduction, the depth of the lensing galaxy/galaxy cluster is negligible compared to the distances between source and observer, so the thin lens approximation can be made. In the thin lens approximation the deflection of the light is instantaneous at the lens. as shown in Figure2.

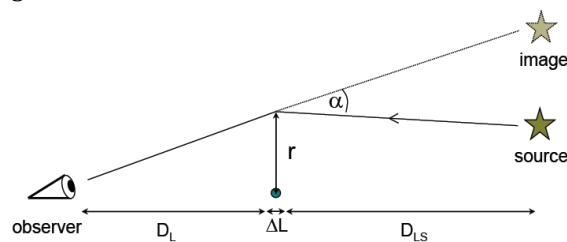


Figure 2: Thin lens approximation

$$\beta = \underline{\theta} - \underline{\alpha'} \quad (4)$$

distances are angular diameter distances

Figure 3: Thin lens approximation in 2D

The two equations used in our investigation to model the lens mass distribution are for isothermal lenses, one being the circular

equation, and the other the elliptical. The equations were found by taking the solution from Schwarzschild (equation 1) and integrating it with the mass profile varying by radius (or other parameters) [26]. So the equations for the deflection angle of an isothermal lens with a circular mass profile whose density decreases proportional to $1/r$ (in Cartesian coordinates) are;

$$\phi_x = x - \frac{mx}{\sqrt{x^2+y^2}} \quad (5)$$

$$\phi_y = y - \frac{my}{\sqrt{x^2+y^2}} \quad (6)$$

And for an elliptical isothermal mass profile the equations are;

$$\phi_x = x - \frac{mq}{\sqrt{1+q^2}} \tan^{-1} \left(\frac{x\sqrt{1+q^2}}{\psi} \right) \quad (7)$$

$$\phi_y = y - \frac{mq}{\sqrt{1+q^2}} \tanh^{-1} \left(\frac{y\sqrt{1+q^2}}{\psi} \right) \quad (8)$$

2.2 MINIMISATION ALGORITHMS

This section details the theory that describes the minimising algorithms compared in this report.

2.2.1 GRID SEARCH

A grid search simply places a number of evenly spaced points throughout parameter-space, finds the figures of merit at each of them and identifies the lowest. The coordinates of this point are then the parameter values for the objective function minimum.

A grid search has two major downfalls. The first is that its accuracy is limited by the distance between points (its resolution). The second is the fact that the number of calculations of parameter-space position merit value increases as the power of parameter-space dimensions (number of parameters). These two problems make the minimisation of large numbers of parameters take an extremely long time, more so if a high degree of accuracy is required. To solve these twin constraints other algorithms can be used.

2.2.2 THE DOWNHILL-SIMPLEX

ALGORITHM

The Downhill-simplex algorithm deals with multidimensional minimisation by finding its way through the topography of parameter-space toward a minimum. It does this in the following sequence of steps. [27] [28]

A simplex is placed at a random starting point in parameter-space. This simplex is an N dimensional shape for N dimensional parameter-space and has polygonal faces. When $N = 2$ the simplex will be a triangle, for $N = 3$ it is a tetrahedron.

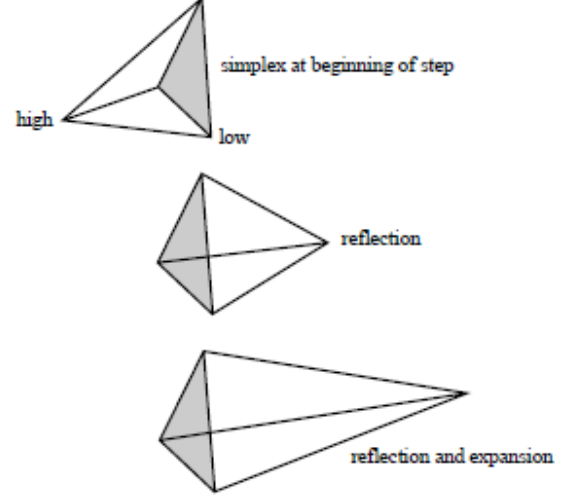


Figure 4: Three dimensional (N=3) simplex reflection and expansion [27]

The simplex has $N + 1$ vertices each of which has its own parameter-space coordinates and hence its own topographical value. When the algorithm starts, these vertices are spread out in parameter-space. The largest valued of the vertices is then reflected through the opposite face of the simplex, conserving the simplex volume (figure 3). If, when this occurs, the new vertex position is lower valued than any of the simplex's other vertices, the simplex will expand. This expansion will be by a factor of the expansion coefficient moving the vertex out by the expansion coefficient times the vertex's distance to the centre of the simplex (figure 4). This expansion will only occur if the new vertex position is lower valued than if expansion had not occurred. Once the process of reflection and expansion is finished it begins again with the newly highest valued vertex.

If at any point no lower valued vertex can be found for reflection, the simplex begins to contract (figure 5). It does this by scaling the distance of the highest valued vertex to the simplex centre by a factor of the contraction coefficient. As with expansion, the new position is only accepted if it is lower valued than the original. If the new position is not lower valued the vertex is moved half the distance to the simplex's lowest valued vertex. The simplex will therefore contract into parameter-space "valley floors", shrinking towards its own lowest point.

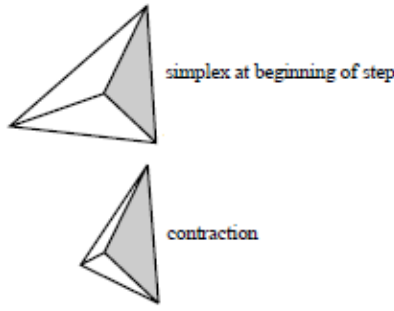


Figure 5: Three dimensional Simplex contraction [27]

The downhill-simplex algorithm stops when the change in value of its lowest valued vertex after a cycle of all vertices moving falls below a predefined tolerance.

2.2.3 THE MARKOV CHAIN MONTE CARLO ALGORITHM

The Markov Chain Monte Carlo (MCMC) method is a commonly used minimisation algorithm which is a fusion of the Monte Carlo method and the Markov Chain method.

The Monte Carlo method was first devised by Ulam and Metropolis in 1949. When used in minimisation it selects random values for the parameters until one of them is the solution. This is not very precise or accurate, and is very slow, not much better than a grid search. [29]

The Markov Chain method was first outlined in a paper in 1906, the title of which translates to "Extension of the law of large numbers on the values that depend on each other" [30]. It is a memoryless stochastic process in which a variable's next value is only based on its current value. The simplest example of a Markov Chain is the one dimensional random walk, in which a point can take a random step either left or right from its current position. Markov Chains however are a much more varied class of algorithms, as they could have different probabilities to take a virtually infinite number of different actions in each step. One option would be to model the behaviour of a woodlouse (which moves faster in the light, and slower in the dark) as a Markov Chain, with the step size larger in areas of higher χ^2 , and smaller in smaller χ^2 , which would cause the point in parameter space to stay in areas of low χ^2 . However Markov Chains are prone to getting stuck in local minima and not finding the global minimum.

A Markov Chain Monte Carlo method incorporates both by taking making a random walk, and taking a random step size, from its current position. If the new position has a lower

χ^2 it will replace the old position, however if χ^2 is higher at the new position it will only take the new value if a random number between zero and one drawn from a uniform distribution is less than $(\chi_2^2 - \chi_1^2)/2$. This method is, like its constituent the Markov Chain, prone to getting stuck in local minima.

2.2.4 GENETIC ALGORITHMS

Genetic algorithms belong to the group known as evolutionary algorithms; algorithms inspired by methods of adaptive change in nature. Genetic algorithms specifically attempt to mimic the natural selection process. They are stochastic algorithms like the Markov chain Monte Carlo algorithm

There are many variations in genetic algorithms. The one used in this report follows these steps. Initially a population of random points with a uniform distribution are generated, each with their own set of parameters. These parameters describe each population member's position in parameter-space and are each the individual genes of that population member.

From the starting population a new generation of positions is derived. To do this the algorithm first values each population member according to whichever function describes parameter-space, in our case χ^2 , and then selects those favourably valued to be parents. Some of the population may be favourably valued enough to be passed on to the next generation without being altered. These are called elite population members. Of those that are not elite, the parents are paired off and genes are randomly selected from each pair to be passed on to their children. A proportion of the child population will be mutants. They will have each of their genes altered via a mutation function. This means that there is a chance to reach otherwise inaccessible parts of parameter-space. The new generation is then evaluated in the same manner and a subsequent population is produced from it. This process is repeated until the termination conditions are met.

There are various termination conditions that can be used with genetic algorithms. The one used for this report was to continue to create successive generations until the change in value of the lowest valued member of the population was smaller than a specific quantity.

2.2.5 SIMULATED ANNEALING

The simulated annealing algorithm is another stochastic algorithm. It simulates the process of heating and slow cooling metal to increase

crystal size and reduce defects. The algorithm therefore has a temperature variable, T .

The algorithm works by generating an initial random point. It then steps from this point in a random direction. The step size is dictated by temperature where the size of the step is equal to the current temperature. The algorithm then values the new parameter-space position with the function to be minimised, χ^2 . If the new position is better/lower valued it becomes the new position. Alternatively if the new parameter-space coordinates have a higher value than those of the previous position the algorithm will only accept the new position with a probability,

$$P = \frac{1}{1 + \exp\left(\frac{\Delta}{\max(T)}\right)}. \quad (9)$$

Δ is the difference in value between new and old positions and T is the current temperature. The acceptance probability is therefore between 0 and 0.5, becoming less likely with either large Δ or small T .

With each step made the algorithm lowers its temperature storing the lowest valued parameter-space coordinates so far. The temperature evolves as

$$T = T_0 \times 0.95^k. \quad (10)$$

k is the annealing parameter. The annealing parameter for each parameter-space dimension,

$$k_i = \log\left(\frac{T_0 \max_j(s_j)}{T_i s_i}\right). \quad (11)$$

s_i is the parameter-space gradient in dimension i . $\max_j(s_j)$ is the maximum gradient of any parameter-space dimension j . This means that the temperature decreases faster in any one parameter when parameter-space has a higher gradient for any of the other parameters.

The algorithm re-anneals after a specified number of points have been accepted. When this happens the annealing parameter is lowered returning the system to a higher temperature in each dimension.

The algorithm terminates when the change in value over a series of points falls below a certain threshold, the maximum number of steps are taken or a set time limit is reached.

2.2.6 NESTED SAMPLING

Nested sampling is a Monte Carlo technique. It has a population of points, the χ^2 of each point is calculated, and then the point with the highest χ^2 is re-plotted with the restriction that the new

point must have a lower χ^2 than it had before [31]. This process repeats until a termination condition is found. The obvious limit of this technique is that as the algorithm proceeds the acceptance rate of new point selection decreases.

In an attempt to overcome this problem Mukherjee et al. in 2006 tried using an ellipsoidal nested sampling technique [32]. This technique proceeds in much the same way as in nested sampling, however the new point at each iteration is selected from inside a multidimensional ellipsoid determined from the current set of points. This means that the acceptance ratio stays higher and increases the efficiency of the code. Figure 6 shows how the ellipsoids may change from iteration to iteration, with the background lines showing the actual contours of the parameter space the algorithm is acting on.

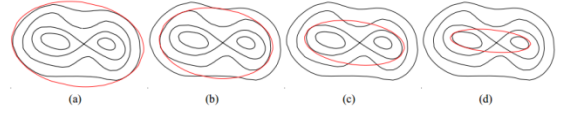


Figure 6: Iteration count increasing from a to d[3].

3. METHODOLOGY

To compare each of the algorithms against each other a model was setup coding in MATLAB or C and each algorithm was applied to it for a variety of different variables.

3.1 MODEL SETUP

For the purposes of the study a source image with known parameters was needed. This allowed for comparisons between the parameters found by minimisation and the actual original values. For this reason a synthetic Gaussian source was created from four parameters. The same values were used across each algorithm once for each minimisation attempt at the same number of parameters tested (one to seven). These parameters were the horizontal and vertical centre position of the Gaussian, horizontal and vertical width of the Gaussian and amplitude/brightness of the Gaussian source. They were a set of evenly distributed randomly generated numbers with predefined boundary conditions. The boundary conditions were Gaussian centre position, $-0.5 \leq x_0 \leq 0.5$ and $-0.5 \leq y_0 \leq 0.5$, Gaussian width $0 < \sigma_x < 1$ and $0 < \sigma_y < 1$ and amplitude $0.5 \leq A \leq 2$. The same boundaries were applied in minimisation when it was a feature of the algorithm. For the minimum values of both

lensing mass and source Gaussian amplitude it was necessary to have values above zero that ensure visible lensing would occur. These were conditions required for the model to represent the strong lensing problem.

The synthetic source was then lensed to create a synthetic image. The lens was described by more evenly distributed randomly generated parameter values. The lens was isotropic and only defined by two parameters. These were the lens mass, M and lens ellipticity, q . q is the ratio of height to width of the lens. The boundaries on these parameters were $0 < M < 1$ and $0 < q < 1$.

To create the lensed image each pixel on the image had to be back traced to source using equations 12 and 13.

$$x_{source} = x_{image} - \frac{Mq}{\sqrt{1-q^2}} \arctan\left(\frac{\sqrt{1-q^2}x_{image}}{\psi(x_{image}, y_{image}, q)}\right) \quad (12)$$

$$y_{source} = y_{image} - \frac{Mq}{\sqrt{1-q^2}} \operatorname{arctanh}\left(\frac{\sqrt{1-q^2}y_{image}}{\psi(x_{image}, y_{image}, q)}\right) \quad (13)$$

Where $\psi(x, y, q) = \sqrt{q^2x^2 + y^2}$

The x_{image} and y_{image} coordinates of each image pixel were placed in these equations, alongside M and q parameters, to find the x_{source} and y_{source} coordinates of the source pixel that corresponded to it. The brightness of the source pixel was then used for that of the image.

$$x_{source} = x_{image} - M \frac{x_{image}}{r(y_{image}, x_{image})} \quad (14)$$

$$y_{source} = y_{image} - M \frac{y_{image}}{r(y_{image}, x_{image})} \quad (15)$$

Where $r(x, y) = \sqrt{x^2 + y^2}$

When there are five or fewer parameters, equations 14 and 15 can be used to back trace pixels and where in the case of the nested Markov chain Monte Carlo sampling, multi-nested Markov chain Monte Carlo sampling and multi-nested genetic Markov chain Monte Carlo sampling algorithms. These two sets of equations are equivalent when q is set to the limit of 1. q cannot be set to 1 however as equations 12 and 13 would not return results. The Grid search, downhill-simplex, MCMC, genetic, simulated annealing and genetic/simplex hybrid algorithms all used equations 12 and 13 with a value of 0.99 for minimisations with parameter numbers fewer than or equal to five.

Randomly generated Gaussian noise was then added to the image to model the instrumentation noise inherent in real world lensing problems. This noise had standard

deviation in amplitude of 0.1. The result was a 100 by 100 pixel image like that seen in figure 7.

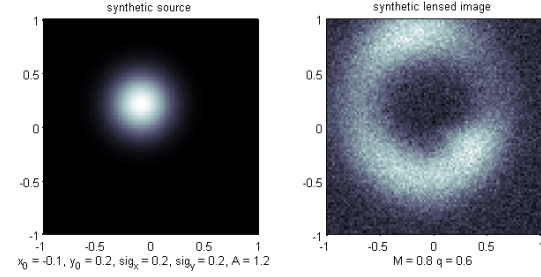


Figure 6: Synthetic source and image

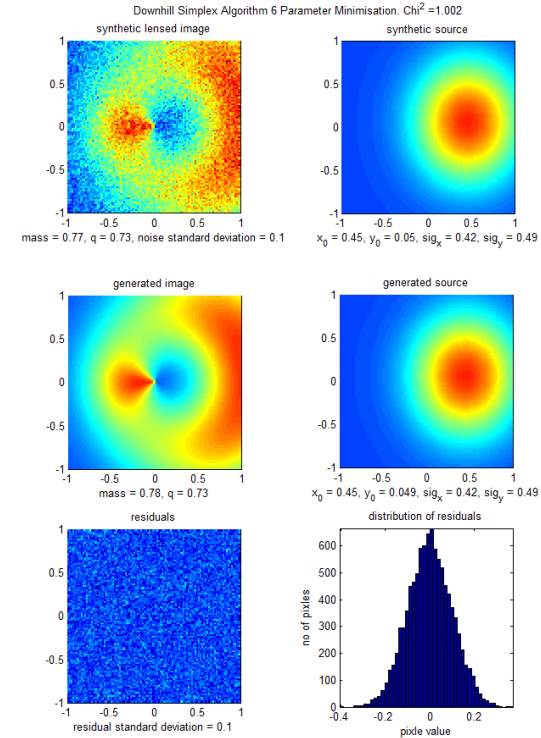


Figure 7: A successful six parameter minimisation

Once a synthetic image was created each minimising algorithm was applied to try to duplicate it by varying the same one to seven parameters. To reflect the real world problem this was done only by comparison between the synthetic image and the model image the algorithm adjusted parameters would generate. This comparison was done with the $\chi^2_{reduced}$ statistic. For each new set of parameters a new model image would be generated and then the following equation applied

$$\chi^2_{reduced} = \sum_i \frac{(f_i^{mod} - f_i)^2}{\sigma^2 N} \quad (16)$$

f_i^{mod} is the flux on the modelled image at pixel i , f_i is the flux on the original synthetic image at pixel i , σ is the standard deviation in noise and N is the total number of pixels. The $\chi^2_{reduced}$

value is smaller with fewer differences between images. This is the value that is minimised. Once the minimum $\chi_{reduced}^2$ was found, the modelled lensed image was subtracted from the synthetic lensed image. The remaining residual differences would then be, for successful minimisation, the same random Gaussian noise applied to the synthetic image in its generation. The residual image would therefore have the same standard deviation in amplitude of 0.1. This value was used to check the success of each minimisation attempt.

3.2 IMPLIMENTATION OF MINIMISATION METHODS

To minimise $\chi_{reduced}^2$ the algorithms described in section 2.2 were applied. Every algorithm was implemented into the code and ten or fifty sets of synthetic image parameters were tested with each. For the following algorithms the number of sets of synthetic image parameter values tested with each number of parameters was fifty: the grid search, downhill-simplex, Markov chain Monte Carlo, genetic, simulated annealing and the hybrid genetic/downhill-simplex algorithms. For the other algorithms the number was ten.

3.2.1 GRID SEARCH

A grid search was applied with two different parameter-space resolutions. The higher resolution run had 101 evenly spaced positions used as values for each parameter. The values were different across parameters, though the same across different numbers of parameters. This setup was applied from 1 to 3 parameters.

The lower resolution run used 11 evenly spaced values for each parameter. This meant it could be run for more parameters, one to five, in a reasonable time.

3.2.2 THE DOWNHILL-SIMPLEX ALGORITHM

The downhill-simplex algorithm was implemented via the inbuilt *fminsearch* MATLAB algorithm. This uses the downhill Nelder Mead simplex algorithm as described by J. C. Lagaris et al. [33]. In this version the coefficients of expansion are, $\chi = 2$ and contraction, $\gamma = \frac{1}{2}$.

No boundaries could be set so nonphysical local minima could be found instead of physical global minima.

The tolerance value of the termination condition was 10^{-6} .

3.2.3 THE MARKOV CHAIN MONTE CARLO ALGORITHM

The Markov chain Monte Carlo algorithm was implemented with boundary conditions that were largely the same as those of the random parameter variables used to generate the synthetic image. The exceptions to this were the lens mass, M and source brightness, A where a lower minimum was in place. The boundary conditions imposed were the source Gaussian centre position, $-0.5 \leq x_0 \leq 0.5$ and $-0.5 \leq y_0 \leq 0.5$, source width $0 < \sigma_x < 1$ and $0 < \sigma_y < 1$, amplitude $0 \leq A \leq 2$, lens mass $0 \leq M \leq 1$ and lens ellipticity $0 < q < 1$.

The standard deviation in step size was set at 0.05 times the distance between the upper and lower boundary of each parameter

For the calculation that determined whether a step was taken when the new position had a higher $\chi_{reduced}^2$ value than that of the old a different probability distribution was used. Instead of the uniform distribution for random number generation described in section 2.2.3 a Gaussian distribution was used.

The termination conditions for the Markov chain Monte Carlo algorithm were twofold. If the algorithm found a set of parameters value at $\chi_{reduced}^2 < 1.05$, twenty more steps were taken. If the coordinates still produced $\chi_{reduced}^2 < 1.05$ after those twenty steps the algorithm stopped. The lowest valued of the last twenty parameter-space coordinates was then used as the minimum. The value of 1.05 was chosen as the minimum $\chi_{reduced}^2$ will be different for any individual minimisation because of the random Gaussian distribution of noise. $\chi_{reduced, min}^2 = 1 \pm \sqrt{2/n}$ where n is the number of pixels in each image. $n = 10\,000$, hence $\chi_{reduced}^2 = 1 \pm 0.14$. The 0.5 tolerance therefore covers more than 3 sigma positive variation from the theoretical mean minimum of 1. Alternately the algorithm would terminate after a total 10 000 steps were taken. If this happened the minimum of the 10 000 previous positions was used as the minimum.

3.2.4 GENETIC ALGORITHM

The genetic algorithm used for this report was the *ga* algorithm provided in the MATLAB global minimisation toolbox.

The population size was kept constant in each minimisation. Its size was fifty unless stated otherwise. The number of elite members in any population was two.

Twenty percent of each child population were mutants. The mutation function was Gaussian with a standard deviation for the

i th parameter coordinate of the k th generation of $\sigma_{k,i}$.

$$\sigma_{k,i} = \sigma_{k,i-1} \left(1 - \frac{k}{G}\right)$$

G was the maximum number of generations, set to 1000. The starting value was, $\sigma_{0,i} = 2$. Every gene in the mutant population was mutated in this fashion.

There were two termination conditions for *ga*. The first was that if the maximum generation count of one thousand was reached the algorithm would stop. The second was that if the change in value of the lowest population member over fifty generations dropped below 10^{-10} the algorithm would stop.

3.2.5 SIMULATED ANNEALING

The simulated annealing algorithm, similarly to the genetic algorithm, was from the MATLAB global minimisation toolbox.

The same boundary conditions were used between genetic and simulated annealing algorithms. They were the source Gaussian centre position, $-0.5 \leq x_0 \leq 0.5$ and $-0.5 \leq y_0 \leq 0.5$, source width $0 < \sigma_x < 1$ and $0 < \sigma_y < 1$, amplitude $0 \leq A \leq 2$, lens mass $0 \leq M \leq 1$ and lens ellipticity $0 < q < 1$.

The starting position of the algorithm could not be specified. Instead the algorithm generated its own set of starting parameters for every individual minimisation from a uniform random distribution with boundary conditions as described above. Re-annealing was set to occur after every one hundred steps

The termination conditions were such that the algorithm ran until the average change in $\chi^2_{reduced}$ was less than 10^{-6} over $500 \times P$ iterations, where P was the number of parameters.

3.2.6 HYBRID GENETIC AND DOWNHILL-SIMPLEX ALGORITHM

A coarse genetic search was followed by a downhill-simplex minimisation. The simplex started at the minimum point of the initial genetic search.

The genetic portion had the same constant population size, mutation function, elite population size and maximum number of generations as the solely genetic algorithm in 3.2.4. The only exceptions to their similarities were a smaller population size of ten and a larger termination tolerance of 10^{-2} .

The downhill-simplex portion of the algorithm only differed from the downhill-simplex algorithm described in section 3.2.2 in that it's starting position in parameter-space

was the minimum valued point found by the genetic part of this combined algorithm.

3.2.7 NESTED MARKOV CHAIN MONTY CARLO SAMPLING

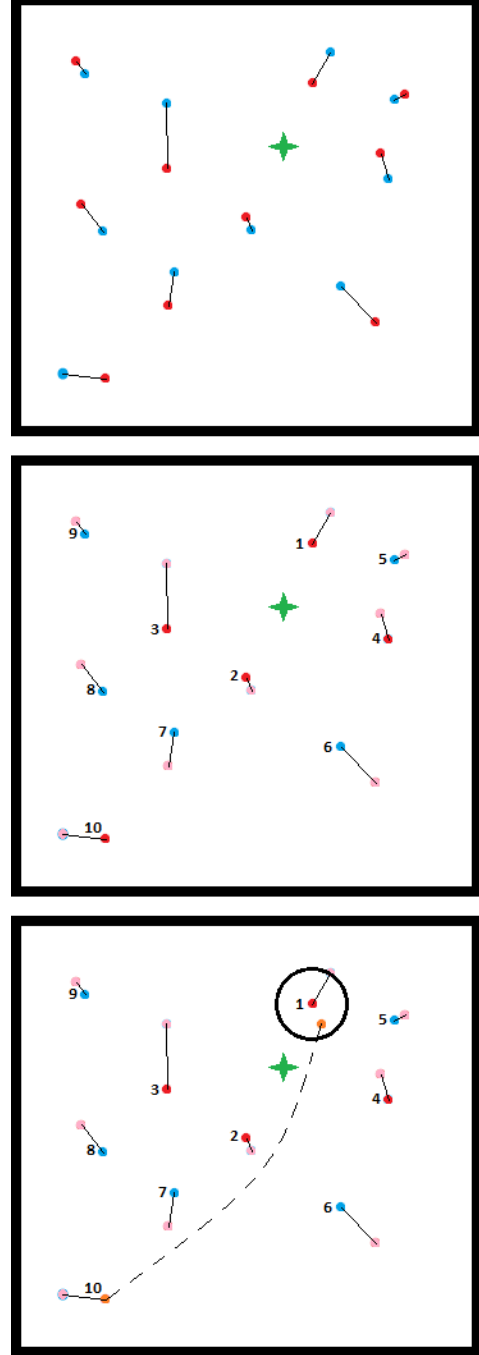


Figure 8: One iteration of NCMCS

Since a major issue with MCMC is that it often gets stuck in the first minima it finds and it cannot effectively sample a large area of the parameter space, it seems logical to combine it with nested sampling. This algorithm was written to attempt to combine MCMC with

nested sampling. The population of points (100 points) was randomly assigned to the parameter space by a uniformly distributed set of random numbers. Each point then took its own step in parameter space as normal in MCMC. For a more thorough explanation of the steps required see section 2.2.3. The points with the maximum and minimum χ^2 were then found. In normal nested sampling the worst point would then be re-plotted in the parameter space from a uniform random number distribution, constrained by it having to have a lower χ^2 than previously. In an attempt to optimise this process the point was re-plotted using a uniform random number distribution within a hypersphere radius of 0.1 of the point with the lowest χ^2 . This means the code was not slowed down by having to re-plot the point over and over until it stumbles upon a point with lower χ^2 . This process then repeated from the MCMC steps of the point population. The code was set to terminate when it either reached 300 iterations (or 400 in the 7 dimensional case), or when it got below $\chi^2 = 2$. Figure 9 shows pictorially how the algorithm proceeds. The red dots in frame 1 are the initial points, the blue the movement assigned randomly, and the green star is the solution. In frame 2 the dots in pink are the point values that are subsequently deleted, the others becoming the new points and being ranked from best (1) to worst (10). The third frame highlights in orange the worst point and a dashed line leads to where it ended up being re-plotted. The black circle around point 1 is the region in which the orange point 10 could have been plotted. This algorithm would probably have been effective without re-plotting the worst point, however continuing with the MCMC walk of a point which is almost certainly nowhere near the solution would have been a waste of computing time. The theory was that by re-plotting the worst point each time there would be less degenerate points, and the population required to thoroughly search the space would be lower.

3.2.8 MULTI-NESTED MARKOV CHAIN MONTY CARLO SAMPLING

The MNMCMCS algorithm was written as a slight variation of the NMCMCS algorithm. It seemed naïve to re-plot only the worst point when in all likelihood many of the points were in areas of parameter space nowhere near the solution. It also seemed naïve to assume that the best point was near the solution, especially in the higher dimensional parameter spaces. Therefore, the MNMCMCS algorithm proceeded as the NMCMCS algorithm up to the re-plotting of the worst

points. At this point the worst 10 points were re-plotted, each one within a hypersphere radius 0.1 from each of the best 10 points. Figure 10 shows how the algorithm proceeds much as NMCMCS for the first 2 frames, but in the third the worst 3 points are re-plotted near the best 3. The theory was that this would mean more of the parameter space that was more likely to hold the solution was more thoroughly searched.

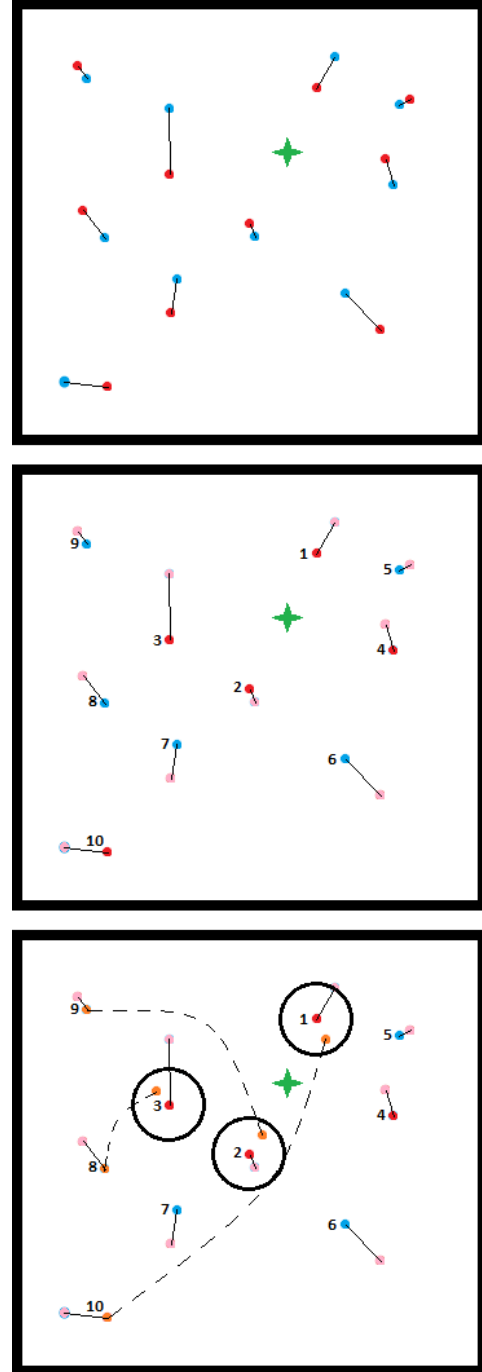


Figure 9: One iteration of MNMCMCS

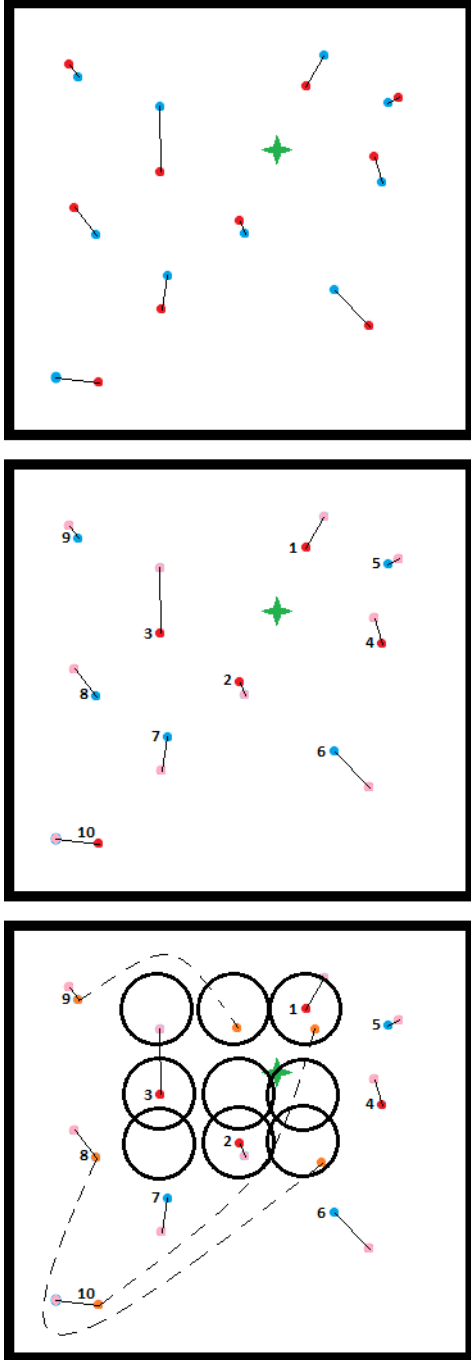


Figure 10: One iteration of MNGMCMCS

3.2.9 MULTI-NESTED GENETIC MARKOV CHAIN MONTY CARLO SAMPLING

The MNGMCMCS algorithm was a further advancement on the NMCMCS algorithm from the MNMCMCS algorithm. Like the MNMCMCS algorithm the worst 10 points were re-plotted, however instead of just being near the best 10 points, the 10 best points were turned into genetic algorithmic parents. The genes of the parents (the parameters) were mixed as though there were 10 genders, so the parents were not paired. All genes of all 10 parents were mixed to

form 10 children (who were previously the worst 10 points). After genetic mixing the children were taken as the centre of the hypersphere in which the new points were plotted. Figure 11 shows how initially the MNGMCMCS algorithm proceeds like the other two nested sampling algorithms. In the third frame the black circles show the areas of the parameter space that the worst 3 points could be plotted to depending on how the genes mix in the breeding stage. The theory was that the genetic mixing would mean the algorithm would search the parameter space even more thoroughly without having to increase the population.

4. RESULTS

The results were obtained by running each of the algorithms on a variety of different parameter-space topography. The order of parameters, one to seven, used to describe this topography and test the algorithms, was $x_0, y_0, \sigma_x, \sigma_y, M, q, A$. For P parameter minimisation, the first P parameters in the above list were the variables in $\chi^2_{reduced}$ minimisation, therefore five parameter minimisation was the minimisation of $\chi^2_{reduced}$ in parameter-space with demotions of source image Gaussian centre and width, x_0, y_0, σ_x and σ_y , as well as the mass of the lens, M . The exception to this was three parameter minimisation, which had the alternate parameter order, x_0, y_0, M . This better reflected real-world problems where lensing mass is more often a sort after parameter than the source width in one dimension.

4.1 GRID SEARCH

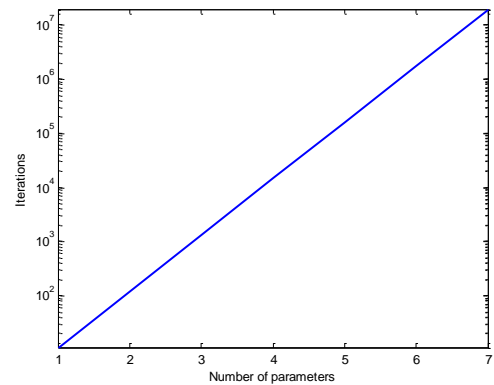


Figure 11: The number of iterations required for the gridsearch algorithm to complete

The grid search always took R^P iterations to complete, where R is the resolution in each of the P perimeter-space dimensions. As the resolution, $R = 11$ for this report, the number of iteration to complete the grid search was that shown in figure 12. The lowest number of iterations is at one parameter and is 11 and the highest is at seven parameters and is 1.95×10^7

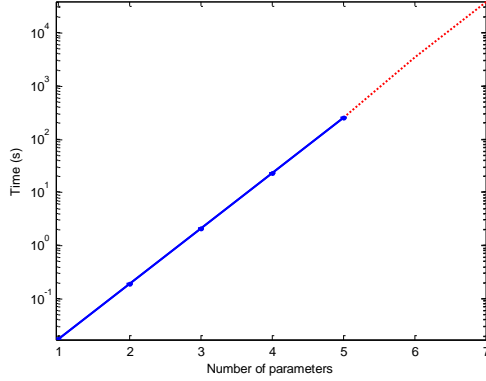


Figure 12: Grid search runtime

The grid search completion time for every parameter number is shown in figure 13, with the minuscule errors being Poisson errors. The six and seven parameter runtimes were found by extrapolation from the previous five as the time to complete would've been very long. The relationship between grid search runtime, t and parameter number, P was found to be $t = 11^P \times 1.56 \times 10^{-3}$ s. 1.56×10^{-3} s being the time for a single iteration to complete.

The shortest runtime was for one parameter at $0.0173 \pm 5 \times 10^{-4}$ s. The largest was 3.861×10^4 s for seven parameters.

4.2 DOWNHILL-SIMPLEX, MCMC, GENETIC AND SIMULATED ANNEALING ALGORITHMS

Successful minimisations were considered to be ones with a final $\chi^2_{reduced} \leq 1.05$, a residual image with pixel brightness distribution standard deviation of 0.1 and values within the boundary conditions $-0.5 \leq x_0 \leq 0.5$, $-0.5 \leq y_0 \leq 0.5$, $0 < \sigma_x < 1$, $0 < \sigma_y < 1$, $0 \leq A \leq 2$, $0 \leq M \leq 1$ and $0 < q < 1$.

The robustness of each algorithm can be described by their success rate. The success rate of each algorithm at each number of parameters is shown in figure 14.

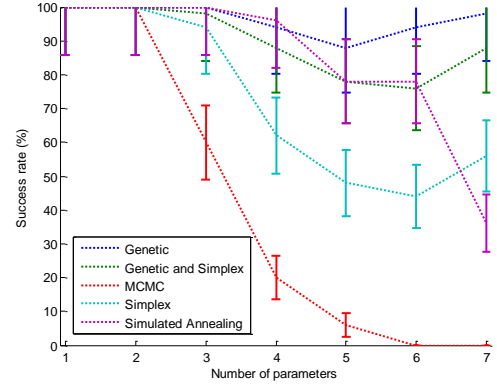


Figure 13: Minimisation success rate of different algorithms

Figure 14 shows that successful minimisation rate decreases for all algorithms as the number of parameters increases, though there is a gain in success rate with the addition of a seventh parameter for the genetic, simplex and hybrid algorithm comprised partly of both. The genetic also increases in success rate with an increase from five up to six parameters going from $88 \pm 13\%$ to $94 \pm 13\%$ and then $98 \pm 14\%$ at seven parameters. The errors shown on this graph are Poisson errors.

The MCMC algorithm is the worst performer across all numbers of parameter minimisation, though it does have a $100 \pm 14\%$ with all the other algorithms for one and two parameter minimisation. At three parameters the algorithm has sunk to a $60 \pm 11\%$ success rate and at four parameters $20 \pm 6\%$. The algorithm hits 0% success rate at six parameters and has the same for 7.

Simulated annealing has the highest success rate from one to four parameters. The genetic algorithm then takes the lead and maintains the highest successful minimisation rate throughout five to seven parameter minimisations, although the two algorithms are only out of each other's calculated errors for six and seven parameter minimisation. At seven parameter minimisation the simulated annealing algorithm becomes much less successful, dropping from $78 \pm 12\%$ at six parameters to $36 \pm 8\%$ at seven.

The simplex algorithm was the next worst for success rate after MCMC. Its lowest rate was $44 \pm 9\%$ at six parameters. The genetic/simplex hybrid algorithm maintained a high success rate with a low point of $76 \pm 12\%$ also at six parameters.

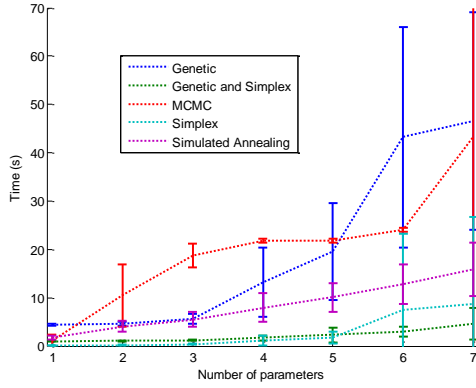


Figure 14: Runtime for successful and unsuccessful minimisation with the different algorithms.

The speed of the algorithms is shown in figure 15. The error bars in this graph are set at the standard deviation in the distribution of the fifty minimisation attempt runtimes for every position.

It can be seen in figure 15 that the minimisation time for all algorithms increases with parameter number. The amount of increase is however quite different. The smallest change was that of the hybrid genetic and simplex algorithm. Its lowest value was 0.98 ± 0.08 s for one parameter minimisation which grew to 4.6 ± 3.2 s at seven parameters. The largest change is in the genetic algorithm going from runtimes of 4.3 ± 0.2 s for one parameter minimisation to 47 ± 23 s at seven parameter minimisation.

For most algorithms the gradient of the change in minimisation runtimes between parameter numbers rose with parameter increase. The Markov chain Monte Carlo algorithm was the exception to this. A termination condition for the MCMC algorithm was a maximum generation count. As it hit this limit more and more often the runtimes levelled off. This resulted in runtimes of 21.7 ± 0.4 s at four parameters, increasing slightly to 23.0 ± 0.4 s at six parameters. At seven parameters the algorithms' variance in runtime dramatically increased with measurements of 43 ± 118 s. This is due to one runtime of 872s. Removing this result from the calculation provides a seven parameter runtime of 26 ± 2 s

The fastest algorithm from one to five parameters was the downhill-simplex, though the standard deviation in times at four and five parameters did overlap with genetic and simplex hybrid algorithm. The overlapping values were 1.2 ± 1.0 s for simplex and 1.7 ± 0.5 s for genetic and simplex at 4 parameters, 1.7 ± 1.2 s for simplex and 2.3 ± 1.7 s for genetic and simplex at five parameters. The fastest algorithm for six and seven parameter minimisation was the simplex and genetic. It had

3.0 ± 1.0 s runtimes for six parameters and 4.6 ± 3.2 s runtimes for seven. However for both six and seven parameters the standard deviation in simplex was very large, completely overlapping the genetic simplex distribution.

The slowest algorithm overall was the genetic. It took the largest amount of time in all but two, three and four parameter minimisation. It had a 4.4 ± 0.2 s minimisation time at one parameter and rose to 47 ± 23 s minimisations at seven parameters. For two to four parameter minimisations the MCMC algorithm took longest with 7.2 ± 6.1 s runtimes for two parameters, 16 ± 5 s runtimes for three and 21 ± 7 s at four.

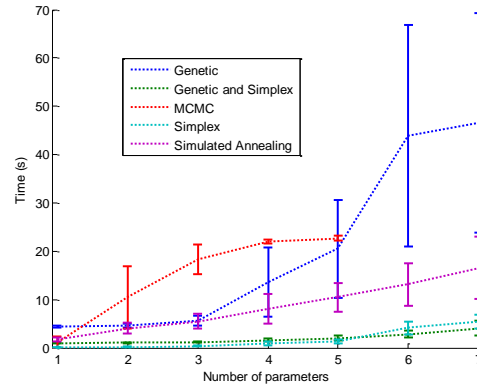


Figure 15: Runtime for successful minimisation with the different algorithms

Isolating just the successful minimisations for each parameter provides the comparative runtimes of only the minimisation attempts that found the correct solution (figure 16). The most notable change between this and the previous graph is in the runtimes of six and seven parameter minimisation for the downhill-simplex algorithm. The values for runtime including unsuccessful minimisation attempts were 8 ± 16 s for six parameters and 8 ± 18 s for seven parameters. Removing unsuccessful runs yielded runtimes of 4.1 ± 1.3 s for six parameters and 5.3 ± 1.4 s for seven parameters.

Other differences between the runtimes with and without unsuccessful minimisations removed include a lack of six and seven parameter values for the MCMC algorithm. This was due to the low success rates of the Markov chain Monte Carlo algorithm at larger numbers of parameters (figure 14). With no successful minimisation attempts at six and more parameters there are no data points to plot.

Figure 17 shows the iteration numbers for successful minimisations. The error bars in this graph are also set at the standard deviation in the distribution of the fifty minimisation attempt runtimes for each plot position.

The graph shows the efficiency of each algorithmic step. The two random walk

algorithms took significantly more steps to minimise $\chi^2_{reduced}$ than the other three, both showing an approximately linear increase in iterations with rising parameter-space dimensions. For simulated annealing this meant a one parameter minimisation iteration count of 860 ± 190 rising to a seven parameter count of 7300 ± 2800 via a $I = 1035.1P - 308.9$ relationship. I is the iteration count and P the number of parameters. In the case of MCMC this is cut off at four parameters as it begins to hit its iteration limit of 10 000. From one to three parameters the relationship is $I = 4406.5P - 3715.9$.

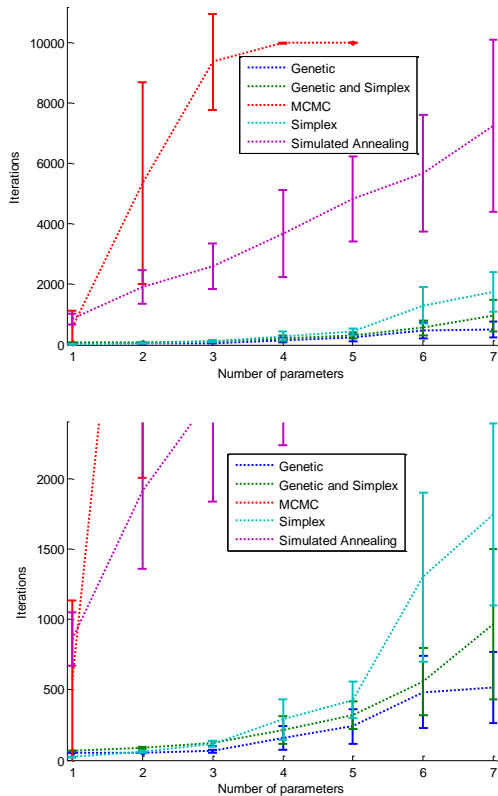


Figure 16: Number of iterations required to successfully minimise $\chi^2_{reduced}$ for different algorithms.

(The bottom graph is the lower section of the top)

The other three algorithms have more powerful individual iterations as fewer are necessary to find a parameter space-minimum. Unlike the simulated annealing and MCMC algorithms they do not show a linear relationship between parameter number and iterations to complete. The algorithm with the fewest iterations; from two to six parameters is also the slowest. It is the genetic algorithm with 53 ± 3 iterations at two parameters rising to 510 ± 250 at seven. Of the non-linear algorithms the simplex takes the most iterations; from four to

seven parameters. At four parameters it takes 290 ± 140 iterations. At seven it takes 1700 ± 600 . Of the three the simplex/genetic hybrid algorithm started with the highest iterations to minimisation but came between the two others at four parameters with 210 ± 99 iterations to successful minimisation. It stayed between them from four up to seven parameters with 960 ± 530 iterations.

4.3 GENETIC ALGORITHM

POPULATION COMPARISON

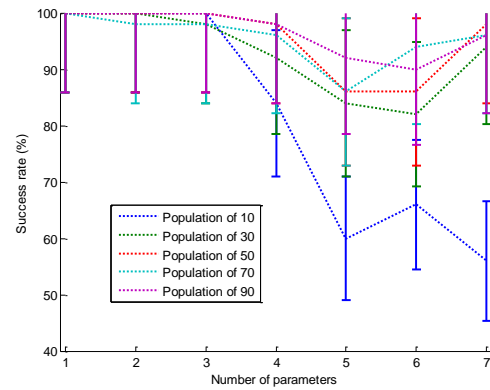


Figure 17: Minimisation success rate for the genetic algorithm at different population sizes.

Results were taken from different population sizes for the genetic algorithm (figures 18 to 20). The errors in success rate are Poisson errors for fifty repetitions of each algorithm at each number of parameters. The errors in iteration and time for minimisation are the standard deviation in the distribution of each point.

Figure 18 shows the minimisation success rate for these different population sizes. The success decreased with parameter number gain for all population sizes between one and five parameters. However between five and six parameters populations of both 10 and 70 rose in success rate, though this was within the error width of each. For the population of 10 the rate went from $60 \pm 11\%$ to $66 \pm 11\%$ and for the population of 70 the rate rose from $86 \pm 13\%$ to $94 \pm 14\%$. For populations of 30, 50, 70 and 90 there was also a rise in success rate between six and seven parameter minimisations. Once again however this increase was matched by an increase in the error of the success rate for each of the population sizes at this number of parameters.

The population size of 10 was the least successful with more than three parameter optimisation. It dropped from $100 \pm 14\%$ at three parameters to $84 \pm 13\%$ at four and finished at $56 \pm 11\%$ at seven parameters. The other populations all remained within error of

each other though larger populations have a slightly higher success rate within that error width. The population of 90 maintained the highest success rate for all but six and seven parameter minimisation where it was lower than 70 and 50 population sizes respectively. At six parameters the population of 90 had a $90 \pm 14\%$ success rate and the population of 70 a $94 \pm 14\%$ rate. At seven parameters the 90 population genetic algorithm achieved a $96 \pm 14\%$ success rate and the population of 50 had a $98 \pm 14\%$ rate.

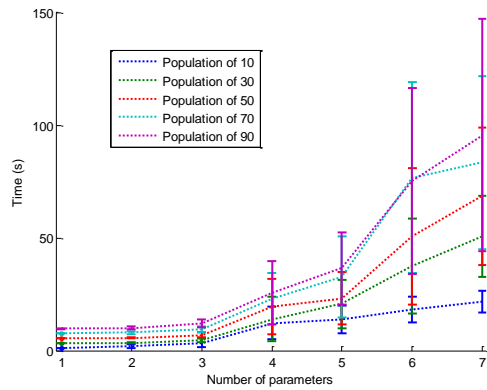


Figure 19: Time for successful and unsuccessful minimisation with the genetic algorithm at different population sizes.

The runtime for each different population size is shown in figure 19. In this plot it can be seen that the runtime for each parameter number increases with an increase in population size and the runtime for any population size increases with increased parameter number.

Figure 13 shows the number of iterations for successful minimisation for the different population sizes tested. As with figure 18 this shows the efficiency of each step. Larger population sizes found solutions in fewer iterations, except between population sizes of 50 and 70 where they have very similar mean values, though apart from the population size of 10 these iteration numbers are largely within the standard deviation of each other. It should also be noted that the for one to three parameter minimisation all the population sizes, bar the population of 10 members, required extremely close numbers of iterations to minimise. All populations taking 51 ± 0 iterations at one perimeter and a range from 70 ± 14 iterations for a population of 30 to 63 ± 9 iterations for a population of 90 at 3 parameters.

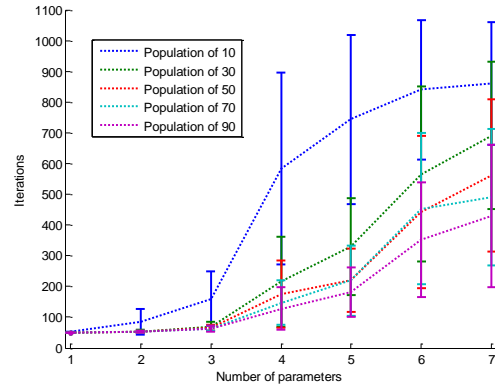


Figure 20: Number of iterations for successful genetic algorithm minimisation at different population sizes

4.4 NESTED MARKOV CHAIN MONTY CARLO SAMPLING ALGORITHMS

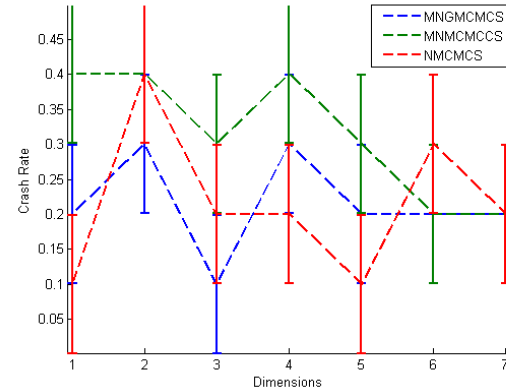


Figure 21: Dimensions vs Crash Rate

Before presenting the findings it is necessary to point out that all the nested sampling algorithms had a bug that could not be found. For some reason sometimes the code would not calculate, at the timeout of the code it would print out the default values of χ^2 set before the simulation. The results of the crashed simulations were filtered out before analysis. It should be pointed out however that the crash rate appears to be constant throughout all results, and not dependant on population or number of dimensions being solved, as can be seen in Figures 21 and 22. Neither did any particular parameter space maps crash more than others, though the reduced number of results may have made the graphs more erratic. If for instance on one particular set of repeats a particularly easy to solve parameter space map crashed then the results for that population or

dimension could be slightly to high, or if a crash occurred whilst one of the harder maps was being solved for the result for that simulation could be too low.

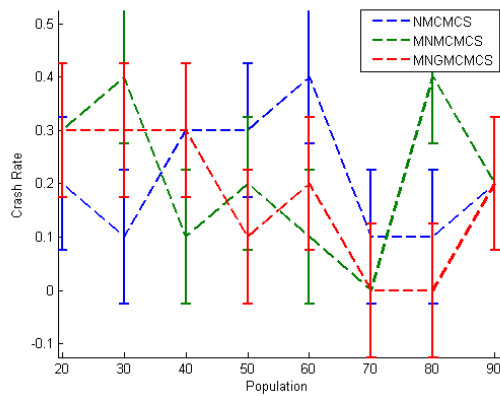


Figure 22: Population vs Crash Rate

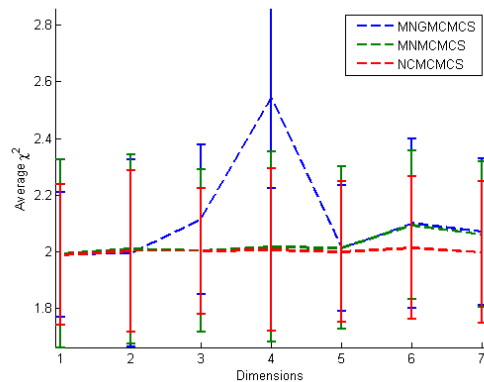


Figure 23: Dimensions vs Average χ^2

First presenting the results for the behaviour and success of the algorithms as the number of dimensions is increased; the χ^2 stays constant as the number of dimensions increases, which is promising as it means the algorithms are still reaching the terminating value as dimensions are increased instead of timing out and struggling to solve the higher dimensional problems. The Algorithm with a genetic component spikes anomalously at 4 dimensions then drops back down again (figure 23). The reason for this is unknown, though it is possible that that particular set of simulations happened to have a lot of the easier maps crash causing the result to be too high. If it is not an anomaly caused by crashes then it could be a characteristic of the way that particular algorithm behaves in 4 dimensions, more

analysis would be required to find a definitive answer.

Figure 24 does not show standard deviation, but since the parameter space maps were created for the algorithms to solve and therefore known it is a measure of the Average deviation of the solution from the values used to generate the map. The results were taken by repeating across different maps for each number of dimensions.

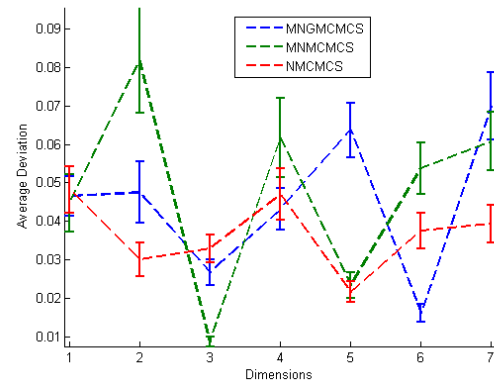


Figure 24: Dimensions vs Average Deviation

The Average deviation does not go above 0.1 (with all dimensional searches normalised so that the searches are between 0 and 1), and has no significant increase as the number of dimensions increases. This shows the algorithms in general are successful, as they are intended as a coarse search to be followed by an algorithm more suited to fine searches, such as a plain MCMC or a Simplex. Both MCMC and Simplex could easily find a very precise and accurate result from that distance from the solution. The fact that the Deviation from the correct answer does not increase with the number of dimensions is also good news, as it means that the algorithms should still be viable at much higher dimensions in terms of robustness.

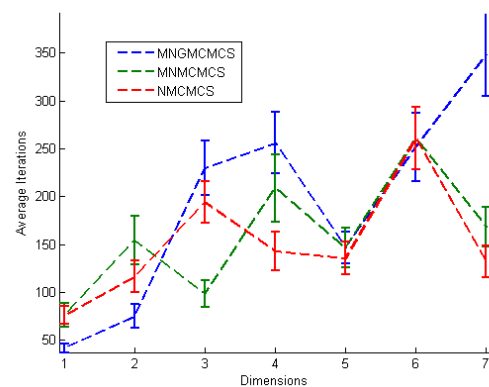


Figure 25: Dimensions vs Average Iterations

The average number of iterations required to reach the termination conditions increases as the number of dimensions increases. This is expected as the population of points is kept constant. As the dimensions are increased the amount of parameter space that has to be searched increases exponentially. It would be expected that the number of iterations required would increase exponentially as a result, however the results are too erratic to say more than there is an increase. Another quirk in the results is that the NCMCMCS and MNCMMCS algorithms find solving in 7 dimensions easier than in 6, the reason for this is not immediately apparent.

The other investigation carried out was the effect of varying the population in each algorithm as this is the main parameter of each Nested algorithm. The experiment was carried out in 4 dimensions, with the population varying from 20 points to 90 points, lower than the 100 used for the dimension experiments. Figure 6 shows the average χ^2 does not vary with the population.

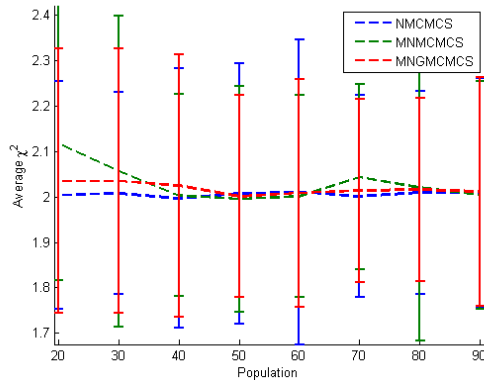


Figure 2618: Population vs average χ^2

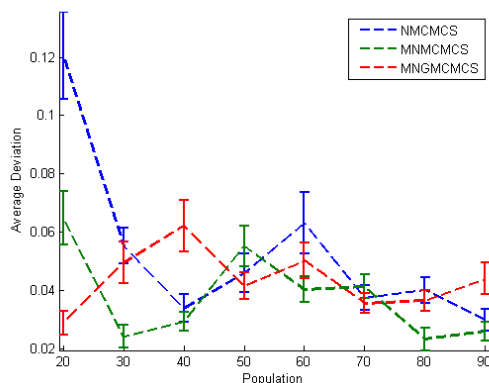


Figure 27: Population vs Average Deviation

The average stays close to 2, the termination condition. So despite the lower

populations the algorithms are still finding low enough χ^2 . It is worth noting that the MNGMCMCS algorithm does not have higher values of χ^2 than the others, which implies that the large result in Figure 23 is indeed an anomaly.

Figure 27 shows that the MNMCMCS and MNGMCMCS algorithms manage to be just as precise and accurate in finding the solutions for the populations tested, showing robustness for reduced populations. However the NCMCMCS algorithm struggles at lower populations, this would be expected as it in theory provides a much less thorough search of the parameter space than the other two as only one of the redundant high χ^2 points is re-plotted to search new areas of the parameter space each iteration, though it is strange that the average χ^2 is not higher in parallel with the higher deviation.

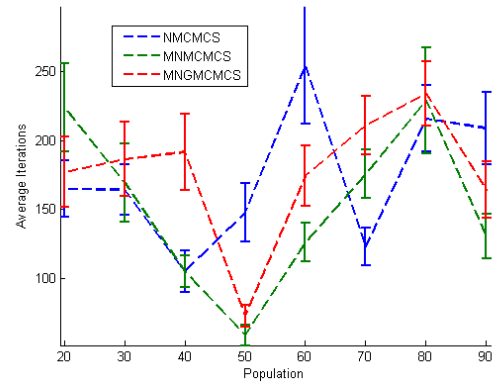


Figure 28: Population vs Average Iterations

The average number of iterations required of the NCMCMCS algorithm appears to fluctuate with no discernable pattern. The MNMCMCS and MNGMCMCS algorithms however show a marked decrease in number of iterations required at a population of 50, both algorithms on average requiring near 50 iterations. This implies that there is an optimum population size for these algorithms, it would be expected that this optimal population size would increase as the number of dimensions increases. As for the reason the algorithm requires higher iterations with more population is a mystery, that is a completely unexpected result. More points plotted in parameter space should mean more of the space is tested in each iteration and therefore less iterations are required before the solution is found.

5. DISCUSSION

From the results above it can be seen that each algorithm would have scenarios that it would be best suited to.

The MCMC algorithm only works with what would likely be considered usable reliability (above 50%) at low numbers of parameters (fewer than three). This success rate at these low parameter numbers is countered however, by its comparatively long runtime. The MCMC is considerably worse at larger numbers of parameters with a success rate diminishing to zero at six parameters (figure 18). It is therefore the recommendation of this report to use other minimisation algorithms in most circumstances.

There is a notable jump in runtime increasing from six to seven parameter minimisation for the MCMC algorithm. This, as mentioned in the results section of this report, is due to one individually large runtime. This may be due to the possibility of additional steepness in parameter-space with every additional dimension. At seven parameters the Markov chain could have found itself trapped within a local minimum with steep enough sides to stop it taking a successful step for the majority of its runtime. This possibility could be dealt with in future uses of the MCMC algorithm by including not just every successful step but every attempted one in the iteration count for termination conditions.

The down-hill simplex algorithm is significantly more successful than the MCMC and the fastest for minimisation in parameter-space of less than five dimensions. Similarly to MCMC however it does drop below the 50% success threshold for larger numbers of parameters hitting a minimum rate of $44 \pm 9\%$ at six parameters. An option to improve the algorithm would be to start multiple simplexes, spread out throughout parameter-space, at the same time. This would likely provide correct solutions more consistently. The time for such an algorithm to complete would be greater but with the initially brief runtime it would still be fast.

The simulated Annealing algorithm maintained a high success rate percentage until it was used to find the minimum in the highest dimension parameter-space. At seven parameters the functions efficiency reduced significantly. This may have been due to the tolerance function not scaling with parameter number. As the parameter number increased so did the size of parameter-space. This would've meant that moving the same distance in parameter-space the parameter values could alter much less whilst failing to find the minimum. If this happened the difference in $\chi^2_{reduced}$ could fall below the tolerance, as

described in section 2.2.5, and the algorithm would terminate.

This would only be true if each step in the seventh additional parameter had a smaller average effect on the overall $\chi^2_{reduced}$ value than any of the other parameters. The seventh parameter is the source Gaussian brightness. Without further analysis it cannot be made certain that it does have a weaker effect on $\chi^2_{reduced}$. The same phenomena however, could also explain the rise in success rate of the genetic, simplex and hybrid algorithms. In each of these algorithms a smoother additional parameter could allow for the routine to find its way around and out of local minima in parameter-space and toward the global minimum.

The most reliable but slowest algorithm over all is the genetic. Its success rate does, however stay within the error of the genetic/simplex hybrid algorithm for all but the six parameter success rate. The genetic/simplex hybrid algorithm has an advantage over the genetic with a fast run time. The algorithm was fast enough to be run multiple times within the time it takes for one genetic algorithm minimisation to complete. With further tuning of the hybrid algorithm its reliability could likely be increased at the cost of some of its small runtime, but this could still result in a both fast and reliable algorithm. If tuning however is not attempted the stochastic nature of the initial genetic part of the algorithm means that repeat runs with the same initial inputs could find better parameter-space minima.

The genetic algorithm was looked at in further detail with a variety of population sizes compared. From these results (section 4.3) there is error in success rate and time measurements that mean that all the population sizes tested above 10 are possibly as successful and fast as each other. There is however a general trend for larger population size to cause an increase in both success rate and runtime, with only the success rate of the population of 70 members markedly out of place at six parameter minimisation beating a population of 90. The lowest population size of 10, though significantly faster than the others, has impractical success rates at high parameter numbers. As the minimisations with the best success rate are likely gained with the largest population size the largest population that time permits should be used.

In terms of real-world applications the semi-linear minimisation used to approach lensing problems usually has a number of parameters that fall within six to eight. In the semi-linear minimisation these parameters

would all describe the lensing mass profile. Parameter space minimisation that takes place to solve real lensing problems is therefore comparable to that found in this report as a minimum of a variant of the χ^2 statistic is sought in both cases, by comparison between modelled and synthetic images. For the simplest one and two parameter minimisations all the algorithms tested were very successful, meaning that only speed is a factor in determining which would be best. The down-hill simplex and grid search algorithms are the fastest at these low parameter numbers. The resolution/runtime trade-off for the grid search algorithm will not be skewed too heavily in the direction of long runtimes at such low numbers of parameters. For larger numbers of parameters genetic is the most robust but with the short runtime of the genetic/simplex hybrid multiple runs of this algorithm will likely have the best outcomes in terms of both speed and accuracy

The three Nested sampling algorithms all show very high success rates for all dimensions. If crashes are ignored (as is reasonable, explained in the results section) their success rates are almost 100% as a coarse search method. If the algorithms were hybridised with one of the better fine searching algorithms (MCMC or Simplex) they should find very precise and accurate results almost all the time, making the nested approach our most robust approach. The actual runtime of the code is much lower than the other algorithms tested, this is due to the fact that the other algorithms were adapted from very well optimised existing MATLAB functions, and the Nested algorithms have been developed from scratch over this year. With more development and optimisation these runtimes could be brought down, and the low number of iterations required implies that the power of the algorithms is very high.

The NMCMCS algorithm shows the least promise as it has to have a higher population, and having more degenerate points. The MNMCMCS algorithm shows the most promise searching thoroughly and being virtually unaffected in terms of robustness by changes in number of dimensions or population size.

The MNGMCMCS algorithm appears worse than the NMCMCMS algorithm, struggling more at higher dimensions, however with further refinement it can be confidently stated that it could be better, especially in much higher dimensional parameter spaces as it more thoroughly searches the parameter space. One possible addition to its functionality would be to instead of placing the worst points in the areas of probability shown in the figure 11, which as shown leaves a lot of the area unsearched to add

an extra piece of code which searches each hypersphere, by continually adding points from outside of the hyperspheres that have higher χ^2 than those already inside, having a Nested MCMC loop followed by a Genetic loop each iteration.

6. SUMMERY AND CONCLUSION

Our investigation shows that the Genetic algorithm and the Nested algorithms are very powerful coarse searching algorithms, not requiring any priors (other than boundary conditions) and still finding very good results. Both could be improved by hybridisation with the better fine search algorithms such as MCMC and Simplex. Although simulating annealing shows strengths in minimising up to 6 parameters the genetic, genetic simplex, simplex and all nested algorithms are consistently more robust. The nested algorithms appear to be the most robust solution to the problem, whilst the Genetic Simplex Hybrid is the most efficient algorithm at higher dimensional (6 and 7) parameter spaces.

Further research could entail discovering the relationship between optimal population size, and dimensions solved for in the nested algorithms. Further refinement and optimisation, removing bugs, and exploring different ways of incorporating Genetic algorithms and possibly simplex algorithms would also be a profitable area of further research. One idea would be to use the nested algorithms for a coarse search, slowly decreasing the population size as the χ^2 decreases until there is the number of points equal to those required for a simplex in that dimension, and then using those as the vertices.

No swarm intelligences were investigated and should be investigated as a solution, as swarm intelligences are a varied group of algorithms. From algorithms that copy the way bees search for new hive locations, to mimicking the echo location of bats. From such a plethora of approaches it is likely that some would solve the problem of minimisation in Gravitational lensing well.

This report presents only a small number of approaches to the problem of multi-dimensional minimisation in the analysis of strong gravitational lensing. There are many different algorithms and approaches already developed and though the research behind this report could not investigate all of them, a more comprehensive investigation could be carried out in the future.

Works Cited

- [1] "Albert Einstein - Biographical," http://www.nobelprize.org/nobel_prizes/physics/laureates/1921/einstein-bio.html, 21 May 2014.
- [2] F. W. Dyson, A. S. Eddington and C. Davidson, "A Determination of the Deflection of Light by the Sun's Gravitational Field, from Observations Made at the Total Eclipse," *Phil. Trans. R. Soc.*, vol. 220, pp. 291-333, 1920.
- [3] F. Zwicky, "On the Masses of Nebulae and of Clusters of Nebulae," *Astrophysical Journal*, vol. 86, p. 217, 1937.
- [4] D. WALSH, R. F. CARSWELL and R. J. WEYMANN, "0957 + 561 A, B: twin quasistellar objects or gravitational lens?," *Nature*, vol. 279, pp. 381-384, 1979.
- [5] S. Refsdal, "The gravitational lens effect," *Monthly Notices of the Royal Astronomical Society*, vol. 128, p. 295, 1964.
- [6] S. Refsdal, "On the possibility of determining Hubble's parameter and the masses of galaxies from the gravitational lens effect," *Monthly Notices of the Royal Astronomical Society*, vol. 128, p. 307, 1964.
- [7] D. Rusin and C. S. Kochanek, "The Evolution and Structure of Early-Type Field Galaxies: A Combined Statistical Analysis of Gravitational Lenses," *The Astrophysical Journal*, vol. 623, pp. 666-682, 2005.
- [8] R. Gavazzi and e. al., "The Sloan Lens ACS Survey. IV. The Mass Density Profile of Early-Type Galaxies out to 100 Effective Radii," *The Astrophysical Journal*, vol. 667, pp. 176-190, 2007.
- [9] G. Soucail, Y. Mellier, B. Fort, G. Mathez and M. Cailloux, "Discovery of the first gravitational Einstein ring - The luminous arc in Abell 370," *ESO Messenger*, pp. 5-6, 1987.
- [10] B. Paczynski, "Giant luminous arcs discovered in two clusters of galaxies," *Nature*, vol. 325, no. 6105, pp. 572-573, 1987.
- [11] G. Soucail, Y. Mellier, B. Fort, G. Mathez and M. Cailloux, "The giant arc in A 370 - Spectroscopic evidence for gravitational lensing from a source at $Z = 0.724$," *Astronomy and Astrophysics*, vol. 191, pp. L19-L21, 1988.
- [12] A. G. Bergmann, V. Petrosian and R. Lynds, "Gravitational lens models of arcs in clusters," *Astrophysical Journal*, vol. 350, pp. 23-35, 1990.
- [13] F. Hammer and F. Rigaut, "Giant luminous arcs from lensing - Determination of the mass distribution inside distant cluster cores," *Astronomy and Astrophysics*, vol. 226, pp. 45-56, 1989.
- [14] G. Mathez, B. Fort, Y. Mellier, J.-P. Picat and G. Soucail, "A new straight arc detected in a cluster of galaxies at $Z = 0.423$," *Astronomy and Astrophysics*, vol. 256, pp. 343-350, 1992.
- [15] R. Pello, B. Sanahuja, J.-F. Le Borgne, G. Soucail and Y. Mellier, "A straight gravitational image in Abell 2390 - A striking case of lensing by a cluster of galaxies," *Astrophysical Journal*, vol. 366, pp. 405-411, 1991.
- [16] M. Pierre, J. F. Le Borgne, G. Soucail and J. P. Kneib, "X-ray analysis and matter distribution in the lens-cluster Abell 2390," *Astronomy and Astrophysics*, vol. 311, pp. 413-424, 1996.
- [17] R. Kayser and T. Schramm, "Imaging procedures for gravitational lenses," *Astronomy and Astrophysics*, vol. 191, pp. 39-43, 1988.
- [18] C. S. Kochanek, C. R. R. D. Blandford and L. A. R. Narayan, "The ring cycle - an iterative lens reconstruction technique applied to MG1131+0456," *Monthly Notices of the Royal Astronomical Society*, vol. 238, pp. 43-56, 1989.
- [19] J. A. Tyson, G. P. Kochanski and I. P. Dell'Antonio, "Detailed Mass Map of CL 0024+1654 from Strong Lensing," *The Astrophysical Journal*, vol. 498, pp. L107-L110, 1998.
- [20] S. Wallington, C. S. Kochanek and R. Narayan, "A Gravitational Lens Inversion Algorithm Using the Maximum Entropy Method," *Astrophysical Journal*, vol. 465, p. 64, 1996.
- [21] S. H. Suyu, P. J. Marshall, M. P. Hobson and R. D. Blandford, "A Bayesian analysis of regularized source inversions in gravitational lensing," *Monthly Notices of the Royal Astronomical Society*, vol. 371, no. 2, pp. 983-998, 2006.
- [22] S. J. Warren and S. Dye, "Semilinear Gravitational Lens Inversion," *The Astrophysical Journal*, vol. 590, pp. 673-682, 2003.
- [23] T. Treu and L. V. E. Koopmans, "Massive Dark Matter Halos and Evolution of Early-Type Galaxies to $z \approx 1$," *Astrophysical Journal*, vol. 611, p. 739-760, 2004.
- [24] M. Barnabe and L. V. E. Koopmans, "A Unifying Framework for Self-consistent Gravitational Lensing and Stellar Dynamics Analyses of Early-Type Galaxies," *The Astrophysical Journal*, vol. 666, p. 726, 2007.
- [25] S. e. al., "Dissecting the Gravitational Lens B1608+656. I. Lens Potential Reconstruction," *Astrophys.J.*, vol. 691, pp. 277-298, 2009.
- [26] C. R. Keeton, "A Catalog of Mass Models for Gravitational Lensing," *arXiv:astro-ph/0102341v2*, 2002.
- [27] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, 2nd ed., Cambridge: Cambridge University Press, 1992.
- [28] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308-313, 1965.
- [29] N. Metropolis and S. Ulam, "The Monte Carlo Method," *Journal of the American Statistical Association*, vol. 44, no. 247, p. 335-341, 1949.
- [30] A. Markov, "Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga," *Izvestiya Fiziko-matematicheskogo obshchestva pri Kazanskom universitete*, no. 15, p. 135-156, 1906.
- [31] J. Skilling, "Nested Sampling," *AIP Conference Proceedings*, no. 735, p. 395-405, 2004.
- [32] M. H. M. B. F. Feroz, "MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics," *Astrophysics Journal*, vol. 398, pp. 1601-1614, 2009.
- [33] J. C. LAGARIAS, J. A. REEDS, M. H. WRIGHT and P. E. WRIGHT, "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM*, vol. 9, no. 1, pp. 112-147, 1998.
- [34] P. Young, J. E. Gunn, J. Kristian, J. B. Oke and J. A. Westphal, "The double quasar Q0957 + 561 A, B - A gravitational lens image formed by a galaxy at $Z = 0.39$," *Astrophysical Journal*, vol. 241, pp. 507-520, 1980.
- [35] D. Ruskin, D. R. Marlow, M. Norbury, W. A. browne, N. Jackson, P. N. Wilkinson, C. D. Fassnacht, S. T. Myers, L. V. E. Koopmans, R. D. Blandford, T. J. Pearson, A. C. S. Readhead and A. G. Bruyn, "The New Two-Image Gravitational Lens System Class B2319+051," *The Astronomical Journal*, vol. 122, pp. 591-597, 2001.
- [36] S. Refsdal, "On the possibility of determining Hubble's parameter and the masses of galaxies from the gravitational lens effect," *Monthly Notices of the Royal Astronomical Society*, vol. 128, pp. 307-310, 1964.
- [37] R. Pelló, B. Fort, J.-P. Kneib, J.-F. Le Borgne, Y. Mellier, I. Appenzeller, R. Bender, L. Campusano, M. Dantel-Fort, R. S. Ellis, A. Moorwood and S. Seitz, "Probing Distant Galaxies with Lensing Clusters," 2000.
- [38] R. Kayser and T. Schramm, "Imaging procedures for gravitational lensing," *Astron. Astrophys.*, vol. 191, pp. 39-43, 1988.
- [39] C. S. Kochanek, R. D. Blandford, C. R. Lawrence and R. Narayan, "The ring cycle - an iterative lens reconstruction technique applied to MG1131+0456," *Monthly Notices of the Royal Astronomical Society*, vol. 238, pp. 43-56, 1989.
- [40] S. Wallington, C. S. Kochanek and D. C. Koo, "Gravitational lens inversions of CL 0024+1654," *Astrophysical Journal*, vol. 441, pp. 58-69, 1995.
- [41] J. A. Tyson, G. P. Kochanski and d. I. P., "Detailed Mass Map of CL 0024+1654 from Strong Lensing," *Astrophysical Journal Letters*, vol. 498, p. L107-L110, 1998.
- [42] S. Wallington, C. S. Kochanek and R. Narayan, "LensMEM: A Gravitational Lens Inversion Algorithm Using the Maximum Entropy Method," *Astrophysical Journal*, vol. 498, pp. 64-72, 1996.
- [43] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Numerical Recipes in FORTRAN 77, 2nd ed., New York: Cambridge University Press, 2001.
- [44] S. J. Warren and S. Dye, "Semilinear Gravitational Lens Inversion," *The Astrophysical Journal*, vol. 590, pp. 673-682, 2003.
- [45] S. Dye and S. Warren, "Decomposition of the visible and dark matter in the Einstein ring 0047-2808 by semi-linear inversion," 2005.
- [46] T. Treu and L. V. Koopmans, "Massive Dark-matter halos and Evolution of Early-type Galaxies to $z=1$," *Astrophys.J.*, vol. 611, pp. 739-760, 2004.
- [47] M. Barnabe and L. V. E. Koopmans, "A Unifying Framework for Self-consistent Gravitational Lensing and Stellar Dynamics Analyses of Early-Type Galaxies," *Astrophys.J.*, vol. 666, pp. 726-746, 2007.
- [48] S. H. Suyu, P. J. Marshall, M. P. Hobson and R. D. Blandford, "A Bayesian analysis of regularised source inversions in gravitational lensing," *MNRAS*, vol. 371, pp. 983-998, 2006.
- [49] S. H. Suyu, P. J. Marshall, R. D. Blandford, C. D. Fassnacht, L. V. E. Koopmans, J. P. McKean and T. Treu, "Dissecting the Gravitational Lens B1608+656. I. Lens Potential Reconstruction," *Astrophys.J.*, vol. 691, pp. 277-298, 2009.
- [50] M. Richey, "The Evolution of Markov Chain Monte Carlo Methods," *The American Mathematical Monthly*, vol. 117, pp. 383-413, 2010.
- [51] MATLAB:2010a, Natick, Massachusetts: The MathWorks Inc, 2010.
- [52] C. R. Keeton, "A Catalog of Mass Models for Gravitational Lensing," 2002.