Solving transcendental equations using Genetic Algorithms

By :Varun Aggarwal
104/ECE/2000
NSIT, Delhi

I. Analytical/Iterative methods to solve transcendental equations

Transcendental equations are equations containing trigonometric, algebraic, exponential, logarithmic, etc. terms. Many analytical/iterative methods are used to solve transcendental equations. Though these methods are capable of solving many transcendental equations they suffer from many common disadvantages. Usually transcendental equations have many solutions in a given range, and analytical methods are not able to find all these roots in a given interval, even when they find several solutions, it is not possible to conclude that the given method has found the complete set of roots/solutions, and has not missed any particular solution. Also, these methods fail in case of misbehaved or discontinuous functions. Hence, though these methods may work very well in some situations, they are not general in nature and need a lot of homework from the Analyst.

An analysis of some common methods used for solving transcendental equations, their disadvantages and cases of failures are discussed below.

a) Newton Raphson method: This is a commonly used method for solving transcendental equations. The method makes use of the slope of the curve at different points. Therefore, if the function is non differentiable at points or has a point of inflexion, the method is not able to find the root. Secondly, if the function changes its slope very quickly (frequently achieves slope of zero), or is discontinuous, cannot be solved by this method. If the function is discrete, the derivative has no meaning for it and this method cannot be used. Also there is no straightforward way to find all the roots in an interval or even ascertain the number of roots in the interval.

b) Bisection method: This method needs two points on the graph such that $f(a)*f(b)<0$. There is no straightforward analytical method to find these points. Another problem lies in choosing the distance between the points a and b. For the method to work, a and b should be close enough, such that the function behaves monotonously in these limits. At the same time, a small difference in values of a and b makes it difficult to search the sample space. An algorithm to ascertain such points for all roots of the equation has to be essentially random in nature and can be another applications of GA. This will be discussed later. Further still, the method fails for discontinuities in function.

c) Method of False Position: This method suffers from same problems as Bisection method.

Hence it can be concluded that analytical methods cannot find all the roots of a transcendental equation reliably.

II. Introduction to Genetic Algorithms

Genetic Algorithms (GAs) were first presented by J. H. Holland in his book "Natural and Artificial Systems" in the year 1975 and developed further by his students. With time, many changes and improvements have been suggested. Here, we discuss the present form of implementation of Genetic Algorithms.

Genetic algorithms are a class of algorithms inspired by evolution. These algorithms encode solutions to a specific problem on a simple chromosome like data structure and apply recombination operators to these structures so as to preserve critical information.

An implementation of a genetic algorithm begins with a population of (typically random) chromosomes. Then these structures are evaluated and allocated reproductive

opportunities in such a way that those chromosomes which represent a better solution to the problem are given more chances to reproduce than those chromosomes which are poorer solutions. The "goodness" of a solution is typically defined with respect to the current population.

The basic steps involved in a GA are the following.

1. Build an initial population of samples (solutions) created randomly or using some initialization method.
2. Calculate the fitness (measure of being provided reproductive opportunities) of all the samples and select individuals for the reproduction process. The selection of the individual is though based on fitness, but it is a probabilistic mechanism. Roulette wheel selection, Rank Tournament Selection, Stochastic Universal Selection are some of the selections used.
3. Apply the genetic operators of crossover, mutations, inversions, etc. to the selected individuals to create new individuals and thus a new generation. Crossover exchanges some of the bits of the two chromosomes and mutation inverts any bit(s) of the chromosome depending on a probability. Crossovers is a distinguishing feature of Genetic Algorithms. Many evolutionary algorithms were earlier used, but they basically worked on mutations and no crossovers took place. In GAs crossovers 'explore' around the already found good solutions and mutations help 'exploiting' the search space for new solutions.

Then again Step2 is followed till the condition for ending the algorithm is reached. Many issues such as encoding of problem, the types of selection operator, the type of fitness functions have been discussed and experimented at large by researchers to reach better results.

III. GAs for the present problem

GAs have been traditionally used for optimization problems. Function optimization implemented by GAs was studied by De Jong [1] in detail and GAs were successful to solve the problem. Solving transcendental equations is also a kind of optimization problem and hence GAs are applicable here.

Also, Melanie Mitchell [2] states, if the space to be searched is large, is known not to be perfectly smooth and unimodal or is not well understood or if the fitness function is noisy and if the task doesn't require a global optimum to be found i.e. quickly finding a sufficient good solution is enough, GAs will have a good chance to be competitive with or other surpassing other weak methods. The present problem has a large search space, a fitness function which might be misbehaved and has more than one solution. All these difficulties indicate that genetic algorithms may be useful in such situations and I will address this problem in this paper.

IV. Implementation of GAs for the present problem

Encoding: There is no strict rule for encoding the solutions to the problem. Generally binary coded solutions are used, though lately, real coded chromosomes are also being used. I have used binary encoding for my experiments. The advantage of using binary encoding is that the maximum number of schemas is investigated [3].

Fitness function: The fitness function tells the algorithm how good a particular solution is. The fitness function chosen for the problem was 1/ (1 + abs(f(x)). The relevance of the fitness function is discussed later.

Selection and reproduction operators: As the present problem is multimodal, the selection and reproduction operator depends on it. These will be discussed later.

The method is kept free from differentiation or any other analytical methods to tackle non-differentiable and discontinuous functions.

V. GAs to tackle Multimodal problems

It is observed that the problem at hand is multimodal in nature i.e. one equation can have many roots. The simple genetic algorithm is capable of searching optimum for a unimodal function, but converges to some local optimum for a multimodal problem. Actually the population drifts towards that optimum whose neighbouring points were seeded in the initial population (typically, generated randomly). It happens so, that the samples neighbouring one of the optimum get more reproductive opportunities. Therefore, there is a better exploitation of this region, which increases the average fitness of these samples, thus giving them more reproductive opportunities every time. Finally these samples dominated the search space and the population converged to that particular solution.

Various methods were devised to solve this multimodal problem, starting from De Jong (the crowding model), followed by Perry (1984), etc. But the study of Goldberg and Richardson on niches (sub domains of a function) and species (stable sub-populations) based on the sharing principle [4] preformed quite well. The paper discusses the sharing methods and analyses a relatively new method, "Clearing Procedure as a niching method" given by A. Petrowski [5].

The sharing function devised by Goldberg and Richardson was the first organized approach for multimodal problems. The sharing method was based on the principle that resources should be shared in all different species. A sharing function was defined which reduced the fitness of a sample depending upon its distance of individuals in the population.

A sharing function can be defined as following

$$S_{ij}(d) = \begin{cases} 1 - (d_{ij} / \sigma)^n & \text{for } d < \sigma \\ 0 & \text{for } d > \sigma \end{cases}$$

where d_{ij} = distance and σ = threshold value of distance.

The fitness of a particular sample is changed as its fitness divided by sum of the values of its sharing with all other samples in the population. This way the fitness of samples, which had dominance in the population, was decreased and thus there reproductive capabilities were limited. Goldberg also observed that mutation was necessary in this procedure for good exploration of the search space.

The method had some limitations. The value of σ was dependent on the problem at hand. The method asked for larger size of subpopulation, otherwise it converged prematurely.

More recently, a clearing procedure was devised by Alain Petrowski [5]. This method allocates the resources of a niche (sub domain of a function) to the best individual of the species rather than sharing it among all the samples of the particular niche as done in sharing. This method worked more successfully than the sharing procedure even with a smaller population.

Implementation of the Clearing Procedure: Alain Petrowski gave the following steps for implementing the clearing procedure. Starting with the fittest sample in the population and equate the fitness of all samples having distance from the sample less than the clearing radius (equivalent to threshold radius) to zero. Then, take the next fittest sample (whose fitness is non-zero) and repeat the clearing procedure with it. In this way, only the fittest samples of each niche are provided with reproductive capabilities. Stochastic Universal Sampling (SUS) was recommended for selection to reduce ‘selection noise’. For the reproductive capabilities, crossover probability was kept equal to 1 and probability for mutation was kept low.

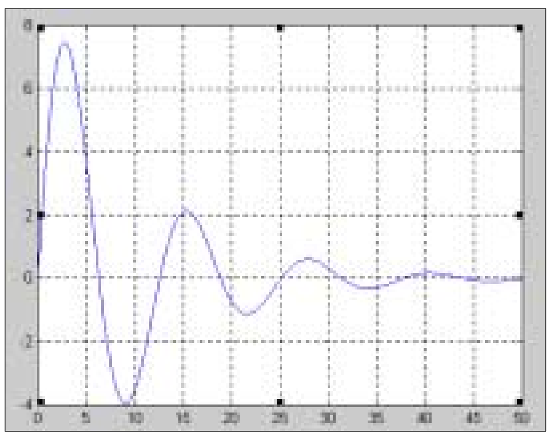
Petrowski also suggested an elitist strategy in the procedure. The elitist strategy used initially by De Jong [1], kept record of the fittest individual of a population. If the new population generated didn’t contain that individual, it was included in the new population incrementing its size by 1. De Jong observed that the strategy worked well with unimodal search space, but degraded the performance of the GA considerably in multimodal search space. Petrowski implemented the elitist strategy with every sub-population separately. This modification improved the clearing strategy significantly.

The working of the clearing procedure is dependent on the value of the clearing radius, which is not known in our problem. Experiments (detailed in the following discussion) were carried out and a strategy was evolved to solve this issue.

Experiments with the clearing procedure:

Coding: Binary Coding was used.
Fitness function: The fitness function used was 1/(1+ abs(f(x)). I earlier tried using 1/f(x), but that led to premature convergence. The fitness of a sample became extremely high if its function value was less than 1, and the sample dominated the whole search space.
Selection operator: Stochastic Universal Selection (SUS) was used. In this process on a wheel, all the samples are placed allotting them space propotional to their fitness. N markers are put around the wheel, where N is the size of the new population. The wheel is spun and samples under each marker are selected.
Clearing Radius: 2
Capacity of niche: 1
Reproduction Operators: Crossover (one point crossover) probability was set to 1 and mutation probability was set to 0.05. One point crossover was used
Distance: Distance was defined as the difference between the decimal equivalents of the samples.
Number of samples considered: 20

Function under test:
$10*e^{(-0.1x)} * \sin(0.5x)$ (Consists of eight solutions in the range 0 to 50)

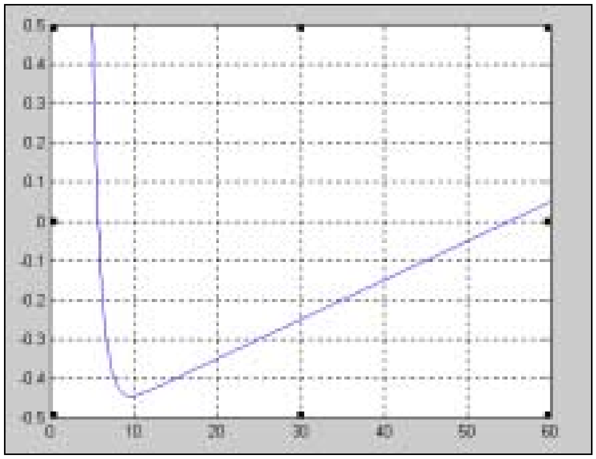


Number of Generations	No of roots found	No of redundant populations	Population size	Accuracy of worst root
10	7	4	80+	App 0.14
20	8	6	150+	App 0.02
30	8	7	240+	App. 0.015

In the earlier generations, the method was able to recognize many niches. But the function under test is deceptive. In the range 40 to 50, though the samples have a very high fitness, only two solution exists. Hence in the later generations, the samples in the region of 40 to 50 start dominating, killing even some of the good subpopulations.
Filtration: The population contained many samples with high fitness but which were not roots. These individuals were present mostly in the range of 40 to 50. It was tough to

filter out roots from such individuals. A scheme utilizing the iterative method is suggested later to counter this problem.

Function under test:
$e^{(5-x)} + x/100 - 0.55$ (The function has 2 roots in the range 0 to 60)



Number of Generations	No of roots found	No of redundant populations	Population size	Accuracy of worst root
10	2	15	120+	App 0.04
20	2	19	250+	App 0.03
30	2	21	400+	App. 0.02

Filtration: Though the presence of a lot of unfit populations made the filtration process lengthy, it was found that the subpopulations containing the 2 roots were the fittest in all the subpopulations and were easily found.

With the clearing radius equal to 2, many unstable populations were observed. Due to the elitist approach, the population size kept on increasing to reach very high value. The population contained large number of subpopulations of nearly the same fitness due to the smooth curve of the function. The same technique worked considerably well with a niche size of 10.

Improvement in Technique
Many different techniques are suggested for speciation (to recognize different subpopulations) in a given population. I devised a new speciation operator to work together with the clearing procedure.
For a sub population to be a fit species, the fitness of the samples on both negative and positive side of the fittest sample should decrease. In the clearing algorithm, we recognize the fittest sample and then build the subpopulation around it. In the new technique, if it is found that the fittest sample in a subpopulation is situated at the edge of the subpopulation, then it is rejected. It is tested whether all the members of a subpopulation are present on one side of the fittest sample. If it is found so, it is rejected i.e the fitness of the best sample of the population is also set to 0.

Some precautions were taken before implementing this procedure. The population was allowed to mature up, before this operator was used (In the present experiment, it was not applied for the first 5 generations). A minimum number of samples of the subpopulation should be available to reject it (set to 7 in our experiments).
The MATLAB code for the new technique together with the clearing procedure is given underneath.
Size: Size of population; *distance(i,j):* distance between the i^{th} and j^{th} sample
σ_c : Clearing Radius; *min_gen*: Minimum number of generations, after which rejection is starts; *min_num_samples*: The minimum number of samples a subpopulation should have before rejection.

```
for i=1:size
   if (fitness(i)>0)
      nbwinners=1;
      count=0;
      same=1;
      for j=i+1: size
         if(fitness(j) > 0 & distance(i,j)< σc )
            if (nbwinners < capacity)
               nbwinner= nbwinner + 1;
            else
               fitness(j) = 0;
               count=count+1;
               sign=0;
               if(distance(i,j)<0)
                  sign=1;
               end
               if(count>1 & same==1)
                  if (sign_prev==sign)
                  else
                     same=0;
                  end
               end
               sign_prev=sign;
            end
         end
      end
      if(n > min_gen)
         if(same==1 & count>= min_num_samples)
                  fitness(i)=0;
         end
      end
   end
end
```

Another third kind of algorithm was also studies. This algorithm followed the same procedure as earlier but had an additional feature of population control. The size of each population was fixed according to the number of subpopulations found (population size =number of subpopulations *6). In the second algorithm suggested, even after rejection of many subpopulations, the total size keeps on increasing, leading to lot of useless number krunching, this algorithm takes care of that.

Experiments with the New Techniques

Function under test:
$10*e^{(-0.1x)} * \sin(0.5x)$ (Consists of eight solutions in the range 0 to 50)

Type of Algorithm	Number of Generations	No of roots found	No of redundant populations	Population size	Accuracy of worst root
1st kind	10	7	4	80+	App 0.14
2nd kind	10	6	4	75+	App 0.2
3rd kind	10	6	6	90+	App. 0.19
1st kind	20	8	6	150+	App 0.02
2nd kind	20	6	3	75+	App 0.02
3rd kind	20	7	3	70+	App. 0.1
1st kind	30	8	7	240+	App. 0.015
2nd kind	30	6	6	200+	App 0.014
3rd kind	30	7	5	90+	App. 0.018

In the above function, the two modifications have little meaning because the niche size is appropriate and therefore there are not many unnecessary subpopulations and the population size remains under control.

In this case, 2nd kind of algorithm shows worst results. On observing the different generations, I came to know that the problem lies with the reproduction operators. I observed that there were many repeated samples in each generation. This fictitiously increased the number of samples in each subpopulation, which may led to its elimination (due to repeated samples, it may seem redundant). A precautionary measure for the same will have to be implemented. The root zero cannot be found by this method as it lies at the edge of the interval, that is why in even best cases the method finds only 7 roots. But the second method was able to considerably reduce the bias of the samples towards the later half of the search space (which has only one root but higher average fitness from the rest of the search space.)

In the third algorithm, a good restriction was imposed on the population size and hence it could be seen that the population remained constant over the generations reducing the time for number krunching. But, worst accuracy becomes poorer because some roots are removed (again due to fictitous increase in their number) and these roots are randomly placed again in later generations.

Function under test:

$e^{(5-x)} + x/100 - 0.55$ (The function has 2 roots in the range 0 to 60)
Results with 1st kind of algorithm. Keeping niche size as 2 (same as before. Inappropriate)

Type of Algorithm	Number of Generations	No of roots found	No of redundant populations	Population size	Accuracy of worst root
1st kind	10	2	15	120+	App 0.04
2nd kind	10	2	13	120+	App 0.02
3rd kind	10	2	12	130+	App. 0.03
1st kind	20	2	19	250+	App 0.03
2nd kind	20	2	14	200+	App 0.01
3rd kind	20	2	11	120+	App. 0.03
1st kind	30	2	21	400+	App. 0.02
2nd kind	30	2	14	220+	App 0.01
3rd kind	30	2	11	100+	App. 0.02

With the first kind of algorithm, it is observed that the population size is increasing indiscriminately. Also the number of redundant subpopulations is increasing. There is a lot of useless computation for redundant subpopulations.

Results with 2nd kind of algorithm: It can be observed from the table that the number of redundant subpopulations remained constant over the generations. Population size came under control.

Using the third kind of algorithm: Here one observes that the population has come to a fixed value too. The number of redundant populations has also got fixed. That makes the algorithm very efficient. But, the problem is same as before: that there are repeated samples, which fictitiously increases the subpopulation size and hence the subpopulation gets destroyed. Once this thing is dealt with, the method will become more robust.

This way, this technique has made Clearing Technique independent of niche size to a very large extent.

Disadvantages: The method stated above suffers from the following disadvantages.

1. To find the roots at the edge of the interval is impossible using this technique. Roots near the edge are also difficult to find. In the above experiment, it is observed that the method surpasses this problem many a times.
2. The method will not find out solutions, if the function is discontinuous in the neighborhood of the fittest sample. A probable solution to the problem is to consider the fitness of the sample at a discontinuity to be 0.

Further improvements: In the above technique, it has been suggested how to get rid of unnecessary subpopulations. It is also possible that the there are two niches present at a distance less than the clearing radius. The following idea can be used to recognize such subpopulations. Inspect the samples of the subpopulation one by one on positive and negative side of the fittest sample in ascending order of distance. Initially the fitness will decrease. If the fitness is found to increase at a particular sample, the following samples should be considered as a different fit subpopulation. The new subpopulation formed should be again investigated in the same way, to recognize for more stable subpopulations.

Disadvantage: The time complexity of the process will be large. In presence of niches at distances less than the clearing radius, they will have to be searched for again and again in each generation.

VI. A Complete Model

It is suggested to divide the infinite search space into intervals, which are successively investigated using genetic algorithms. The issue is to decide the order of investigation of the intervals. One idea is to start from the mid interval, i.e. the interval containing the y-axis. Then alternatively intervals on the left hand and the right hand side should be investigated. It is so advised that the choice of doing the search between the right hand or left hand side of the graph can be inspired by the results of the investigation of the preceding intervals on that side. Though all intervals should be checked, their order can be selected in the above-mentioned way. It is also advised to consider slightly overlapping intervals, for lowering the discrimination of the samples at the edge of an interval.

We have to also determine whether an interval contains no roots so that its investigation can be stopped. To do so, we should check that after some minimum number of generations (preferably 10, on basis of the above experiments), if the best sample of the population has a fitness lower than a given value (preferably 0.990), then the interval should be rejected. As we are following the elitist strategy, the probability of loss of any good interval is extremely low.

Another innovative idea is to decide upon the intervals using GAs. A large interval can be considered for the application of the GA. The clearing radius can be set equal to half the interval size, in which we will finally apply the GA. The clearing procedure should be then applied, using the New Selection Operator [6], so that no subpopulation dies. This way we can get an order in which we can investigate the interval for finding solutions. This is just an intuitive method and needs to be studied/experimented further.

GA combined with Iterative methods

While analyzing the application of GAs to the present problem, it was observed, that, though GAs are able to search for all the available roots in a interval, their accuracy is lower than the accuracy of the roots found by iterative methods. Also the issue of filtration of roots from a population was not fully resolved.

Therefore it is suggested to use GAs combined with iterative method. In a stable subpopulation, we know that the fitness first increases, becomes maximum and then decreases. This is same as the function value changing its sign in the interval (over which the subpopulation exists). Therefore bisection method can be used to find out the root to the maximum accuracy. For using this method, the size of the interval has to be noted down in the last generation. This will give us values for a and b such that (f(a)*f(b) < 0). By using this method, roots can be found with better accuracy and filtering problem will be solved. Newton-Raphson method shouldn't be used, because it employs differentiation and will not work for discrete functions and also fails on encountering non-differentiable points in the function.

Disadvantages:

1. The method will not work if there is a discontinuity in the function near the root of the equation.
2. If the equation has a repeated root, this method will fail.

VII. Conclusion

It is concluded that GAs can be helpful in finding solutions for transcendental equations. Experiments were conducted using clearing procedure and a modification of the clearing procedure to find roots of transcendental equations in a given interval. They worked efficiently and were able to find all roots of the given transcendental equation. Other methods such as the Island Model [7], the Sequential Niche Technique [8], etc. can also be tried to solve the problem.

A scheme is suggested to break the search space into intervals that can be tested by GAs. Another scheme using GAs to determine the order of investigation of intervals is also suggested for further studies. A hybrid algorithm, using both genetic algorithm and iterative method, is also suggested for better accuracy at the loss of generalization. The schemes developed above can be also implemented on parallel machines by allotting different intervals to different machines.

Acknowledgement

I thank the following people for giving me suggestions and answering my queries through email.

1. Alain Petrowski, National Institute of Telecommunication, France
2. Roger Wainwright, University of Tulsa, USA
3. David Andre, University of California, Berkley, USA
4. Alife Rennard, GA expert
5. Martin Butz, ILLIGAL, University of Illinois, Urbana, USA
6. Everett Carter, California, USA

Bibliography

1. De Jong, “An Analysis of the Behavior of a Class of Genetic Adaptive Systems”, 1975
2. Melanie Mitchell “An Introduction to Genetic Algorithms” (text), 1996
3. David E. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning” (text), 1989
4. David E. Goldberg, J. Richardson, “Genetic Algorithms with sharing for multimodal Optimization”, 1987
5. Alain Petrowski, “A Clearing Procedure as a Niching Method for Genetic Algorithms”, 1996
6. Alain Petrowski, “A New Selection Operator Dedicated to Speciation”, 1997
7. M. Bessaou, A. Petrowski, P. Siarry, “Island Model Combining with Speciation for Multimodal Optimization”,
8. D. Beasly, D. R. Bull, R. R. Martin, “A sequential Niche technique for multimodal function optimization”, 1993