Proceedings of the 6th WSEAS Int. Conf. on EVOLUTIONARY COMPUTING, Lisbon, Portugal, June 16-18, 2005 (pp29-35)

On the Solution of Ill-Conditioned Systems of Linear and Non-Linear Equations via Genetic Algorithms (GAs) and Nelder-Mead Simplex Search

Nikos E. Mastorakis
Military Institutes of University Education (ASEI)
Hellenic Naval Academy, Terma Hatzikyriakou 18539,
Piraeus, GREECE

http://www.wseas.org/mastorakis

Abstract: - Solution of Ill-Conditioned Systems of Linear and/or Non-Linear Equations are tested via Genetic Algorithms. The method is compared with other methods via specific numerical examples. Directions for future research are also provided.

Key-Words: - Ill-Conditioned Systems, Non – linear equations, genetic algorithms, numerical solutions

1 Introduction

Ill-Conditioned Systems arise in many problems in modeling and simulation of physical, engineering, socio-economic and biological systems.

Solving Systems of Linear Equations or Solving Systems of non-Linear Equations is now always an easy task, due to ill-conditioning of these systems in many times.

We recall that a system of linear equations

$AX = B$

is called ill- Conditioned

$\det A \approx 0$

Also, we recall that a system of non-linear equations $f(X) = 0$ is called ill-conditioned at the point X^* (where X^* is a point-solution of $f(X) = 0$, i.e. $f(X^*) \approx 0$) if $\det J(f(X^*)) \approx 0$ in a closed set containing X^* where J is the Jacobian matrix of $f(X) = 0$

Ill-Conditioned Systems are very sensitive to roundoff errors and, therefore, may pose problems during computation of the solution [6]. During computing process, these errors induce small changes in the coefficients which, in turn, result a large error in the solution [6].

In this paper, we attempt to overcome this problem by using evolutionary computing. We shall need the following definitions from the Genetic Algorithms practice:

Fitness function is the objective function we want to minimize.

Population size specifies how many individuals there are in each generation. We can use various Fitness Scaling Options (rank, proportional, top, shift linear, etc…[16]), as well as various Selection Options (like Stochastic uniform, Remainder, Uniform, Roulette, Tournament)[16].

Fitness Scaling Options: We can use scaling functions. A Scaling function specifies the function that performs the scaling. A scaling function converts raw fitness scores returned by the fitness function to values in a range that is suitable for the selection function. We have the following options: *Rank Scaling Option:* scales the raw scores based on the rank of each individual, rather than its score. The rank of an individual is its position in the sorted scores. The rank of the fittest individual is 1, the next fittest is 2 and so on. Rank fitness scaling removes the effect of the spread of the raw scores. *Proportional Scaling Option:* The Proportional Scaling makes the expectation proportional to the raw fitness score. This strategy has weaknesses when raw scores are not in a "good" range. *Top Scaling Option:* The Top Scaling scales the individuals with the highest fitness values equally. *Shift linear Scaling Option*: The shift linear scaling option scales the raw scores so that the expectation of the fittest individual is equal to a constant, which you can specify as Maximum survival rate, multiplied by the average score.

We can have also option in our Reproduction in order to determine how the genetic algorithm creates children at each new generation. For

example, *Elite Counter* specifies the number of individuals that are guaranteed to survive to the next generation.
Crossover combines two individuals, or parents, to form a new individual, or child, for the next generation.

Crossover fraction specifies the fraction of the next generation, other than elite individuals, that are produced by crossover.

Scattered Crossover: Scattered Crossover creates a random binary vector. It then selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent, and combines the genes to form the child.

Mutation: Mutation makes small random changes in the individuals in the population, which provide genetic diversity and enable the GA to search a broader space.
Gaussian Mutation: We call that the Mutation is Gaussian if the Mutation adds a random number to each vector entry of an individual. This random number is taken from a Gaussian distribution centered on zero. The variance of this distribution can be controlled with two parameters. The Scale parameter determines the variance at the first generation. The Shrink parameter controls how variance shrinks as generations go by. If the Shrink parameter is 0, the variance is constant. If the Shrink parameter is 1, the variance shrinks to 0 linearly as the last generation is reached.

Migration is the movement of individuals between subpopulations (the best individuals from one subpopulation replace the worst individuals in another subpopulation). We can control how migration occurs by the following three parameters.

Direction of Migration: Migration can take place in one direction or two. In the so-called "Forward migration" the nth subpopulation migrates into the (n+1)'th subpopulation. while in the so-called "Both directions Migration", the nth subpopulation migrates into both the (n-1)th and the (n+1)th subpopulation.
Migration wraps at the ends of the subpopulations. That is, the last subpopulation migrates into the first, and the first may migrate into the last. To prevent wrapping, specify a subpopulation of size zero.

Fraction of Migration is the number of the individuals that we move between the subpopulations. So, Fraction of Migration is the fraction of the smaller of the two subpopulations that moves. If individuals migrate from a subpopulation of 50 individuals into a population of 100 individuals and Fraction is 0.1, 5 individuals (0.1 * 50) migrate. Individuals that migrate from one subpopulation to another are copied. They are not removed from the source subpopulation. *Interval of Migration* counts how many generations pass between migrations.

The Nelder-Mead simplex algorithm appeared in 1965 and is now one of the most widely used methods for nonlinear unconstrained optimization [13]÷[16]. The Nelder-Mead method attempts to minimize a scalar-valued nonlinear function of n real variables using only function values, without any derivative information (explicit or implicit). The Nelder-Mead method thus falls in the general class of direct search methods.

In this paper, we try to solve Ill-Conditioned Systems using Genetic Algorithm. The results are not satisfactory. So, after the end of the Genetic Algorithm, we continue with and the Nelder-Mead Simplex Search. The method is outlined in Session 2 with specific examples.

2 Numerical Results for Systems of Linear Equations

Example 1:

Consider the system of equations

$$2x + 3y = 6$$

$$2.0001\,x + 3.0001\,y = 5.9999$$

Note that: $\det(A) = -0.0001$

We consider the error

$$e = \left|6 - (2x + 3y)\right| + \left|5.9999 - (2.0001\,x + 3.0001\,y)\right|$$

First Method for minimizing e:

Our first method uses a GA with fitness function the aforementioned error, where we use

- Initial Population: 20 individuals;
- Fitness Scaling: We use Rank Scaling Option

- Selection: Tournament selection with Tournament size 4
- Reproduction: Elite Counter 2, Crossover Fraction 0.8.
- Mutation: We use Gaussian Mutation with Scale 1 and Shrink 1.
- Crossover: Scattered
- Migration: Both Direction Migration with Fraction of Migration 0.2 and Interval of Migration 20
- Number of Generations: 1000

So, we run our algorithm via *Matlab* ([16]) for 1000 generations and we take various solutions as in the following table. The initial range for each variable *x, y* is [0,1].

x	y	e
2.2908	0.4727	$3.7635*10^{-4}$
-0.4602	2.3068	$2.8466*10^{-4}$
4.8028	-1.2019	$4.6009*10^{-4}$
-0.8790	2.5860	$5.9722*10^{-4}$
3.8601	-0.5736	$5.0179*10^{-4}$
1.5613	0.9591	$3.5203*10^{-4}$
0.4839	1.6775	$7.7749*10^{-4}$
2.2391	0.5073	$5.3724*10^{-4}$
0.6567	1.5620	$5.3182*10^{-4}$
1.6299	0.9134	$3.5433*10^{-4}$
1.1832	1.2111	$3.3943*10^{-4}$
3.4816	-0.3212	$4.1604*10^{-4}$
-4.9566	5.3045	$5.9877*10^{-4}$
2.7632	0.1578	$3.9210*10^{-4}$

Unfortunately, our GA does not give a satisfactory and unique result with 1000 generations (iterations).

We compare our first method of GAs with the following method of Nelder-Mead.

***Second Method for minimizing** e:*

We use the method of Nelder-Mead with various initial values. We run the Nelder-Mead algorithm via *Matlab* ([16]).

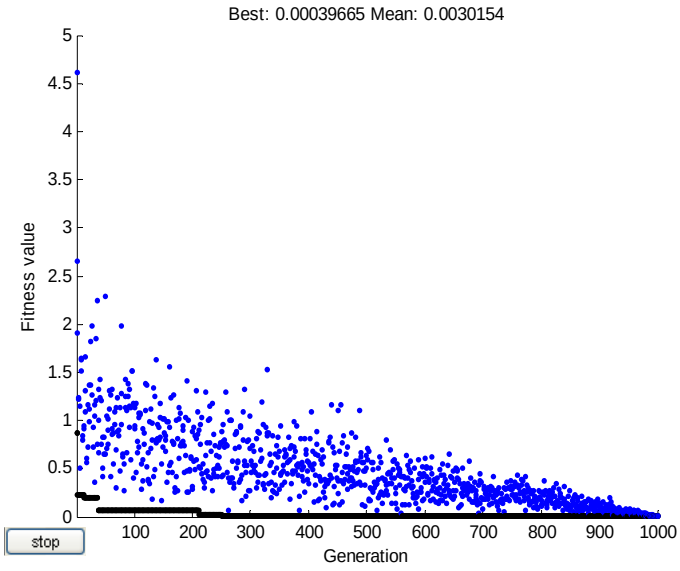
x_0	y_0	x	y	e
0	0	0.2920	1.8053	$3.0973*10^{-4}$
0	4	0.0006	1.9995	$3.0001*10^{-4}$
4	0	2.9993	0.0004	$3.9996*10^{-4}$
1	1	-1.3936	2.9290	$2.5355*10^{-4}$
1	2	0.4908	1.6728	$3.1636*10^{-4}$
2	1	-0.7538	2.5025	$2.7487*10^{-4}$

The Nelder-Mead method does not provide also satisfactory results.

***Third Method for minimizing** e (Hybrid Method):*

We use the first method, but after 1000 generations (iterations) we continue with Nelder-Meed (via *Matlab*, ([16]).
The results are much better, because in each case and independently on the initial values we obtain: $x=-9$ and $y=8$

So, using the aforementioned GA with random initial population, we find after 1000 generations $x=-2.8999$ and $y=0.0666$ and $e=3.9665*10^{-4}$
Using these values $x=-2.8999$, $y=0.0666$ for Nelder-Meed we find $x=-9$, $y=8$ and $e=8.0856*10^{-19}$



Example 2:

Consider again the system

$$2x+3y=6$$

$$2.0001\,x+3.0001\,y=5.9999$$

and the error

$$e=(6-(2x+3y))^2+(5.9999-(2.0001\,x+3.0001\,y))^2$$

First Method for minimizing e:

The error e can be considered as fitness function in our GA where we use:

- Initial Population: 20 individuals
- Fitness Scaling: We use Rank Scaling Option
- Selection: Tournament selection with Tournament size 4
- Reproduction: Elite Counter 2, Crossover Franction 0.8.
- Mutation: We use Gaussian Mutation with Scale 1 and Shrink 1.
- Crossover: Scattered
- Migration: Both Direction Migration with Fraction of Migration 0.2 and Interval of Migration 20
- Number of Generations: 1000

We run again our algorithm via *Matlab* ([16]) for 1000 generations and we take various solutions as in the following table. . The initial range for each variable *x*, *y* is [0,1].

x	y	e
3.8601	-0.5736	$2.1777*10^{-7}$
1.9899	0.6733	$1.0787*10^{-7}$
0.7546	1.4970	$3.9951*10^{-7}$
0.4230	1.7181	$1.7562*10^{-7}$
4.6694	-1.1130	$1.0479*10^{-7}$
5.4876	-1.6587	$1.5078*10^{-7}$
-0.5143	2.3429	$1.8869*10^{-7}$
-0.4182	2.2787	$1.1411*10^{-7}$
-1.9146	3.2763	$2.8597*10^{-7}$
-0.7863	2.5240	$3.7120*10^{-7}$

Second Method for minimizing e:

We use the method of Nelder-Mead with various initial values. We run the Nelder-Mead algorithm via *Matlab* ([16]).

x_0	y_0	x	y	e
0	0	0.3305	1.7796	$4.8399*10^{-8}$
0	10	0.0008	1.9995	$4.8500*10^{-8}$
0	1	0.0012	1.9992	$4.5207*10^{-8}$
1	0	2.9968	0.0020	$9.5757*10^{-8}$

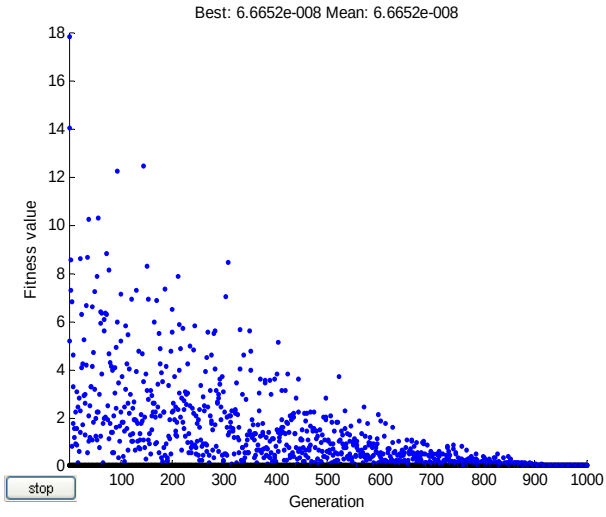
The Nelder-Mead method does not provide also satisfactory results in every case.

Third Method for minimizing e (Hybrid Method):

We use the first method, but after 1000 generations (iterations) we continue with Nelder-Meed (via *Matlab*, ([16]).
The results are much better, because in each case and independently on the initial values we obtain:
$x=-9$ and $y=8$

Thus, using the aforementioned GA with random initial population, we find after 1000 generations
$x=0.8644$ and $y=1.4237$ and $e=6.6652*10^{-4}$

Using these values $x=0.8644$, $y=1.4237$ as initial values for Nelder-Meed we find $x=-9$, $y=8$ and $e=1.0587*10^{-18}$



3 Numerical Results for Systems of Non-Linear Equations

Example 3:

Suppose the system of non-linear equations:

$$x^2 - 2x + 3y = -1$$

$$2x^2 - 3.9999x + 6.0001y = -1.9999$$

therefore $\det J(f(X^*)) \approx 0$. This creates problems in the numerical solution of the afore mentioned system

To overcome these difficulties we consider the error

$$e = \left|x^2 - 2x + 3y + 1\right| + \left|2x^2 - 3.9999x + 6.0001y + 1.9999\right|$$

First Method for minimizing e:

As first method we try to run the Genetic Algorithm with the aforementioned parameters.

So, we run our algorithm via *Matlab* ([16]) for 1000 generations and we take various solutions as in the following table. The initial range for each variable *x*, *y* is also [0,1].

x	y	e
1.5704	-0.1084	$5.3803*10^{-4}$
0.9379	-0.0013	$2.0284*10^{-4}$
-0.0998	-0.4033	$7.7005*10^{-4}$
0.7587	-0.0195	$6.7849*10^{-4}$
-0.0773	-0.3869	$1.1607*10^{-4}$
1.7718	-0.1986	$2.9938*10^{-4}$
0.4306	-0.1080	$1.3845*10^{-4}$
-0.5223	-0.7725	$2.0829*10^{-4}$
1.9471	-0.2990	$2.7950*10^{-4}$
-0.7527	-1.0240	$3.3292*10^{-4}$
2.3876	-0.6418	$1.5481*10^{-4}$
1.0577	-0.0011	$9.3678*10^{-5}$
1.8430	-0.2368	0.0010
1.6076	-0.1232	$8.6899*10^{-4}$

Second Method for minimizing e:

We use the method of Nelder-Mead with various initial values. We run the Nelder-Mead algorithm via *Matlab* ([16]).

x_0	y_0	x	y	e
0	0	0.1866	-0.2205	$7.3412*10^{-5}$
0	1	0.0057	-0.3295	$1.1253*10^{-4}$
0	2	0.0049	-0.3300	$1.5000*10^{-4}$
2	1	0.6420	-0.0427	$2.0056*10^{-5}$
2	2	2.4808	-0.7309	$3.7497*10^{-5}$
-1	-1	-0.8720	-1.1681	$1.5202*10^{-4}$

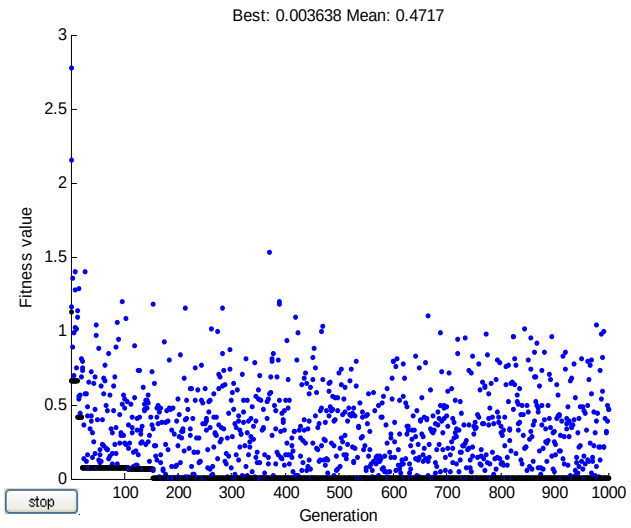
Third Method for minimizing e (Hybrid Method):

We use the first method, but after 1000 generations (iterations) we continue with Nelder-Meed (via *Matlab,* ([16]).

Unfortunately, the results in this system of non-linear equations are not so good as in the case of linear equations. This case is faced by increasing the mutation. We use *uniform mutation* with 0.3 rate.

That means that the algorithm will select a fraction of the vector entries of an individual for mutation, where each entry has a probability of 0.30 of being mutated. In the second step, the algorithm replaces each selected entry by a random digit 0 or 1. After this change, the results of our hybrid method are also impressive, as we have convergence to x=1 and y=0.

So, using the aforementioned GA (*uniform mutation* with 0.3 rate) starting from random initial population, we find after 1000 generations
$x = 0.99682$, $y = 0.00355$ and
$e = 9.2060*10^{-6}$.



Note the result of mutation in the previous figure.

In the sequel, we use these values $x = 0.99682$, $y = 0.00355$ as starting points for our Nelder-Meed algorithm and we find solution
$x \approx 1$ and $y \approx 0$

Example 4:

Here, we consider the system of non-linear equations:

$$x^2 - 2x + 3y = -1$$

$$2x^2 - 3.9999x + 6.0001y = -1.9999$$

with the error

$$e = (x^2 - 2x + 3y + 1)^2 + (2x^2 - 3.9999x + 6.0001y + 1.9999)^2$$

First Method for minimizing e:

We start again minimizing the error e (fitness function) using the Genetic Algorithm with the aforementioned parameters. We run our algorithm via *Matlab* ([16]) for 1000 generations. Some solutions are given in the following table. The initial range for each variable *x, y* is also [0,1].

x	y	e
0.3532	-0.1395	3.5571*10-7
2.0725	-0.3835	3.8563*10-8
-0.8753	-1.1722	2.5894*10-8
1.2539	-0.0215	1.0702*10-9
1.5044	-0.0848	7.3618*10-9
2.4817	-0.7318	1.1828*10-9
-0.2560	-0.5258	2.7406*10-8
-1.1089	-1.4826	1.7773*10-6
2.4342	-0.6856	2.9775*10-8
0.7235	-0.0256	7.8382*10-7
2.5727	-0.8246	4.1497*10-7
-0.4219	-0.6739	1.6545*10-8
-1.3301	-1.8097	5.0446*10-8
2.8456	-1.1354	2.7533*10-9

Second Method for minimizing e:

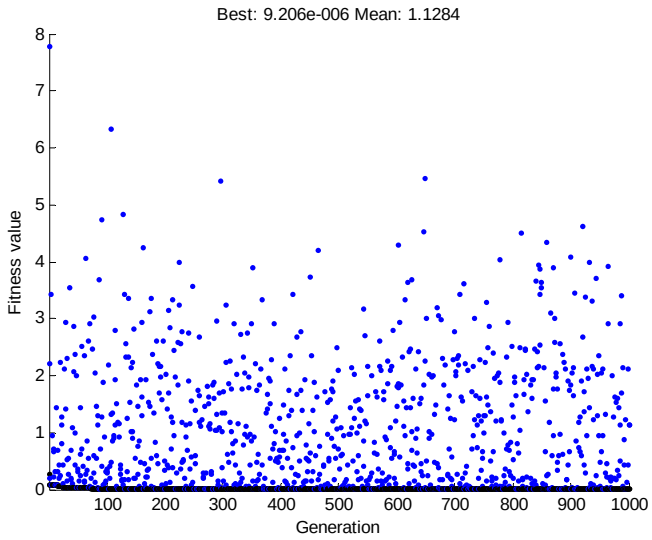
We can use again the method of Nelder-Mead with various initial values. We run the Nelder-Mead algorithm via *Matlab* ([16]).

x_0	y_0	x	y	e
0	0	0.1866	-0.2205	3.6829*10-9
0	1	0.0057	-0.3295	8.9839*10-9
0	2	0.0049	-0.3300	1.1985*10-8
2	1	0.6420	-0.0427	3.1230*10-10
2	2	2.4808	-0.7309	1.1247*10-9
-1	-1	-0.8720	-1.1681	1.7154*10-8

Third Method for minimizing** e (Hybrid Method)**:

We use the hybrid method where we change the mutation. We use *uniform mutation* with 0.3 rate. The results of our hybrid method are also wonderful, as we have convergence to x=1 and y=0. In this case, we have the error:

So, using the aforementioned GA (*uniform mutation* with 0.3 rate) starting from random initial population, we find after 1000 generations
$x = 0.99977$, $y = 0.00068$ and
$e = 2.0767 * 10^{-5}$.



Note the result of mutation in the previous figure.

In the sequel, we use these values $x = 0.99977$, $y = 0.00068$ as starting points for our Nelder-Meed algorithm and we find solution
$x \approx 1$ and $y \approx 0$

4 Conclusion

In this paper, an Hybrid Method (GA + Nelder Meed) is proved to be a strongest tool for solving of Ill-Conditioned Systems of Linear and/or Non-Linear Equations. The method is outlined via specific numerical examples. Some previous work of the author can be found in [7]÷[10].

References:

[1] Goldberg D.E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Second Edition, 1989

[2] Grefenstette J.J., Optimization of control parameters for Genetic Algorithms, IEEE Trans. Systems, Man and Cybernetics, SMC 16, Jan/Feb 1986, pp. 128

[3] Eberhart R., Simpson P. and Dobbins R. (1996), *Computational Intelligence PC Tools*, AP Professionals.

[4] Kosters W.A., Kok J.N. and Floreen P., Fourier Analysis of Genetic Algorithms, Theoretical Computer Science, Elsevier, 229, 199, pp. 143-175.

[5] Angel Fernando Kuri-Morales, "Solution of Simultaneous Non-Linear Equations using Genetic Algorithms", WSEAS Transactions on SYSTEMS, Issue 1, Volume 2, January 2003, pp.44-51

[6] E. Balagusuramy, *Numerical Methods*, Tata McGraw Hill, New Delhi, 1999

[7] Nikos E. Mastorakis, "Solving Non-linear Equations via Genetic Algorithms". Proceedings of the 6th WSEAS Conference on Evolutionary Computing, Lisbon, Portugal, June 16-18, 2005.

[8] Gonos I.F., Mastorakis N.E., Swamy M.N.S.: "A Genetic Algorithm Approach to the Problem of Factorization of General Multidimensional Polynomials", IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, Part I, Vol. 50, No. 1, pp. 16-22, January 2003.

[9] Mastorakis N.E., Gonos I.F., Swamy M.N.S.: "Design of 2-Dimensional Recursive Filters using Genetic Algorithms", IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, Part I, Vol. 50, No. 5, pp. 634-639, May 2003.

[10] Mastorakis N.E., Gonos I.F., Swamy M.N.S.: "Stability of Multidimensional Systems using Genetic Algorithms", IEEE Transactions on Circuits and Systems, Part I, Vol. 50, No. 7, pp. 962-965, July 2003.

[11] Andrew Curtis and Roel Snieder: "Reconditioning inverse problems using the genetic algorithm and revised parameterization", Geophysics, Vol. 62, No. 4.pp. 1524–1532, July-August 1997.

[12] Ralf Östermark, "Solving Irregular Econometric and Mathematical Optimization Problems with a Genetic Hybrid Algorithm", Computational Economics, Volume 13 , Issue 2, pp. 103 - 115 , April 1999.

[13] Lagarias, J.C., J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," SIAM Journal of Optimization, Vol. 9 Number 1, pp. 112-147, 1998.

[14] J. A. Nelder and R. Mead, "A simplex method for function minimization", Computer Journal, 7 , 308-313, 1965

[15] F. H. Walters, L. R. Parker, S. L. Morgan, and S. N. Deming, Sequential Simplex Optimization, CRC Press, Boca Raton, FL, 1991.

[16] *Matlab*, Version 7.0.0, by Math Works, Natick, MA, 1994 http://www.mathworks.com