This presentation introduces a new evolutionary algorithm that can be used to find starting values (SVs) for nonlinear regression (NLR) problems. The method is called a Stochastic Funnel Algorithm (SFA). The SFA is introduced through its operation in the SSE/parameter space of a NLR model. The SFA uses a relaxed trust region in the log10 space of the parameters of an NLR model. The SFA takes explicit advantage of the general shape of the Sum-of-Squares Error (SSE) of a data/model set to explore and exploit that space and eventually converge to the region of the best parameter estimates.
Using a Stochastic Funnel to find NLR Starting Values

Adrian O’Connor
Sharjah Men’s College
Sharjah
United Arab Emirates
adrian.oconnor@hct.ac.ae

Dr. Junbin Gao
Charles Sturt University
School Acc & Comp Sci.
Bathurst, Australia
jbgao@csu.edu.au

Dr. John Louis
Charles Sturt University
School Comp. & Math.
Wagga Wagga, Australia
jlouis@csu.edu.au

Abstract
This presentation introduces a new evolutionary algorithm that can be used to find starting values (SVs) for nonlinear regression (NLR) problems. The method is called a Stochastic Funnel Algorithm (SFA). The SFA is introduced through its operation in the SSE/parameter space of a NLR model. The SFA uses a relaxed trust region in the log_{10} space of the parameters of an NLR model. The SFA takes explicit advantage of the general shape of the Sum-of-Squares Error (SSE) of a data/model set to explore and exploit that space and eventually converge to the region of the best parameter estimates.

Keywords
Nonlinear Regression, Starting Values, Evolutionary Algorithm, Relaxation, Global Optimization.

Introduction
Most methods used to find parameters for nonlinear regression models are local searches and these require S.V’s (S.V’s) to initiate the search. S.V’s need to be such that the nonlinear searches can get to the global optimal parameter set of the model. In the hyper-space of the SSE this is often not a trivial problem. The finding of S.V’s is often an ad hoc procedure and not a great deal has been written on the topic.

Ratkowsky [1] has discussed the issue of S.V’s and devotes a chapter to the issue. Gallant [2] devotes about a page to the issue while still noting that S.V’s are important. Bates & Watts [3], and Draper & Smith [4], and others refer to the variety of methods used to find S.V’s. Other texts [5], [6] for example, though considered classics of nonlinear regression, devote little space to the issue of S.V’s. [1], [2], [3], [4] highlight the need for good S.V’s in nonlinear regression searches and describe the methods they have used to find S.V’s for various models. Those methods are various and no one method appears to suit all models. It would seem obvious that a single method for finding S.V’s for all nonlinear models is desirable. Up until this stage the methods of calculus and linear algebra have been used to make ancillary models that can be used for S.V’s. This is not always easy, though when it can be done the S.V’s are generally considered to be good. In the past these same ancillary methods were used to get final parameter estimates, until at least the 1960’s and the ready availability of computing resources [6a].

Evolutionary methods for solving optimization problem, inspired by models of Darwinian evolution [5], have become popular in recent times [6], [7], [8], [9], [10]. These methods are robust and have been shown to be widely applicable. Rather than looking for a single solution evolutionary methods look for populations of solutions. These populations are allowed to breed and the best solutions are kept to multiply. The weakest solutions tend to be culled. This process is called Natural Selection (Darwin’s term) [11] or Survival of the Fittest (Spencer’s term) [11]. The process continues and converges on a solution set, eventually stopping when some termination criteria are met. It is towards these evolutionary methods that this work has progressed in the search for a single method that can be universally applied to find good S.V’s. The search appears to have reached a satisfactory conclusion with the SFA.

The Environment of the SSE
It is useful to start with a brief introduction of the SSE for linear models and expand the discussion to nonlinear models.
A linear model with additive error can be written as,

\[ y_i = f(x_i, \hat{\beta}) + \epsilon_i \]  

The data is \( \{x_i, y_i\} \), the parameters of the model are given by the vector \( \hat{\beta}^* \), and \( f(.) \) is a linear function of parameters (\( \hat{\beta} \)). A linear function is one where the first derivative of the model with respect to the parameters is a constant. Associated with this model is the Sum-of-Squares Error (SSE), which is a function of the data, the model and the parameters. Given a data set and a model the SSE can then be written as a function only of the parameters.

\[ SSE(\hat{\beta}) = \sum_{i=1}^{n} (y_i - f(x_i, \hat{\beta}))^2 \]

For a linear model this is a quadratic function and forms a paraboloid in the SSE/parameter space. Parameters of such linear models can be uniquely and quickly found by solving what are known as the normal equations for the model/data combination [4].

A nonlinear regression (NLR) model with additive error can also be described by,

\[ y_i = f(x_i, \hat{\beta}) + \epsilon_i \]  

The data is \( \{x_i, y_i\} \), the parameters of the model are given by the vector \( \hat{\beta}^* \), but \( f(.) \) is now a nonlinear function of parameters. This means that for at least some of the parameters of the model the first derivative is not a constant but a function of at least some of the parameters. Associated with this model is the Sum-of-Squares Error (SSE), which is a function of the data, the model and the parameters. Given a data set and a model the SSE can again be written as a function only of the parameters.

\[ SSE(\hat{\beta}) = \sum_{i=1}^{n} (y_i - f(x_i, \hat{\beta}))^2 \]

For a nonlinear model this is not a quadratic equation. It can have asymptotes and local stationary points. The determination of the best parameter set, which minimizes the SSE, cannot, generally, be solved analytically through the normal equations. Iterative methods are required and many of these have their origin in Newton-type methods of optimization. These are localized methods that require a starting point. From the starting point a method will generally fall towards that local minimum of the SSE whose zone of convergence it happens to fall within. The local minimum may or may not be the global optimal minimum, where one finds the optimal parameter set \( \hat{\beta}^* \).

The best possible model is found at \( \hat{\beta}^* \) and at this point the SSE is a global minimum, for both linear and nonlinear models. From this we can say,

\[ SSE(\hat{\beta}^*) \geq SSE(\hat{\beta}) \quad \forall \hat{\beta} \neq \hat{\beta}^* \]

The value of the SSE for any point \( \hat{\beta}^* \) that is not \( \hat{\beta}^* \) will be greater than or equal to the SSE at \( \hat{\beta}^* \). For a linear model with one parameter we find that,

\[ \text{Prob}\left(\frac{SSE(\hat{\beta}^*) - SSE(\hat{\beta}^*)}{|\hat{\beta} - \hat{\beta}^*|} \leq |\hat{\beta} - \hat{\beta}^*|\right) = 1 \]

This simply says that as we get closer to \( \beta^* \) the SSE gets smaller, and approaches the value of the SSE at \( \beta^* \).

For a linear model with more than one parameter, or a nonlinear model, we are reduced to an inequality,

\[ \text{Prob}\left(\frac{SSE(\hat{\beta}^*) - SSE(\hat{\beta}^*)}{|\hat{\beta} - \hat{\beta}^*|} \leq |\hat{\beta} - \hat{\beta}^*|\right) \geq 0 \]

This is saying that the closer we are to the minimum the less the SSE, but not necessarily always. Linear models may have an SSE that is ellipsoidal in cross-section. For nonlinear models we find that in some local neighborhoods of the SSE, there can be basins and/or peaks. Notwithstanding these ellipsoids, basins and peaks, the Intermediate Value Theorem tells us that for any point \( \hat{\beta} \neq \hat{\beta}^* \) there will be a point \( \hat{\beta}^* \) within the space \([\hat{\beta}^-, \hat{\beta}^+]\) that will have an SSE that is smaller than \( SSE(\hat{\beta}^+) \).

Using a random search in the space \([\hat{\beta}^-, \hat{\beta}^+]\) will eventually find such a point. Eqn. 6 is a direct
result of Eqn. 5 and is the necessary and sufficient condition for Darwinian evolution to occur in the fitness landscape provided by the SSE. Necessary because Darwinian evolution works in a fitness landscape and the SSE is the measure of fitness in this case. It is sufficient because that is all that is required.

One can also say that,

\[
\text{Prob}\left( \frac{\text{SSE}(\hat{\beta}^*) - \text{SSE}(\hat{\beta}^*)}{|\hat{\beta}^* - \hat{\beta}^*|} \leq |\beta^* - \beta^*| \right) \rightarrow 1 \quad \forall |\beta - \beta^*| \leq \epsilon \rightarrow 0 \quad 8
\]

This statement alludes to the pseudo-parabolic shape of the SSE in the region of the optimal parameter set. The closer one is to \( \hat{\beta}^* \) then the more likely a point that is even closer to \( \hat{\beta}^* \) will have an SSE that is lesser, because the SSE is shaped like a parabola in this region. For linear models we can say that \( \epsilon \) can be as large as we like for equation 8 to hold, because the SSE is parabolic. Nonlinear models have a non-parabolic SSE and the value of \( \epsilon \) may need to be quite small for equation 8 to hold.

For nonlinear models the SSE can have features that challenge NLR search algorithms. These may be asymptotes, both horizontal and vertical, and local optima such as basins or hills. In the presence of such features it is wise to have good S.V’s, because poor S.V’s can lead to poor results. Poor results can be convergence to a wrong solution set, failure to converge, overflow or underflow errors. Having good S.V’s normally means S.V’s that are near the optimal parameter set but it can mean a point from which a classical NLR search algorithm can follow the SSE continuously downwards in a path to the minimum.

**Evolutionary Algorithms (EA) & the SFA**

If \( \hat{\beta}^* \) exists, and is unique, and there is an SSE associated with the model, then we have a nonrandom landscape (SSE) which can be explored and exploited to determine \( \hat{\beta}^* \). In the language of biology, we have a population of phenotypes that we search to find “good” genotypes and eventually the best genotype (parameter set) for a model/data set. Equations 7 & 8 describe the necessary conditions for an evolutionary process to negotiate the landscape of the SSE and find the minimum of the SSE. A random search will eventually find that minimum by utilizing the generally downward sloping nature of the SSE, or by converging to some trust region of the parameters.

Evolutionary Algorithms such as Genetic Algorithms (GA) and Evolution Strategies (ES) are global optimizers and, given sufficient time/generations, will eventually find the best solution for the parameters vector. These methods use recombination and/or mutation to generate and perturb offspring solutions at each iteration or generation. GAs employ a diploid process to create offspring from parental pairs. ES use a haploid process to create offspring from individual parents. Both methods look at the value of the fitness function (the phenotype), which for NLR is the SSE, to decide which candidates are deemed fit enough to breed from and which, because of poor fitness, will be culled. The value of the fitness function is the key for both these methods. Neither looks explicitly at the dispersion of parameter values in a current generation. These methods rely on the non-random nature of the SSE as described in equations 7 & 8 to force their random selection to converge to a solution set. That solution set may be considered as the “answer” or the best solution from that set may be considered the “answer”. Descriptions of GA and ES are given in Eiben & Smith [5].

A new EA is introduced in this paper that will also converge to best parameter solution sets for NLR models. This is called a Stochastic Funnel Algorithm (SFA). The SFA uses the features described in Eqns. 7 & 8 to negotiate the landscape of the SSE described in Eqn. 4. In contrast to GA and ES methods the SFA makes explicit use of the dispersion of the current population of solutions (the genotypes) to determine where the next generation of solutions will be placed.

The rationale of the SFA presumes that a nonlinear model can be described with positive parameters, though the sign in front of the parameter may be plus or minus. In the absence of any other knowledge it is reasonable to presume that any parameter then will have a value that falls in the interval [1E-06, 1E+06]. To start the SFA we change this space to a \( \log_{10} \) space (hence the need for positive parameter values) and then drop uniformly distributed random values across this \( \log_{10} \) space. 100 random values are distributed across \([\log_{10}(1E-06), \log_{10}(1E+06)]\). Each order of magnitude will then have about 8 values. At this stage the SFA search is blind and no one area of the search space can be favored, so it seems reasonable to pepper the space with values. Each of these points has an SSE associated with it.
The 100 points are then sorted on the basis of ascending SSE. The best 20 are kept and the rest are culled. This process is called elitism. These 20 points now help to define a trust region. By choosing the minimum and maximum of each parameter from the best 20 we define a box constrained region. This forms the basis of a search space for the next generation. The minimum and maximum are now used to relax the trust region and hence ensure that the SFA does not converge prematurely. The new search space is defined with Dynamic Relaxation \( R_D \) and Static Relaxation \( R_S \). \( R_D \) causes the search to adapt to the landscape that it finds itself in. If no better estimates than the current best 20 are found in a generation then the search space expands. \( R_S \) maintains variety in the solutions developed and helps to ensure that the SFA does not converge prematurely. It also allows to SFA to creep towards a solution in the latter stages of a SFA search. The values for \( R_D \) and \( R_S \) have been set empirically to 20% and 20. These values were selected after extensive testing to determine the minimum values for these parameters whilst ensuring that the SFA did not prematurely converge. The search space is, at any iteration, for any particular parameter,

\[
[\text{Max} (1E-06, \beta_{\min} - R_D(\beta_{\max} - \beta_{\min}) - R_S), \text{ Min} (1E+06, \beta_{\max} + R_D(\beta_{\max} - \beta_{\min}) + R_S)].
\]

The max/min prescription ensures that the search remains in \([1E-06, 1E+06]\), as mentioned earlier. This space is converted to a \( \log_{10} \) space and 100 random values are dropped into the space. Each order of magnitude in that space will then have approximately the same number of values. These 100 offspring values are combined with the 20 parent values, sorted and the best 20 kept again. The values for the maximum and minimum then help define the next generations search in \( \log_{10} \) space. The process continues until some termination criteria are met. In this work the criterion is the convergence of the SFA until the best value and the 20th best value of each parameter are the same; all the best 20 for each parameter are the same.

The SFA was developed in response to the need for a single method for finding S.V’s that could be reliably used as a front-end for NLR search algorithms. It is easy to understand, code, and apply. Here the SFA is demonstrated in Microsoft-EXCEL where nearly all its operations are achieved through keyboard macros.

The convergence behavior of the SFA is demonstrated below on the familiar BoxBOD data and model taken from Bates & Watts [4]. The data and model are shown below.

\[
y_i = \beta_1 (1 - \exp(-\beta_2 \cdot x_i)) + \varepsilon_i
\]

<table>
<thead>
<tr>
<th>( X )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y )</td>
<td>109</td>
<td>149</td>
<td>149</td>
<td>191</td>
<td>213</td>
<td>224</td>
</tr>
</tbody>
</table>

Table 1. BoxBOD data from Bates & Watts [4].

The best parameter estimates for the two parameters \( \beta_1 \) and \( \beta_2 \) are known to be (213.8, 0.547). S.V’s can be determined by transforming the deterministic relationship from Equation 9 by differentiating and then taking a log transform.

\[
\frac{dy}{dx} = \beta_1, \beta_2 \exp(-\beta_2 \cdot x)
\]

\[
\ln\left(\frac{dy}{dx}\right) = \ln(\beta_1, \beta_2) - \beta_2 \cdot x
\]

The gradient \( dy/dx \) can be interpolated from the data and the slope and intercept of equation 8 can be estimated using simple linear regression (SLR). Estimates of the parameters of the BoxBOD model can be then determined. These estimates can be used as S.V’s. The estimates are (155, 0.188). Using the SFA estimates of the parameters are (213.8, 0.547), the same as the best estimates; good S.V’s indeed. Two runs of the SFA are demonstrated on the BoxBOD data below. The vertical axis is the scale for the maximum and minimum values of the best 20 for each parameter at each generation. The initial search space is \([1E-06, 1E+06]\). Convergence is rapid and good S.V’s are available within 20 to 30 generations of the SFA.

The SFA has been developed and tested using the models shown in Table 2 below. For all models the SFA was able to converge to the region of best parameter estimates. Some models converged faster than others.
Figure 1. 2 runs of the SFA on the BoxBOD model/data set.

Table 2. Models used in development of Stochastic Funnel Algorithm

<table>
<thead>
<tr>
<th>Model name &amp; source,</th>
<th>Description</th>
<th>Iterations to $\beta^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear–author</td>
<td>$y = \beta_1 + \beta_2 \cdot x + \varepsilon$</td>
<td>$O(30)$</td>
</tr>
<tr>
<td>Asymptote-author</td>
<td>$y = \frac{1}{\beta - x} + \varepsilon$</td>
<td>$O(30)$</td>
</tr>
<tr>
<td>Asymptote2-author</td>
<td>$y = \frac{1}{(\beta_1 - x) \cdot (\beta_2 - x)} + \varepsilon$</td>
<td>$O(500)$</td>
</tr>
<tr>
<td>BoxBOD-NIST**</td>
<td>$y = \beta_1 \cdot (1 - \exp(-\beta_2 \cdot x)) + \varepsilon$</td>
<td>$O(100)$</td>
</tr>
<tr>
<td>DanWood-NIST**</td>
<td>$y = \beta_1 \cdot x^{\beta_2} + \varepsilon$</td>
<td>$O(100)$</td>
</tr>
<tr>
<td>Logistic-various</td>
<td>$y = \frac{\beta_1}{1 + \beta_2 \cdot \exp(-\beta_3 \cdot x)} + \varepsilon$</td>
<td>$O(200)$</td>
</tr>
<tr>
<td>MGH09-NIST*</td>
<td>$y = \frac{\beta_1 \cdot (x^2 + x \cdot \beta_2)}{x^2 + x \cdot \beta_3 + \beta_4} + \varepsilon$</td>
<td>$O(1000)$</td>
</tr>
<tr>
<td>MGH10-NIST**</td>
<td>$y = \beta_1 \cdot \exp\left(\frac{\beta_2}{x + \beta_3}\right) + \varepsilon$</td>
<td>$O(30 000)$</td>
</tr>
</tbody>
</table>
**Misra1a-NIST**

\[ y = \beta_1 \ast (1 - \exp(-\beta_2 \ast x)) + \varepsilon \]

O(300)

**Gompertz-Ratkowsky,**

\[ y = \beta_1 \ast \exp(-\beta_2 \ast \exp(-\beta_3 \ast x)) + \varepsilon \]

O(10000)

**Rat42-NIST**

\[ y = \frac{\beta_1}{1 + \exp(\beta_2 - \beta_3 \ast x) + \varepsilon} \]

O(1000)

**Rat43-NIST**

\[ y = \frac{\beta_1}{(1 + \exp(\beta_2 - \beta_3 \ast x))^{1/\beta_4} + \varepsilon} \]

O(500)

<table>
<thead>
<tr>
<th>2 input/1 output</th>
<th>Artificial Neuron-author***</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ y = \frac{1}{1 + \exp(-w_1 \ast x_1 - w_2 \ast x_2 + b)} ]</td>
<td>[ O(10) ]</td>
</tr>
</tbody>
</table>

**Inversion of 2x2 matrix-author****

\[
\begin{bmatrix}
 a & b \\
 c & d \\
\end{bmatrix} \begin{bmatrix}
 e & f \\
 g & h \\
\end{bmatrix} = \begin{bmatrix}
 1 & 0 \\
 0 & 1 \\
\end{bmatrix}
\]

\[ SSE = (1 - (ae + bg))^2 + (0 - (af + bh))^2 + (0 - (ce + dh))^2 + (1 - (cf + dh))^2 \]

O(30)

*The 1 and 2-Asymptote models were created to demonstrate the challenges in terms of vertical and horizontal asymptotes that can exist in the SSE/parameter space.

**NIST model/data sets have been chosen as these models have presented convergence challenges to some common NLR algorithms and are used as reference set to compare NLR algorithms. The NIST is the National Institute for Science and Technology. Site is [http://www.itl.nist.gov/div898/strd/](http://www.itl.nist.gov/div898/strd/)

*** This model is able to simulate 14 of the 16 possible output configurations of a two input Boolean device. XOR and XNOR cannot be simulated using a single neuron, which is a well known result.

Initial search space was [1E-6, 1E+6], no logs used.

**** The left matrix is the matrix to be inverted. The right matrix is its inverse. The task is to determine the elements e, f, g, and h that yield the identity matrix on the right hand side. No logarithms used, as the elements can be positive or negative.

**Pseudo-code for Stochastic Funnel Algorithm**

The pseudo-code here is for a SFA which generates 100 offspring in a generation from the information supplied by the 20 best candidates, on the basis of SSE. Implicit is Dynamic and Static Relaxation which in this work has been empirically determined at 20% and ±20. The SFA uses a uniform random number generator to initially generate values in the space [1E-06, 1E+06] using 10^8(RANDBETWEEN(-60000, +60000)/10000). This ensures approximately equal numbers of candidates in each order of magnitude with a precision of about 4 places, depending on the size of the candidates. This limits the absolute accuracy of the SFA but then it is meant to find S.V’s, not actual final parameter estimates. Because the SFA starts across 12 orders of magnitude one finds that, at least in its early stages of operation, the best 20 candidates can range across several orders of magnitude. To ensure coverage of each order of magnitude in the trust region of the search the maximum and minimum of the best 20 are converted to logs.

Fig.2. Pseudo code for Stochastic Funnel Algorithm

Create initial population of 100 random points in log(parameter) search space.

DO UNTIL termination criteria.

Use Elitism to determine best 20.

Determine min and max for each parameter.

Use Relaxation to define trust region.

Convert to log space

Generate 100 random offspring in trust region

Combine 20 parents and 100 offspring

OD

**Conclusion**

This work comes from the search for a single method that can be universally applied to find S.V’s for nonlinear regression models. The search has, at this stage, resulted in the development of the...
Stochastic Funnel Algorithm in its current implementation. The SFA has been successfully applied to a number of NLR models to find S.V’s. The S.V’s developed with the SFA have been within 3 significant digits accuracy of the known final values of the models, much better accuracy than pre-existing methods. The SFA takes between $O(10)$ and $O(10^4)$ generations to converge. This variety in convergence speed appears to reflect the correlation between parameters that can exist when sampling parameter estimates near the optimal solution, rather than as a result of the nonlinearity of the model. The SFA is robust and, because it doesn’t use calculus, it can be applied to a wide variety of optimization problems. With the ever increasing speed of PCs the issue of a large number of generations to convergence decreases, as it always has.

The SFA is constrained to searching for positive parameter values because the search space is logarithmic. By using a more sophisticated random number generator this constraint may be eliminated and a wider ranging search applied. That is, a random number generator that can generate positive and negative values, yet still ensure coverage of each of the orders of magnitude that each parameter may possibly take.

The SFA has been used to determine the weights and bias for a single artificial neuron. It appears to provide an alternative to backpropagation in training neural networks. The SFA has also been used to determine the elements of the inverse of a 2x2 matrix. Its use in NLR and these two examples just mentioned suggest that the SFA may provide a non-calculus based, and hence more robust, method for solving optimization problems, as well as being used as a front-end for NLR algorithms.

The Stochastic Funnel Algorithm provides a new method for a random search to “descend with modification” [7], using a trust region that provides a framework of selection so that improvement (lesser SSE) occurs “through natural selection” [7], and good starting values determined.

**Bibliography**
