A Real-Coded Genetic Algorithm with System Reduction and Restoration for Rapid and Reliable Power Flow Solution of Power Systems

Asst. Prof. Dr. Hassan Abdullah Kubba
Electrical Engineering Department
College of Engineering/ University of Baghdad
E-mail: hassankubba@yahoo.com

Alaa Suheib Rodhan
M.Sc. Electrical Engineering
college of Engineering/ University of Baghdad
E-mail: **alaa.rodhan@gmail.com**

ABSTRACT

The paper presents a highly accurate power flow solution, reducing the possibility of ending at local minima, by using Real-Coded Genetic Algorithm (RCGA) with system reduction and restoration. The proposed method (RCGA) is modified to reduce the total computing time by reducing the system in size to that of the generator buses, which, for any realistic system, will be smaller in number, and the load buses are eliminated. Then solving the power flow problem for the generator buses only by real-coded GA to calculate the voltage phase angles, whereas the voltage magnitudes are specified resulted in reduced computation time for the solution. Then the system is restored by calculating the voltages of the load buses in terms of the calculated voltages of the generator buses, after a derivation of equations for calculating the voltages of the load busbars. The proposed method was demonstrated on 14-bus IEEE test systems and the practical system 362-busbar IRAQI NATIONAL GRID (ING). The proposed method has reliable convergence, a highly accurate solution and less computing time for on-line applications. The method can conveniently be applied for on-line analysis and planning studies of large power systems.

Keywords: Load flow analysis, Load modeling, Power system modeling, Real Coded Genetic algorithms, Simulation, Voltage measurement

الحل السريع والموثوق لسريان الحمل الكهربائي بأستخدام الخوارزمية الجينية ذات التشفير الحقيقي مع أختزال الشبكة وأعادتها

أ.م. د. حسن عبدالله كبة
قسم الهندسة الكهربائية
كلية الهندسة/جامعة بغداد

علاء سحيب روضان
قسم الهندسة الكهربائية
كلية الهندسة/جامعة بغداد

الخلاصة

يقدم البحث طريقة عالية الدقة لحساب سريان الحمل الكهربائي و تقليل أحتمالية الانتهاء في الحدود الدنيا المحلية بأستخدام الخوارزمية الجينية ذات التشفير الحقيقي مع أختزال الشبكة وأعادتها وتم تطوير الطريقة المقترحة (الخوارزمية الجينية ذات التشفير الحقيقي) لتقليل زمن الحساب الكلي بتقليل حجم النظام الى عدد محطات التوليد فقط، بعد أختزال عدد محطات الأحمال في النظام الحقيقي أو الواقعي لتقليل الزمن اللازم للحساب ، ومن ثم يتم حساب زاوية طور الفولتية بعد تحديد مقدار الفولتية لكل محطة توليد بأستخدام الخوارزمية الجينية ذات التشفير الحقيقي ، بعد ذلك يتم أعادة تمثيل النظام ككل وحساب مقدار وزاوية طور الفولتيات لكل محطات الأحمال بأستخدام النتائج المستحصلة لمقدار وزاوية طور الفولتيات لكل محطات التوليد بعد أشتقاق المعادلات المطلوبة لحساب مقدار وزاوية طور فولتيات لكل محطات الأحمال بصيغة مقدار وزاوية فولتيات محطات التوليد ، الطريقة المقترحة تم تطبيقها للعمل على الشبكة الوطنية العراقية.الطريقة المقترحة عالية الدقة، موثوقة والزمن اللازم للوصول الى الحل قليل وكذلك ممكن تطبيقها في دراسات التحليل والتخطيط للأنظمة الكهربائية كبيرة الحجم وأثناء أشتغال المنظومة .

الكلمات الرئيسية: تحليل تدفق الحمل ، تمثيل الحمل ، تمثيل نظام القدرة الكهربائية ، الخوارزمية الجينية ذات التمثيل الحقيقي ، المحاكاة ، وحساب الفولتية.

1. INTRODUCTION

The power flow problem, which is to determine the power system static states (voltage magnitudes and voltage phase angles) at each busbar to find the steady state operating condition of a system, is very important and the most frequently carried out study by electrical power utilities for power system on-line operation, planning and control. The mathematical formulation of the electrical power flow problem results in a set of non-linear algebraic equations. The optimization numerical methods such as Newton-Raphson method or the artificial intelligence methods such as Genetic Algorithm (GA) are applied to solve the power flow problem. The power flow problem has multiple solutions, **Kubba 1991**.The numerical methods and some of the artificial intelligence methods suffer from the local minima problem. Also there are many criteria which should be taken into consideration such as the speed of solution, storage requirement and the degree of solution accuracy. With increasing computer speeds, researchers are increasingly applying artificial and computational intelligence techniques, especially in power system problems. These methods offer several advantages over traditional numerical methods. Among these techniques is that of *genetic algorithm*. Genetic algorithms (*GAs*) are efficient stochastic search techniques that emulate natural phenomena. They have been used successfully to solve a wide range of optimization problems. Because of existence of local minima, these algorithms offer promise in solving large-scale problems. A genetic algorithm mimics *Darwin's evolution process* by implementing *"survival of the fittest"* strategy. Genetic algorithm solves linear and nonlinear problems by exploring all regions of the search space and exponentially exploiting promising areas through *selection*, *crossover*, and *mutation* operations. They have been proven to be an effective and flexible optimization tool that can find optimal or near-optimal solutions, **Wong, et al., 1999**. In this study, an improved genetic algorithm solution of the load flow problem is presented in order to minimize the total *active and reactive power mismatches* of the given systems, a *real-coded* genetic algorithm has been implemented. The proposed method has been demonstrated on a typical test system, and was used to solve the Iraqi National Grid load flow problem.

2. THE REAL-CODED (CONTINUOUS) GENETIC ALGORITHM (RCGA)

The binary genetic algorithm is conceived to solve many optimization problems that stump traditional techniques. But, the attempting to solve a problem where the values of the variables are continuous and want to define them to the full machine precision. In such a problem, each variable requires many bits to represent it. If the number of variables is large, the size of the chromosome is also large. In principle, any conceivable representation could be used for encoding the variables. When the variables are naturally quantized, the binary genetic algorithm fits nicely. However, when the variables are continuous, it is more logical to represent them by *floating-point numbers, i.e., real number*. In addition, since the binary genetic algorithm has its precision limited by the binary representation of variables, using floating-point numbers instead easily allows representation to the machine precision. This continuous genetic algorithm also has the advantage of requiring less storage than the binary genetic algorithm because a single floating-point number represents the variable instead of N_{bits} integers. The continuous genetic algorithm is inherently faster than the binary genetic algorithm, because the chromosomes do not have to be decoded prior to the evaluation of the *cost function* (*objective function*), **Ippolito, et al., 2006**. Since the continuous GA is implemented using floating point numbers, i.e., real numbers we have called this as Real-Coded GA (RCGA).

3. MATHEMATICAL DESCRIPTION & COMPONENTS OF A CONTINUOUS GENETIC ALGORITHM (RCGA)

The real-coded genetic algorithm is very similar to the binary genetic algorithm, but the primary difference is the fact that variables are no longer represented by bits of zeros and ones, but instead by floating-point real numbers over whatever range is deemed appropriate. However, this simple fact adds some nuances to the application technique that must be carefully considered. In particular, we will present the RCGA operators, which are used in this research.

3.1 The Variables and Cost Function

A cost function generates an output from a set of input variables (a chromosome). The cost function may be a mathematical function, or from experiment. The objective is to modify the output in some desirable fashion by finding the appropriate values for the input variables. The goal is to solve some optimization problem where we search for an optimum (minimum) solution in terms of the variables of the problem. The term fitness is extensively used to designate the output of the *objective function* in the genetic algorithm literature. Fitness implies a maximization problem. Fitness has a closer association with biology than the term cost, and thus we have adopted the term cost, since most of the optimization literature deals with minimization, hence cost. They are equivalent. If the chromosome has N_{var} variables (a 2N-dimensional optimization problem) given by $(b_1, b_2, \ldots\ldots, b_{Nvar})$ where N is the number of buses, then the chromosome is written as an array with $(1\times N_{var})$ elements so that:

$$chromosome = [b_1, b_2, b_3, \ldots\ldots\ldots, b_{Nvar}] \tag{1}$$

In power flow problem, the chromosome is written in terms of the voltages magnitudes and voltages phase angles variables of all the buses as follows:

$$chromosome = [V_1, V_2, \ldots., V_N, \theta_1, \theta_2, \ldots\ldots, \theta_N] \tag{1.1}$$

In this case, the variable values are represented as floating-point numbers. Each chromosome has a cost found by evaluating the cost function (f) at the variables ($V_1, V_2, \ldots., V_N, \theta_1, \theta_2, \ldots.., \theta_N$).

$$cost = f(chromosome) = f(b_1, b_2, \ldots, b_{Nvar}) \tag{2}$$

Equations (1) and (2) along with applicable constraints constitute the problem to be solved. Our primary problem in this research is the continuous functions introduced below. The two cost functions are:

$$\Delta P_i = P_i^{sp} - V_i \sum_{k=1}^{N} V_k \left(G_{ik} \cos\theta_{ik} + B_{ik} \sin\theta_{ik}\right) \tag{3}$$

Where P_i^{sp} is the specified active power at bus i, eqn.3 is for ″PV″ (generator buses), and ″PQ″ (load buses),

$$\Delta Q_i = Q_i^{sp} - V_i \sum_{k=1}^{N} V_k \left(G_{ik} \sin\theta_{ik} - B_{ik} \cos\theta_{ik}\right) \tag{4}$$

Where Q_i^{sp} is the specified reactive power at bus i, eqn.4 is for PQ buses only, Where $\theta_{ik} = \theta_i - \theta_k$ and,(ΔP_i) is the mismatch active power at bus (i) and (ΔQ_i) is the mismatch reactive power at bus (i). (V_i, V_k, θ_i, θ_k) are the voltage magnitude and angle at buses (i) and (k) respectively, which are the variables of the two cost functions and (N) is the number of buses, **Kubba, 2008.**

3.2 Variable Encoding, Precision, and Bounds

Here, the difference between binary and continuous genetic algorithms is shown. It is no longer needed to consider how many bits are necessary to represent accurately a value. Instead, (V) and (θ) have continuous values that are limited between appropriate bounds which are in our problem, $0.9 \leq V \leq 1.1$ and $-20 \leq \theta \leq 20$. Since the genetic algorithm is a search technique, it must be limited to exploring a reasonable region of variable space. Sometimes, this is done by imposing a constraint on the problem. If one does not know the initial search region, there must be enough diversity in the initial population to explore a reasonably sized variable space before focusing on the most promising regions.

3.3 Initial Population

The genetic algorithm starts with a group of chromosomes known as the *population*. A matrix represents the population with each row in the matrix being a $(1\times N_{var})$ array (chromosome) of continuous values. Given an initial population of N_{ind} chromosomes, the full matrix of $(N_{ind}\times N_{var})$ random values is generated. All variables are normalized to have values between 0 and 1, the range of a uniform random number generator. The values of a variable are "unnormalized" in the cost function. If the range of values is between b_{lo} and b_{hi}, then the unnormalized values are given by:

$$b=(b_{hi}-b_{lo})b_{norm}+b_{lo} \tag{5}$$

where, b_{hi} is highest number in the variable range, b_{lo} is lowest number in the variable range, and b_{norm} is normalized value of variable. This society of chromosomes is not a democracy; the individual chromosomes are not all created equal. Each one's worth is assessed by the cost function. So at this point, the chromosomes are passed to the cost function for evaluation. In this research, we had used a population size (initial population) of 20 individuals (chromosomes) for 14-bus IEEE system power flow solution and 500 individuals for 362-bus Iraqi National Grid (ING) power flow solution which depends on the number of variables for each system. These population sizes are kept constant throughout the whole solution process.

3.4 Natural Selection

Survival of the fittest translates into discarding the chromosomes with the higher costs. First, the N_{ind} costs and associated chromosomes are ranked from lowest cost to highest cost. Then, only the best are selected to continue, while the rest are deleted. The selection rate, X_{rate}, is the fraction of N_{ind} that survives for the next step of mating. The number of chromosomes that are kept each generation is:

$$N_{keep}=X_{rate}.N_{ind} \tag{6}$$

Natural selection occurs each generation or iteration of the algorithm. Of the N_{ind} chromosomes, only the top N_{keep} survive for mating, and the bottom $(N_{ind} - N_{keep})$ are discarded to make room for the new offspring. Deciding how many chromosomes to keep is somewhat arbitrary. Letting only a few chromosomes survive to the next generation limits the available genes in the offspring. Keeping too many chromosomes allows bad performers a chance to contribute their traits to the next generation. We use 50% (X_{rate}=0.5) in the natural selection process. Another approach to natural selection is called *thresholding* (Truncation Selection) is used in this research. In this approach, all chromosomes that have a cost function lower than some truncation threshold survive. The threshold must allow some chromosomes to continue in order to have parents to produce offspring. Otherwise, a whole new population must be generated to find some chromosomes that pass the test. At first, only a few chromosomes may survive. In later generations, however, most of the chromosomes will survive unless the threshold is changed. An attractive feature of this technique is that the population does not have to be sorted.

3.5 Selection

In this process, two chromosomes are selected from the mating pool of N_{keep} chromosomes to produce two new offspring. Pairing takes place in the mating population until $(N_{ind} - N_{keep})$ offspring are born to replace the discarded chromosomes. Pairing chromosomes in a genetic algorithm can be as interesting and varied as pairing in an animal species. Two types of selection are used in this research, which are:

3.5.1. Rank-weighted roulette wheel: This approach uses a uniform random number generator to select chromosomes. The row numbers of the parents are found using:

$ma = ceil\ (N_{keep} * rand(1, N_{keep}/2))$
$pa = ceil\ (N_{keep} * rand(1, N_{keep}/2)),$

Where *ceil* rounds the value to the next highest integer and *rand* generates arrays of random numbers whose elements are uniformly distributed in the interval (0, 1). This approach is problem independent and finds the probability from the rank of the chromosome. Rank weighting is slightly more difficult to program than the other selection types. Small populations have a high probability of selecting the same chromosome. The probabilities only have to be calculated once. We tend to use rank weighting because the probabilities do not change each generation. This approach of selection had been used in 14-bus IEEE-system.

3.5.2. Tournament selection: Another approach that closely mimics mating competition in nature is to randomly pick a small subset of chromosomes (two or three) from the mating pool, and the chromosome with the lowest cost in this subset becomes a parent. The typical value accepted by many applications is k =2 (so-called tournament size). The tournament repeats for every parent needed. Thresholding and tournament selection make a nice pair, because the population never needs to be sorted. Tournament selection works best for large population sizes because sorting becomes time-consuming for large populations. Each of the parent selection schemes results in a different set of parents. As such, the composition of the next generation is different for each selection scheme. Rank-weighted Roulette-wheel and tournament selection are standard for most genetic algorithms. It is very difficult to give advice on which selection scheme works best. In our problem, we follow the *roulette-wheel* and *tournament* parent selection procedures for 14-bus IEEE-system and 362-bus ING respectively, **Younes, and M. Rahli, 2006.**

3.6 Crossover (Recombination)

As for the binary algorithm, two parents are chosen, and the offspring are some combination of these parents. Many different approaches have been tried for crossing over in continuous genetic algorithm. The simplest methods choose one or more points in the chromosome to mark as the *crossover points*. Then the variables between these points are merely swapped between the two parents. For example, consider the two parents to be:

$parent\ 1 = [b_{m1}, b_{m2}, b_{m3}, b_{m4}, b_{m5}, b_{m6}, \ldots\ldots, b_{mNvar}]$
$parent\ 2 = [b_{d1}, b_{d2}, b_{d3}, b_{d4}, b_{d5}, b_{d6}, \ldots\ldots, b_{dNvar}]$

Crossover points are randomly selected (at points (3, 4)), and then the variables in between are exchanged:

$offspring\ \ 1 = [b_{m1}, b_{m2}, b_{d3}, b_{d4}, b_{m5}, b_{m6}, \ldots\ldots, b_{mNvar}]$
$offspring\ \ 2 = [b_{d1}, b_{d2}, b_{m3}, b_{m4}, b_{d5}, b_{d6}, \ldots\ldots, b_{dNvar}]$

The extreme case is selecting N_{var} points and randomly choosing which of the two parents will contribute its variable at each position. Thus, one goes down the line of the chromosomes and, at each variable, randomly chooses whether or not to swap information between the two parents. This method is called *uniform crossover*:

$offspring\ \ 1 = [b_{m1}, b_{d2}, b_{m3}, b_{m4}, b_{d5}, b_{m6}, \ldots\ldots, b_{dNvar}]$
$offspring\ \ 2 = [b_{d1}, b_{m2}, b_{d3}, b_{d4}, b_{m5}, b_{d6}, \ldots\ldots, b_{mNvar}]$

The problem with these point crossover methods is that no new information is introduced; each continuous value that was randomly initiated in the initial population is propagated to the next generation, only in different combinations. Although this strategy works fine for binary representations, there is now a continuum of values, and in this continuum we are merely interchanging two data points. These approaches totally rely on mutation to introduce new genetic material. The blending methods remedy this problem by finding ways to combine variable values from the two parents into new variable values in the offspring. A single offspring variable value b_{new} comes from a combination of the two corresponding parents variable values:

$$b_{new}=\beta b_{mn}+(1-\beta)b_{dn} \qquad (7)$$

Where, β is a random number on the interval [**0,1**], b_{mn} = n^{th} variable in the mother chromosome, b_{dn} = n^{th} variable in the father chromosome.

The same variable of the second offspring is merely the complement of the first (*i.e.* replacing β by 1 – β). If β = 1, then b_{mn} propagates in it's entirely and b_{dn} dies. In contrast, if β = 0, then b_{dn} propagates in it's entirely and b_{mn} dies. When β = 0.5, the result is an average of the variables of the two parents. This method has been demonstrated to work well on several interesting problems. Choosing which variables to blend is the next issue. Sometimes, this linear combination process is done for all variables to the right or to the left of some crossover point, **Woon, 2004**. Any number of points can be chosen to blend, up to N_{var} values where all variables are linear combinations of those of the two parents. The variables can be blended by using the same β for each variable or by choosing different β's for each variable. These blending methods effectively combine the information from the two parents and choose values of the variables between the values bracketed by the parents; however, they do not allow introduction of values beyond the extremes already represented in the population. Of course, the factor (0.5) is not the only one that can be used in such a method. Heuristic crossover is a variation where some random number β is chosen on the interval [0, 1] and the variables of the offspring are defined by:

$$b_{new} = \beta(b_{mn}-b_{dn})+b_{dn} \qquad (8)$$

Variations on this theme include choosing any number of variables to modify and generating different β for each variable. This method also allows generation of offspring outside of the values of the two parent variables. Sometimes, values are generated outside of the allowed range. If this happens, the offspring is discarded and the algorithm tries another β. In our problem, we want to find a way to closely mimic the advantages of the binary genetic algorithm scheme. It begins by randomly selecting a variable c in the first pair of parents to be the crossover point, **Yin, 1993**:

$$c= round\ up\ \{random*N_{var}\} \qquad (9)$$

Where, (round up) is rounding mode that rounds to the nearest allowable quantized value. We'll let: parent 1 = [b_{m1}, b_{m2}, ……, b_{mc}, ……, b_{mNvar}] parent 2 = [b_{d1}, b_{d2}, ……, b_{dc}, ……, b_{dNvar}], Where (m) and (d) subscripts discriminate between the *mom* and *dad* parent. Then, the selected variables are combined to form new variables that will appear in the children:

$$b_{new1} = b_{mc} - \beta\ (b_{mc} - b_{dc})$$
$$b_{new2} = b_{dc} + \beta\ (b_{mc} - b_{dc})$$

Where, β is also a random value between 0 and 1. The final step is to complete the crossover with the rest of the chromosome as in binary genetic algorithm:

offspring 1 = [b_{m1}, b_{m2}, ……, b_{new1}, ……, b_{dNvar}]
offspring 2 = [b_{d1}, b_{d2}, ……, b_{new2}, ……, b_{mNvar}]

If the first variable of the chromosomes is selected, then only the variables to the right of the selected variable are swapped. If the last variable of the chromosomes is selected, then only the variables to the left of the selected variable are swapped. This method does not allow offspring variables outside the bounds set by the parent unless $\beta > 1$, **Younes, and Rahli, 2006-Jain, and Martin1, 1998**.

3.7 Mutation

Random mutations alter a certain percentage of the genes in the list of chromosomes. If care is not taken, the genetic algorithm can converge too quickly into one region of the *cost surface*. If this area is in the region of the *global minimum*, that is good. However, some functions, such as the one we are modeling, have many *local minima*. If nothing is done to solve this tendency to converge quickly, it may end up in a local rather than a global minimum. To avoid this problem of overly fast convergence (premature convergence), the routine is forced to explore other areas of the cost surface by randomly introducing changes, or mutations, in some of the variables. Mutation points are randomly selected from the ($N_{ind} \times N_{var}$), total number of genes in the population matrix.

Increasing the number of mutations increases the algorithm's freedom to search outside the current region of variable space. It also tends to distract the algorithm from converging on a popular solution. With the process of the crossover and mutation taking place, there is a high chance that the optimum solution could be lost as there is no guarantee that these operators will preserve the fittest string. To counteract this, *elitist* models are often used. In an elitist model, the best individual in the population is saved before any of these operations take place. After the new population is formed and evaluated, it is examined to see if this best structure has been preserved. If not, the saved copy is reinserted back into the population. The genetic algorithm then continues on as normal, **Ibrahim, 2005- Vasconcelos, et al., 2002.**

4. PROPOSED TECHNIQUE

In the proposed method the load busbars are eliminated, retaining only generator busbars for the iterative process. The system equations in terms of generator busbars and load busbars can be written as:

$$\begin{bmatrix} I_G \\ I_L \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} V_G \\ V_L \end{bmatrix} \tag{10}$$

If the voltage of the Kth load busbar is initially assumed to be $V_{Lk} = 1.0\angle 0^o$. Then the current in the busbar to the load is:

$$I_{Lk} = \frac{P_{Lk} - jQ_{Lk}}{V*_{Lk}} \tag{11}$$

From the second row of eqn. 10, we have

$$V_L = -Y_{22}^{-1}\, Y_{21}\, V_G + Y_{22}^{-1}\, I_L \tag{12}$$

Substituting eqn. 12 in the first row of eqn. 10, we get

$$I_G = Y_{11}\, V_G + Y_{12}(-Y_{22}^{-1}\, Y_{21}\, V_G + Y_{22}^{-1}\, I_L\,) \tag{13}$$

The above equation is written as

$$I_G = Y_{GG}\, V_G + Y_{GL} I_L \tag{14}$$

Where: $Y_{GG} = Y_{11} - Y_{12}\, Y_{22}^{-1}\, Y_{21}$ and $Y_{GL} = Y_{12} Y_{22}^{-1}$ (15)

From eqn. 14, the i^{th} generator busbar is:

$$I_i = \sum_{k=1}^{m} Y_{ik}\, V_k + a_i \text{ , for } i = 1,2,\ldots,m. \quad (16)$$

Where a_i is the i^{th} element of the column vector A given by

$$A = Y_{GL}\, I_L \quad (17)$$

The complex power at the busbar is

$$S_i = V_i^*\, I_i = V_i^* \sum_{k=1}^{m} Y_{ik}\, V_k + V_i^*\, a_i$$

for i=1,2,……,m. (18)

The real power injection at the busbar is

$$P_i = Re\, S_i = \sum_{k=1}^{m} e_i\, (e_k G_{ik} - f_k B_{ik}) + \sum_{k=1}^{m} f_i\, (e_k B_{ik} + f_k G_{ik}) + L_i$$

for i=1,2,….,m. (19)

Where $L_i = e_i c_i + f_i d_i$ (20)

(L_i) can be considered as an equivalent local load at generator busbar i due to elimination of the load busbars **, Mithulananthan, et al., 2004** .

5. COMPUTER ALGORITHM OF THE PROPOSED METHOD

The computer algorithm for the proposed method is as follows:

1. Read the lines data and form the nodal admittance matrix.

2. Read the busbars data, such as the specified active power, voltage magnitude of the generator buses, specified active and reactive power of the load buses, slack bus voltage, and initial estimate of the voltage of the load buses, assuming (1.0 p.u., 0.1 MW/MVAr)

3. Eliminate the load busbars and reduce the network to the size of that of the generators busbars.

4. Compute (I_L) using **Eqn. (11)** for all load buses, form the column vector (A) given by **Eqn. (17)**, then form (L_i) assuming (e_i) equal to the specified values, and (f_i) initially is zero.

5. Execute the Real-Coded Genetic Algorithm on the generator buses only to find the most recent value of the voltages, implementing all the GA operators such as Selection with Rank-Weighting Roulette Wheel, Tournament selection with truncation threshold, Single-point Crossover with blending method, and Mutation (rate of Mutation=0.2), we use initial population of 20 chromosomes for 14-bus IEEE system and 500 chromosomes for Iraqi National Grid (ING) system. At each generation (iteration) of the GA, we calculate the most recent values of (V_L) from **Eqn. (12)**, (I_L) from **Eqn. (11)** and (L_i), then calculate (P_i) from **Eqn. (19)**.

6. Convergence Test: The mismatch active powers for the generator buses (cost function) are calculated at each GA generation (iteration) according to the following equation:

$$\Delta P_i = P_i^{sp} - P_i^{cal} \text{ , for } i=1,2,\ldots,m. \quad (21)$$

When the mismatch active powers (cost function) for all generator buses except the slack bus are less than a small tolerance value (usually 0.001), 0.1MW/MVAR then the solution has converged.

7. Restore the system and calculate the load busbars voltages using **Eqn. (12)**.

8. Print results and end.

6. IMPLEMENTATIONS AND RESULTS

Two test systems were used to demonstrate the performance of the proposed method, namely:
1. 14 busbars IEEE International test system, the lines and buses data are present in, **Kubba, 1991.** The ″14- bus″ test system consists of: 1 slack bus, 4 generator buses (PV) and 9 load buses(PQ).
2. The Iraqi National Grid (ING) which consists 362 busbars, 1 slack bus, 29 generator buses (PV) and 332 load buses (PQ) , **Al-Bakri, 1994** .

The load flow solution using real-coded genetic algorithm programs with and without the method of Reduction and Restoration have been developed by the use of MATLAB version 7, and tested with a Pentium 4, 3GHz (Cache 2M) PC with 2GB RAM. **Table 1** illustrates the power flow solution for a 14-bus IEEE test system using conventional RCGA with two objective functions, which are the mismatch active and reactive powers at each bus according to its constraints except the slack bus. The sum of weighted cost multi-objective functions is used. The most straightforward approach to multi-objective optimization is to weight each function and add them together, Abido**, 2003**.

$$cost = \sum_{i=1}^{h} w_i f_i \tag{22}$$

Where f_i is the cost function (i), w_i is the weighting factor, *h is the number of objective functions,* and

$$\sum_{i=1}^{h} w_i = 1. \tag{23}$$

Implementing this multiple objective optimization approach in a real-coded genetic algorithm only requires modifying the cost function to fit the form of **Eqn. (22)** and does not require any modification to the genetic algorithm. Thus, **Eqn. (22)** becomes:

$$cost = wf_1+(1-w)f_2 \tag{24}$$

Where f_1 and f_2 are the mismatch active and reactive powers respectively, and have the same rank of importance. This approach is adopted in this research for its simplicity, easy of programming and gives us the required accuracy. Here, (w) is chosen to be (0.5), **Riccieri, and Falcao, 1999**.
Because of the stochastic nature of the genetic algorithm process, each independent run will probably produce a different number of generations and consequently the computation time and the best amongst these should be chosen. The best of the 10 implementations runs are shown in the tables. The total computation time was 7.156 sec. **Table 2** illustrates the power flow solution of the same IEEE test system using RCGA with the method of system Reduction and Restoration (Proposed Method). Since, we only retain the generator buses for the GA process, so a single objective function (mismatch active power) is needed. The total computation time for the whole load flow solution was 0.18 second. The power flow solution results for the Iraqi National Grid (362-bus) by using RCGA with the method of system reduction and restoration were tabulated in **Table 3** and **Table 4**. Since the proposed method (RCGA with system Reduction and Restoration) implements the complete cycles of the genetic algorithm on the generator busbars only which are the first thirty buses of the system, then **Table 3** shows the results and number of generations for each generator busbar and the power flow solution for the total Iraqi National Grid are presented. **Table 4** shows the voltages of load buses which are calculated after restoring the system, also the mismatch active and reactive powers of load buses are presented. The total computation time with conventional RCGA method was more than 72 hours, while the total computation time for the proposed method was 519 seconds for the whole load flow solution of 362-bus ING with the same accuracy. A ranked-weighted roulette wheel and Tournament selection process were used for 14-bus IEEE and ING respectively**. Figure1** shows the evaluation process of the genetic algorithm for bus 2 of 14-bus IEEE system, the dotted curve represents the

minimum cost of the solution (chromosome) which is converged with 15 generations and the solid curve represents the average value of the costs amongst generations versus the number of generations.

7. CONCLUSIONS

The proposed method which had presented in this paper is very much faster than the simple real-coded genetic algorithm, since the system is reduced to the size of that of the generator busbars which for any realistic system is small as we see for the 362-bus Iraqi National Grid, only 30 buses are generator busbars. We must take into consideration that the main drawback of the genetic algorithm is the large computation time. So, this contribution is especially for GA as an optimization technique. The objective function (cost function) for the generator buses is only the mismatch active power, so that multi-objective function techniques are not needed. Thus, it can be concluded that the proposed method is suitable for on-line implementation for small and medium-scale power systems and it can be used for planning study for large-scale systems. The proposed method has reliable convergence and high accuracy of solution. Whereas the traditional numerical techniques (Gauss-Seidel, Newton-Raphson, Fast decoupled,…etc.) use the characteristics of the problem to determine the next sampling point (*e.g.* gradient, linearity and continuity), genetic algorithm makes no such assumptions. Instead, the next sampled point is determined based on stochastic sampling or decision rules rather than on a set of deterministic decision rules. Genetic algorithms with the method of system reduction and restoration have been used to solve difficult problems with objective functions that possess properties such as *continuity, differentiability* and so forth. Also, whereas the traditional numerical techniques mentioned above use single point at a time to search the problem space, genetic algorithm uses a population of candidate solutions for solving the problem, thus reducing the possibility of ending at a local minima.

8. REFERENCES

Al-Bakri, ″A Study of Some Problems on the IRAQI NATION AL GRID and Establishing a Method Algorithm for Load Flow,″ *M.Sc Thesis , University of Baghdad*, 1994.

H. A. Kubba, A. S. Hassan, and T. Krishnaparandhama, ″Comparative Study of Different Load Flow Solution Methods,″ *Al-Muhandis, Refe- reed Scientific Journal of Iraqi Engineers Society*, Vol. 107, pp. 25-46, December 1991.

H. A. Kubba, ″ An Efficient and more Reliable Second Order Load Flow Solution Method,″ *Journal of Association for the Advancement of Modeling & Simulation Techniques in Enterprices*, 2009.

H. A. Kubba, Omar R, and Soltani j" A multi-objective genetic algorithm for a rapid and efficient load flow solution for electrical power systems." Proceedings of the international conference on modelling and simulation, Petra, Jordan; 18– 20 November 2008. p.14–9.

H. T. Yang, P.C. Yang, and C. L. Huang, ″A Parallel Genetic Algorithm Approach to Solving the Unit Commitment Problem: Implementation on the Transporter Networks,″ *IEEE Transactions on Power Systems*, Vol. 12, No. 2, pp. 661-668, May 1997.

J. A. Vasconcelos, R. L. S. Adriano, D. A. G. Vieira, G. F. D. Souza, and H. S. Azevedo, ″NSGA with Elitism Applied to Solve Multi-Objective Optimization Problems,″ *Journal of Microwaves and Optoelectronics*, Vol. 2 No. 6, pp. 59-69, December 2002

K. P. Wong, A. Li, and T. M. Law, ″Advanced Constrained Genetic Algorithm Load Flow Method,″ presented in *IEEE Proceedings on genera tor, transmission, distribution*, Vol. 146, No. 6, November 1999.

L. C. Woon, "Genetic Algorithm for Load Flow Solution Techniques," *Master of engineering (electrical), Universiti Technologi Malaysia*, 2004, http://www.sps.utm.

L. Ippolito, A. La Cortiglia, and M. Petrocelli, "Optimal Allocation of Facts Devices by Using Multi-Objective Optimal Power Flow and Genetic Algorithms," *International journal of emerging electric power systems*, Vol. 7, No. 2, pp. 1-19, 2006.

L. C. Jain, and N. M. Martin, "Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications," CRC Press, CRC Press LLC, 1998.

M. Younes, and M. Rahli, "On the Choice Genetic Parameter with Taguchi Method Applied in Economic Power Dispatch," *Leonardo Journal of Sciences*, issue 9, pp. 9-24, July/December 2006.

M. A. Abido, "Environmental/Economic Power Dispatch Using Multi- Objective Evolutionary Algorithms," *IEEE Trans. Power Syst.*, Vol. 18, No. 4, pp.1529–1537, Nov. 2003.

N. Mithulananthan, O. Than, and Le Van Phu, " Distributed Generator Placement in Power Distribution System Using Genetic Algorithm to Reduce Losses," *Thammasat Int. J. Sc. Tech.*, Vol. 9, No. 3, pp. 55- 62 July/September 2004.

O. F. Riccieri, and D. M. Falcao, "Meter Placement Method for State Estimation using Genetic Algorithms," *Intelligent System Application to Power Systems (ISPA), Rio de Janeiro, Brazil*, pp. 360-364, April 1999.

S. B. M. Ibrahim, "The PID Controller Design Using Genetic Algorithm," *A dissertation submit ted to University of Southern Queensland, Faculty of engineering and surveying, Electrical and Electronics Engineering*, October 2005.

T. Bouktir, L. Slimani, and M. Belkacemi, "A Genetic Algorithm for Solving the Optimal Power Flow Problem," *Leonardo journal of sciences*, pp. 44-58, January/June 2004.

X. Yin., "Application of Genetic Algorithm to Multiple Load Flow Solution Problem in Electrical Power Systems," presented in *IEEE Proceedings of the 32nd conference on decision and control, San Antonio, Texas,* December 1993.

NOMENCLATURE

N = number of busbars in the system.
m = number of generator busbar in the system.
V_G = M-dimensional vector of voltages of generator busbars.
I_G = M-dimensional vector of currents of generator busbars.
V_L = (N-M) dimensional vector of voltages of load busbars.
I_L = (N-M) dimensional vector of currents of load busbars.
Y= admittance matrix of order NxN.
Y_{11}, Y_{12}, Y_{21}, Y_{22} = sub-matrices of Y of appropriate order.
V_k^* = conjugate of kth busbar voltage V_k.
e_k , f_k = inphase and quadrature components of V_k.
c_i , d_i = real and imaginary parts of a_i.
sp = specified value.
cal = calculated.
G_{ik}, B_{ik} = real and imaginary parts of the admittance Y_{ik}

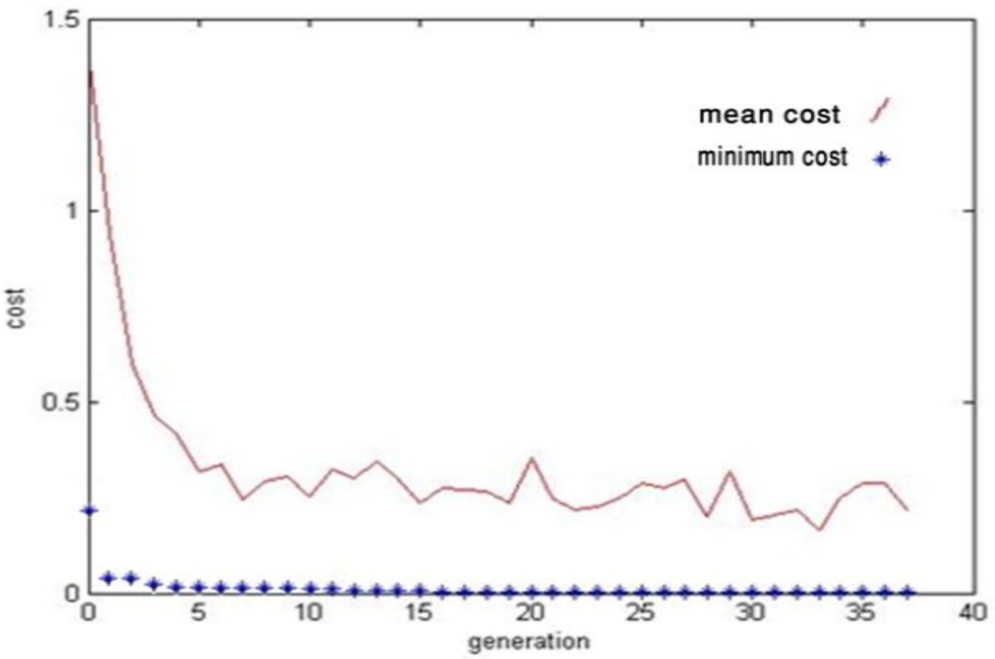

Figure1 Evaluation process for busbar (2), 14- bus IEEE test system

Table 1 Power Flow Solution (14-Bus IEEE) Test System with accuracy (0.001p.u.), using RCGA Without Reduction & Restoration

Bus No	Active power mismatch	Reactive power mismatch	Voltage Magnitude (p.u)	Voltage Angle (deg)	No. of generations
1	Slack	Slack	1.06	0.00	—
2	0.000329	PV	1.045	3.2117	17
3	0.000131	PV	1.010	-4.3582	7
4	0.000484	PV	1.070	-6.1436	21
5	0.000890	PV	1.090	-12.423	47
6	0.000798	0.000481	1.057131	6.30252	95
7	0.000365	0.000060	1.0773818	-4.6541	107
8	0.000222	0.000773	1.0565362	-1.7120	193
9	0.000185	0.000682	1.0456395	1.44081	172
10	0.000273	0.000322	1.045163	-9.0031	18
11	0.000950	0.000223	1.057696	-5.4828	90
12	0.000411	0.000535	1.061725	7.67754	43
13	0.000770	0.000521	1.0482889	-11.028	29
14	0.000209	0.000762	1.0588537	-3.3446	47
Total	Computational	Time:			7.156 sec.

Table 2 Power Flow Solution (14-Bus IEEE) Test System with accuracy (0.001p.u.), using RCGA With Reduction & Restoration

Bus No.	Active power mismatch	Reactive power mismatch	Voltage magnitude(p.u)	Voltage angle(deg.)	No. of Generations
1	Slack	Slack	1.06	0.00	–
2	0.000373	PV	1.045	3.2117	15
3	0.000130	PV	1.010	-3.35826	5
4	0.000374	PV	1.070	-6.10062	21
5	0.000890	PV	1.090	-11.0235	40
6	0.000678	0.000444	1.0476131	6.30252	-
7	0.000360	0.0006	1.0573888	-4.6999	-
8	0.000223	0.000788	1.065092	-2.71203	-
9	0.000109	0.0005	1.0558895	1.63081	-
10	0.000223	0.000321	1.0551690	-9.03316	-
11	0.000850	0.000221	1.0476990	-4.08283	-
12	0.000407	0.000546	1.0762725	6.60054	-
13	0.000660	0.000512	1.0482889	-11.0208	-
14	0.000205	0.000769	1.0688837	-3.34469	-
Total	Computational	Time:			0.18 sec.

***Table 3** Power Flow Solution For "IRAQI NATIONAL GRID" with accuracy (0.001p.u.), using RCGA with the method of Reduction and Restoration (Only the Generator Busbars)

Bus No.	Active power mismatch (p.u)	Reactive power mismatch (p.u)	Voltage magnitude (p.u)	Voltage Angle(deg)	No. of Generations
1	Slack	Slack	1.04	0	—
2	0.0005	PV	1	18.3805	262
3	0.00021334	PV	1	2.8233	57
4	0.0008081	PV	1	-9.5400	319
5	0.00011245	PV	1	13.6445	521
6	0.00043106	PV	1	-11.8520	34
7	0.0018487	PV	1	4.1875	500
8	0.00066843	PV	1	7.5529	244
9	0.00023882	PV	1	12.3150	30
10	0.00016648	PV	1	4.0006	134
11	0.0003391	PV	1	-19.7704	88
12	0.00045458	PV	1	-6.3530	266
13	0.00013682	PV	1	4.5221	424
14	0.00058912	PV	1	-6.9794	76
15	0.00054176	PV	1	-8.1968	353
16	0.00021063	PV	1	13.5898	42
17	0.00078201	PV	1	4.5766	39
18	$2.4477*10^{-6}$	PV	1	11.1094	41
19	0.00090163	PV	1	7.0672	47
20	0.00089409	PV	1	-7.0275	9
21	0.00037127	PV	1	-3.2876	159
22	0.00014522	PV	1	-10.7986	24
23	0.00093387	PV	1	2.0421	216
24	0.00084462	PV	1	9.0268	47
25	0.00038532	PV	1	2.9669	17
26	0.00023586	PV	1	3.8338	52
27	$7.2047*10^{-6}$	PV	1	-6.8666	88
28	0.00011686	PV	1	0.0252	50
29	0.00026843	PV	1	7.3612	333
30	0.0005791	PV	1	9.1833	134

*Total computing time (Genetic Algorithm without the method of Reduction and Restoration): more than 72 hours.
*Total computing time (Genetic Algorithm with the method of Reduction and Restoration): 519 sec., this time is for the total load flow solution of 362-bus ING system.

Table 4 Power Flow Solution For "IRAQI NATIONAL GRID" (Load Buses) After System Restoration

Bus No.	Active power mismatch (p.u)	Reactive power mismatch (p.u)	Voltage magnitude (p.u)	Voltage Angle(deg.)
31	0.0003	0.0002	1.02247	12.3782
32	0.000310	0.00013264	1.03356	-12.5347
33	0.000761	0.00088425	0.955482	-18.8922
34	0.00048	0.000007	0.981021	-3.14901
35	0.00082	0.00019414	0.969731	-7.5778
36	0.0007	0.0004	0.999866	0.40992
37	0.00094	0.00059979	0.99142	-0.108361
38	0.00072	0.0000028	1.02766	2.41241
39	0.00010	0.000031	0.951085	8.90456
40	0000073	0.000082	0.978972	7.68687
41	0.000003	0.00014068	0.954069	-2.09315
42	0.00056231	0.00034232	0.951974	10.2677
43	0.00015186	0.00084486	0.956778	9.06423
44	0.00093275	0.00060546	0.955108	-13.2829
45	0.00085845	0.0000039	1.00216	-8.97243
46	0.000025	0.00022034	0.955224	-7.08481
47	0.0005849	0.00025364	0.96548	-8.2154
48	0.0006	0.0009	0.96959	0.0359168
49	0.00046407	0.00068482	1.01733	8.11071
50	0.00064014	0.00040881	0.96882	16.9887
51	0.00081139	0.00073374	1.01725	12.1607
52	0.00063582	0.00036833	0.999238	0.015595
53	0.00064585	0.00073222	0.95709	7.0377
54	0.00039461	0.00031144	1.02049	4.97693
55	0.00050197	0.00051873	0.95678	18.5084
56	0.0006757	0.00066047	1.02961	13.4216
57	0.00048624	0.00046999	1.01989	13.0202
58	0.00026657	0.00078447	1.03154	-4.56727
59	0.00020212	0.00087574	0.998382	-0.133244
60	0.00099704	0.00040061	0.969316	-9.13924
61	0.0002	0.0005	1.09633	-3.3891
62	0.00062866	0.00048481	1.01166	3.0274
63	0.000034	0.00050729	1.00118	11.0695

64	0.00052304	0.0006	0.956323	0.126932
65	0.00039148	0.00027503	0.959269	-3.93418
66	0.0003338	0.00048038	0.959048	5.45821
67	0.00045369	0.00074015	1.03691	8.05995
68	0.00077681	0.00053799	0.962027	-0.556099
69	0.00027057	0.00061177	1.0712	5.11859
70	0.00040663	0.00054506	0.959477	-13.4194
71	0.00015602	0.00084029	1.0151	4.6136
72	0.00085314	0.0000044	0.9505	-9.4481
73	0.00082757	0.00056498	0.9587	14.8084
74	0.0000066	0.00028761	1.0205	-0.5232
75	0.00045213	0.00089712	0.9606	16.254
76	0.00026312	0.00099152	1.0235	-0.4562
77	0.0003	0.0002	0.9989	-15.9269
78	0.00060173	0.00081968	0.9569	9.3835
79	0.00097288	0.00093161	0.9615	-9.6373
80	0.00030775	0.00026214	1.0172	-11.3539
81	0.00028197	0.0003666	0.9567	10.958
82	0.0001	0.0007	0.9709	-9.6128
83	0.00017501	0.00031678	0.9538	0.5662
84	0.00060463	0.00055344	0.9689	12.4227
85	0.00049479	0.00025939	0.9917	-17.9393
86	0.00074739	0.000061	0.9976	11.2733
87	0.00073692	0.00062359	1.0459	3.6541
88	0.0002	0	0.9665	7.1323
89	0.000081	0.00089968	1.0154	-0.9266
90	0.00051316	0.00093613	1.0007	-15.3443
91	0.000031	0.00072166	1.0017	18.7953
92	0.00062835	0.00047428	0.9977	12.0889
93	0.0000047	0.00047299	1.0034	11.0409
94	0.00031952	0.00085516	0.9855	1.9537
95	0.00028333	0.00050642	1.0368	18.9362
96	0.0008504	0.00096063	1.0821	-15.1242
97	0.00038635	0.00026579	1.0445	-5.8019
98	0.00069584	0.00012365	0.9702	-8.2354
99	0.0007	0.0009	0.9562	16.788
100	0.00010717	0.0000085	0.9727	-10.4169
101	0.0006727	0.00077553	0.9557	8.5754
102	0.00053227	0.00015283	1.0827	-10.9894
103	0.00044692	0.00081571	0.9729	0.6583
104	0.00016048	0.000073	1.0319	0.9266
105	0.000025	0.00047994	0.9625	-2.6931
106	0.000048	0.00079433	0.9284	-10.2436
107	0.00084716	0.00015576	0.9641	4.4674
108	0.00068356	0.000055	1.0552	1.2291
109	0.00026455	0.00064387	1.0413	11.6445
110	0.00058321	0.00015476	1.02897	9.3654
111	0.00071167	0.00048416	1.089	16.4406
112	0.00094005	0.00097359	0.9736	12.9073
113	0.00084364	0.00061624	1.0071	19.928
114	0.00021032	0.00017141	0.9618	3.9724

115	0.00084827	0.00097626	0.9523	0.8467
116	0.00083	0.000547	0.9523	4.5623
117	0.00047	0.0007656	1.0258	-15.256
118	0.000455	0.00072	0.96136	16.2351
119	0.00092	0.0009962	0.992564	-9.2541
120	0.00012	0.0004547	1.03654	0.06541
121	0.001	0.0008	0.9618	2.2392
122	0.00013234	0.00093924	0.9884	11.3193
123	0.00096448	0.0005687	0.9523	-7.3612
124	0.00081065	0.0000031	1.0828	-6.4949
125	0.00030709	0.000083	0.9998	-11.0119
126	0.000048	0.00071905	0.9618	3.1288
127	0.00096313	0.00098343	0.9757	-4.9987
128	0.00093397	0.00077209	1.0843	-0.3262
129	0.00062329	0.00064654	1.0182	12.9015
130	0.00094189	0.00037681	0.9539	-8.8013
131	0.00014215	0.00057406	0.9571	1.6474
132	0.00041524	0.0005684	0.9583	10.7085
133	0.00020296	0.00046611	0.9638	6.3202
134	0.00037946	0.00062747	0.9987	9.103
135	0.00095745	0.00082409	0.9785	-1.7189
136	0.00027374	0.00046965	0.9706	1.4742
137	0.00020975	0.00085845	0.971	7.2577
138	0.00039402	0.00048299	1.0127	7.7591
139	0.00035319	0.00036285	0.9632	5.5677
140	0.0006075	0.0009984	0.95154	6.3214
141	0.0008155	0.0002237	0.95214	-14.2365
142	0.00011	0.0009845	1.0564	0.98745
143	0.0002734	0.0002717	0.9654	3.2145
144	0.0001812	0.000567	0.9628	-17.149
145	0.0004911	0.0006331	0.9752	-4.2187
146	0.0005208	0.000486	0.9962	0.05871
147	0.0005119	0.000886	1.0547	2.0154
148	0.0002753	0.0004816	0.9614	13.2974
149	0.0004286	0.0004054	1.0893	1.5647
150	0.0002753	0.0001582	0.9512	6.2354
151	0.0007	0	0.9544	3.5375
152	0.00013698	0.00027886	1.007	-0.8344
153	0.00020208	0.00023032	0.9965	-0.5292
154	0.00040253	0.00020923	0.9701	5.2631
155	0.00046085	0.00096044	0.9731	5.815
156	0.00029987	0.000035	0.9753	4.4393
157	0.00080446	0.00053457	0.9485	-8.8882
158	0.00035469	0.00015957	0.9687	13.5985
159	0.0003943	0.000029	0.9585	-1.0752
160	0.00073661	0.00017162	0.9279	13.6502
161	0.00065884	0.00028923	0.9548	-6.1351
162	0.00035578	0.00029898	0.9587	-2.3739
163	0.00075604	0.00092751	0.9521	2.898
164	0.000035	0.00038695	1.0067	8.9077
165	0.00099806	0.00071947	1.0691	-10.2774
166	0.00012425	0.000052	0.955	-8.262

167	0.00092219	0.00019175	0.9984	13.4656
168	0.00076628	0.00081641	0.9775	17.2191
169	0.00092571	0.000057	0.973	0.6329
170	0.00054766	0.00069843	0.9653	0.4959
171	0.00099334	0.00079967	0.9611	10.6166
172	0.00049608	0.00022697	0.9946	-1.0459
173	0.00055849	0.00042155	0.9421	-4.9939
174	0.00048618	0.000055	1.0338	3.6292
175	0.00014611	0.00072444	0.9827	10.898
176	0.00051741	0.00064903	0.9628	6.5469
177	0.00071172	0.00036796	0.9539	-1.5042
178	0.00019378	0.000039	0.9512	6.5171
179	0.00060168	0.0008397	1.0102	-7.9275
180	0.00059232	0.00022898	0.9634	6.5271
181	0.0001	0.0002	1.0054	4.2552
182	0.00044569	0.000618	1.0125	-6.0294
183	0.00055934	0.00041766	0.9865	3.0486
184	0.00020306	0.00096151	0.9548	7.0504
185	0.00088243	0.00045767	0.9413	-0.5076
186	0.00069729	0.00070518	0.9528	9.3272
187	0.000044	0.000333	0.9537	4.4252
188	0.000013	0.00044238	0.9582	2.137
189	0.00050756	0.00076265	0.9543	4.8151
190	0.00066192	0.00075837	0.9592	19.2249
191	0.0002513	0.00044497	0.9583	0.8323
192	0.00020302	0.00080563	0.9555	2.266
193	0.00030057	0.00094464	1.0024	-17.265
194	0.00046703	0.00042967	0.9994	11.0973
195	0	0	0.9507	18.5179
196	0.00093542	0.000085	0.9738	10.7321
197	0.00051919	0.00062756	0.9848	1.4284
198	0.00013828	0.00084404	1.067	6.9025
199	0.00052945	0.0002323	1.027	-17.4657
200	0.00054335	0.00038057	0.9445	-15.5822
201	0.00061187	0.00075132	0.9957	-18.0073
202	0.00096005	0.00091358	0.9778	-2.2563
203	0.00079194	0.00022214	0.9552	-3.3525
204	0.00026439	0.00095589	0.9802	0.5317
205	0.00056456	0.00087312	0.9946	18.2447
206	0.00054948	0.00055895	0.9555	8.0271
207	0.00077144	0.001	0.9546	3.8581
208	0.00075889	0.00027959	1.0049	8.4842
209	0.00056002	0.00040952	1.0468	-13.798
210	0.0006285	0.00057178	0.9503	-5.8999
211	0.00019025	0.00095178	0.9528	-5.8964
212	0.00059117	0.00056099	0.9572	-0.4872
213	0.00037677	0.00016316	0.9714	17.869
214	0.00088221	0.00063269	0.9555	-11.1881
215	0.00065945	0.00093676	0.9865	-2.8314
216	0.00087711	0.00089287	0.988	-0.0197
217	0.00093717	0.00011334	0.9586	-9.8487

218	0.00048357	0.0005098	1.0667	16.3454
219	0.000074	0.00050823	0.9713	8.419
220	0.00023902	0.00097386	0.9738	8.5508
221	0.00020321	0.00085756	1.0454	15.2306
222	0.00011518	0.00095031	1.0223	6.236
223	0.0009373	0.00031901	0.9452	1.1682
224	0.00063283	0.00047536	0.99	-4.223
225	0.00019285	0.000067	0.9426	-12.6611
226	0.000078	0.00023635	0.9599	-3.5077
227	0.00054555	0.000043	0.963	5.5577
228	0.0007854	0.000023	0.9515	2.3654
229	0.00021656	0.00064519	1.0196	-10.1812
230	0.00045404	0.00078516	1.0707	-17.4119
231	0.00026644	0.00016696	0.9708	-3.5614
232	0.00032806	0.0001135	1.0165	-17.2337
233	0.0008132	0.00026348	1.0561	0.7447
234	0.00049081	0.00047598	0.9979	6.2127
235	0.00042958	0.00037536	0.9598	3.4219
236	0.00099515	0.00014891	1.0225	6.0696
237	0.00084	0.00054738	0.9921	-3.3138
238	0.00019869	0.0005315	1.0221	5.5529
239	0.00069827	0.00070673	0.9505	-0.6187
240	0.00073886	0.0006194	0.9452	14.327
241	0.000012	0.00047631	1.0173	-11.6858
242	0.00052499	0.000039	0.945	-2.4259
243	0.00044988	0.00058049	0.9533	16.2874
244	0.00033758	0.00014153	0.9937	6.6786
245	0.00068628	0.00084693	0.9597	-4.9778
246	0.0004596	0.00037358	0.9885	9.4525
247	0.00051515	0.00045483	0.9546	4.6063
248	0.00032144	0.00068358	1.0714	7.7469
249	0.00043791	0.00041416	0.9966	-0.7901
250	0.00028647	0.00022116	0.9556	-2.497
251	0.0007	0.0003	0.95315	1.1601
252	0.000095	0.00054734	0.95733	-8.8202
253	0.00079549	0.00077536	0.98058	-5.4324
254	0.00032922	0.00065235	1.0836	-17.168
255	0.000085	0.00037146	0.959	8.3101
256	0.00094778	0.000036	0.98041	-19.965
257	0.00078819	0.00080088	0.95607	-13.439
258	0.0001091	0.00023452	1.0162	-4.7631
259	0.0008952	0.00047521	0.9842	1.5236
260	0.0009	0.0003	0.9574	-2.2963
261	0.0004	0.0008	1.0025	-3.0548
262	0.00066294	0.000013	0.9891	-1.1725
263	0.00048644	0.0001791	0.94889	8.6261
264	0.00015727	0.00054346	0.95678	-18.469
265	0.00033058	0.00075487	0.9502	8.8319
266	0.00060066	0.00054603	1.0306	6.2556
267	0.00052942	0.00042094	1.0529	14.874

268	0.00088617	0.00069782	1.0911	-0.47424
269	0.0004741	0.00045141	1.0366	5.3008
270	0.00034368	0.00003	1.0076	-13.849
271	0.001	0.0005	1.052	5.453
272	0.00012802	0.00052728	1.0162	-0.30575
273	0.00092958	0.00022451	0.9701	-1.8693
274	0.000074	0.00096697	0.99916	-0.018933
275	0.00072001	0.00030789	0.99985	14.765
276	0.00063061	0.00031637	1.0273	3.0481
277	0.00095612	0.000019	1.0046	-7.9187
278	0.00068346	0.00018341	0.95131	3.5014
279	0.0004	0.0007	0.9827	-15.4074
280	0.000053	0.00026706	1.0474	-14.6568
281	0.00035039	0.00018333	1.0241	-4.0515
282	0.0004435	0.00023507	0.964	7.934
283	0.000093	0.00028498	0.9423	-3.4764
284	0.00049599	0.000075	1.0241	9.6195
285	0.001	0.0002	0.9153	14.1537
286	0.00087371	0.00037862	0.912	-7.0177
287	0.00036547	0.00042879	0.9512	-5.2367
288	0.0008	0.0009	1.0031	-2.86
289	0.00076513	0.00057713	0.9934	1.0543
290	0.0003	0	0.9345	1.6056
291	0.00072	0.00029	0.9136	-6.1
292	0.0001864	0.00039723	0.9789	-10.8992
293	0.00012961	0.00028	0.976	-9.4627
294	0.00082929	0.00050417	1.0373	13.1924
295	0.00021605	0.00030426	0.935	7.205
296	0.00087376	0.00048	0.9006	-16.6558
297	0.00044415	0.00067544	0.9385	11.6025
298	0.001	0.0006	0.962	17.4074
299	0.0002	0.0007	0.9108	4.4968
300	0.0006	0.00023838	0.9914	-7.3521
301	0.00021	0.00077513	0.9295	-3.7163
302	0.00060162	0.00080401	0.9947	11.3228
303	0.00083567	0.00077209	1.0395	5.0055
304	0.00031004	0.00057108	1.0016	9.8521
305	0.00034026	0.00077	0.9113	-19.9432
306	0.0005	0.0001	0.9574	9.9693
307	0.00082069	0.00072388	0.9175	-13.1628
308	0.00078933	0.00012857	0.9248	-2.9038
309	0.00047276	0.00056897	0.915	3.2155
310	0.00051363	0.00048865	0.9966	1.6473
311	0.0005	0.0005	0.9932	5.0575
312	0.00045	0.000501	0.9141	10.7649
313	0.0009075	0.00069747	0.9307	-9.4417
314	0.00059802	0.00083978	0.9472	-2.4195
315	0.00090878	0.00080899	0.9357	-11.4126
316	0.0004	0.0002	1.0021	12.7353
317	0.00079075	0.00051686	0.9498	14.4289
318	0.00061	0.00095154	0.9573	9.1572

319	0.00033893	0.00020428	0.9459	18.9635
320	0.00073845	0.00032157	0.9683	-11.3016
321	0.0008	0.0009	0.911	4.4709
322	0.00089603	0.00069157	0.9326	1.409
323	0.00062809	0.00088	1.0397	15.4222
324	0.00075763	0.00013265	1.0234	7.1817
325	0.00034566	0.00027051	0.9664	-8.6569
326	0.00068725	0.00094826	1.0478	-0.6129
327	0.00098619	0.00011221	0.9242	10.1916
328	0.00078934	0.00017901	0.9116	-14.6921
329	0.00053819	0.00015669	1.0548	16.6016
330	0.00078625	0.00091999	1.0757	12.2295
331	0.0004	0.0002	0.9059	-8.4352
332	0.00068151	0.00040504	0.9118	-6.1132
333	0.00030002	0.00088672	0.9854	-18.1334
334	0.00031717	0.00071215	1.0076	-10.6794
335	0.00078608	0.00072425	0.9117	5.1871
336	0.00057	0.00029	0.9127	-3.4872
337	0.00035894	0.00064906	0.9453	0.4137
338	0.00054271	0.00040283	0.9783	19.1633
339	0.0004782	0.00019485	0.9615	7.4683
340	0.00019572	0.00018078	1.0245	8.2345
341	0.0002	0.0001	0.9202	15.3665
342	0.00071403	0.00057652	0.9462	5.148
343	0.00041883	0.00016	0.9531	8.1265
344	0.00050634	0.00018	1.0171	-5.4776
345	0.00083654	0.00070874	0.9417	3.9911
346	0.00045	0.00046	1.0612	4.6101
347	0.00068	0.0005833	0.954	4.7473
348	0.00070461	0.00033	1.0019	14.4764
349	0.00015384	0.00074427	0.9648	6.3369
350	0.00057962	0.00013103	0.9975	4.7811
351	0.0009	0.0005	0.9572	17.3267
352	0.00071164	0.00015677	0.9489	-7.1507
353	0.0007123	0.00071494	0.9817	-10.7106
354	0.00074257	0.00035	0.9236	-3.1823
355	0.00083908	0.00066149	0.9129	-2.6764
356	0.00084867	0.00028569	1.019	-3.3266
357	0.00030526	0.00062	0.9299	-12.2346
358	0.00073194	0.00088	0.9218	-12.8536
359	0.00013875	0.00012092	0.9438	-9.7586
360	0.00048704	0.00048171	0.9875	1.5297
361	0.00062871	0.00068218	1.023	-5.9336
362	0.00059909	0.00069565	0.9912	-1.5367