“APPLICATION OF GENETIC ALGORITHM FOR LOSS REDUCTION IN DISTRIBUTION SYSTEMS”

SUBMITTED IN THE PARTIAL FULFILLMENT FOR THE DEGREE OF

MASTER OF ENGINEERING

(POWER APPARATUS AND SYSTEM)

(2002-2005)

SUBMITED BY

SATVIR SINGH DESWAL (03/PAS/2002)

(UNIVESITY ROLL NO: 4192)

UNDER THE GUIDENCE OF

Dr.N.K.JAIN(Astt. Proff.)

ELECTRICAL ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING

DELHI COLLEGE OF ENGINEERING

UNIVERSITY OF DELHI

BAWANA ROAD, NEW DELHI

CERTIFICATE

This is to certify that this dissertation titled "**Application of Genetic Algorithm for loss reduction in Distribution Systems** "being submitted by **SATVIR SINGH DESWAL (03/PAS/2002)** of DELHI College of Engineering in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering is a bonafide work carried out under our guidance and supervision.

Dr.N.K.JAIN
Assistant Professor
Electrical Engineering
Deptt.
Delhi Collage of
Engineering

ACKNOWLEDGEMENTS

I am highly indebted and express my deep sense of gratitude to my worthy and reverend guide **Dr.N.K.JAIN , Assistant Professor**, Deptt.of Electrical Engineering , Delhi College of Engineering, New Delhi for this valuable guidance and help extended by this worthiness, which have enabled me to carried out the work successfully. His cooperation came forth to help me out of my difficulties before it was even called for by me. His precious suggestion and devotion have encourage me all through to the advantage of making my carrier bright.

I shall be failing in my duty if I do not appreciate and applauded the services of and the time devoted by honor'able **Prof.Parmood Kumar**, Head of The Deptt. of Electrical Engineering. Without the blessings, good wishes and kind help, this project would not have been accomplished.

My family had a big roll in ensuring that I successfully complete the project. Their constant encouragement and unwavering support have been a great source of strength for me. I am grateful to my friends for their valuable help and good wishes which went a long way in fulfilling this task.

Satvir Singh Deswal

(Er.No.03/PAS/2002 , Univ. R.No.4192)

Department of Electrical Engineering
Delhi College of Engineering ,New
Delhi

CONTENTS

APPENDIX :

- ANEXTURE-A(Transmission & Distribution losses of various countries)
- ANEXTURE-B(Transmission & Distribution losses of various states of India)
- BLOCK DIAGRAM-GENERAL
- BLOCK DIAGRAM-SELECTION STRATEGY
- BLOCK DIAGRAM-CROSS OVER STRATEGY
- BLOCK DIAGRAM-MUTATION STRATEGY
- BLOCK DIAGRAM-OPTIMIZATION STRATEGY
- BLOCK DIAGRAM SHOWING COMPLETE PROCEDURE FOLLOWED IN GA.
- PROGRAM LISTING

ABSTRUCT

The idea of applying the biological principle of natural evolution to artificial systems, introduced more than three decades ago, has seen impressive growth in the past few years. Evolutionary algorithms have been successfully applied to numerous problems from different domains, including optimization, automatic programming, machine learning, economics, operations research, ecology, population genetics, studies of evolution and learning, and social systems . In this study we will only consider genetic algorithms.

As its name suggests, a Genetic Algorithm (GA) is a biologically inspired search heuristic which produces a *population* of random solutions (called *chromosomes*) to a given problem and iteratively applies genetic operators on this population to evolve better and better solutions over successive *generations*. GAs are probabilistic searching methods which use implicitly parallel directed random exploration of the search space to produce near-optimum solutions over time.

One of the greatest attributes of GAs is that they are capable of "learning" – that is, they modify future solutions based on the successes and failures of past solutions.

Also, they are capable of adapting to changes over time. Therefore, GAs are considered to be in the realm of Artificial Intelligence. While a GA may never produce the absolute optimum solution, it is mathematically likely to get very close using a fraction of the computational requirements of an exhaustive deterministic search.

The distribution system is considered not only as one of the important part of the electric power system but one of the most complicated systems created by the mankind. It constitutes the link between electricity utilities and consumers. Usually, it suffers from unbalanced feeder structures and unbalanced loading which affects system power quality and electricity price.

This presentation introduces a genetic based algorithm (G.A) to determine the states of the switches for minimum loss configuration . The problem of feeder configuration can be looked upon as an optimization problem, where the objective function reflects the different goals that the individual utilities may pursue.

The algorithm can be directed to minimize the losses which are a major sign of better power quality. Also, the operator has the ability to direct it to minimize the active power loss. A radial distribution system is used to demonstrate the capability of the proposed G.A along with load flow studies.

CHAPTER-1

INTRODUCTION

1.1 Introduction & Engineering applications of Optimization

The ever increasing demand to lower the production cost to withstand competition has prompted to look for rigorous methods of decision making, such as optimization methods, to design and produce products both economically and efficiently. Optimization techniques, having reached a degree of maturity over the past several years are being used in wide spectrum in industries. With rapidly advancing computer technology, computers are becoming more powerful and correspondingly, the size of complexity of problems being solved using optimization techniques is also increasing. Optimization methods coupled with modern tools of computer – aided design are also being used to enhance the creative process of conceptual and detailed design of system.

Various techniques are used to speed up the convergence of optimization problems. In this dissertation, an optimization technique, generic algorithm which can perform dynamical and possess adaptive features has been presented. The detail of the same has been discussed in the next chapters and later and the numerical calculations have been included in last chapter. Also the results are completed with other techniques.

Optimization in its broadest sense is applied to solve any engineering problem. Some typical applications are given below:

1 Design of aircraft and aerospace structures for minimum weight
2 Finding the optimal trajectories of space vehicles.
3 Design of civil engineering structures frames, foundations, bridges, towers, chimneys and dams for minimum cost.
4 Minimum-weight design for structures for earth quake, wind and other types of random loading.
5 Design of water resource systems for maximum benefits.
6 Data compression and virtual channel enhancement.
7 Optimum design of linkages, cams, gears, machine tools and other mechanical components.
8 Selection of machining conditions in metal-cutting processes for minimum production cost.
9 Design of material handling equipment such as conveyers, trucks and cranes for minimum cost.
10 Design of pumps, turbines and heat transfer equipments for maximum efficiency.
11 Optimum design of electrical machinery such as motor, generator and transformers.
12 Optimum design of electrical works.

13 Shortest route taken by sales person visiting various cities during one route.

14 Optimal production planning, controlling and scheduling.

15 File allocation in distributed systems.

16 Energy conservation.

17 Design of transporter networking.

18 Path routing

19 Planning of maintenance and replacement of equipment to reduce operating cost.

20 Inventory control.

21 Robot path allocation.

22 Genetic algorithm is used to search for a number of hidden layers are neutral network solution and to design a starting set of weights to the networks.

23 Planning the best strategy to obtain maximum profit in the presence of a competitor.

24 Optimization of membership functions of fuzzy logic control.

1.2. METHODS OF OPTIMIZATIONS

OPTIMAZATION

Mathematical Programming Techniques

1. Calculus methods
2. Calculus of variations
3. Non-linear programming
4. Geometric programming
5. Quadratic programming
6. Linear programming
analyses)
7. Dynamic programming
8. Integer programming
9. Stochastic programming
10. Separable programming
11. Multiobjective programming
12. Network methods
    CPM & PERT
13. Game theory
14. Simulated annealing
15. Genetic algorithm
16. Neural Network

Stochastic Process Techniques

1. Statistical decision Theory
2. Markov Processes
3. Queuing theory
4. Renewal theory
5. Simulation methods
6. Reliability theory
7. Simulated annealing
8. Genetic Algorithm
9. Neural Network

Statistical Methods

1. Regression analyses
2. Cluster Analyses
3. Pattern Recognition
4. Design of Experiment
5. Discriminate Analyses
(Factor

- The mathematical programming techniques are useful in finding the minimum of a function of several variables under a prescribed set of constraints.
- The stochastic process techniques are used to analyze problems which are described by a set of random variable having known probability distribution.
- The statistical methods enable one to analyze the experimental data and build empirical models to obtain the most accurate representation of the physical situation.

Several factors are considered in deciding a particular method to solve a given optimization problem as:

(1) The type of problem to be solved.

(2) The availability of ready made computer program.

(3) The calendar time required for the development.

(4) The accuracy of the solution.

(5) The available knowledge of the efficiency of the method.

(6) The programming language and the quality of coding desired.

(7) The ease with which the program is used and its output is interpreted.

1.3 Comparison of Genetic Algorithm with other Techniques

The GA differs substantially from more traditional search and optimization methods. The four most significant differences are:

- Gas searches a population of points in parallel, not a single point.
- Gas use probabilistic transition rules, not deterministic ones.
- Gas works on an encoding of the parameter set rather than the parameter itself (except where real-valued individuals are used)
- GA's do not require derivative knowledge, only objective function & corresponding fitness, levels influence the direction of search

It is important to note that GA can provide a number of potential to a given problem and the choice of the final solution is left to the user, in cases where a particular problem does not have a unique solution, for e.g. in multi objective optimization where the result is usually a family of Pareto-optimal solutions. The GA is potentially useful for identifying these alternative solutions simultaneously.

1.4 Application Areas of GAs

When Would You Use a Genetic Algorithm?

GAs are not guaranteed to find the global optimum solution to a problem, but they are generally good at finding "acceptably good" solutions to problems in "acceptably quickly". Where specialized techniques exist for solving particular problems, they are likely to out-perform GAs in both speed and accuracy of the final result, so there is no black magic in evolutionary computation. Therefore GAs should be used when there is no other known efficient problem solving strategy.

Applications

Genetic algorithms are used in solving problems in the areas of cellular automata, fuzzy logic, image registration , communications network configuration , simulation modeling and optimization , time-tabling , multiobjective workforce scheduling , time constraint scheduling of limited resources , and combinatorial optimization. The most widely studied combinatorial task is traveling salesman problem. Bin packing problems are also widely studied . They have been utilized in playing games such as SimCity, SimEarth; in biology, chemistry and medicine; circuitry design and computer engineering; network routing for the telephone company; to detect computer viruses; for military artificial intelligence applications; military guidance and deciphering applications; art and music. GAs have been shown to be able to out-perform conventional optimization techniques of difficult, discontinuous, multimodal, noisy functions.

1.5 ADVANMTAGE OF Genetic Algorithm

Genetic algorithm works according to the principles of natural genetics on a population of string structures representing the problems variables. Three operators reproduction, crossover and mutation – are used to create new and hopefully better populations. The basic differences of GA's with most of the traditional optimization methods are that GA's use a coding of variables instead of variables directly, a population of points instead of a single point and stochastic operators instead of deterministic operators. All these features make GA – search robust, allowing them to be applied to a wide variety of problems.

GA is powerful and versatile search and optimaization method applicable to a broad range of activities.

GA is the global optimization search method. It requires a little knowledge of mathematics i.e. it is single search method.

1.6 DISSERTATION ORGANISATION

The material of this dissertation has been arranged in six chapters, references. The contents of chapters are briefly outlined as indicated below:

Chapter-1 provides the introduction of optimization and advantages of genetic algorithm.

Chapter-2 Gives introduction about history of optimization techniques, biological background ,Covers the concept of genetic algorithm , principle of working and brief survey of previous work done on genetic algorithm.

Chapter-3 Brief introduction of Power losses in Transmission and Distribution & various strategies concerned to the power losses.

Chapter-4 The application of Genetic Algorithm(along with load flow studies) for reduction of losses in Distribution Systems.

- **Future Scope**

CHAPTER-2

Literature Review – Genetic Algorithms

2.1. Introduction

Genetic Algorithms are nondeterministic stochastic search/optimization methods that utilize the theories of evolution and natural selection to solve a problem within a complex solution space. They are computer-based problem solving systems which use computational models of some of the known mechanisms in evolution as key elements in their design and implementation. They are a member of a wider population of algorithm, Evolutionary Algorithms (EA). The major classes of EAs are: genetic algorithms, evolutionary programming, evolution strategies, classifier system, and genetic programming. They all share a common conceptual base of simulating the evolution of individual structures via processes of selection, mutation, and reproduction.The processes depend on the perceived performance of the individual structures as defined by an environment. Gases maintain a population of structures that evolve according to rules of selection and other operators that are referred to as "search operators" such as recombination and mutation. Each individual in the population receives a measure of it's fitness in the environment. Reproduction focuses attention on high fitness individuals, thus exploiting the available fitness information. Recombination and mutation perturb those individuals, providing general heuristics for exploration. Although simplistic from a biologist's viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

GAs are not guaranteed to reach the global optimum, but they are generally good at finding an acceptable solution during an acceptable amount of time. They are mainly designed to solve optimization problems. However, when cooperating with other techniques it can also deal with problems with constrains. It is so robust that it can be applied to a wide range of problem areas. It also has good performance when solving some difficult problems which no existing specialized techniques can perform well on. Even if such specialized techniques exist, improvements could be made by hybridizing them with a GA.

2.1.1 History

Idea of evolutionary computing was introduced in 1960s by I. Rechenberg in his work "Evolution strategies. His idea was then developed by other researchers. Genetic Algorithms (GAs) were invented by John Holland and developed by him and his students and colleagues. This lead to Holland's book ***"Adaptation in Natural and Artificial Systems"*** published in 1975. Holland was not so much interested in optimization, but in adaptation. He investigated the genetic algorithm with decision theory for discrete domains. Holland emphasized the importance of recombination in large populations. Simply said, solution to a problem solved by genetic algorithms is evolved. In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method "genetic programming" (GP).

2.1.2 Biological Background

All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serves as a model for the whole organism. A chromosome's characteristic is determined by the genes. Each gene has several forms or alternatives which are called alleles, producing differences in the set of characteristics associated with that gene. The set of chromosome is called the genotype, which defines a phenotype (the individual) with a certain fitness.

During reproduction, first occurs recombination (or crossover). Genes from parents form in some way the whole new chromosome. The new created offspring can then be mutated. Mutation means, that the elements of DNA are a bit changed. This changes are mainly caused by errors in copying genes from parents. The fitness of an organism is measured by success of the organism in its life. According to Darwinian theory the highly fit individuals are given opportunities to "reproduce" whereas the least fit members of the population are less likely to get selected for reproduction, and so "die out".

2.2. Basic Principles and How They Work

Based on a natural phenomenon called "the survival of the fittest", only the fittest individuals survive and reproduce. The reproduction process happens in the gene pool. New combinations of genes are generated from previous ones by exchanging segments of genetic material among chromosomes (known as crossover"). Then a new gene pool is created. Repeated selection and crossover cause the continuous evolution of the gene pool and the generation of individuals that survive better in a competitive environment.

2.2.1 Simple Genetic Algorithm and Basic Principles

The first person who proposed genetic algorithms (GAs) as computer programs that mimic the evolutionary process in nature is Holland, in early 1970s. His genetic algorithm is commonly called the Simple Genetic Algorithm or SGA, shown in figure-1.

```
Simple Genetic Algorithm()
{
    initialize population;
    evaluate population;
    while termination criterion not reached
    {
        select solution for next population;
        perform crossover and mutation;
        evaluate population;
    }
}
```

Figure 1: Simple Genetic Algorithm

GAs operate on **encoded representations** of the solutions, equivalent to those chromosomes of individuals in nature. It is assumed that a potential solution to a problem may be represented as a set of parameters and encoded as a chromosome. In the SGA, Holland encoded the solutions as strings of bits from a binary alphabet.

A **fitness function** must be provided for evaluating each string. Each solution is associated with a fitness value, based on the fitness function, to reflect how good it is.

Selection models nature's survival-of-the-fittest mechanism. In principle, individuals from the population are copied to a "mating pool", with highly fit individuals being more likely to receive more than one copy, and unfit individuals being more likely to receive no copies. The size of the mating pool is equal to the size of the population. In the SGA, a fitter string receives a higher number of offspring and thus has a higher chance of surviving in the next generation. In the proportionate selection scheme, a string f_i with fitness value f / f_i is allocated offspring, where f is the average fitness value of the population. The SGA uses the **roulette wheel selection scheme** to implement proportionate selection. Each string is allocated a sector of a roulette wheel with the angle subtended by the sector at the center of the wheel equaling $2\Pi f_i / f$.

A string is allocated an offspring if a randomly generated number in the rage 0 to 2Π falls in the sector corresponding to the string.

The **reproduction** phase of GA is simulated through a **crossover** mechanism. The simplest method of crossover is to cut the chromosomes of two individuals at some randomly chosen position, and then exchange their “head” and “tail” segments, known as 1-point crossover. Usually not all pairs of individuals are selected for mating. The **crossover rate** being applied is typically between 0.6 and 1.0. If crossover is not applied, offspring are produced simply by duplicating the parents. Another operation, called **mutation**, causes sporadic and random alteration of the bits of strings, which is a direct analogy from nature and plays the role of regenerating lost genetic materials. It is applied to offspring after crossover. Another parameter, **mutation rate**, gives the probability that a bit will be flipped. **Convergence** is the progression towards increasing uniformity in the gene pool. A gene is said to have converged when 95% of the population share the same value .

2.2.2 Pseudo-Code for Genetic Algorithms

The following is a pseudo-code for general genetic algorithm approach:

0. **[Representation]** Define a genetic representation of the system.
1. **[Start]** Generate random population of n chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness of each chromosome in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete
   - 3.1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
   - 3.2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
   - 3.3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
   - 3.4. **[Accepting]** Place new offspring in a new population

4. **[Replace]** Use new generated population for a further run of algorithm
5. **[Test]** If the end condition is satisfied, stop, and return the best solution in current population
6. **[Loop]** Go to step 2 As you can see, the pseudo-code very general.

There are many things that can be implemented differently in various problems. First question is how to create chromosomes, what type of encoding to choose. In connection with this is the choice of the two basic operators of GA, which are crossover and mutation. Furthermore, selection of parents from the current solution is also to be clearly defined.

2.2.3 Encoding

The chromosome should in some way contain information about solution which it represents. The most used way of encoding is a binary string. In binary encoding every chromosome is a string of bits, 0 or 1. The chromosome then could look like this:

Chromosome 1: 1101100100110110

Chromosome 2: 1101111000011110

Each chromosome has one binary string. Each bit in this string can represent some characteristic of the solution. Or the whole string can represent a number.

Encoding depends on the problem and also on the size of instance of the problem. Of course, there are many other ways of encoding. Permutation encoding, value encoding, and tree encoding are among the many other encoding systems used in GA. These and many other encoding schemes are discussed in most of the references given at the end.

2.2.4 Selection

According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods how to select the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others. Two of these are briefly given, namely, roulette wheel selection and rank selection:

Roulette Wheel Selection: Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel (pie chart) where all chromosomes in the population are placed in according to their normalized fitness. Then a random number is generated which decides the chromosome to be selected. Chromosomes with bigger fitness values will be selected more times since they occupy more space on the pie.

Rank Selection: The previous selection will have problems when the fitnesses differs very much. For example, if the best chromosome fitness is 90% of all the roulette wheel then the other chromosomes will have very few chances to be selected. Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population). After this all the chromosomes have a chance to be selected. But this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones. When creating new population by crossover and mutation, we have a big chance that we will loose the best chromosome. Elitism is a method, which first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution.

2.2.5 Crossover and Mutation

Selection alone cannot introduce any new individuals into the population, i.e., it cannot find new points in the search space. These are generated by genetically-inspired operators, of which the most well known are **crossover and mutation.** Crossover is sometimes referred to as recombination, too. The crossover and mutation are the most important part of a genetic algorithm. The performance of the algorithm is mainly influenced by these two operators. Usually, there is a predefined probability of procreation via each of these operators. Traditionally, these probability values are selected such that crossover is the most frequently used, with mutation being resorted to only relatively rarely. This is because the mutation operator is a random operator and serves to introduce diversity in the population. The kind of operator to be applied to each member of the gene pool is determined by random choice based on these probabilities. Of the two operators, mutation involves only a single parent and result in the creation of a single offspring. The standard crossover operator called simple crossover has numerous variants such as partially-mapped, position-based, order-based, sub tour chunking, cyclic, acyclic, inversion, and edge-recombination crossovers. All of these involve two parents. Depending on operator and problem context, each generates either one or two offspring. Crossover takes two individuals, and cuts their chromosome strings at some randomly chosen position, to produce two "head" segments, and two "tail" segments. The tail segments are then swapped over to produce two new full-length chromosomes. The two offspring each inherit some genes from each parent. This is known as single point crossover. Crossover is not usually applied to all pairs of individuals selected for mating. A random choice is made, where the likelihood of crossover being applied is typically between 0.6 and 1.0. If the crossover is not applied, offspring are produced simply by duplicating the parents. This gives a chance of passing on its genes without the disruption of crossover. Mutation is applied to each child individually after crossover. It randomly alters each gene with a small probability (typically 0.001). The traditional view is that crossover is more important of the two techniques for rapidly exploring a search space. Mutation provides a small amount of random search, and helps ensure that no point in the search space has a zero probability of being examined. For binary encoding the crossover can look like this (| is the crossover point):

Chromosome 1 : 11011 | 00100110110

Chromosome 2 : 11011 | 11000011110

Offspring 1 : 11011 | 11000011110

Offspring 2 : 11011 | 00100110110

And mutation can produce the following offsprings:

Offspring 1 : 1101111000011110

Offspring 2 : 1101100100110110

Mutated offspring 1 : 1100111000011110

Mutated offspring 2 ::1101101100110110

2.2.6 Introductory Example

Let us consider the following simple example, demonstrating the genetic algorithm's workings. The population consists of 4 individuals, which are binary-encoded strings (genomes) of length 8. The fitness value equals the number of ones in the bit string, with a crossover probability of 0.7, and a mutation probability of 0.001. The initial (randomly generated) population might look like this:

Chromosome Alleles Fitness

A 00000110 2

B 11101110 6

C 00100000 1

D 00110100 3

Using fitness-proportionate selection we must choose 4 individuals (two sets of parents), with probabilities proportional to their relative fitness values. In our example, suppose that the two parent pairs are {B,D} and {B,C} (note that A did not get selected as our procedure is probabilistic). Once a pair of parents is selected, crossover is effected between them with probability 0.7, resulting in two offspring. Suppose, in our example, that crossover takes place between parents B and D at the (randomly chosen) first bit position, forming offspring E=10110100 and F=01101110, while no crossover is effected between parents B and C, forming offspring that are exact copies of B and C. Next, each offspring is subject to mutation with probability 0.001 per bit. For example, suppose offspring E is mutated at the sixth position to form E'=10110000, offspring B is mutated at the first bit position to form B'=01101110, and offspring F and C are not mutated at all. The next generation population, created by the above operators of selection, crossover, and mutation is therefore:

Chromosome Alleles Fitness

E' 10110000 3

F 01101110 5

C 00100000 1

B' 01101110 5

Note that in the new population, although the best individual with fitness 6 has been lost, the average fitness has increased. Iterating this procedure, the genetic algorithm will eventually find a perfect string, i.e., with maximal fitness value of 8.

2.3 How GAs work

While GAs have been applied for a large number of optimization problems, there is no accepted "general theory" which explains exactly why GAs have the properties they do. Although a very clear picture of the workings of GAs has not yet emerged, there are several hypotheses having been put forward which can partially capture the essence of GA mechanics[35] .

2.3.1 Schemata and the Schema Theorem

A schema is a pattern describing a subset of strings with the same gene value at certain positions. For example, a schema 11*** represents strings with 1s in the first two positions, and 11000 is an instance of this schema. The order of a schema is the number of fixed positions it contains. The **defining length** of a schema is the distance between the outmost fixed positions. For example, the order of **1*0 is 2, and the defining length is 3. If an individual has high fitness, it is due to the fact that it contains good schemata. It is more likely to find better solutions by passing good schemata to the next generation. Thus, Holland showed that the best way to explore the search space is to allocate reproductive trials to individuals in proportion to their fitness value relative to the rest of the population, so that good schemata receive an exponentially increasing number of trials in successive generations. This is called **schema theorem**. He also showed that the number of schemata being processed in each generation is of the order 3 n , where n is the population size. This capacity of GAs, known as **implicit parallelism**, arises from the fact that a string simultaneously represents l 2 (where l is the number of bit positions in a string) different schemata (because for each position, it can be fixed or not).

2.3.2 Building Block Hypothesis

Try to visualize the GA's search for the optimal string as a simultaneous competition among schemata to increase the number of their instances in the population. We can describe the optimal string as the juxtaposition of schemata with short defining lengths and high average fitness values. Such schemata are called **building blocks**. According to Goldberg[21] , the power of GAs lies in their ability to find good building blocks. Building-block hypothesis assumes that strings with high fitness values can be located by sampling building blocks with high fitness values and combining the building blocks effectively, and this is most done by crossover operation. However it is not always true that the juxtaposition of good building blocks yields good strings. Depending on the objective function, very bad strings can be generated when good building blocks are combines. Such objective functions are called **GA-deceptive functions**. It happens when there is interaction (often referred to as **epistasis**) between genes. That is, the contribution of a gene to the fitness depends on the value of other genes in the chromosome. Thus, a successful coding scheme encourages the formation of building blocks by ensuring that related genes are close together on the chromosome, while there is little interaction between genes.

2.3.2 Exploration and Exploitation

A good search algorithm must use two techniques to find a global optimum: exploration for new and unknown areas in the search space, and exploitation to make use of knowledge found at visited points. However these two techniques are contradictory, and a good search algorithm must find a tradeoff between them.

Holland[27] showed that GAs combine both exploration and exploitation at the same time in an optimal way. This may be theoretically true, but in practice there are inevitably problems, because Holland made certain simplifying assumptions: infinite population, the fitness function accurately reflecting the utility of a solution, and no interaction between genes. However the first assumption can never be satisfied in practice, and thus GAs are doomed to have stochastic errors. One such problem, which is also found in nature, is that of **genetic drift**[6]. For the second and third assumptions, they may be satisfied in a laboratory test, but are harder to satisfied for real world problems.

2.4 Practical Aspects

When theories go into practice, we need to consider far more than those theoretical aspects described above. Besides, most of the steps in the traditional GA can be implemented using a number of different algorithms.

2.4.1 Initial Population

The initial population may be generated randomly, or through some heuristic methods[25].

2.4.2 Fitness Function

The fitness function is the most crucial aspect of GAs, along with the coding scheme used. Grefenstette[24] sought an ideal set of parameters for a GA but concluded that within fairly wide margins, parameter settings were not critical. What is critical in the performance of a GA is the fitness function and the coding scheme used. A general rule to construct a fitness function is that it should be able to reflect the value of a chromosome in a real way. However, the "real" value of a chromosome is usually not good enough for guiding a genetic search. When coming up with a combinatorial optimization problem, where there are many constraints, most points in the search space represent invalid chromosome and hence have the real value zero. In this case, a better fitness function should be defined in terms of how good it is at leading us towards valid chromosomes.

Cramer [Cra85] suggested that if the natural goal of the problem is all or nothing, better results could be obtained if we invent meaningful subgoals and reward them.

Another approach is to use **penalty function**, which represents how poor the chromosome is, and construct the fitness as (constant – penalty). Richardson et al[30] states that those that represent the amount by which the constraints are violated are better than those simply based on the number of constraints violated. Good penalty functions can be constructed from the expected completion cost, which is how much an invalid chromosome will "cost" to turn it into a valid one. We will talk about more on this issue later when applying GAs to constraint satisfaction problems.

2.4.3 Fitness Range Problems

As the population converges during the process of a genetic algorithm, so the range of fitness in the population reduces. Similar to some other search algorithms, it is also possible for GAs to converge on a local maximum: when the genes from a few comparatively highly fit but not optimal individuals rapidly come to dominate the population. Only mutation remains to explore new space. However it simply performs a slow, random search[22] . This phenomenon is known as **premature convergence**, and is mainly because the population is not infinite. The basic idea to deal with this problem is to control the number of reproductive opportunities each individual gets, to prevent any "super-fit" individuals from suddenly taking over. The converse problem to premature convergence is **slow fishing**. It is due to insufficient gradient in the fitness function to push the GA towards the maximum.

2.4.4 Parent Selection Techniques

We have already seen the parent selection method in SGA. In order to avoid those problems mentioned in the previous section, several selection techniques have been proposed[2]. We can categorize them into two groups: **explicit and implicit fitness *remapping***.

Explicit fitness remapping includes fitness scaling, windowing, and ranking. In **fitness scaling**, the maximum number of reproductive trials allocated to an individual is set to a certain value, typically 2.0. This is achieved by subtracting a suitable value from the raw fitness score, then dividing by the average of the adjusted fitness values. However, the presence of just one super fit individual can lead to over compression. Besides, if the fitness function is too flat, genetic drift will become a problem.

Fitness windowing is used in Grefenstette's GENESIS GA package [23] . This is similar to fitness scaling, except that the amount to be subtracted is chosen differently. The minimum fitness in each generation is recorded, and the amount to be subtracted is the minimum fitness in the previous n generations, where n is typically 10.

In **fitness ranking**, individuals are sorted in order of raw fitness, and then new fitness values are assigned according to rank. This may be done either linearly[2] or exponentially[9] . Fitness ranking can cease over compression problem. In general, several experiments have shown fitness ranking is superior to fitness scaling[2].

In **implicit fitness remapping**, it fills the mating pool without passing through the intermediate stage of remapping the fitness. **Tournament selection [8]** is a typical method of implicit fitness remapping. The simplest form is **binary tournament selection.** We randomly pick pairs of individuals from the population, and copy the one with higher fitness into the mating pool. Another related replace method is **steady-state replacement** [9,11,37] . Instead of replacing the whole population between generations, only a few (typically two) individuals are replaced. This model may be more similar to what happens in nature, by giving rise to competition between parents and their children.

Goldberg & Deb[19] compare 4 different schemes: proportionate selection, fitness ranking, tournament selection, and steady state selection, and conclude that by suitable adjustment of parameters, they will give similar performances.

2.5 Variants and Current Research Topics

Several variants of GAs have been proposed and some problems have also been raised [5] . In this section we will explore some main research topics of GAs.

2.5.1 Crossover Techniques

As mentioned above, SGA uses 1-point crossover, where mating chromosomes are cut once. Other crossover techniques have also been devised, often involving more than one cut point. In **2-point crossover**, chromosomes are regarded as loops by connecting the ends together. Two cut points decide a segment, and two chromosomes exchange the segment. It performs the same task as 1-point cross over, but more general. Researchers now agree that 2-point crossover is generally better than 1-point crossover, because a looped chromosome may contain more building blocks. More-then-two-point crossover may be possible, but DeJong[14] concluded[2] that 2-point crossover gives an improvement, but adding further crossover points reduces the performance of the GA. However, an advantage of having more crossover points is that the problem space can be searched more thoroughly.

In **uniform crossover**, each gene in the offspring is created by copying the corresponding gene from either parent, according to a randomly generated crossover mask. Syswerda [37] argues that uniform crossover is the best crossover method, because under uniform crossover, schemata of a particular order are equally likely to be disrupted, irrespective of their defining lengths. Therefore the total amount of schemata disruption is lower. For example, the performance of GAs using 2-point crossover drops dramatically if the recommendations of the building block hypothesis[3] are not adhered to. However, uniform crossover still performs well in this case.

Researchers have done several experiments in order to prove which is the best crossover method. Eshelman el al[17] showed that no overall winner emerged. Spears and DeJong[31] say that 2-point crossover will perform poorly when the population has largely converged, because the segments exchanged are likely to be identical. A possible way to deal with this problem is to choose two new cross points again when identical offspring are produced. DeJong and Spears [16] conclude that this modified 2-point crossover is best for large populations, but the increased disruption of uniform crossover is beneficial if the population size is small. Many other crossover techniques have been suggested. One is that the GA adaptively learns which sites should be favored for crossover. This information is recorded in a **punctuation string**, which is part of the chromosome and can be passed on to the offspring[12,28] . Another one is called **partially matched crossover** (PMX) for use in order-based problems[21] (such as the traveling salesperson problem). In PMX the order of genes are crossed instead of values.

2.5.2 Inversion and Reordering

The order of genes on a chromosome is critical for the building block hypothesis to work effectively. Thus techniques for **reordering** the positions of genes have been suggested.

Inversion [27] is one of such techniques and works by reversing the order of genes between two randomly chosen potions within the chromosome. In fact, reordering is inspired by nature. There are many mechanisms by which the arrangement of the chromosomes may evolve (known as **karyotypic evolution**) [MS89] so that organisms can easily adapt to new conditions as the environment changes. However, for the majority of GA applications, the environment is static. Hence reordering is of little importance in these cases.

2.5.3 Epistasis

Epistasis is the interaction between different genes in a chromosome. When there is little interaction between genes, tasks can be solved efficiently by simple techniques, such as hill-climbing, and do not require a GA. When there is strong interaction, GAs can outperform simple techniques. However, according to the building block hypothesis, one of the basic requirements of GAs to be successful is low epistasis. Thus we need to know whether we can either avoid it, or develop a GA which works well with high epistasis.

In a GA, if schemata which are not contained in the global optimum increase more rapidly than those which are, the GA will be mislead away from the global optimum. This is known as **deception**, which is a special case of epistasis, and is difficult to solve. It can be tackled in two ways: as a coding problem or a GA theory problem. In the theory part, Davis and Coombs point [13] out that GAs have been made to work even in domain of high epistasis. Davidor[10] also points out that present-day GAs are only suitable for problems of medium epistasis. If the epistasis is too high, GAs will not be effective; if it is too low, GAs will be outperformed by simpler techniques. In the coding part, Beasley, Bull, and Martin [5] presented a technique called **expansive coding** for designing reduced-epistasis representations.

2.5.4 Hamming Cliffs and Gray Codes

Most optimization problems have continuous variables that assume real values. A common way for encoding continuous variables in the binary alphabet is to encode each variable with a fixed number of binary bits, and concatenate all strings together. A drawback of it is the presence of **Hamming cliffs** – the hamming distances between the binary codes of adjacent integers. For example, 01111 and 10000 are the integer representations of 15 and 16 respectively, but have a hamming distance of 5. **Gray codes** suggested alleviating the problem by ensuring that the codes for adjacent integers always have a Hamming distance of 1. However, the Hamming distance does not monotonously increase with the difference in integer values, and it introduces Hamming cliffs at other levels.

2.5.5 Mutation and Naive Evolution

Do we really need to do crossover in GA? Actually, biologists see mutation as the main source for evolutionary change [26]. Schaffer et al [31] suggest that "**naive evolution**" (just selection and mutation) performs a hill climb-like search which can be powerful without crossover. Later in another paper [32] they found that crossover gives much faster 15 evolution than a mutation only population, but mutation generally finds better solutions than a crossover-only regime. Spears [36] further suggests a suitable modified mutation operator can do everything that crossover can do. Eshelman [18] also states "the key to naive evolution's success is the use of Gray coded parameters, making search much less susceptible to amming cliffs". He believes that naive evolutions is a much more powerful algorithm than many people in the GA community have been willing to admit.

2.5.6 Adaptation

Using dynamically variable crossover or mutation rate (operator probabilities) might help adaptation. Davis[9,11] describes an adaptive technique that a weighting figure is allocated to each operator, based on its performance over the past 50 matings. Credits are given to those operators which can produce better offspring. However it may reward operators which simply locate local optimum. Some researchers vary the mutation probability by decreasing it exponentially during a run[1,7]. Unfortunately there is no clear reason why this should lead to an improvement.

2.5.7 Distributed and parallel GAs

Distributed GAs distributed a large population into a number of weakly interacting subpopulations, and each evolves independently. To ensure global competition, the best chromosomes of the subpopulations are exchanged. **Parallel GAs** are parallel implementations of the sequential GA to speed execution.

2.5.8 Knowledge-based Techniques

Some researchers have advocated designing new operators using domain knowledge[11] to make each GA more task-specific. For example, Davidor designed **"analogous crossover"** for his task in robotic trajectory generation. It used local information in the chromosome to decide which crossover sites would yield unfit offspring. Domain knowledge can also be applied in designing **local improvement operators**[34], or performing **heuristic initialization** of the population to make search begins with some reasonably good point[25]. Goldberg[21] described techniques of knowledge–directed crossover and mutation, and the hybridization of GAs with other search techniques[11] .

2.5.9 Redundant Value Mapping

If a binary representation is used, and the number of values of a gene is not a power of 2, some of the binary codes are redundant and not correspond to any valid gene value. A number of solutions are briefly mentioned by DeJong[15]：

- Discard the chromosome as illegal.
- Assign the chromosome low fitness.
- Map the invalid code to a valid one. (remapping)

There are several ways of achieving remapping: fixed remapping (an invalid gene is always mapped to another specific valid gene), random remapping, or probabilistic remapping (every gene value is remapped to one of the valid values in a probabilistic way).

2.6 Comparison with Other Techniques

Most research into GAs has so far concentrated on finding empirical rules for getting them to perform well. There is no accepted "general theory" which explains exactly why GAs have the properties they do. Nevertheless, several hypotheses have been put forward which can partially explain the success of GAs. Holland's Schema theorem was the first rigorous explanation of how GAs work.

According to Goldberg, the power of the GAs lies in it being able to find good building blocks. However, both theorems have been criticized in recent time.
There are three main types of traditional or conventional search method: calculus-based, enumerative, and random. Calculus-based methods are also referred to as gradient methods. These methods use the information about the gradient of the function to guide the direction of search. If the derivative of the function cannot be computed, because it is discontinuous, for example, these methods often fail. Such methods are generally referred to as hill climbing. Enumerative methods work within a finite search space, or at least a discredited infinite search space. The algorithm then starts looking at objective function values at every point in the space, one at a time.
Random search methods are strictly random walks through the search space while saving the best.
GAs differ from conventional optimization/ search procedures in that:

1. They work with a coding of the parameter set, not the parameters themselves.
2. They search from a population of points in the problem domain, not a singular point.
3. They use a payoff information as the objective function rather than derivatives of the problem or auxiliary knowledge.
4. They utilize probabilistic transition rules based on fitness rather than deterministic one.

We can see that both the enumerative and random methods are not efficient when you have a significantly large search space or significantly difficult problem, as in the realm of NP-Complete problems. The calculus-based method are inadequate when you are searching a "noisy" search space (one with numerous peaks). Calculus-based methods also depend upon the existence of derivatives or well-defined slope values. But, "the real world of search is fraught with discontinuities, vast multimodal noisy searchspaces."

Simulated Annealing: This technique was invented by Kirkpatrick in 1982. Starting from a random point in the search space, a random move is made. If this move tales us to a higher point, it is accepted. If it takes us to a lower point, it is accepted only with probability p(t), where t is time. The function p(t) begins close to 1, but gradually reduces towards zero.

A genetic algorithm, as a search process, differs in one important aspect from simulated annealing and tabu-search. At each iterative step a number of different solutions are generated and carried over to the next step. In simulated annealing and tabu-search, only a single solution is carried over from one iteration to the next. Hence simulated annealing and tabu-search may be regarded as special cases of genetic algorithms with a population size equal to 1.

2.6 Genetic Algorithm Operators

The third decision to make in implementing a genetic algorithm is what genetic operator to use. The decision depends greatly on the encoding strategy. Here I will discuss crossover and mutation mostly in the context of bit-string encoding and I will mention a number of other operators that have been proposed in GA literature.

(1) Crossover

It could be said that the main distinguishing feature of a GA is the use of crossover. Single point crossover is the simplest form: a single cross-over position is chosen at random and the parts of two parents after the crossover position are exchanged to form two offspring. The idea here is, off course, to recombine building blocks (schemas) on different strings. Single point crossover has some shortcomings, though. For one thing, it cannot combine all possible schemas. For example, it cannot in general, combine instances of 11*****1 and *****11** to form an instance of 11**11*1. Likewise, schemas with long define lengths are likely to be destroyed in the single point crossover. Eshelman,Caruana, and Schaffer(1989) call this " position bias" : the schemas that can be created or destroyed by the crossover depend strongly on the location of the bits in the chromosomes. Single-point crossover assumes that short low order schemas are the functional building blocks of strings but one generally does not in advanced what ordering will group functionally related bits together. This was the purpose of inversion operator and other adaptive operators above. Eshelman, Caruana, and Schaffer also point out that there may not be any way to put all functionally related bits close together on a string since particular bits might be more crucial in more than one schema. They point out further that the tendency of a single point crossover to keep short intact can lead to preservation of hitchhikers- bits that are not part of the desired schema but which, by being closed to the string hitchhike along with the beneficial schema as it reproduces. (This was seen in "Royal Road" experiments, described above in chapter 4) Many people have also noted that single point crossover treats some loci preferentially the segments exchanged between two parents always contain the end points of the string.

To reduce positional bias and this end point effect many GA practitioners use two point crossover in which two positions are chosen at random and the segments between them are exchanged. Two point crossover is less likely to disrupt schemas with large defining lengths and can combine more schemas than single point crossover. In addition, the segments are exchanged that do not necessarily contain the end points of the strings. Again, there are schemas that two point crossovers cannot combing. GA practitioners have experimented with different number of crossover points (in one method, the number of crossover points for each parents is chosen from a Poisson distribution whose mean is the function of length of chromosome). Some practitioners believe strongly in the superiority of "parameterized uniform crossover" in which an exchange happens at each bit position with probability p (typically $0.5 <= p <= 0.8$). Parameterized uniform crossover has no position bias. Any schemas contained at different positions in the parents can potentially be recombined in the offspring. However, this lack of position bias can prevent co adapted alleles from ever forming in the population, since parameterized uniform crossover can be highly disruptive of any schema.

Given these (any the many other variants of crossover found in the GA literature), which one should you use? There is no simple answer, the success or failure of a particular crossover operator depends in complicated ways on the particular fitness function, encoding and other details of the GA. It is still a very important open problem to fully understand these interactions. There are many papers in GA literature quantifying aspects of various crossover operation (Position bias , disruption potential, ability to create different schemas in one step, and so on),but these do not gibe definitive guidance on when to use which type of crossover. There are also many papers in which the usefulness of different types of crossover is empirically compared, but all these studies produce conflicting results. Again, it si hared to glean general conclusions. It is common in recent GA applications to use either two point crossover or parameterized uniform crossover with p=0.7-0.8.

For the most part, the comments and references above deal with crossover in the context of bit-string encoding, through some of them apply to other types of encoding as well. Some types of encoding require especially require especially defined crossover and mutation operators- for example, the tree encoding used in genetic programming, or encoding for problems like the Traveling Salesman problems (in which the task is to find correct ordering for allocation of object)

Most of the comments above also assume that crossover' stability to recombine highly fit schemas is the reason it should be useful. Giben some of the challenges we have seen to the relevance of schemas as an analyst tool for understanding GAs, one might ask if we should not consider the possibility that crossover is actually useful for some entirely different reason (e.g. it is in sense a "macro mutation" operator that simply allows for large jumps in the search space). I must leave this question as an open area of GA research for interested readers to explore. (Terry Jones (1995) has performed some interesting, though preliminary, experiments attempting to tease out the different possible roles of crossover in Gas).

Its answer must shed light on the question of why recombination is useful for real organisms (if indeed it is) -controversial and still open question in revolutionary biology.

(2). Mutation

A common view in the GA community, dating back to Hollan's book Adaptation in Natural and Artificial Systems, is that crossover is the major instrument of variation and innovation in GA's, with mutation insertion the population against permanent fixation at any particular locus and thus playing more of a background role. This differs from the traditional positions of other evolutionary computation methods, such as evolutionary programming and early versions of evolution strategies, in which random mutation is the only source of variations.(later versions of evolution strategies have included a form of crossover.)

However, the appreciation of the role of mutation is growing as the GA community attempts to understand how GA's solve complex problems. Some comparative studies have been performed how GA's solve complex problems. Some comparative studies have been performed on the power of mutation versus mutation and crossover have the same ability for "disruption" of existing schemas, crossover is a more robust "constructor" of new schemas. Muhlenbein(1992, p. 15), on the other hand, argues that in many cases a hill climbing strategy will work better than a GA with crossover and that " the power of mutation has been underestimated in traditional genetic algorithms." AS we saw in the Royal Road experiments, it is not a choice between crossover , mutation and selection that is all important. The correct balance also depends on details of the fitness function and the encoding. Furthermore, crossover and mutation vary in relative usefulness over the course of a run. Precisely how all this happens still needs to be elucidated. In my opinion the most promising aspect for producing the right balances over the course of a run is to find ways for the GA to adapt its own mutation and crossover rated during a search. Some attempts at this will be described below.

2.8 Other Operators and Mating Strategies

Though most GA applications use only crossover and mutation, many other operators and strategies for applying them have been exploded in the GA literature. These include inversion and gene doubling (discussed above) and several operators for preserving diversity in the population.

For example, De Jong(1975) experimented with a "crowding" operator in which a newly formed offspring replaced the existing individual most similar to itself. This prevented too many similar individual ("crowds") from being in the population at the same time. Gold Berg and Richardson(1987) accomplished a similar result using an explicit "fitness sharing function: each individual's fitness was decreased by the presence of other population members, where the amount of decrease due to each other population member was an explicit increasing function of similarity between two individuals. Thus, the individual that were similar to many other individuals were punished and the individuals that were different were rewarded., Goldberg and Richardson showed that in some cases this could include appropriate "speciation" allowing the population members to converge on several peaks in the fitness landscape rather than a similar effect could be obtained without the presence of an explicit sharing function.

A differ way to promote diversity is to put restrictions on mating. For example if only sufficiently individual are allowed to mate, distinct “species"(mating groups) will tend to form. This approach has been studied by Deb and Goldberg(1989). Eshelman(1991)and Eshelman Schaffer(1991) used to opposite tack: they disallowed mating between sufficiently similar individuals ("incest"). Their desire was not to form species but to keep entire population as diverse as possible. Holland (1975)and Booker(1985) have suggested using "mating tags"-only those individual's with atching tags are allowed to mate (a kind of "sexual selection" procedure). Theses tags would. in a principle, evolve along with the test of the chromosomes to adaptively implement appropriate restriction on mating. Finally, there have been some experiments with spatially restricted mating (see, e.g., Hills 1992): the population evolves on a spatial lattice, and individuals are likely to mate only with individuals in their spatial neighborhood. Hills found that such a scheme helped preserve diversity by maintaining spatially isolated species, with innovations largely occurring at the boundaries between species.

2.9 Parameters for Genetic Algorithm

The fourth decision to make in implementing the genetic algorithm is how to set the values for various parameters, such as population size, crossover rate and mutation rate. These parameters typically interact with one another nonlinearly so that they cannot be optimized one at a time. There is a great deal of discussion of parameters settings and approaches to a parameters adaptation in the evolutionary computation literature- too much to study or even list. There are no conclusive results on what is the best, most people use what has worked well in previously reported cases. Here I will review some of the experimental approaches people have taken to find the "best" parameter setting.

De Jong (1975) performed on early systematic study of how varying parameters affected the GA's on-line and off-line search performances on a small suite of test functions. "on-line" performance at time 't' is the average fitness of all the individuals that have been evaluated over the t evaluations steps. The off-line performance at time t si the average value over t evaluation steps, of the best fitness that has been setup to each evaluation step. De Jong's experiments indicated that the best population size was 50-500 individual's, the best single point cross-over rate was ~0.6 per pair of parents, and the best mutation rate was 0.001 per bit. These settings (along with De Jong's test suite) became widely used in the GA community, even though it was not clear how well the GA would perform with theses setting on problems outside De Jong's test suite. Any guidance was gratefully accepted.

Somewhat later, Grefenstette(1986)noted that , since the GA could be used as an optimization procedure , it could be used to optimize the parameters for another GA!(A similar study was done by Bramlette(1991). In Grefenstette's experiments, th e"meta-leve GA 'evolved a population of 50 GA parameter sets for the problems in De Longs test suite. Each individual encoded six GA parameters: population size, Crossover rate, mutation rate, generation gap, scaling window, and selection strategy (enlist or non-enlist). The fitness of an individual was a function of the on-line or off-line performance of GA using the parameters encoded by that individual. The meta-level GA itself used De Jong's parameter settings. The fittest individual for on- line

Performances set the population size to 30, the crossover rate to 0.95 , the mutation rate to 0.01 ,and the generation gap to 1, and used enlist selection. These parameters gave a small but significant improvement in on-line performance over De Jong's settings. Notice that Grefenstette's results call for a smaller population and higher crossover and mutation rates than De Jong's for off-line performance. This was an interesting experiment, but again, in view of the specialized test suite, it is not clear how generally these recommendations hold. Others have shown that there are many fitness functions for which these parameters settings are not optimal.

Scaffer, Caruana, Eshelman, and das(1989) spent over a year of CPU time systematically testing a wide range of parameters set was the on-line performance of a GA with those parameters on a small set of numerical optimization problems (including some of De Jong's functions) encoded with gray coding. Scaffer et al. found that the best settings for population size, crossover rate, and mutation rate were independent of the problem in their test suite. These settings were similar to those found by Grefenstette: population size 20-30, crossover rate 0.75-0.95, and a mutation rate 0.005-0.01. It may be surprising that a very small population size was better, especially in light of their studies that have argued for larger population sized(e.g, Goldberg 1989d), but this may be due to the on- line performance measure : since each individual ever evaluated contributes to the on- line performance , there is a large cost for evaluating a large population.

Although Grefenstette and Scaffer et al. found that a particular setting of parameters worked best for on-line performance on their test suites, it seems unlikely that any general principles about parameter setting can be formulated o prori, in view of the variety of problem types, encodings, and performance criteria that are possible in different applications. Moreover, the optimal population size, crossover rate, and mutation rate likely change over the course of a single run. Many people feel that the most promising approach is to have the parameters values adapt in real time to the outgoing search. There have been several approaches to self-adaptation of GA parameters. For example, this has long been a focus of research in the evolution strategies community, in which parameters such as mutation rate are encoded as part of the *chromosome.* Here I will describe Lawrence Davis's approach to self adaptation of operator rates(Davis 1989,1991).

Davis assigns to each operator a "fitness" which is a function of how many highly fit individuals that operator has contributed to created over the last several generations. Operators gain high fitness both for directly creating good individuals and for "setting the stage" for good individuals to be created (that is creating ancestors of good individuals). Davis tested this method in the context of a steady- state GA. Each operator (e.g., crossover, mutation) starts out with the same initial fitness. At each time step a single operator is chosen probabilistically (on the basis of its current fitness) to create anew individuals, which replaces allow fitness member of the population. Each individual I keep a record of which operator created it. If I has fitness better than the current best fitness, then i receives some created for the operator that created it, as do 1's parents, grandparents, and so on, back to a prescribed level of ancestor. The fitness of each operator over a given time interval is a function of its previous fitness and the sum of the credits received by all the individuals created by the operator during that time period. (The frequency with which operator fitness are updated is parameter of the method). In principle, the dynamically changing to keep up with the actual usefulness at different stages of the search, causing the GA to use them at appropriate rates at different times. As far as I know , this ability for the operator fitness to keep up with the actual usefulness of the operator has not been tested directly in any way, though Davis showed that this method improved the performance of a GA an some problems (including , it turns out, Montana and Davis's project on evolving weights for neutral networks).

A big question then, for any adaptive approach to setting parameters- including Davis's- is this: How well does the rate adaptation of parameter settings match the rate of adaptation in the GA population? The feedback for settings parameters comes from the population's success or the failure fitness function, but it might be difficult for this information to travel fast enough for the parameter settings to stay up to date with the population's current state. Very little work has been done on the measuring these different rates of adaptation and how well they match in different parameter-adaptation experiments. The most important to be done in order to get self-adaptation methods to work well.

2.10 PREVIOUS WORK ON GENETIC ALGORITHM

Linear programming is an optimization method applicable for the solution of the problems in which the objective function and the constraints appear as linear functions of the decision variables. At least four nobel prizes were awarded for contributions related to linear programming. When the Nobel Prize in economics was awarded in 1975 jointly to L.V. Kantorovich of the former Soviet Union and T.C. Koopmans of the United states, the citation of the prize mentioned there contributions on the application of linear programming to the economic problem of allocating resources.

Although genetic algorithms were first presented systematically by Holland the basic idea of analyses and design based on the concept of biological evolution is found in the work of Rechenberg.

In design optimization of electric motors by genetic algorithm is discussed. Anup Kumar et. Al. has analyzed the technique based on genetic algorithm for file allocation on the distributed system. in the use of genetic algorithms in search and optimization is given. Ramarathnam et. Al. deals with the comparative study of minimization techniques for optimization of induction motor design. gives the details of genetic algorithms have investigated the future paths for integer programming and links to artificial intelligence. Liepins et. Al. deals with the genetic algorithm foundations and applications. Based on modern control theory Fosha and Elgard have developed an optimal controller that provides better transient response. Gupta have used GA for the minimization of total intracell moves in cellular manufacturing.

2.11 FUTURE SCOPE OF WORK

The extensive study carried out create a thrust to augment the scope into the field of

1. Multiple criteria optimization.
2. Design of genetic algorithm based on fuzzy logic controllers and systems, where the membership function can be optimized using this technique.

3. Development of user friendly software of optimization of single and Multi-objective problems using genetic algorithms

2.12 Conclusion

Genetic algorithms are original systems based on the supposed functioning of the Living. The method is very different from classical optimization algorithms.

1. Use of the encoding of the parameters, not the parameters themselves.
2. Works on a population of points, not a unique one.
3. Use the only values of the function to optimize, not their derived function or other auxiliary knowledge.
4. Use probabilistic transition not determinist ones.

It's important to understand that the functioning of such an algorithm does not guarantee success. we are in a stochastic system and a genetic pool may be too far from the solution, or for example, a too fast convergence may halt the process of evaluation. these algorithms are nevertheless extremely efficient, and are used in fields as diverse as stock exchange. Production scheduling or programming of assembly robots in the automotive industry.

3.13 References :

[1]. D.H. Ackley. An empirical study of bit vector function optimization. In L. Davis, editor, Genetic Algorithms and simulated annealing,

[2]. J.E. Baker. Adaptive selection methoeds for genetic algorithms. In J.J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms.

[3]. D.Deasley, D.R. Bull, and R.R. Martin. An over view of genetic algorithms.

[4] D.Deasley, D.R. Bull, and R.R. Martin. Reducing epistasis in combinatorial problems by expansive coding. In S. Forrest, editor, Proceedings of the Fifth International Conference on Genetic algorithms

[5]. D.Deasley, D.R. Bull, and R.R. Martin. An over view of genetic algorithms.

[6]. L. Booker. Improving search in genetic algorithms. In L. Davis, editor, Genetic Algorithms and Simulated Annealing.

[7]. M.F. Bramlette. Initialization, mutation and selection methods in genetic algorithms for function optimization. In R.K. Belew and L.B. Booker, editor, Proceedings of the Fourth International Conference on Genetic Algorithms.

[8]. A. Brindle. Genetic algorithms for function optimization.

[9]. L. Davis. Adapting operator probabilities in genetic algorithms. In J.D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, 1989

[10]. Davidor. Epistasis variance: Suitability of a representation to genetic algorithms. Complex Systems.

[11]. L.Davis. Handbook of Genetic Algorithms.

[12]. Y. Davidor. A genetic algorithm applied to robot trajectory generation.

[13]. L. Davis and S. Coombs. Genetic algorithms and communication link speed design: theoretical considerations.

[14]. K. DeJong. The Analysis and behavior of a Class of Genetic Adaptive Systems

[15]. K. DeJong. Genetic algorithms: A 10 year perspective.

[16]. K. DeJong and W.M. Spears. An analysis of the interacting roles of populations size and crossover in genetic algorithms.

[17]. L.J. Eshelman, R. Caruna, and J.D. Schaffer. Biases in the crossover landscape.

[18]. Larry J. Eshelman. Bit-climbers and naïve evolution. GA-Digest.

[19]. D.E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms.

[20]. D.E. Goldberg. Alleles, loci, and the TSP. In J.J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms.

[21]. D.E. Goldberg. Genetic Algorithms in search, optimization and machine learning.

[22]. D.E. Goldberg. Sizing populations for serial and parallel genetic algorithms.

[23]. J.J. Grefenstette. GENESIS: A system for using genetic search procedures.

[24]. J.J. Grefenstette. Optimization of control parameters for genetic algorithms. IEEE Trans .

[25]. J.J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms.

[26]. D.L. Hartl. A primer of population genetics. Sinauer Associates Inc., 1988.

[27]. J.H. Holland. Adaptation in Natural and Artificial Systems.

[28]. J.H. Holland. Genetic algorithms and classifier systems: foundations and future directions.

[29]. S.J. Louis and G.J.E. Rawlins. Designer genetic algorithms: Genetic algorithms in structure design.

[30]. Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some guidelines for generic algorithms with penalty functions.

[31]. J.D. Schaffer, R.A. Caruna, Eshelman L.J., and R. Das. A study of control parameters affecting online performance of genetic algorithms of function optimization.

[31]. A.M. Spears and K. DeJong. An analysis of multi-point crossover.

[32]. J.D. Schaffer and L.J. Eshelman. On crossover as an evolutionarily viable strategy.

[33]. J.D. Schaffer and A. Morishma. An adaptive crossover distribution mechanism for genetic algorithms.

[34]. J.Y. Suh and D. Van Gucht. Incorporating heyristic information into genetic search.

[35]. M Srinivas & L. Patnaik. Genetic Algorithms: a Survey, Computer .

[36]. William M. Spears. Crossover or mutation?

[37]. G. Syswerda. Uniform crossover in genetic algorithms.

CHAPTER-3

POWER LOSSES IN TRANSMISSION AND DISTRIBUTION

3.1 Introduction

In India, average T & D (Transmission & Distribution) losses, have been officially indicated as 23 percent of the electricity generated. However, as per sample studies carried out by independent agencies including TERI, these losses have been estimated to be as high as 50 percent in some states. In a recent study carried out by SBI Capital Markets for DVB, the T&D losses have been estimated as 58%. This is contrary to claims by DVB that their transmission and distribution losses are between 40 and 50 percent. With the setting up of State Regulatory Commissions in the country, accurate estimation of T&D Losses has gained importance as the level of losses directly affects the sales and power purchase requirements and hence has a bearing on the determination of electricity tariff of a utility by the commission.

3.2 Components of T&D losses

Energy losses occur in the process of supplying electricity to consumers due to technical and commercial losses. The technical losses are due to energy dissipated in the conductors and equipment used for transmission, transformation, sub-transmission and distribution of power. These technical losses are inherent in a system and can be reduced to an optimum level. The losses can be further sub grouped depending upon the stage of power transformation & transmission system as Transmission Losses (400kV/220kV/132kV/66kV), as Sub transmission losses (33kV /11kV) and Distribution losses (11kV/0.4kv). The commercial losses are caused by pilferage, defective meters, and errors in meter reading and in estimating unmetered supply of energy.

3.3 Level of T& D Losses

The officially declared transmission and distribution losses in India have gradually risen from about 15 percent up to the year 1966-67 to about 23 percent in 1998-99. The continued rising trend in the losses is a matter of serious concern and all out efforts are required to contain the them. According to a study carried out by Electric Power Research Institute (EPRI) of the USA some time back, the losses in various elements of the T&D system usually are of the order as indicated below: -

System Elements	Power losses (%)	
	Minimum	Maximum
Step-up transformers & EHV transmission system	0.5	1.0
Transformation to intermediate voltage level, transmission system & step down to sub-transmission voltage level	1.5	3.0
Sub-transmission system & step-down to distribution voltage level	2.0	4.5
Distribution lines and service connections	3.0	7.0
Total Losses	7.0	15.5

The losses in any system would, however, depend on the pattern of energy use, intensity of load demand, load density, and capability and configuration of the transmission and distribution system that vary for various system elements. According to CEA vide its publication (July 1991) 'Guidelines for Reduction of Transmission and Distribution Losses' it should be reasonable to aim for total energy losses in the range of 10-15% in the different states in India. The enclosed Annexure-B indicates the rising trend of T&D losses in the various states in the past. This can be compared with T&D losses in the other countries indicated in the enclosed Annexures-A. A glimpse of this Annexure indicates that in most developed countries the T&D losses are less than 10 percent.

3.4 Reasons for or high T&D Losses

Experience in many parts of the world demonstrates that it is possible to reduce the losses in a reasonably short period of time and that such investments have a high internal rate of return. A clear understanding on the magnitude of technical and commercial losses is the first step in the direction of reducing T&D losses. This can be achieved by putting in place a system for accurate energy accounting. This system is essentially a tool for energy management and helps in breaking down the total energy consumption into all its components. It aims at accounting for energy generated and its consumption by various categories of consumers, as well as, for energy required for meeting technical requirement of system elements. It also helps the utility in bringing accountability and efficiency in its working.

3.5 Reasons for high technical losses

The following are the major reasons for high technical losses in our country: -

- Inadequate investment on transmission and distribution, particularly in sub-transmission and distribution. While the desired investment ratio between generation and T&D should be 1:1, during the period 1956 -97 it decreased to 1:0.45. Low investment has resulted in overloading of the distribution system without commensurate strengthening and augmentation.
- Haphazard growths of sub-transmission and distribution system with the short-term objective of extension of power supply to new areas.
- Large scale rural electrification through long 11kV and LT lines.
- Too many stage of transformations.
- Improper load management.
- Inadequate reactive compensation
- Poor quality of equipment used in agricultural pumping in rural areas, cooler air-conditioners and industrial loads in urban areas.

3.6 Reasons for or commercial losses

Theft and pilferage account for a substantial part of the high transmission and distribution losses in India. Theft / pilferage of energy is mainly committed by two categories of consumers i.e. non-consumers and bonafide consumers. Antisocial elements avail unauthorized/unrecorded supply by hooking or tapping the bare conductors of L.T. feeder or tampered service wires. Some of the bonafide consumers willfully commit the pilferage by way of damaging and / or creating disturbances to measuring equipment installed at their premises.Some of the modes for illegal abstraction or consumption of electricity are given below:

- Making unauthorized extensions of loads, especially those having “H.P.” tariff.
- Tampering the meter readings by mechanical jerks, placement of powerful magnets or disturbing the disc rotation with foreign matters.
- Stopping the meters by remote control.
- Willful burning of meters.
- Changing the sequence of terminal wiring.
- Bypassing the meter.
- Changing C.T.ratio and reducing the recording.
- Errors in meter reading and recording.
- Improper testing and calibration of meters.

3.7 T&D losses in restructure SEBs

Some states have embarked on programs of power sector reforms and have taken steps to restructure their SEBs (State Electricity Boards). The reforming states that were reporting T&D losses of around twenty percent before restructuring process suddenly reported higher losses after carrying out detailed studies of their system. For example, before restructuring its power sector, Orissa reported 23 percent loss, after restructuring, T&D loss were shown to be 51 percent. In AP where these losses were of the order of about 25 percent before restructuring, it is now estimated to be around 45 percent after restructuring. Haryana has now estimated its losses at 40 percent and Rajasthan at 43 percent against earlier level of 32 percent and 26 percent respectively.

3.8 Regulatory concerns

In the absence of a realistic estimate of T&D losses, it is not possible for the regulatory commissions to correctly estimate the revenue requirements and also avoid the situation where the consumers pay for the inefficiencies of the utilities.

In order to determine an appropriate tariff, the first step is to determine the justified cost incurred by the entity. This would provide an indication of the revenue requirement, which in turn is the basis of any tariff design. The regulator has therefore to be very careful about how losses are worked out.

The aim of the regulator must be to encourage the utility to make every effort to reduce losses while at the same time ensuring that those conditions applied which threaten the viability of the utility are not applied.

3.9 Barriers in private sector participation

The lack of realistic estimates of T& D losses acts as a disincentive for private sector participation in power distribution as the party can not have an idea of the realistic revenue potential of the area being privatized.

3.10 Unmetered supply

Unmetered supply to agricultural pumps and single point connections to small domestic consumers of weaker sections of the society is one of the major reasons for commercial losses. In most states, the agricultural tariff is based on the unit horsepower (H.P.) of the motors. Such power loads get sanctioned at the low load declarations. Once the connections are released, the consumers get into the habit of increasing their connected loads, without obtaining necessary sanction, for increased loading, from the utility. Further estimation of the energy consumed in unmetered supply has a great bearing on the estimation of T&D losses on account of inherent errors in estimation. Most of the utilities deliberately overestimate the unmetered agricultural consumption to get higher subsidy from the State Govt.and also project reduction in losses. In other words higher the estimates of the unmetered consumption, lesser the T&D loss figure and viceversa. Moreover the correct estimation of unmetered consumption by the agricultural sector greatly depends upon the cropping pattern, ground water level, seasonal variation, hours of operation etc.

To increase the food output, almost all the State Governments show benevolence to farmers and arrange supply of electric power for irrigation to the farmers at a nominal rate, and in some States, without charges at all. In view of this, most Electricity Boards supply power to agriculture sector and claim subsidy from the State Govt. based on energy consumption.

Since the energy supplied to the agriculture sector is a generous gesture by the State Govt., all the electricity boards have eliminated energy meters for agriculture sector services. The absence of energy meters provides ample opportunities to SEBs to estimate average consumption in agriculture sector at a much higher value than the actual. In the absence of energy meters, most of the SEBs resort to fudging consumption figures to include not only the under estimated T&D Losses but also energy theft from their system. The extent of fudging is more in the States where agricultural activity is high. The benefit derived by these boards is not only the extent of subsidy from the respective States but also self praise, by showing much less T&D losses. Further the boards are ignoring the inefficiency in operating the distribution system by blaming the agricultural supply for all ills and raising the tariff of other consumers.

Most of the methods being employed by SEBs for estimating the unmetered energy consumption are as follows: -

- Load factor based estimation.
- Estimation based on feederwise theoretical calculation of losses.
- Estimation based on readings of meters installed at all the Distribution Transformers located on a feeder.

However, none of the these methods provide correct estimation of unmetered consumption.

3.11 Measures for reducing technical losses

3.11.1 Short term measures

- Identification of the weakest areas in the distribution system and strengthening /improving them so as to draw the maximum benefits of the limited resources.
- Reducing the length of LT lines by relocation of distribution sub stations/ installations of additional distribution transformers (DTs).
- Installation of lower capacity distribution transformers at each consumer premises instead of cluster formation and substitution of DTs with those having lower no load losses such as amorphous core transformers.
- Installation of shunt capacitors for improvement of power factor.

3.11.2 Long term measures

- Mapping of complete primary and secondary distribution system clearly depicting the various parameters such as conductor size line lengths etc.
- Compilation of data regarding existing loads, operating conditions, forecast of expected loads etc.
- Carrying out detailed distribution system studies considering the expected load development during the next 8-10 years.
- Preparation of long-term plans for phased strengthening and improvement of the distribution systems along with associated transmission system.
- Estimation of the financial requirements for implementation of the different phases of system improvement works.
- Formulation of comprehensive system improvement schemes with detailed investment program so as to meet system requirement for first 5 years period.

3.12 Measures for reducing non-technical losses

According to the International Utilities Revenue Protection Association. (IURPA), research carried out on utilities worldwide indicates that service quality, customer relationships, and overall service satisfaction can minimize revenue losses. This has been demonstrated in Pakistan where rampant power theft has contributed financial crisis for WAPDA (Water & Power Development Authority). The World Bank and Asian Development Bank which had supplied the bulk of WAPDA's development loans wanted the authority to recover its unpaid dues, cut power theft and reduce its T&D Losses. Accordingly WAPDA was forced to raise power rates.

But instead of improving the financial situation, this action resulted in increased financial crisis of WAPDA due to increased incidence of theft and unpaid bills. In view of this, the authority applied extreme measures to curb power theft. The Chairman of the authority (a serving army officer) deployed 35,000 troops to tackle the crisis. The troops were instructed to identify and arrest people responsible for power theft. As a result of this more than 36 military courts began trying cases of power theft. There are a range of methods being employed by utilities the world over to mitigate power theft. Some of these measures are given below.

- Set up vigilance squads to check and prevent pilferage of energy.
- Severe penalties may be imposed on those tampering with the meter seals etc.
- Energy audits should be introduced and personal responsibility should be fixed on the district officers (executive engineers) for energy received and energy sales in each area.
- Installation of tamper-proof meter boxes and use of tamper-proof numbered seals.
- Providing adequate meter testing facilities. A time bound program should be chalked out for checking the meters, and replacement of defective meters with tested meters.

3.13 Initiatives required

Keeping the above in view it is very essential that immediate steps are initiated to have an assessment of the realistic T &D losses in each of the states and that immediate steps are taken to reduce the same in a systematic manner by all the players in the field.

- The central or the state governments should draw plans to provide financial support to the utilities for installations of meters on at least all the distribution transformers in a phased manner.
- It should be made obligatory for all the big industries as well as the utilities to carry out energy audit of their system to identify high loss areas and take remedial measures to reduce the same.
- Schemes for incentive awards to utilities who are able to reduce T&D losses beyond a certain pre-fixed limit.
- The financial institutions should be encouraged to provide easy loans to utilities for taking remedial measures to reduce the T&D losses.
- Publicity campaigns should be carried out to make the consumer aware of the high penalties on the unauthorized use of electricity.
- Utilities should prepare realistic power Master Plans for their systems to develop a strategy to meet the growing electricity demands of the different sectors of the state's economy over the next 15 years.

3.14 Issues for discussion

1. **Status of metering and steps required for early installation of the same**

   In view of the financial assistance being provided by the Central government for installation of meters, the feasibility of achieving the proposed targets can be an important issue for discussion.

2. **Mitigating power theft**

   Indian Electricity Act 1910 has been amended through Sections 39 and 39A to make theft of energy and its abetment as a cognizable offence with deterrent punishment of upto 3 years imprisonment.

Theft of electric power is a problem experienced in varying degrees by all electric utilities. The impact of theft is not limited to loss of revenue, it also effects power quality resulting in low voltage and voltage dips.

Adequacy of the existing measures to curb power theft could be an issue for discussion.

3. **Implementation of energy audits schemes**

It should be obligatory for all big industries and utilities to carry out Energy Audits of their system. Further time bound action for initiating studies for realistic assessment of the total T&D Losses into technical and non-technical losses has also to be drawn by utilities for identifying high loss areas to initiate remedial measures to reduce the same. The realistic assessment of T&D Loss of a utility greatly depends on the chosen sample size which in turn has a bearing on the level of confidence desired and the tolerance limit of variation in results. In view of this it is very essential to fix a limit of the sample size for realistic quick estimates of losses.

4. **Setting of bench marks for yearly reduction of T&D losses (technical and non-technical)**

3.15 References

1. Power Ministers Conference February 2000. – Agenda Notes
2. Electric Power International – Fall, 1999.
3. Guidelines for Energy Audit in Power Systems, CEA, New Delhi May 1992
4. P. N Khare MSEB, Power Theft – A Root Cause of T& D Losses
5. Mr.M.H.P. Rao 'Power Sector bogged down by T&D Losses' in Financial Express,dated July 9 ,1999

CHAPTER-4

APPLICATION OF GENETIC ALGORITHM FOR LOSS REDUCTION IN DISTRIBUTION SYSTEMS

4.1 Introduction

Distribution Systems are the networks that transport the electric energy from bulk substations or sources to many services or loads. In most cases distribution system is radially structured because it has some advantages over meshed network , such that lower short circuit and simpler switching and protecting equipment . On the other hand, it provides lower reliability. Generally, network reconfiguration is needed to provide service to as many consumers as possible following fault condition, or during planned outages for maintenance purposes, reduce system losses and balance the loads to avoid overloading of network elements. During normal operating conditions, networks are reconfigured for two purposes:

(a) Loss reduction to reduce overall system power loss

(b) load balancing to relieve network overloads.

Many techniques have been proposed for solving feeder reconfiguration problem through switching operation. For example, Goswami et al. [1] presented a heuristic algorithm utilizing the concept of optimal flow pattern for the minimum loss configuration of distribution feeders. Jin-Cheng et al. [2] proposed a solution algorithm ,based on a loss reduction formula and a line flow updating formula for the network reconfiguration problem. In [3], the developed algorithm is based on partitioning the distribution network into groups of load buses, such that the line section losses between the groups of nodes are minimized. M.S. Kandil et al. [4] presented an approach based on heuristic search strategies to determine the switching actions for minimum loss configuration and/or transformers load balancing. The authors of [5] proposed a network reconfiguration algorithm based on branch exchange for load balancing. S.I.Mohamed et al. [6] used artificial neural network (ANN) to reconfigure the feeder that reduces the active power losses.

Feeder reconfiguration through switching operation is a complicated combinatorial optimization problem. Genetic algorithms have recently been used to solve many difficult engineering problems and are particularly effective for combinatorial optimization problems with large and complex search spaces. In this paper, a G.A is presented for multi-objective programming to solve the reconfiguration problem. Five objectives are considered in conjunction with network constraints.

The G.A is basically a stochastic searching algorithm. It is capable of solving non-smooth, non-continuous and non-differentiable problems for parallel computation to find global or near global optimal solutions. The results of the case studies demonstrate the effectiveness of the solution algorithm and proved that the G.A is suitable to solve this kind of problems.

4.2 Introduction-Load Flow Analysis

Load flow (or power flow) analysis is the determination of current , voltage, active power and reactive voltamperes at various points in the power system operating under normal steady state or static conditions. Load flow studies are made to plan the best operation and control of the existing system as well as to plan the future expansion to keep pace with the load growth .Such studies help in ascertain the effects of new loads , new generating stations, new lines and new interconnections before they are installed. The prior information serves to minimize the system losses and to provide a check on the system stability.

The mathematical formulation of load flow problem results in a set of algebraic non-linear equations . A lot of calculation work is involved in the solution of these equation of these equations.Hand computations are very tedious and time consuming. Now a days digital computers, because of greater flexibility, economy accuracy and quiker operation, have practically replaced network analysers for the solution of load flow problems.

4.3 Problem Formulation

Distribution feeders contain a number of switches that are normally closed and others that are normally open. Under normal operating conditions, **distribution** engineers periodically reconfigure **distribution** feeders by opening and closing of switches in order to increase networks reliability and/or reduce line losses. In this section, the feeder reconfiguration problem is formulated as a multi-objective optimization problem, which can be solved efficiently using load flow studies and G.A [7].

Objective Functions

(a) Minimize the Total Power Loss(in lines) in the sample system:

Min

$$TP_{\ell oss} = \sum_{\substack{p,q=1 \\ p \neq q}}^{n} \{(P_{pq}-jQ_{pq})+(P_{qp}-jQ_{qp})\}$$

where:

n is the number of buses, pq depict the line connecting bus p& q.
P is the real power flow & Q is the reactive power flow.

4.4 Solution Algorithm For Feeder Reconfiguration

The selection of an optimum configuration among discrete numerous switching options requires solution of a complicated combinatorial optimization problem. Load flow studies along with G.A have recently proved as an effective tool for solving this type of problems with large and complex search spaces. The search of any G.A starts with a random generation of a population of strings. Each string is divided into a number of sub strings equals the number of the problem variables. Each sub string consists of a number of genes to present one of the variables in a certain coding system. Fig.(1) depicts the flow chart of the proposed G.A approach.

4.5 Application

To show the validity, and efficiency of the load flow studies along with proposed G.A, it is tested on the **distribution** system shown in Fig.(2). This system includes Two generators, five buses including one slack bus ,7 branches and 14 switches. The system data are

illustrated in tables (1) and (2). The Gauss-Seidal method is used using Y_{BUS}, with acceleration factor of 1.4 and 1.4 and tolerances of 0.0001 and 0.0001 per unit for the real and imaginary components of voltage.

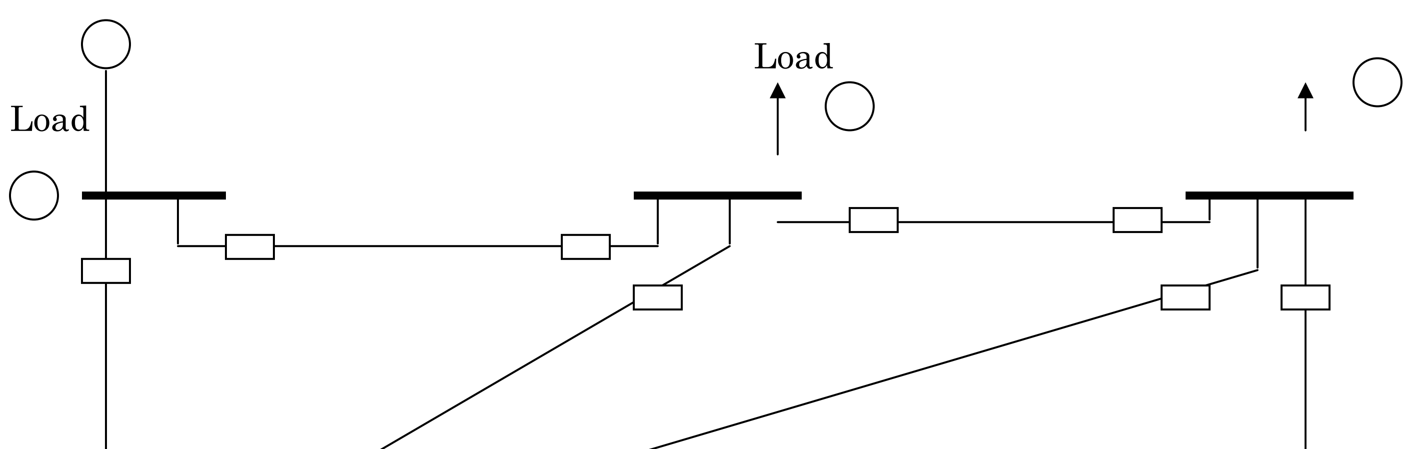


Fig.(2) Sample system (Network)

4.6 Flow Chart of the Gauss-Seidal iterative method

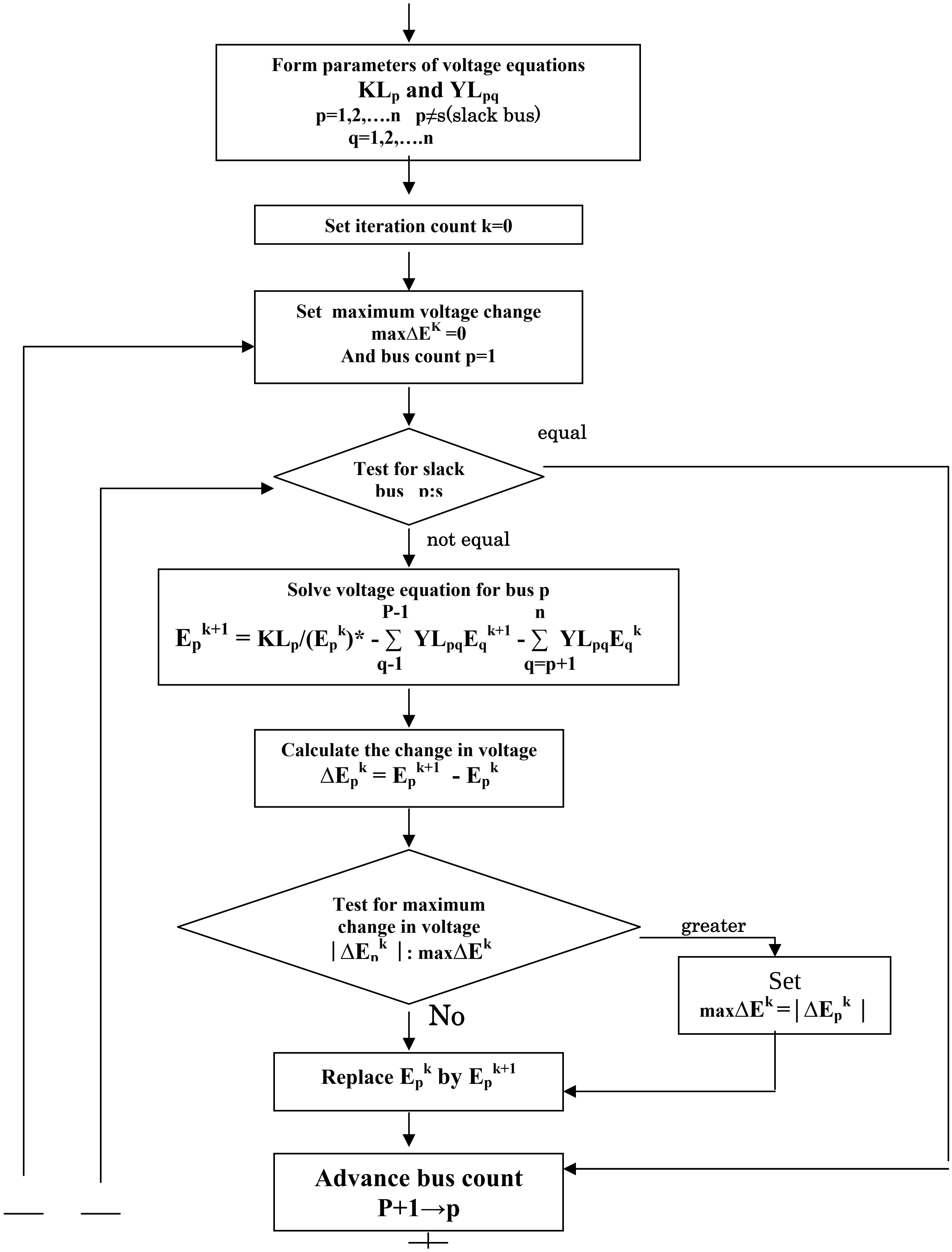
START

↓

Form the admittance matrix Y_{BUS}

↓

Assume bus voltages $E_P^{(0)}$
Where p=1,2,….n
p≠s(slack bus)


Form parameters of voltage equations
KL$_p$ and YL$_{pq}$
p=1,2,….n p≠s(slack bus)
q=1,2,….n
Set iteration count k=0
Set maximum voltage change
maxΔE^K =0
And bus count p=1
Test for slack
bus p:s
equal
not equal
Solve voltage equation for bus p
$E_p^{k+1} = KL_p/(E_p^k)^* - \sum_{q-1}^{P-1} YL_{pq}E_q^{k+1} - \sum_{q=p+1}^{n} YL_{pq}E_q^k$
Calculate the change in voltage
$\Delta E_p^k = E_p^{k+1} - E_p^k$
Test for maximum
change in voltage
$|\Delta E_p^k| : \max\Delta E^k$
greater
Set
$\max\Delta E^k = |\Delta E_p^k|$
No
Replace E_p^k by E_p^{k+1}
Advance bus count
P+1→p

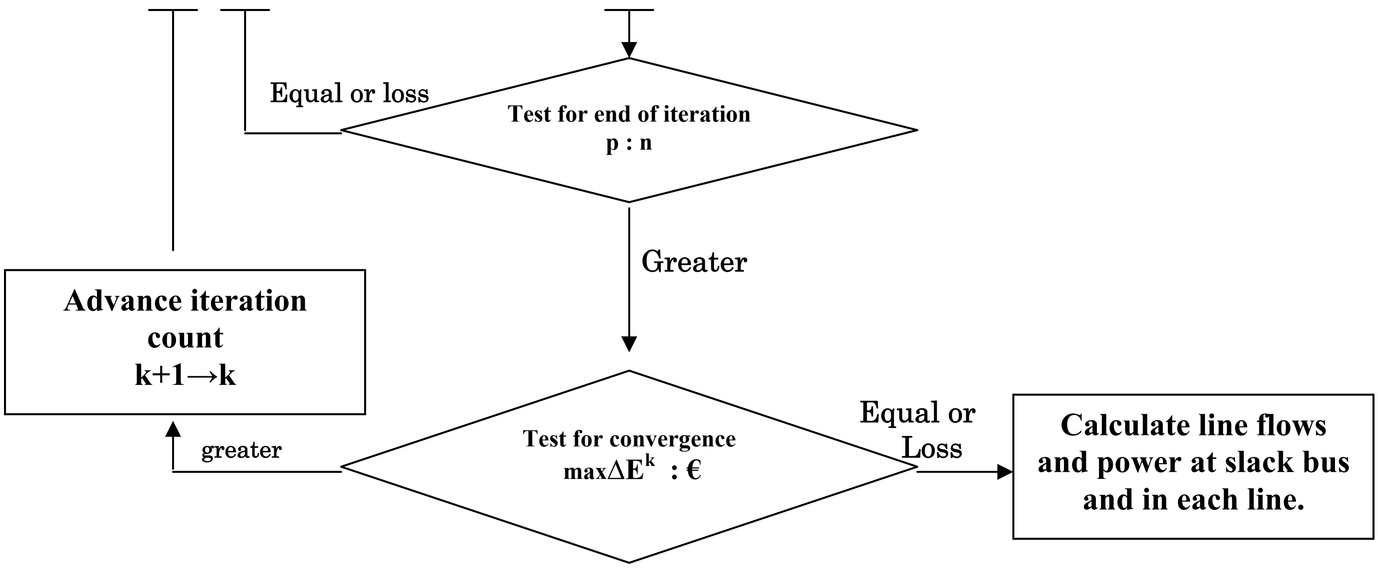


Table-1

(Impedances and line charging for sample system)

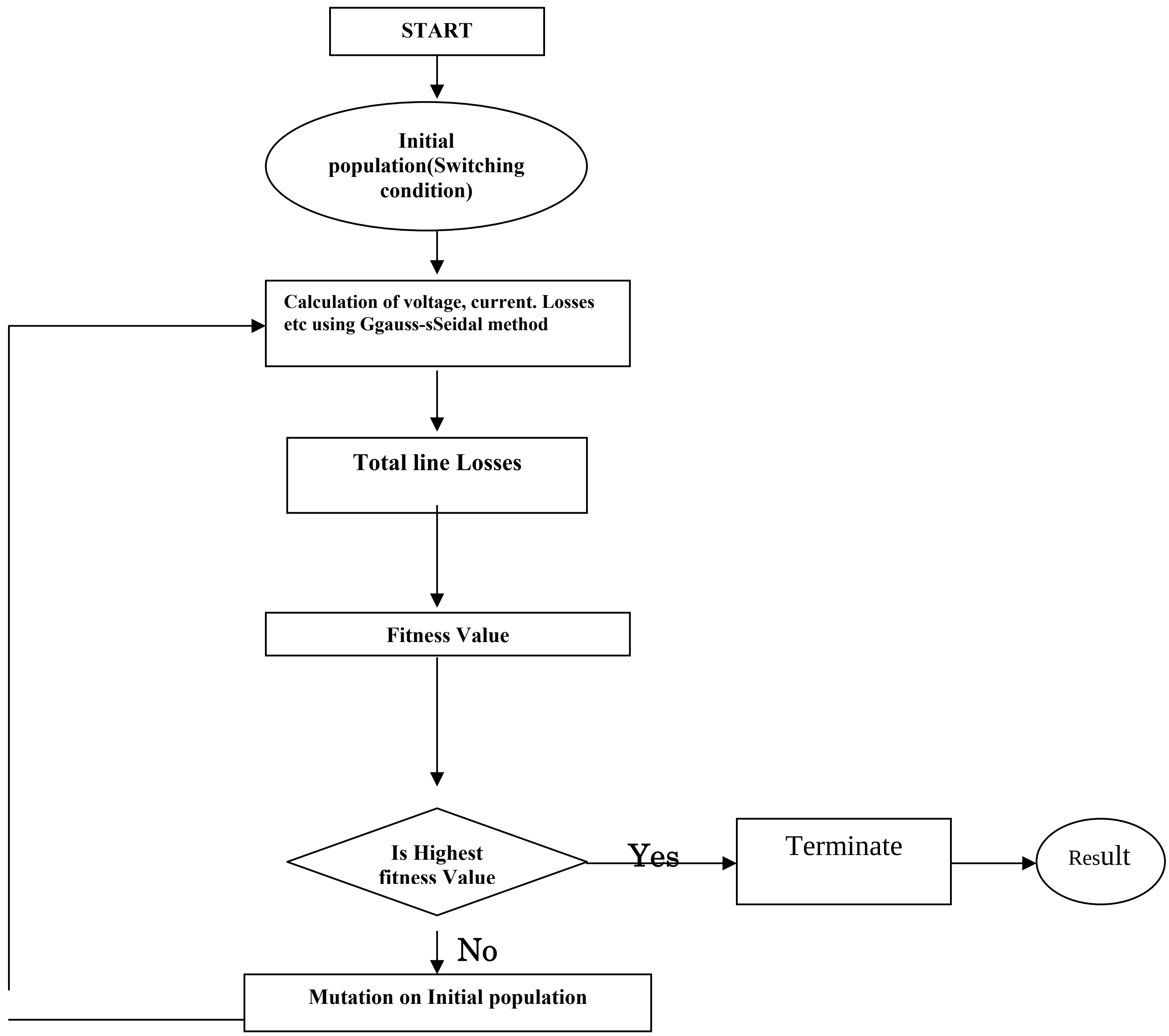
Bus Code(p-q)	Impedance(Z_{pq})	Line Charging($y'_{pq}/2$)
1-2	0.02+j0.06	0.0+j0.030
1-3	0.08+j0.24	0.0+j0.025
2-3	0.06+j0.18	0.0+j0.020
2-4	0.06+j0.18	0.0+j0.020
2-5	0.04+j0.12	0.0+j0.015
3-4	0.01+j0.03	0.0+j0.010
3-5	0.08+j0.24	0.0+j0.025

Table(2)

(Scheduled generation and loads and assumed bus voltages for sample system)

Bus Code(p)	Assumed Bus Voltage	Generation		Load	
		P(Kw)	Q(Mvar)	P(Kw)	Q(Mvar)
1	1.06+j0.0	0	0	0	0
2	1.0+j0.0	40	30	20	10
3	1.0+j0.0	0	0	45	15
4	1.0+j0.0	0	0	40	5
5	1.0+j0.0	0	0	60	10

4.7 Flow Chart of the proposed genetic algorithm application



** No cross over because switch is either in the condition of ON or OFF.

4.8 RESULT

The selection of a particular system has numerous losses in the lines , when all of them are connected. It is being thought that, the lines connected in the system, somehow if withdrawn by using some technique(G.A), then the effect of that on the losses, bus voltages, solution time is compared. The table-(3) shows all the possible combinations and the losses in the line. It shows that the fitness value= $1/[\text{error}]^2$ is 22.12 in iteration No.0 , when all the lines are connected .

The fitness value is maximum i.e 28.94743 in the iteration No.25 , when the bus lines connecting 3-4, 4-5 are removed. Though the losses have been reduced but only consideration is to have line loading with in limits.

4.9 Conclusion

A load flow studies along with Genetic algorithm approach has been presented to solve the above problem of reducing line losses. Numerical results of two generators, five buses including one slack bus ,7 branches and 14 switches distribution system showed the efficiency and capability of load flow studies along with Genetic algorithm in solving this type of problem.The algorithm can be directed easily by the experience of the operator to minimize the total active power losses in bus lines.

4.10 References ：

[1] S.K.Goswami and S.K.Basu, "A New Algorithm For the Reconfiguration of Distribution Feeders for Loss Minimization", IEEE Trans.

[2] Jin – Cheng Wang, Hsiao-Dong Chiang and Gary R.Darling, "An Efficient Algorithm For Real-Time Network Reconfiguration in Large Scale Unbalanced Distribution Systems", IEEE Trans.

[3] R.J.Sarfi, M.M.A.Salama and A.Y.Chikhani, "Distribution System Reconfiguration For Loss Reduction: An Algorithm Based on Network Partitioning Theory" , IEEE Trans.

[4] M.S.Kandil , A.M.Riad , A.El-Hosseiny and S.M. Abou Zaid, " Artificial Intelligence Approach For Loss Reduction and Load Balancing in Distribution Systems".

[5] M.A.Kashem , V.Ganapathy and G.B.Jasmon , "Network Reconfiguration For Load Balancing in Distribution Networks" .

[6] S.L.Mohamed ,E.A.Mohamed and A.R.Abu El Wafa, " Distribution Network Reconfiguration For Power Loss Reduction Using Artificial Neural Network".

[7] Tsai-Hsiang Chen and Jeng-Tyan Cherng, "Optimal Phase Arrangement of Distribution Transformers connected to a Primary Feeder For System Unbalance Improvement and Loss Reduction Using a Genetic Algorithm ", IEEE Trans.

[8] Mesut E.Baran and Felix F.Wu, "Network Reconfiguration In Distribution Systems For Loss Reduction and Load Balancing", IEEE Trans.

[10] Referance for the load flow studies from the Book □ Power System Engineering□ by Nagrath.I.J& Kothari.D.P.

[11] Referance for the load flow studies from the Book □ Electrical Power System□ by Wadhwa.C.L.

[12] Referance for the load flow studies from the Book □ Computer Methods in Power System Analysis□ by Stagg.G.W & Ahmed H.El-Abiad.

[13] Referance for the load flow studies from the Book □ Electrical Power System□ by Ashfaq Husain.

[14] Referance for the load flow studies from the Book Power System Analysis by J.J.Grainger & W.D.Stevenson.Jr.

FUTURE SCOPE OF THE WORK DONE

Many techniques have been proposed for solving feeder reconfiguration problem through switching operation. Genetic algorithm have recently been used to solve many different engineering problems with large and complex search spaces and hence can be presented for multi-objective programming to solve the feeders reconfiguration problem . The G.A is basically a stochastic searching algorithm. It is capable of solving non-smooth, non-continuous and non-differentiable problems for parallel computation to find global or near global optimal solutions.

In this work load flow studies along with genetic algorithm is applied on five bus radial distribution network for feeder reconfiguration for reducing the line losses and it can be extended for large number of busses.

The genetic algorithm can further be applied to minimize the total active power losses and at the same time improving or minimizing total complex power, average voltage drop, neutral current of the transformer and total voltage unbalance factor which are a major sign of better power quality.

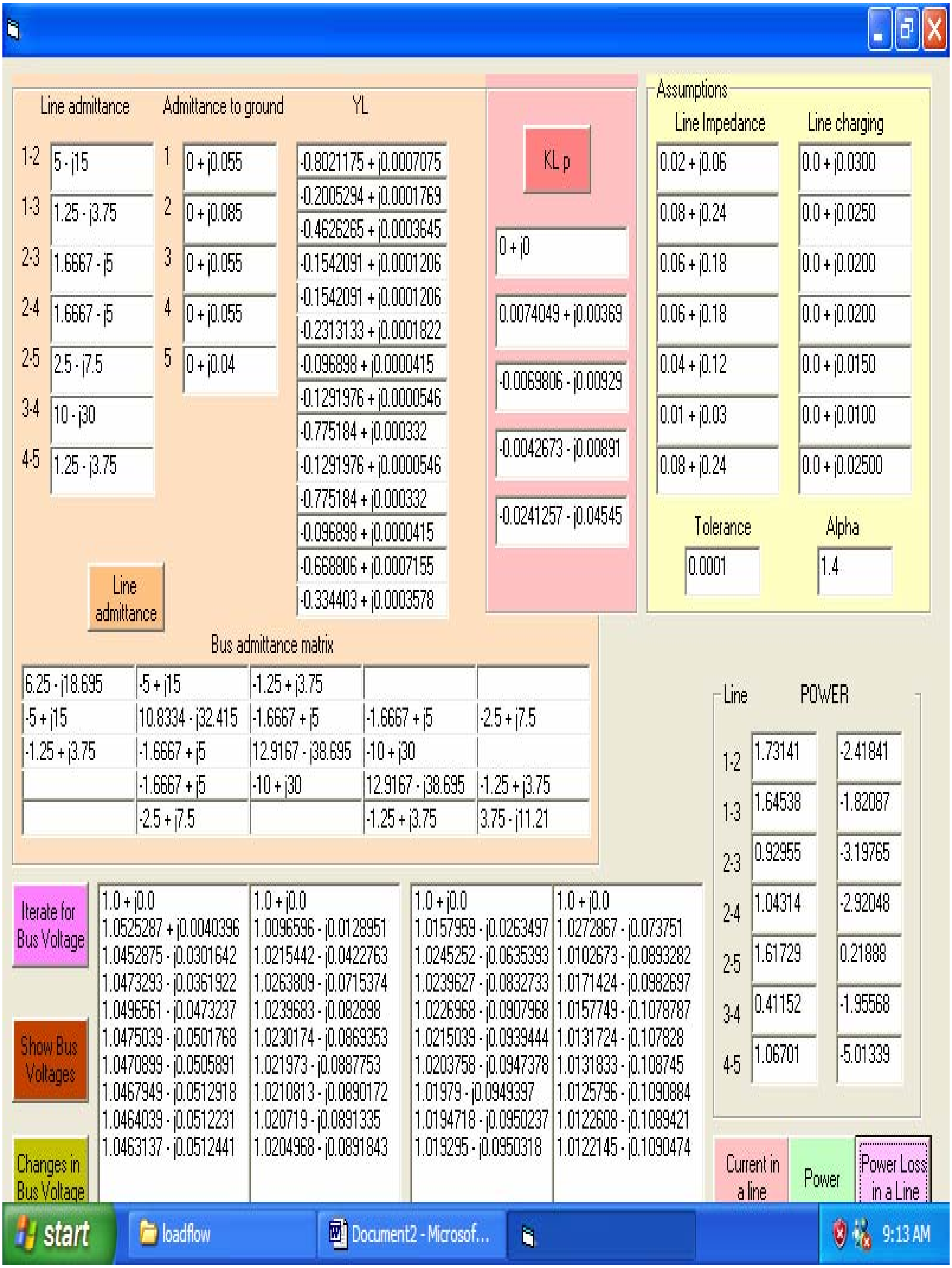
Table (3) – Value of the Fitness function under different switching conditions.

Iteration No.	Bus Code No. Out	Switching Condition (Population condition)	Total Line Loss(MW)	Error = (Losses/generation)	F(x) = $[error]^2$	Fitness function =1/f(x)
0	**Nil**	**11 11 11 11 11 11 11**	**8.4453**	**0.2111325**	**0.04457**	**22.4331**
1	1-2	00 11 11 11 11 11 11	24.24159	0.60603	0.3672	2.7226
2	1-3	11 00 11 11 11 11 11	11.49377	0.287344	0.08256	12.1114
3	**2-3**	**11 11 00 11 11 11 11**	**8.66023**	**0.21650**	**0.046874**	**21.3334**
4	**2-4**	**11 11 11 00 11 11 11**	**8.87871**	**0.221967**	**0.04926**	**20.2964**
5	2-5	11 11 11 11 00 11 11	14.67087	0.36677	0.134521	7.4337
6	3-4	11 11 11 11 11 00 11	8.64845	0.216211	0.046747	21.3916
7	**4-5**	**11 11 11 11 11 11 00**	**7.45505**	**0.18367625**	**0.034736**	**28.788848**
8	1-2 & 2-3	00 11 00 11 11 11 11	23.42726	0.5856815	0.343022	2.9152
9	1-2 & 2-4	00 11 11 00 11 11 11	23.04726	0.57618	0.33198	3.01218
10	1-2 & 2-5	00 11 11 11 00 11 11	29.06147	0.726536	0.527855	1.8944
11	1-2 & 3-4	00 11 11 11 11 00 11	35.75294	0.89382	0.7989	1.25168
12	1-2 & 4-5	00 11 11 11 11 11 00	23.4573	0.5864325	0.534390	2.9077
13	1-3 & 2-3	11 00 00 11 11 11 11	13.7196	0.34299	0.11764	8.5002
14	1-3 & 2-4	11 00 11 00 11 11 11	13.5102	0.337755	0.11407	8.7658
15	1-3 & 2-5	11 00 11 11 00 11 11	21.38612	0.53465	0.28585	3.4982
16	1-3 & 3-4	11 00 11 11 11 00 11	11.05464	0.276366	0.07637	13.0927
17	1-3 & 4-5	11 00 11 11 11 11 00	10.1760	0.2544	0.06471	15.4512
18	2-3 & 2-4	11 11 00 00 11 11 11	10.65631	0.26640	0.07097	14.0898
19	2-3 & 2-5	11 11 00 11 00 11 11	17.70669	0.44266	0.19595	5.1032
20	**2-3 & 3-4**	**11 11 00 11 11 00 11**	**8.47305**	**0.21182**	**0.044870**	**22.28642**
21	**2-3 & 4-5**	**11 11 00 11 11 11 00**	**7.59496**	**0.1898**	**0.036052**	**27.73760**
22	2-4 & 2-5	11 11 11 00 00 11 11	19.81572	0.495393	0.245414	4.0747
23	2-4 & 3-4	11 11 11 00 11 00 11	12.45372	0.31134	0.09693	10.3162
24	**2-4 & 4-5**	**11 11 11 00 11 11 00**	**7.78433**	**0.1946**	**0.037872**	**26.4044**
25	**3-4 & 4-5**	**11 11 11 11 1100 00**	**7.43495**	**0.18587**	**0.03454**	**28.94743**
26	1-2 , 2-3 & 2-4	00 11 00 00 11 11 11	23.06087	0.57652	0.332377	3.0086

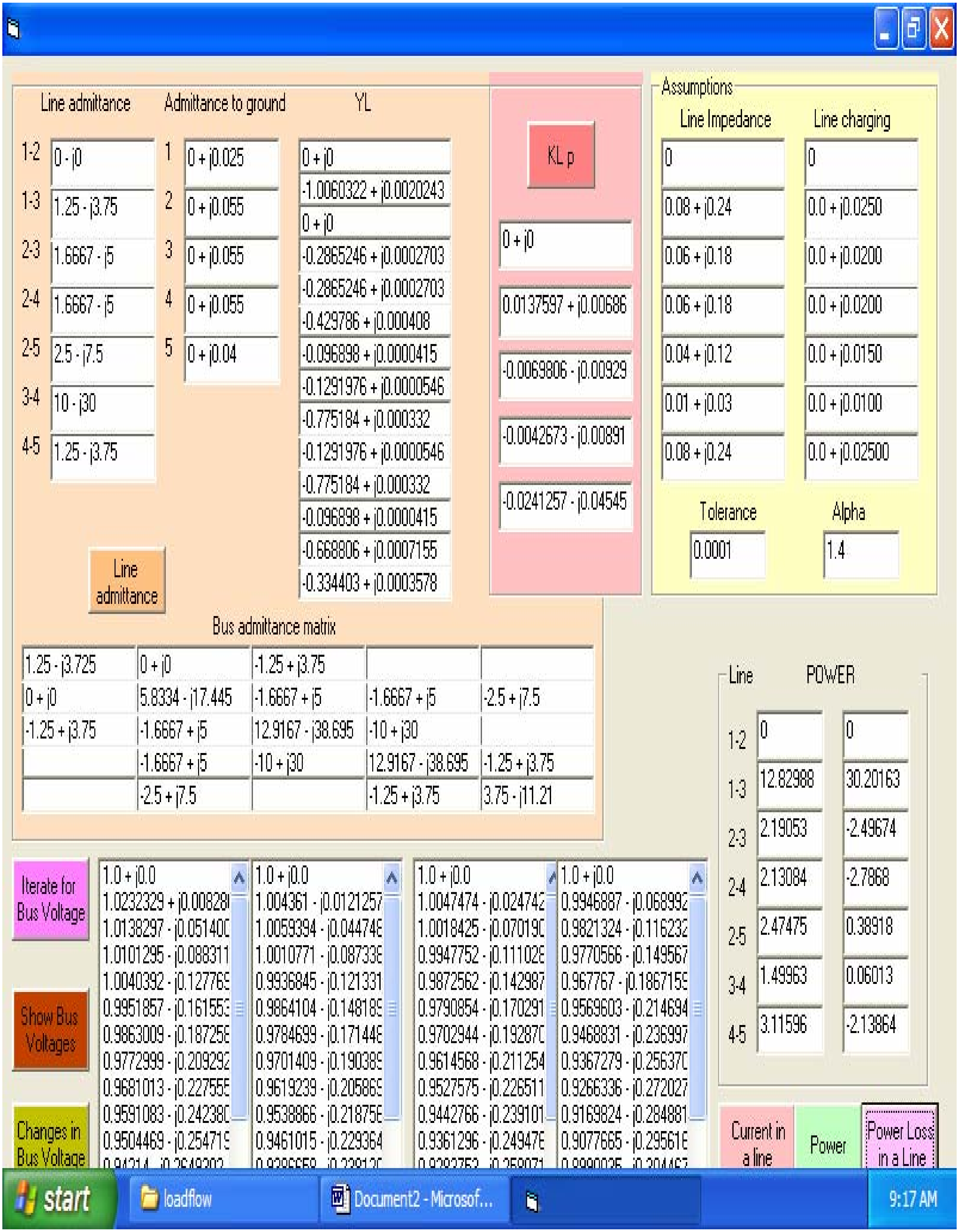
27	1-2 , 2-3 & 2-5	00 11 00 11 00 11 11	28.94309	0.723577	0.523564	1.90998
28	1-3, 3-4 & 2-5	11 00 11 11 00 00 11	25.58099	0.63952	0.40899	2.445036
29	1-2, 2-3 & 4-5	00 11 00 11 11 11 00	23.7110	0.5927	0.351383	2.8458

Iteration No.	**Bus Code No. Out**	**Switching Condition (Population condition)**	**Total Line Loss(MW)**	**Error = (Losses/generation)**	**F(x) = $[error]^2$**	**Fitness function =1/f(x)**
30	1-2, 2-4 & 2-5	00 11 11 00 00 11 11	30.13966	0.75349	0.567749	1.76134
31	1-2, 2-4 & 3-4	00 11 11 00 11 00 11	123.99255	3.0998	9.6088	0.104
32	1-2, 2-4 & 4-5	00 11 11 00 11 11 00	23.09618	0.57740	0.3333	2.9994
33	1-3, 2-3 & 2-4	11 00 00 00 11 11 11	24.5678	0.61419	0.37723	2.6508
34	1-3, 2-3 & 2-5	11 00 00 11 00 11 11	38.99766	0.9749	0.95051	1.0520
35	1-3, 2-3 & 4-5	11 00 00 11 11 11 00	12.3635	0.323408	0.10459	9.5608
36	1-3, 2-4 & 2-5	11 00 11 00 00 11 11	43.803	1.0950	1.1992	0.83387
37	1-3, 2-4 & 4-5	11 00 11 00 11 11 00	12.8254	0.32063	0.1028	9.7269
38	2-3, 2-4 & 2-5	11 11 00 00 00 11 11	53.682	1.3420	1.8011	0.55521
39	2-3, 2-4 & 3-4	11 11 00 00 11 00 11	12.24169	.30604	0.09366	10.6767
40	2-3, 2-4 & 4-5	11 11 00 00 11 11 00	10.70789	0.26769	0.07166	13.9544
41	1-3, 3-4 & 4-5	11 00 11 11 11 00 00	9.72739	0.24318	0.05913	16.909366
42	2-5 & 3-4	11 11 11 11 00 00 11	21.38728	0.5346	0.285884	3.4979119

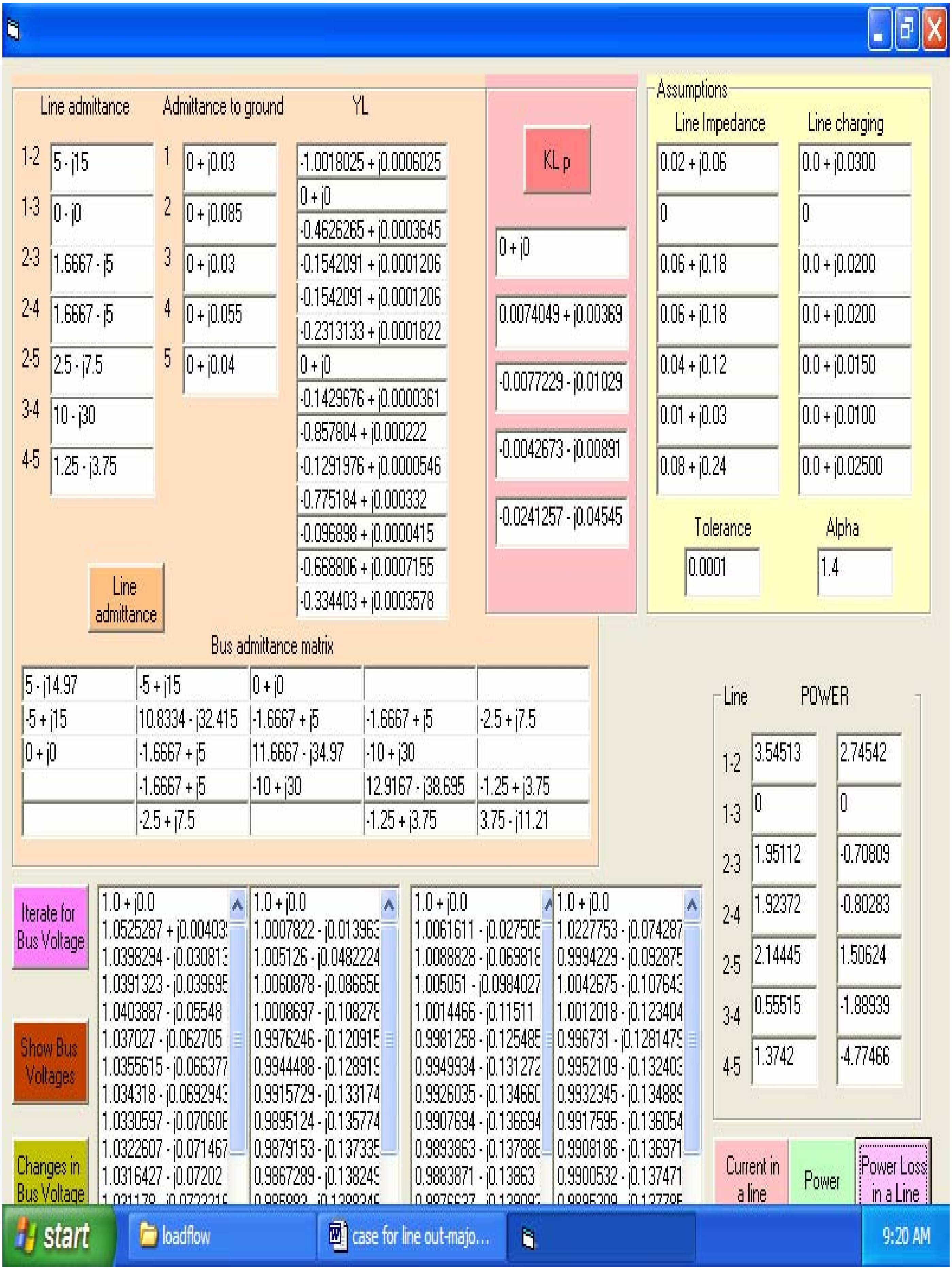
Case-0(When all Lines in)



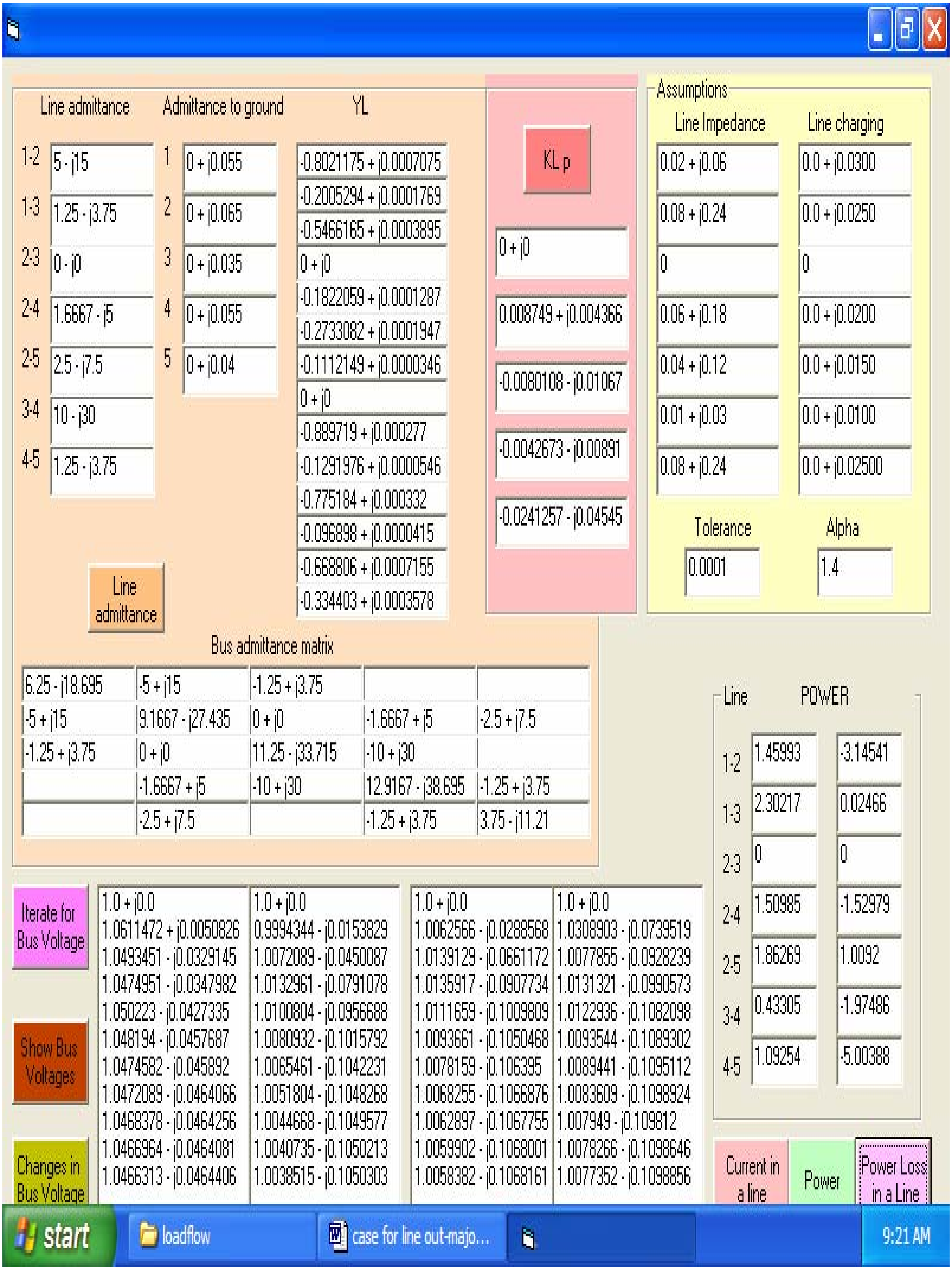
Case-1(When Line 1-2 is out)



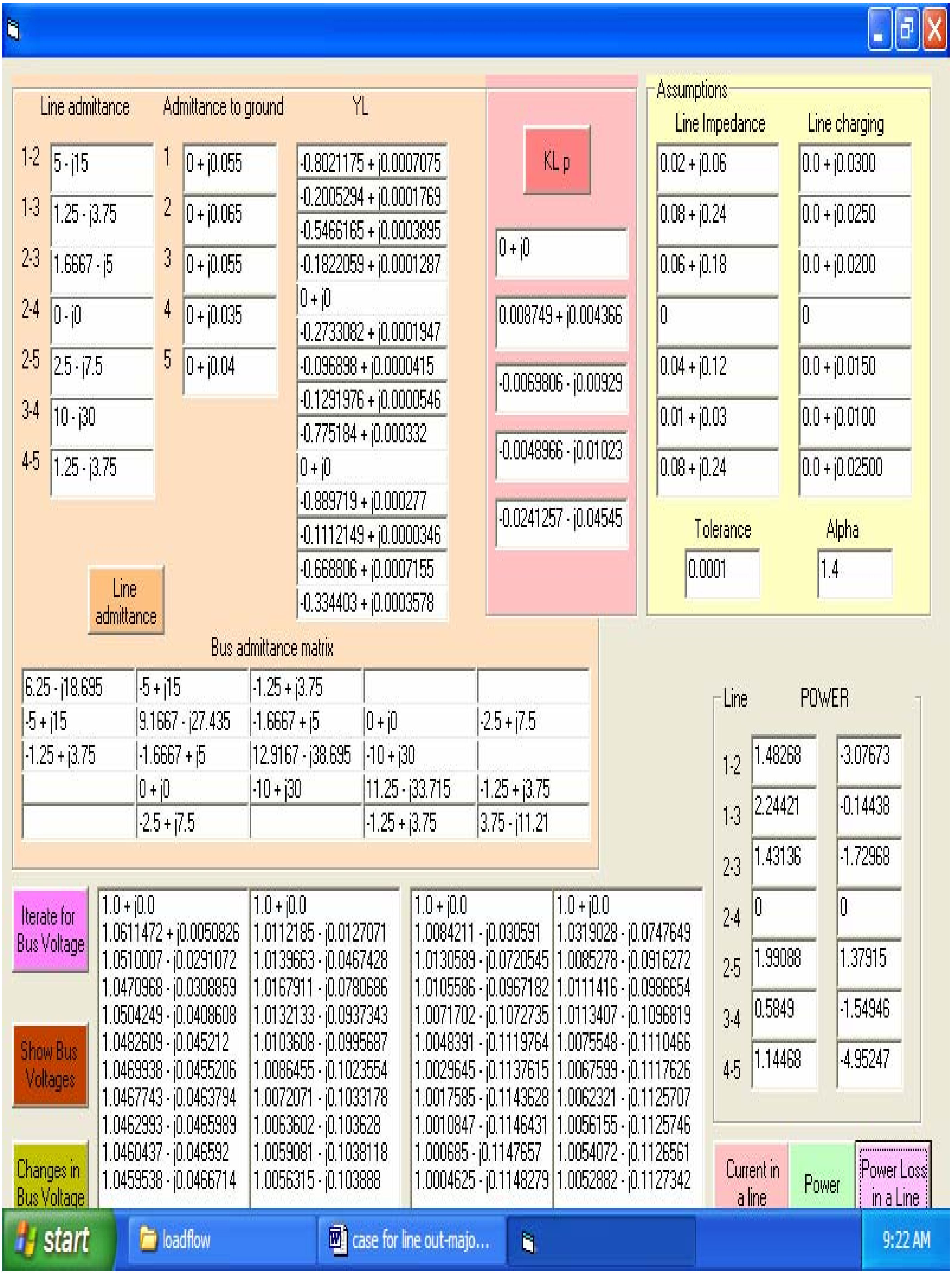
Case-2(When Line 1-3 is out)



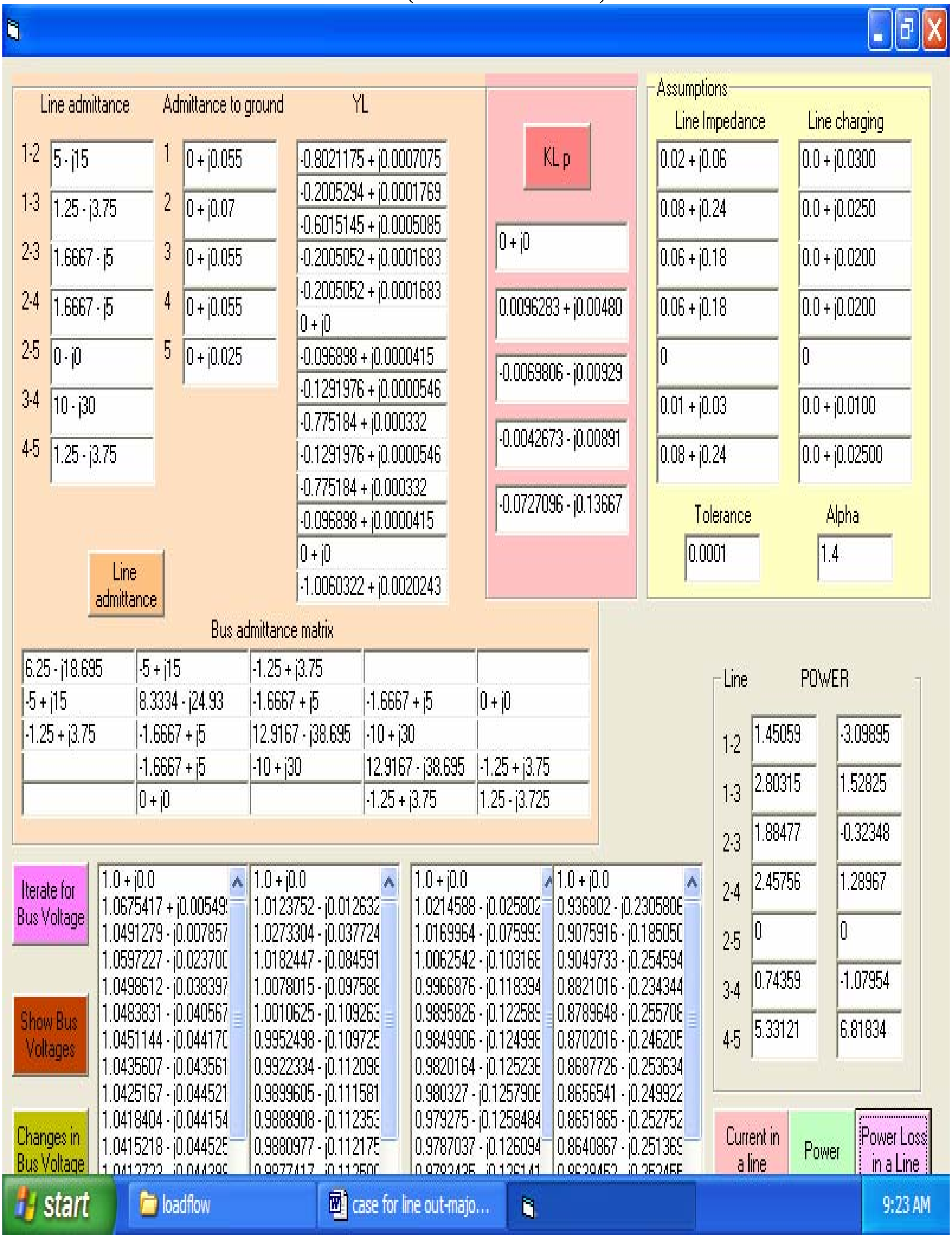
Case-3(When Line 2-3 is out)



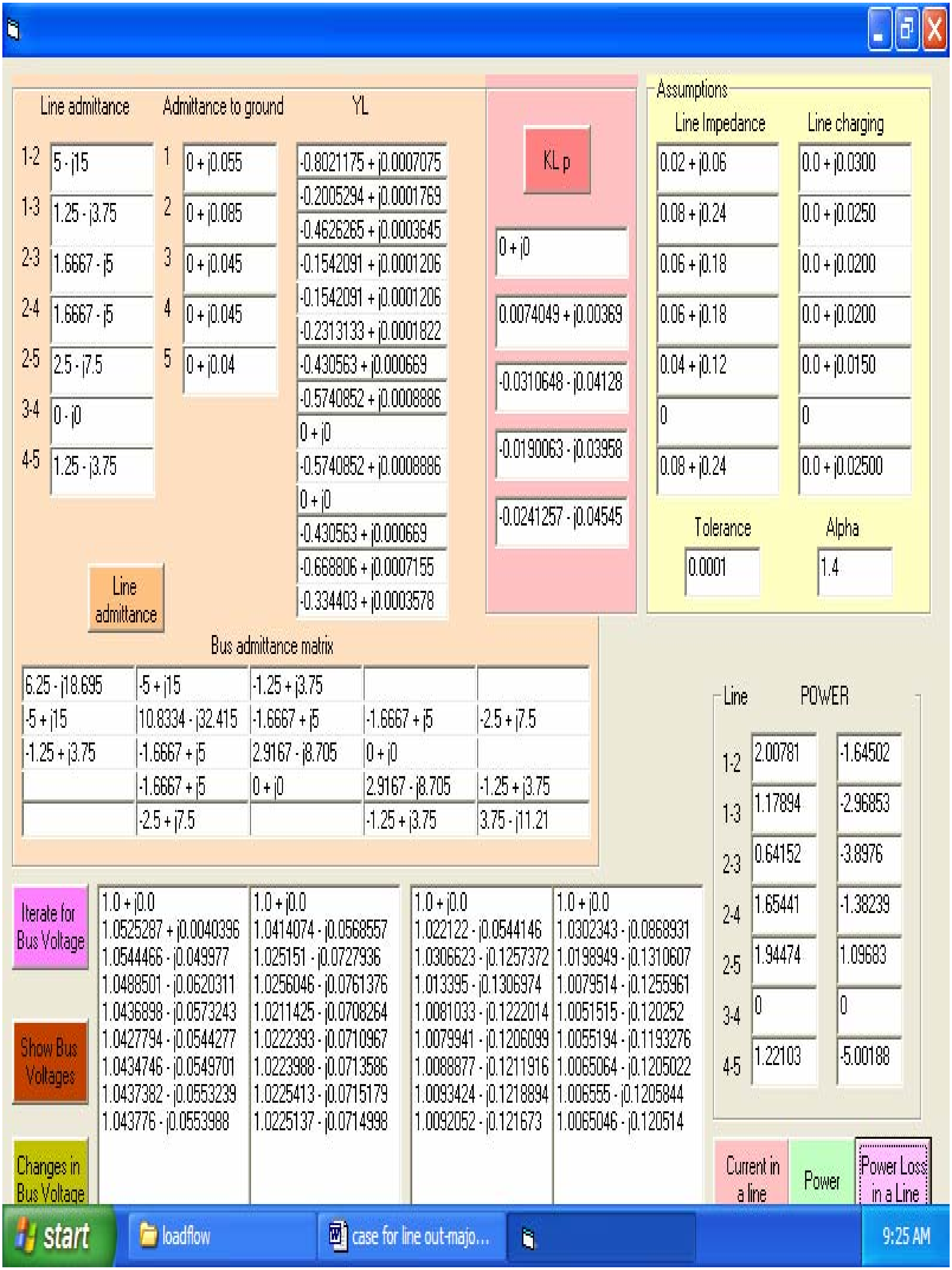
Case-4(When Line 2-4 is out)


Line admittance
Admittance to ground
YL
1-2 5 - j15
1-3 1.25 - j3.75
2-3 1.6667 - j5
2-4 0 - j0
2-5 2.5 - j7.5
3-4 10 - j30
4-5 1.25 - j3.75
1 0 + j0.055
2 0 + j0.065
3 0 + j0.055
4 0 + j0.035
5 0 + j0.04
-0.8021175 + j0.0007075
-0.2005294 + j0.0001769
-0.5466165 + j0.0003895
-0.1822059 + j0.0001287
0 + j0
-0.2733082 + j0.0001947
-0.096898 + j0.0000415
-0.1291976 + j0.0000546
-0.775184 + j0.000332
0 + j0
-0.889719 + j0.000277
-0.1112149 + j0.0000346
-0.668806 + j0.0007155
-0.334403 + j0.0003578
Line admittance
KL p
0 + j0
0.008749 + j0.004366
-0.0069806 - j0.00929
-0.0048966 - j0.01023
-0.0241257 - j0.04545
Assumptions
Line Impedance
Line charging
0.02 + j0.06
0.0 + j0.0300
0.08 + j0.24
0.0 + j0.0250
0.06 + j0.18
0.0 + j0.0200
0
0
0.04 + j0.12
0.0 + j0.0150
0.01 + j0.03
0.0 + j0.0100
0.08 + j0.24
0.0 + j0.02500
Tolerance
0.0001
Alpha
1.4
Bus admittance matrix
6.25 - j18.695 | -5 + j15 | -1.25 + j3.75 | |
-5 + j15 | 9.1667 - j27.435 | -1.6667 + j5 | 0 + j0 | -2.5 + j7.5
-1.25 + j3.75 | -1.6667 + j5 | 12.9167 - j38.695 | -10 + j30 |
| 0 + j0 | -10 + j30 | 11.25 - j33.715 | -1.25 + j3.75
| -2.5 + j7.5 | | -1.25 + j3.75 | 3.75 - j11.21
Line POWER
1-2 1.48268 -3.07673
1-3 2.24421 -0.14438
2-3 1.43136 -1.72968
2-4 0 0
2-5 1.99088 1.37915
3-4 0.5849 -1.54946
4-5 1.14468 -4.95247
Iterate for Bus Voltage
Show Bus Voltages
Changes in Bus Voltage
1.0 + j0.0
1.0611472 + j0.0050826
1.0510007 - j0.0291072
1.0470968 - j0.0308859
1.0504249 - j0.0408608
1.0482609 - j0.045212
1.0469938 - j0.0455206
1.0467743 - j0.0463794
1.0462993 - j0.0465989
1.0460437 - j0.046592
1.0459538 - j0.0466714
1.0 + j0.0
1.0112185 - j0.0127071
1.0139663 - j0.0467428
1.0167911 - j0.0780686
1.0132133 - j0.0937343
1.0103608 - j0.0995687
1.0086455 - j0.1023554
1.0072071 - j0.1033178
1.0063602 - j0.103628
1.0059081 - j0.1038118
1.0056315 - j0.103888
1.0 + j0.0
1.0084211 - j0.030591
1.0130589 - j0.0720545
1.0105586 - j0.0967182
1.0071702 - j0.1072735
1.0048391 - j0.1119764
1.0029645 - j0.1137615
1.0017585 - j0.1143628
1.0010847 - j0.1146431
1.000685 - j0.1147657
1.0004625 - j0.1148279
1.0 + j0.0
1.0319028 - j0.0747649
1.0085278 - j0.0916272
1.0111416 - j0.0986654
1.0113407 - j0.1096819
1.0075548 - j0.1110466
1.0067599 - j0.1117626
1.0062321 - j0.1125707
1.0056155 - j0.1125746
1.0054072 - j0.1126561
1.0052882 - j0.1127342
Current in a line
Power
Power Loss in a Line
start
loadflow
case for line out-majo...
9:22 AM

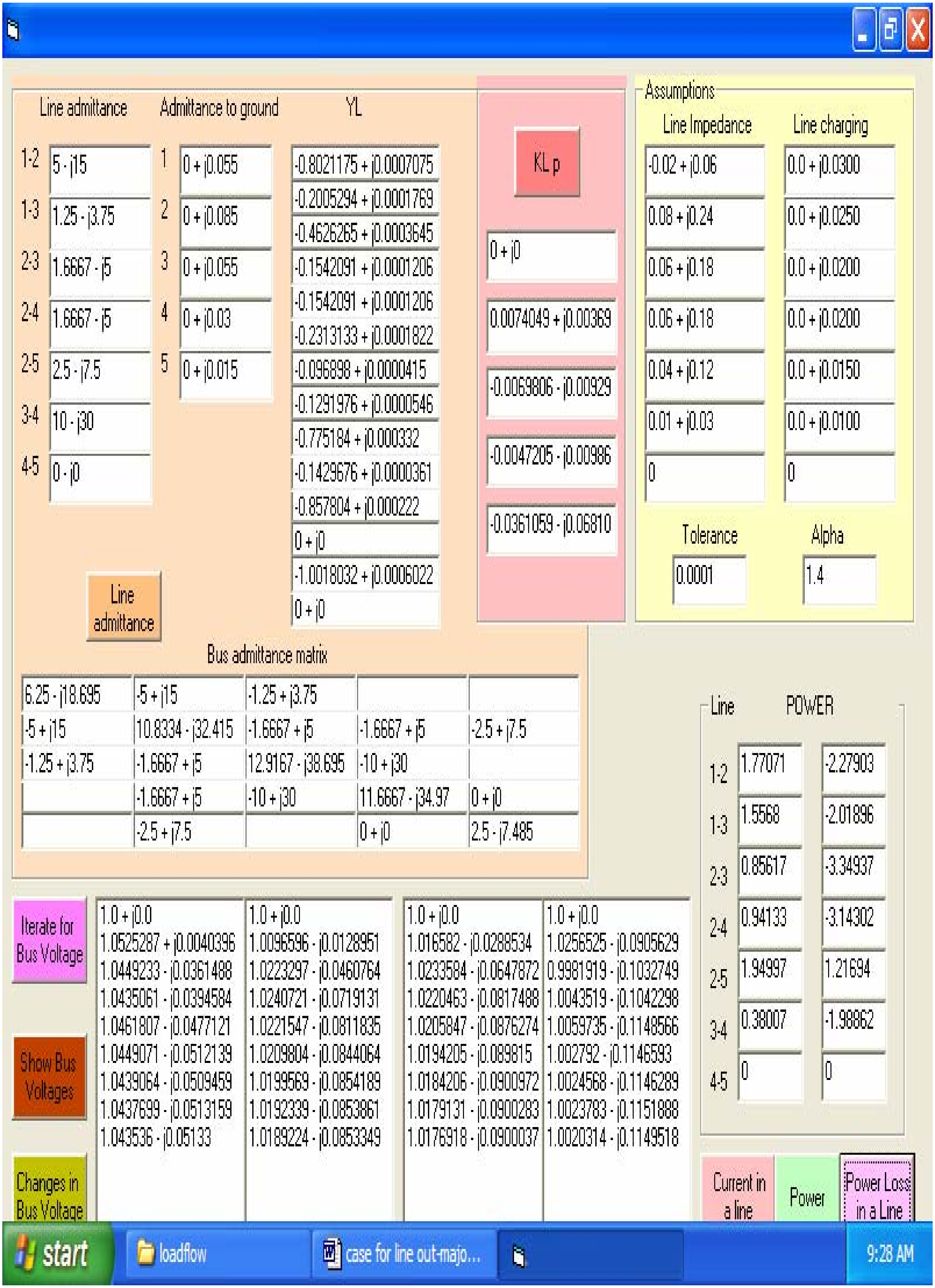
Case-5(When Line 2-5 is out)



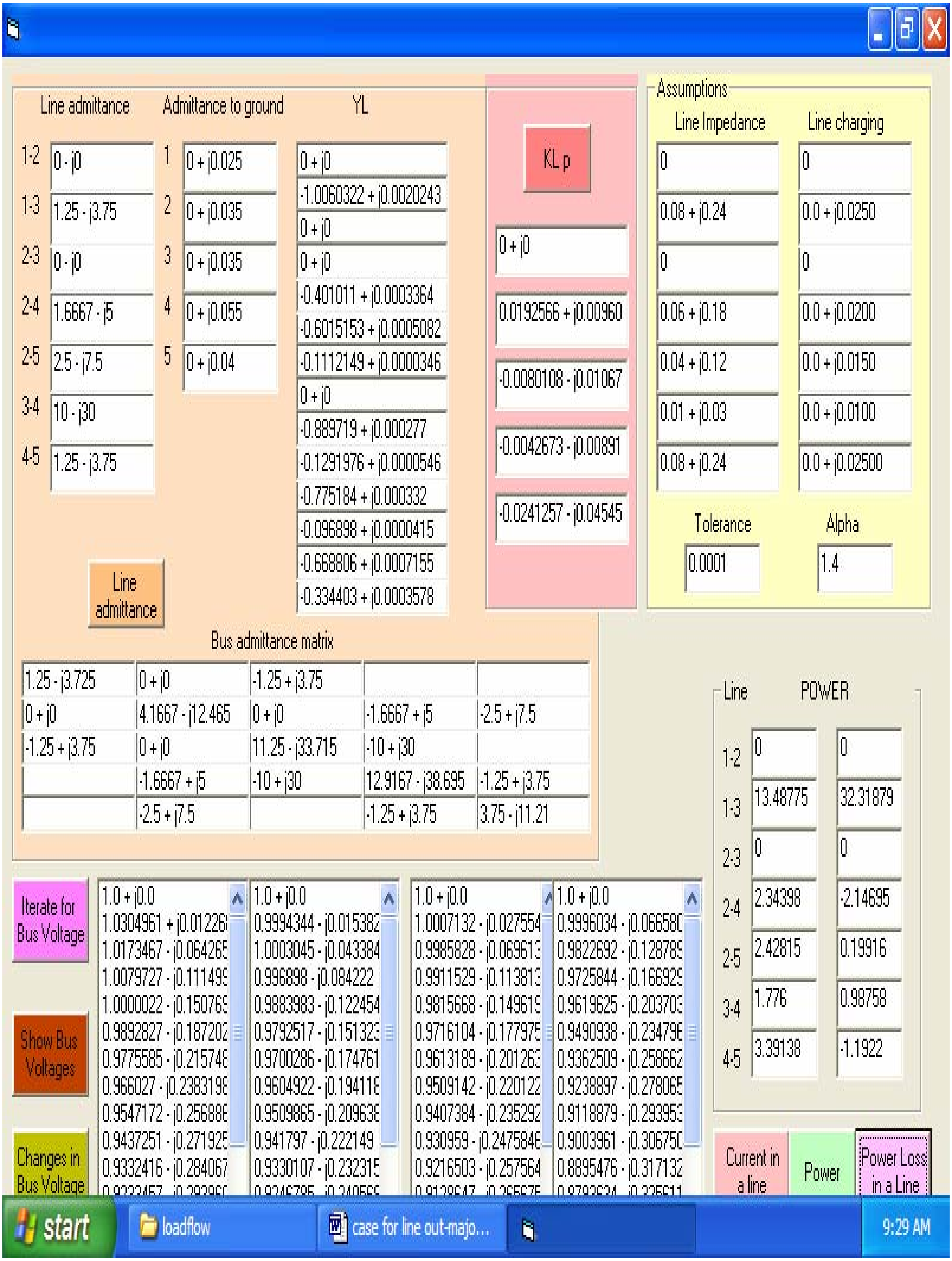
Case-6(When Line 3-4 is out)



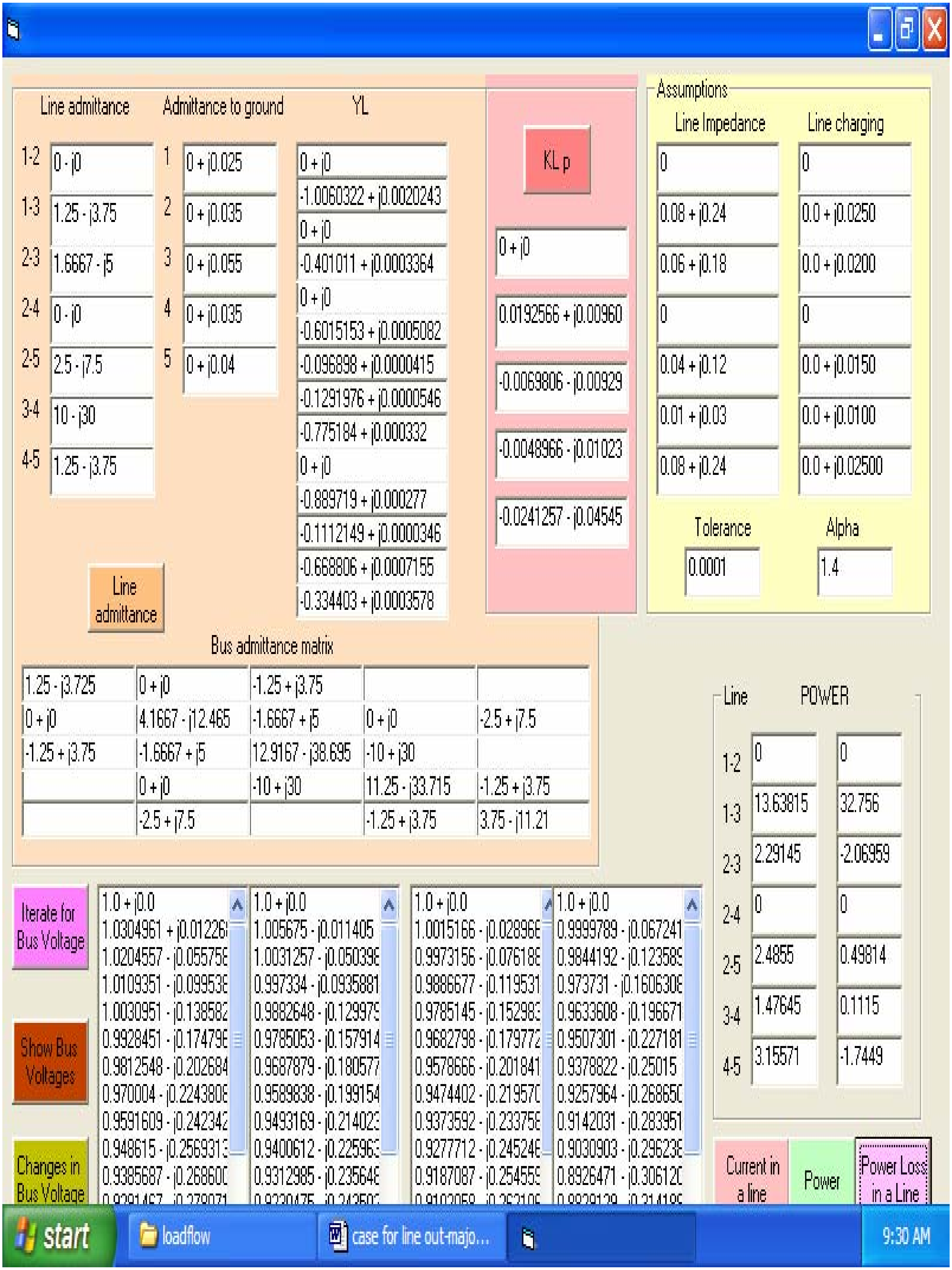
Case-7(When Line 4-5 is out)



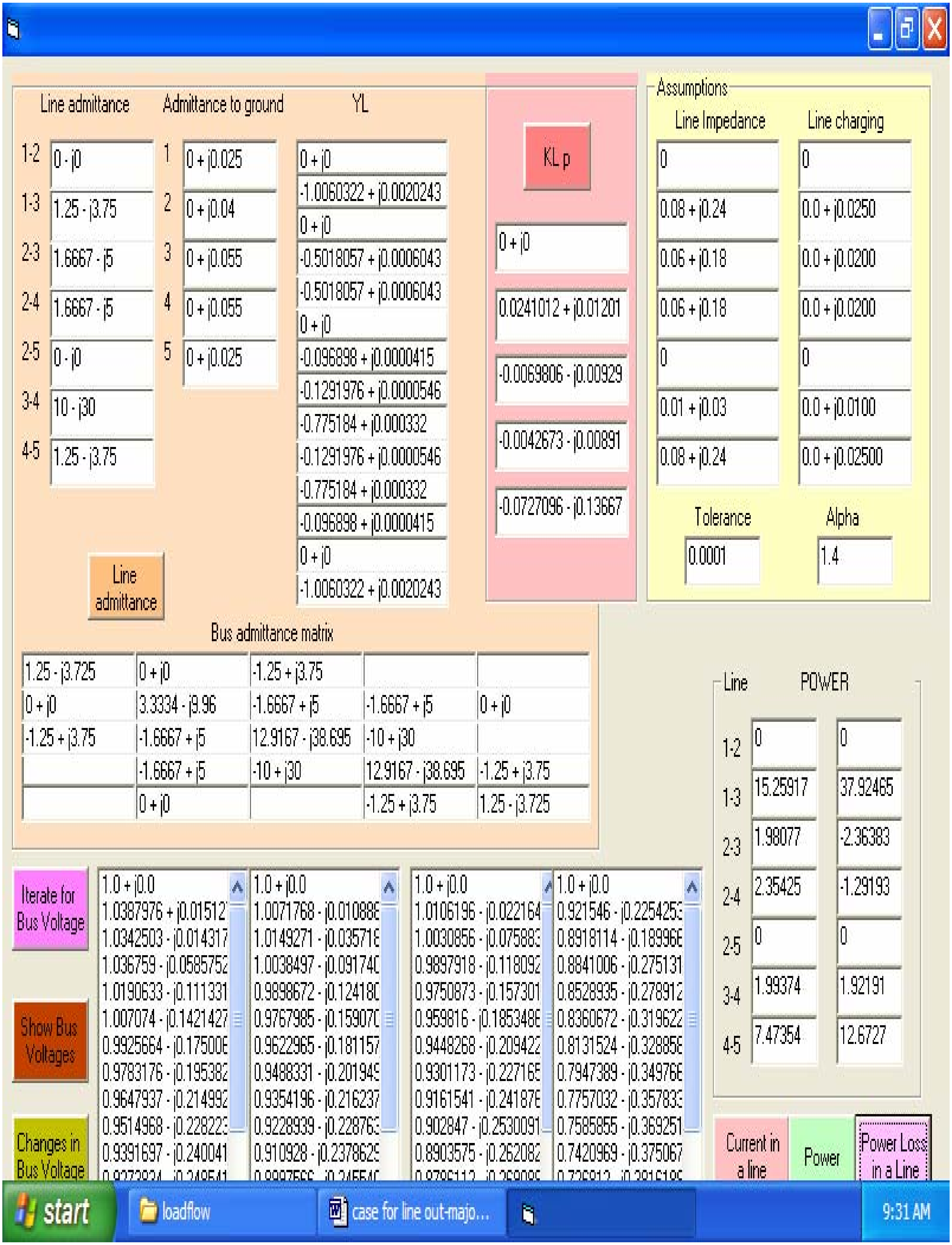
Case-8(When Lines 1-2 & 2-3 are out)



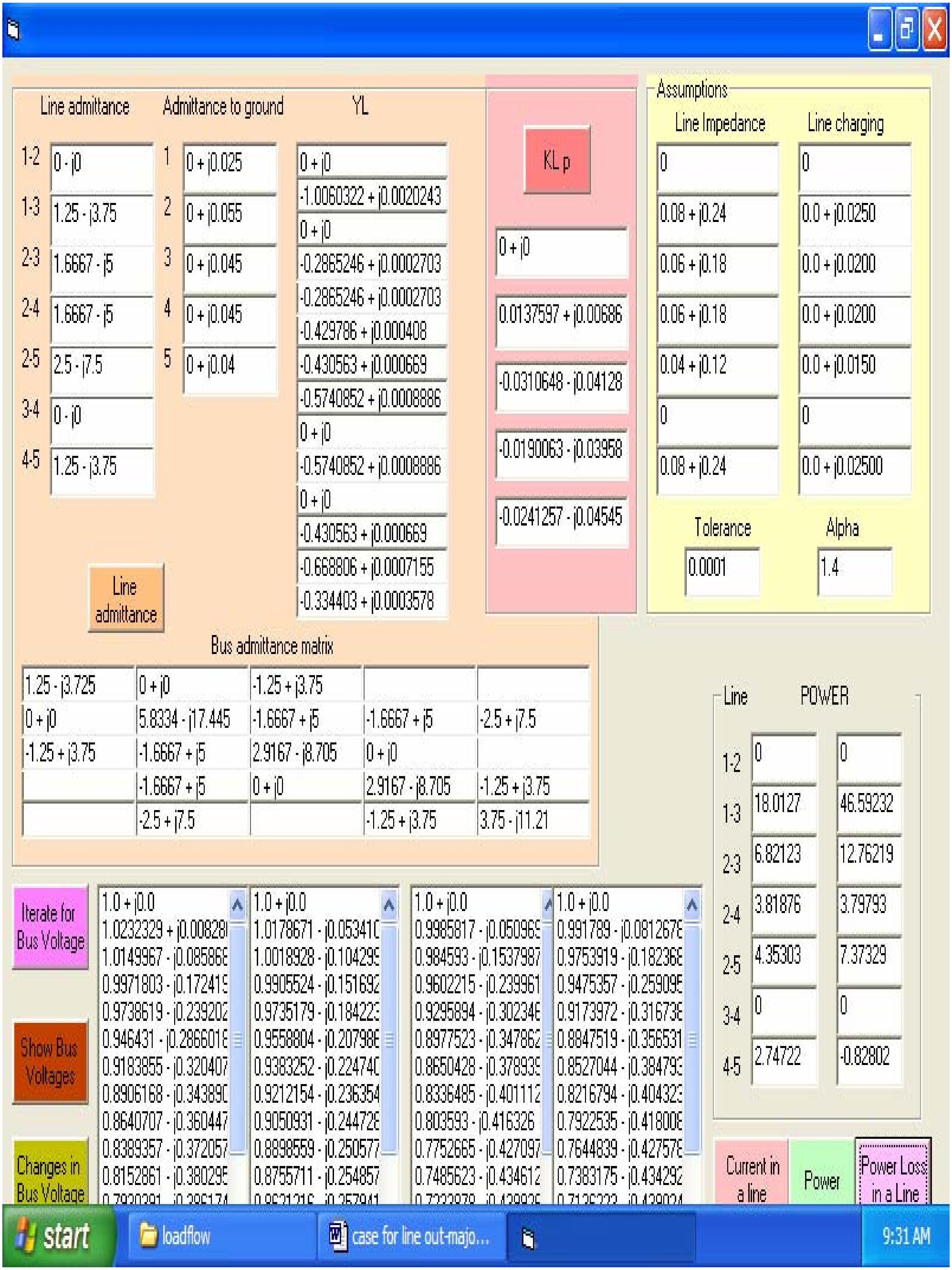
Case-9(When Lines 1-2 & 2-4 are out)



Case-10(When Lines 1-2 & 2-5 are out)

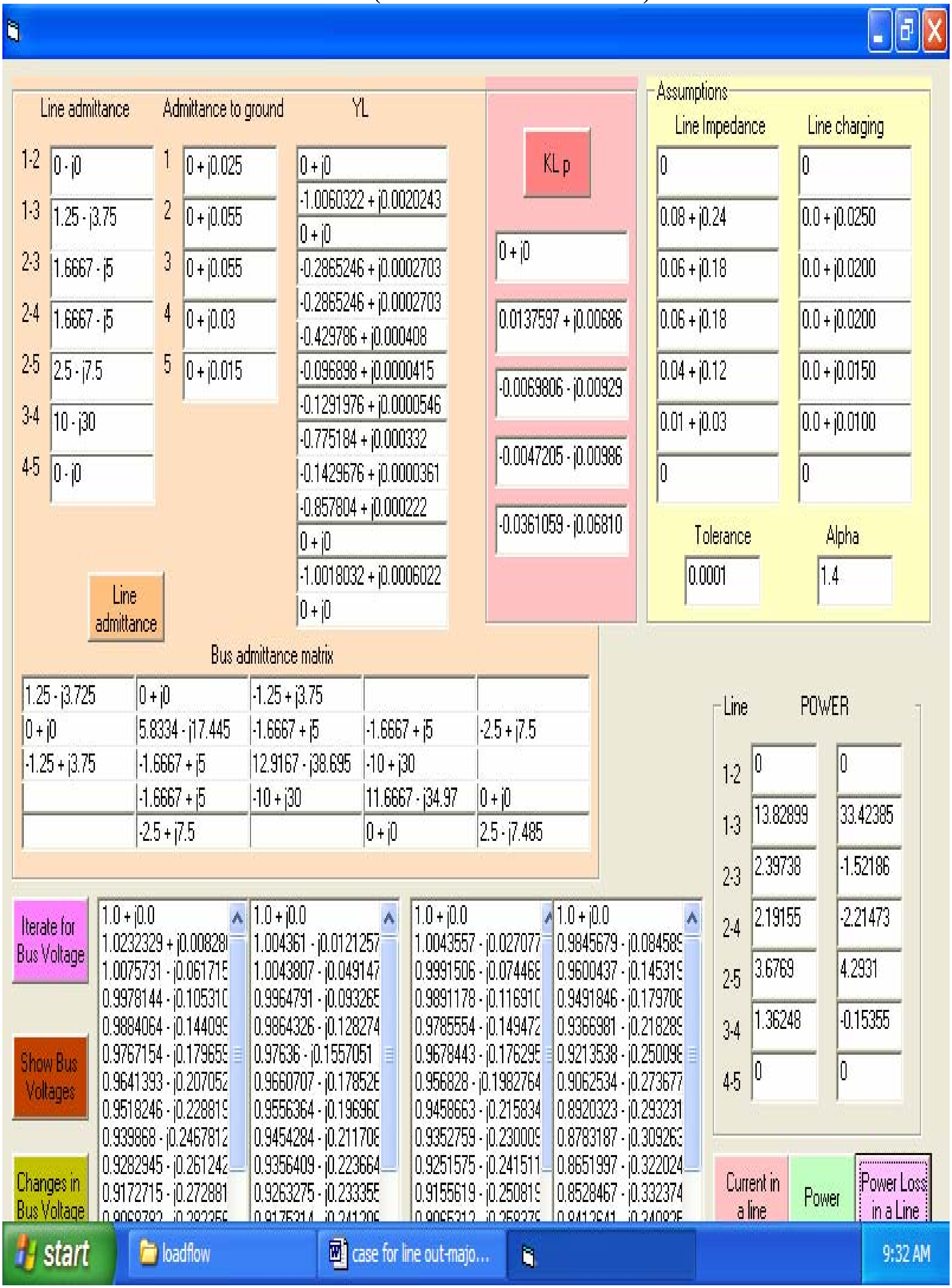


Case-11(When Lines 1-2 & 3-4 are out)

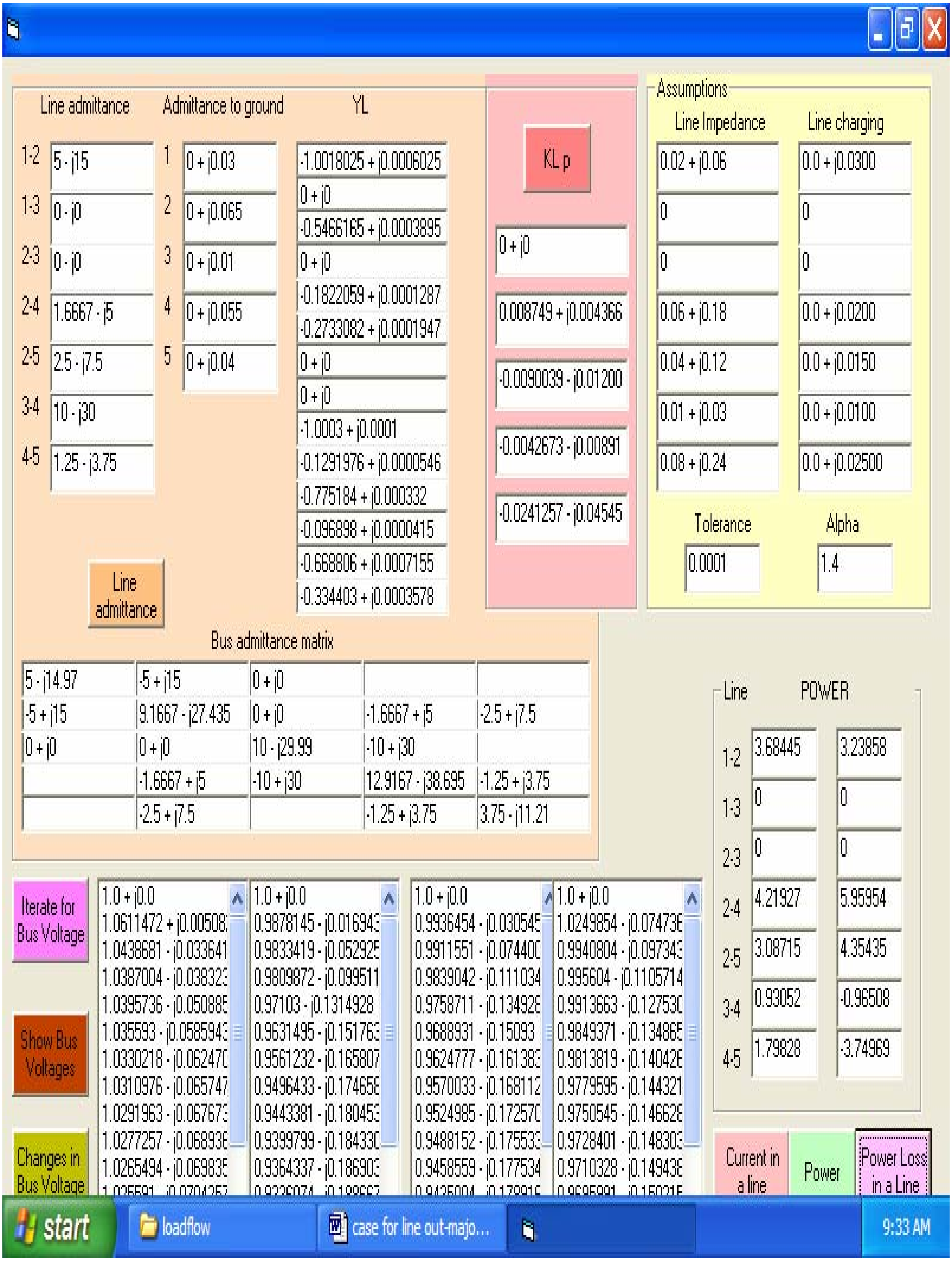


Line admittance
Admittance to ground
YL
1-2 0 - j0
1-3 1.25 - j3.75
2-3 1.6667 - j5
2-4 1.6667 - j5
2-5 2.5 - j7.5
3-4 0 - j0
4-5 1.25 - j3.75
1 0 + j0.025
2 0 + j0.055
3 0 + j0.045
4 0 + j0.045
5 0 + j0.04
0 + j0
-1.0060322 + j0.0020243
0 + j0
-0.2865246 + j0.0002703
-0.2865246 + j0.0002703
-0.429786 + j0.000408
-0.430563 + j0.000669
-0.5740852 + j0.0008886
0 + j0
-0.5740852 + j0.0008886
0 + j0
-0.430563 + j0.000669
-0.668806 + j0.0007155
-0.334403 + j0.0003578
KL p
0 + j0
0.0137597 + j0.00686
-0.0310648 - j0.04128
-0.0190063 - j0.03958
-0.0241257 - j0.04545
Assumptions
Line Impedance
Line charging
0
0
0.08 + j0.24
0.0 + j0.0250
0.06 + j0.18
0.0 + j0.0200
0.06 + j0.18
0.0 + j0.0200
0.04 + j0.12
0.0 + j0.0150
0
0
0.08 + j0.24
0.0 + j0.02500
Tolerance
0.0001
Alpha
1.4
Line admittance
Bus admittance matrix
1.25 - j3.725 | 0 + j0 | -1.25 + j3.75 | |
0 + j0 | 5.8334 - j17.445 | -1.6667 + j5 | -1.6667 + j5 | -2.5 + j7.5
-1.25 + j3.75 | -1.6667 + j5 | 2.9167 - j8.705 | 0 + j0 |
| -1.6667 + j5 | 0 + j0 | 2.9167 - j8.705 | -1.25 + j3.75
| -2.5 + j7.5 | | -1.25 + j3.75 | 3.75 - j11.21
Line POWER
1-2 0 0
1-3 18.0127 46.59232
2-3 6.82123 12.76219
2-4 3.81876 3.79793
2-5 4.35303 7.37329
3-4 0 0
4-5 2.74722 -0.82802
Iterate for Bus Voltage
Show Bus Voltages
Changes in Bus Voltage
1.0 + j0.0
1.0232329 + j0.00828
1.0149967 - j0.085868
0.9971803 - j0.172419
0.9738619 - j0.239202
0.946431 - j0.286601
0.9183855 - j0.320407
0.8906168 - j0.343890
0.8640707 - j0.360447
0.8389357 - j0.372057
0.8152861 - j0.380295
1.0 + j0.0
1.0178671 - j0.053410
1.0018928 - j0.104299
0.9905524 - j0.151692
0.9735179 - j0.184223
0.9558804 - j0.207986
0.9383252 - j0.224740
0.9212154 - j0.236354
0.9050931 - j0.244728
0.8898559 - j0.250577
0.8755711 - j0.254857
1.0 + j0.0
0.9985817 - j0.050965
0.984593 - j0.1537987
0.9602215 - j0.239961
0.9295894 - j0.302346
0.8977523 - j0.347862
0.8650428 - j0.378939
0.8336485 - j0.401112
0.803593 - j0.416326
0.7752665 - j0.427097
0.7485623 - j0.434612
1.0 + j0.0
0.991789 - j0.0812678
0.9753919 - j0.182368
0.9475357 - j0.259095
0.9173972 - j0.316738
0.8847519 - j0.356531
0.8527044 - j0.384793
0.8216794 - j0.404323
0.7922535 - j0.418008
0.7644839 - j0.427578
0.7383175 - j0.434292
Current in a line
Power
Power Loss in a Line
start
loadflow
case for line out-majo...
9:31 AM

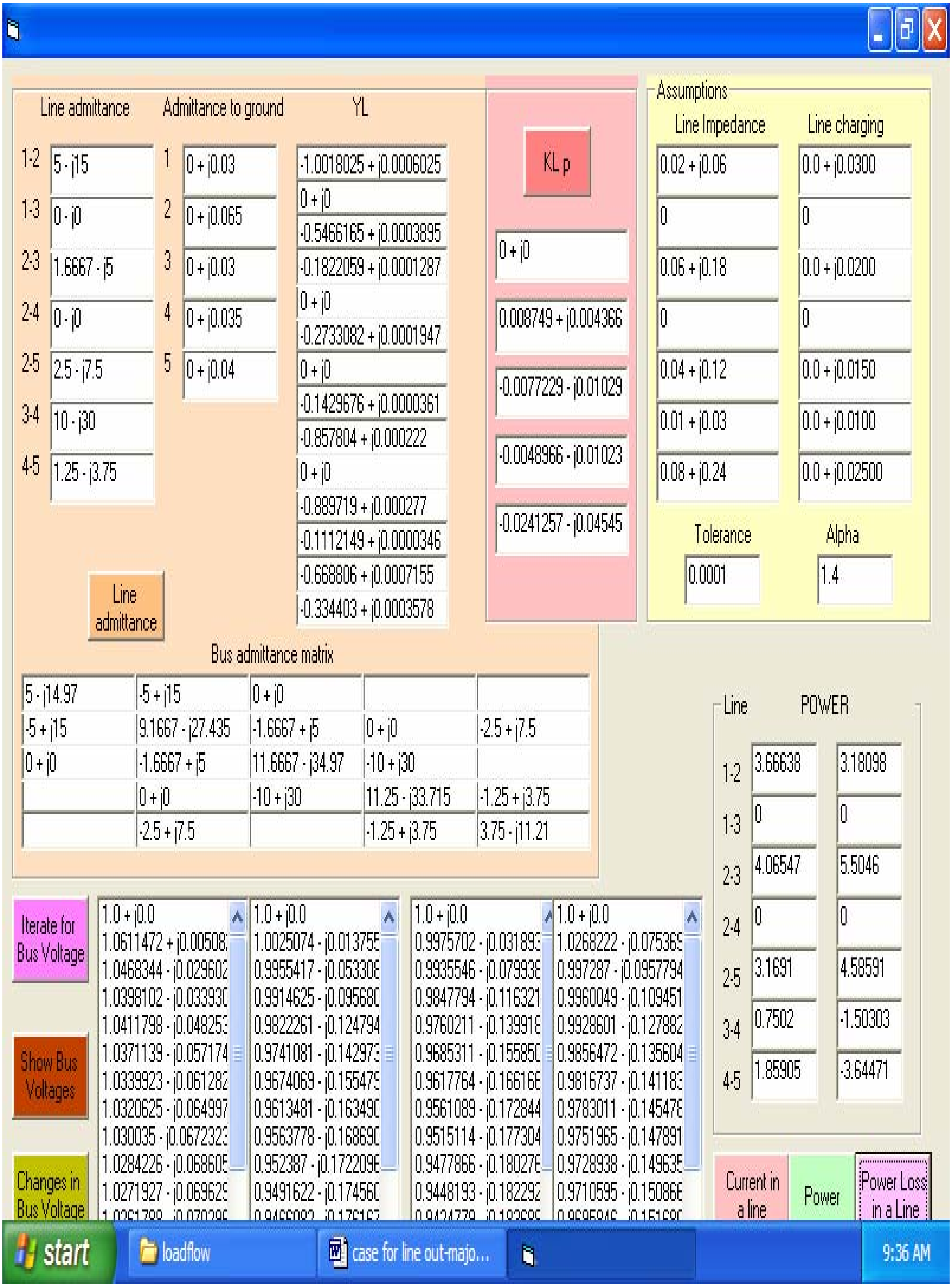
Case-12(When Lines 1-2 & 4-5 are out)



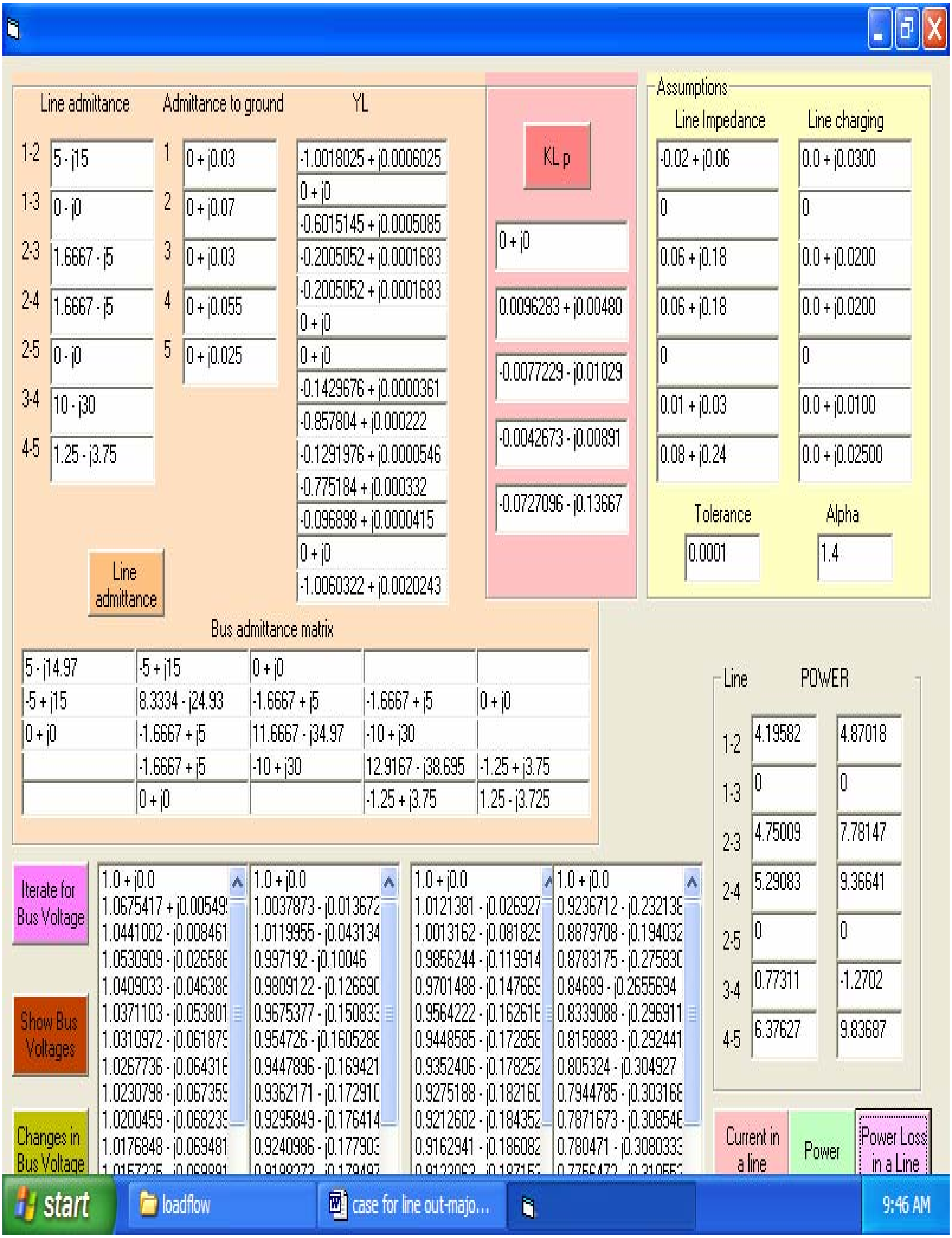
Case-13(When Lines 1-3 & 2-3 are out)



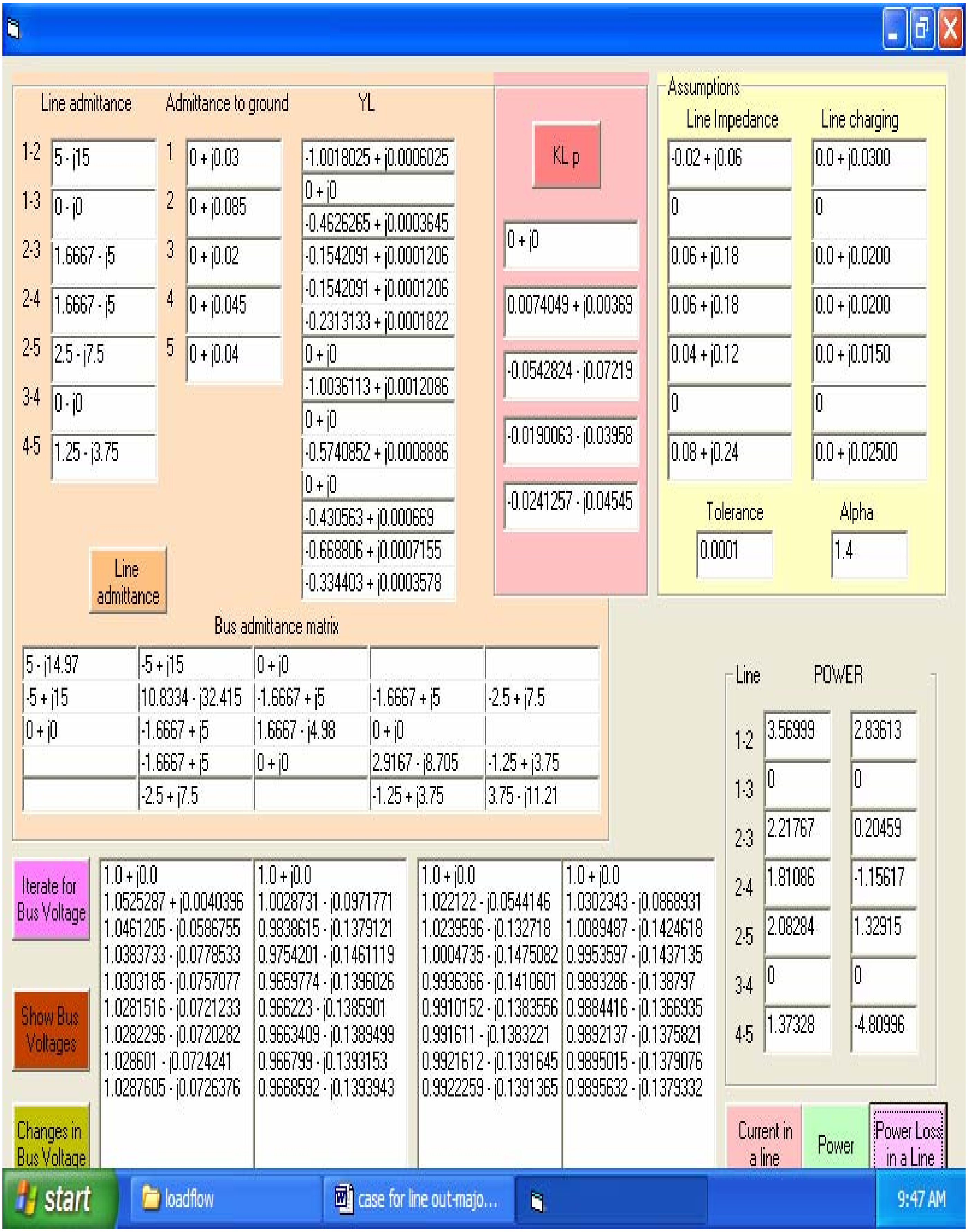
Case-14(When Lines 1-3 & 2-4 are out)



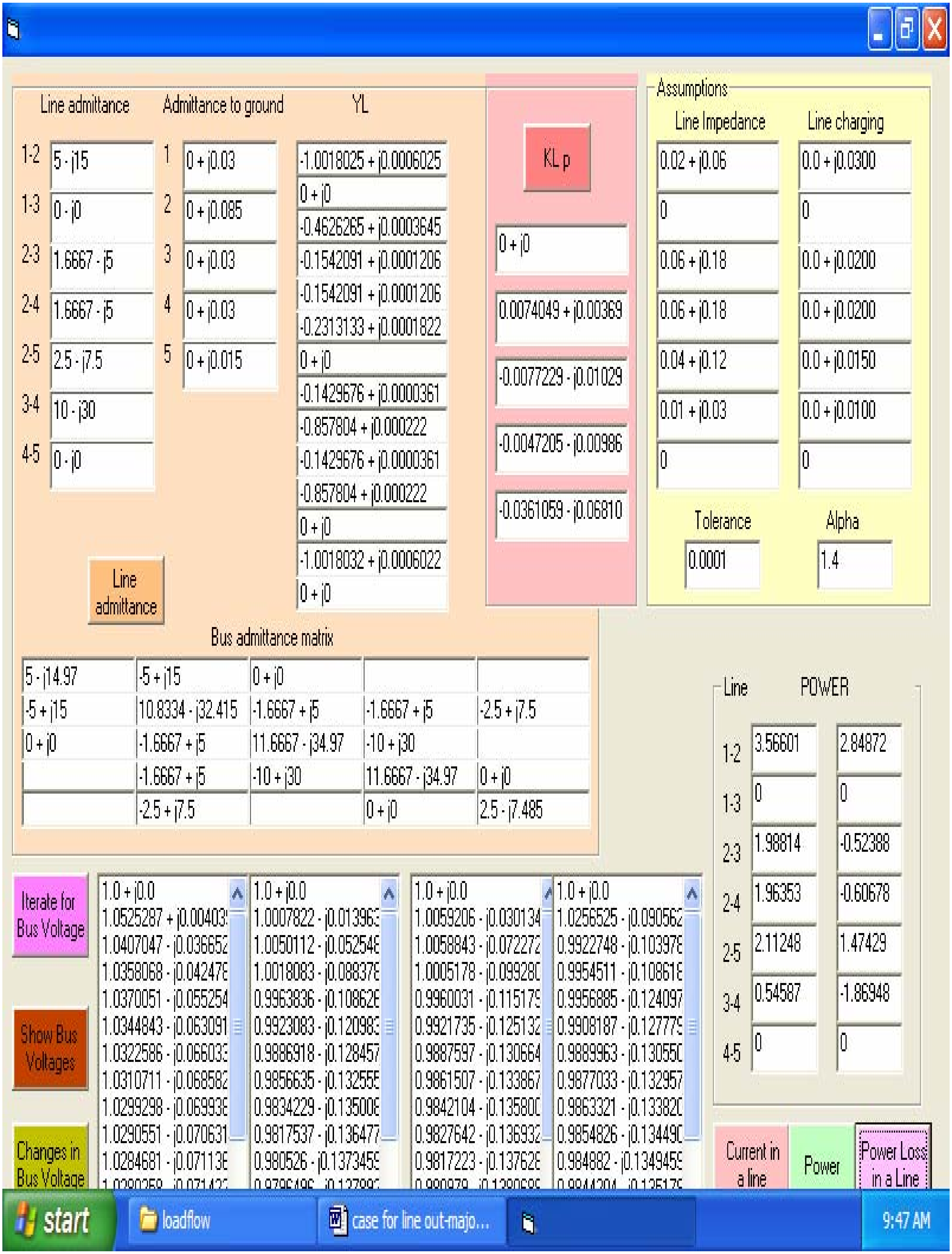
Case-15 (When Lines 1-3 & 2-5 are out)



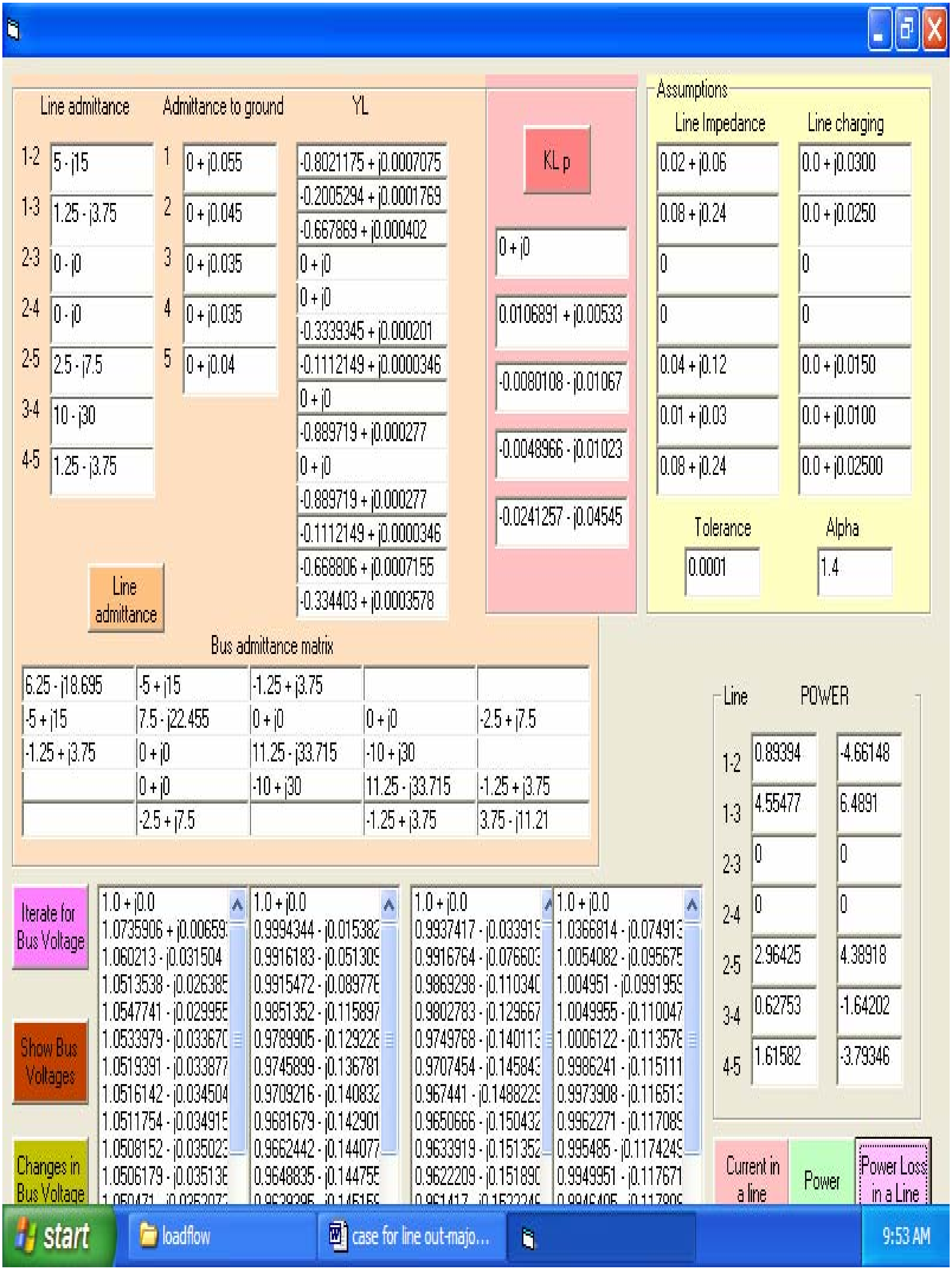
Case-16(When Lines 1-3 & 3-4 are out)



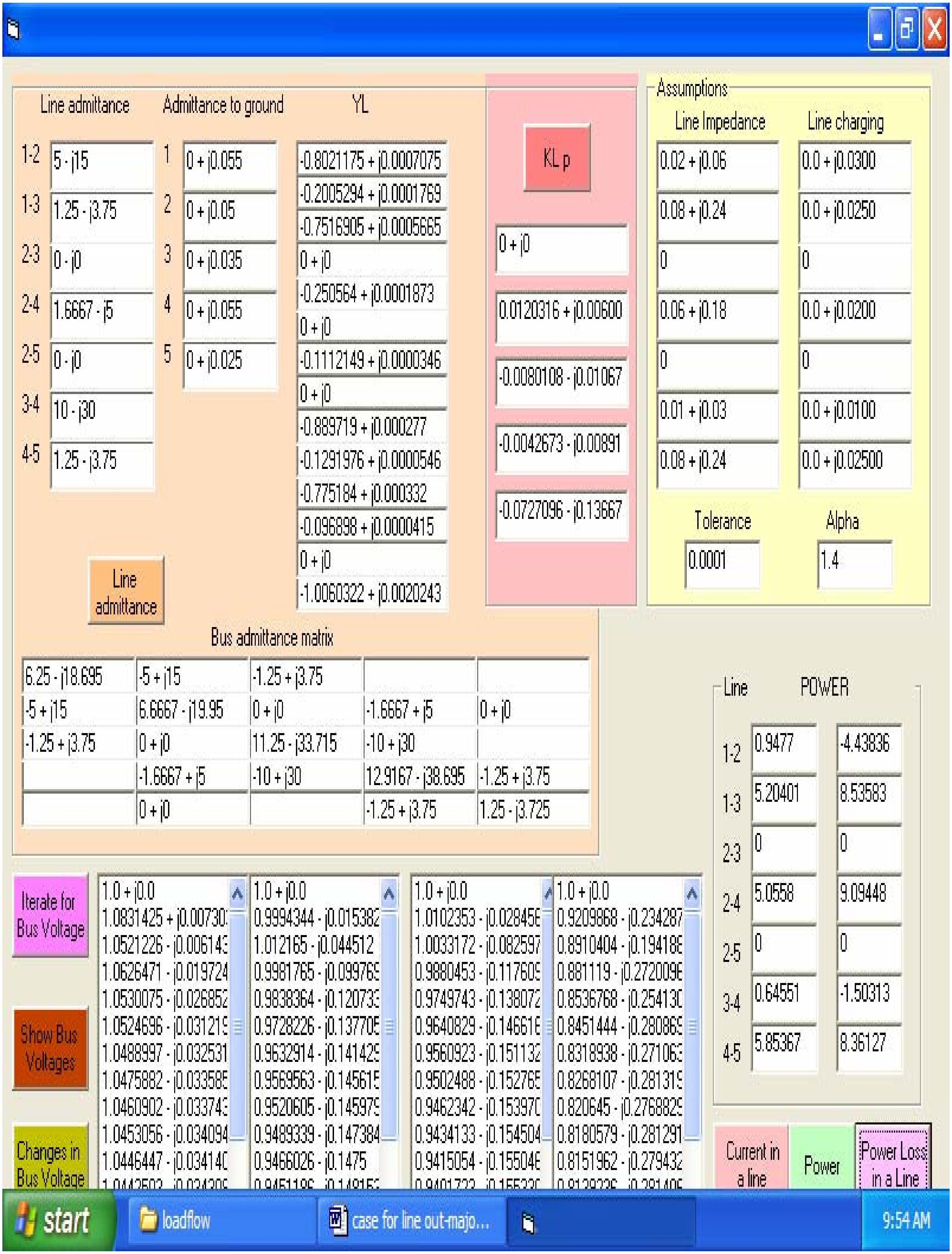
Case-17(When Lines 1-3 & 4-5 are out)



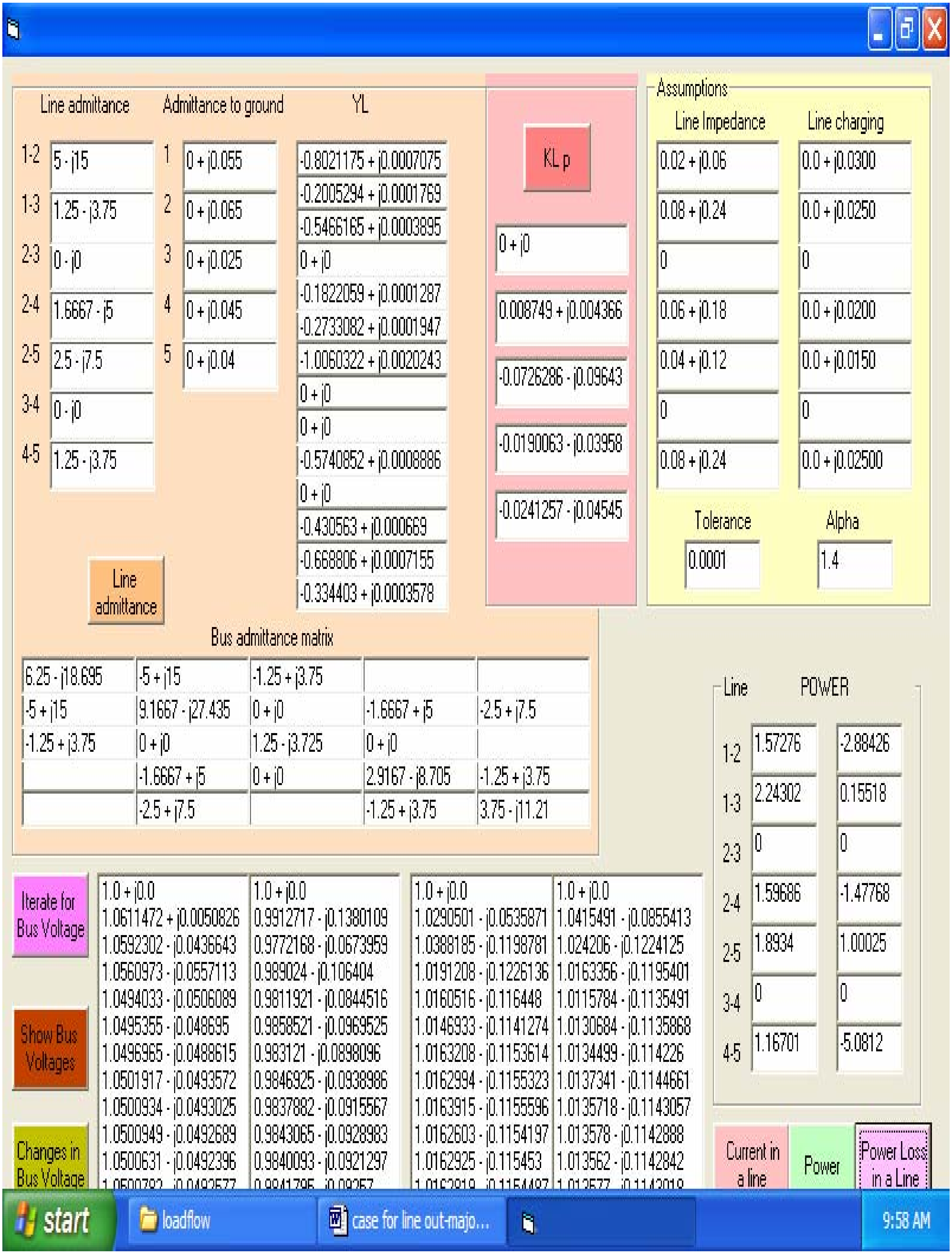
Case-18(When Lines 2-3 & 2-4 are out)



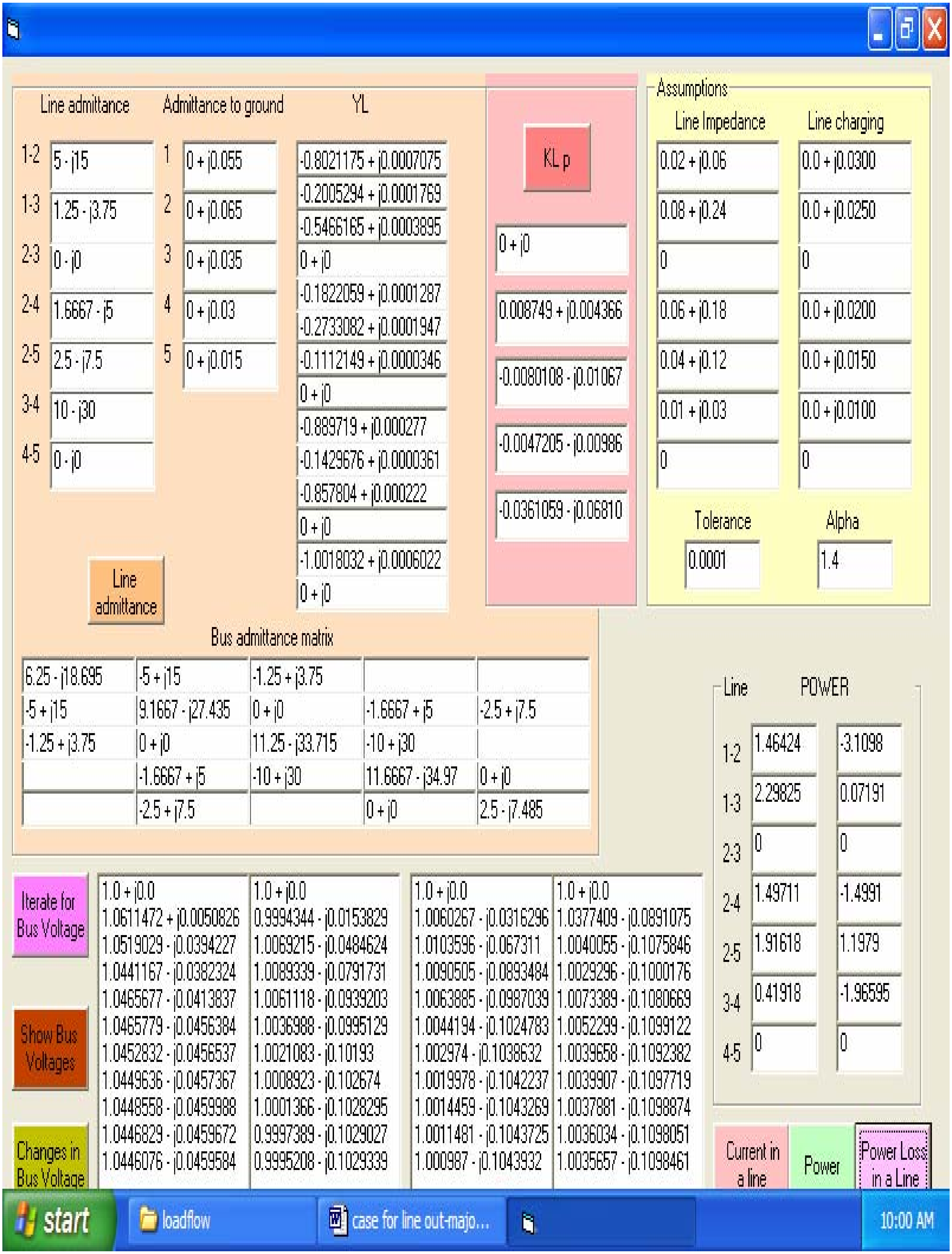
Case-19(When Lines 2-3 & 2-5 are out)



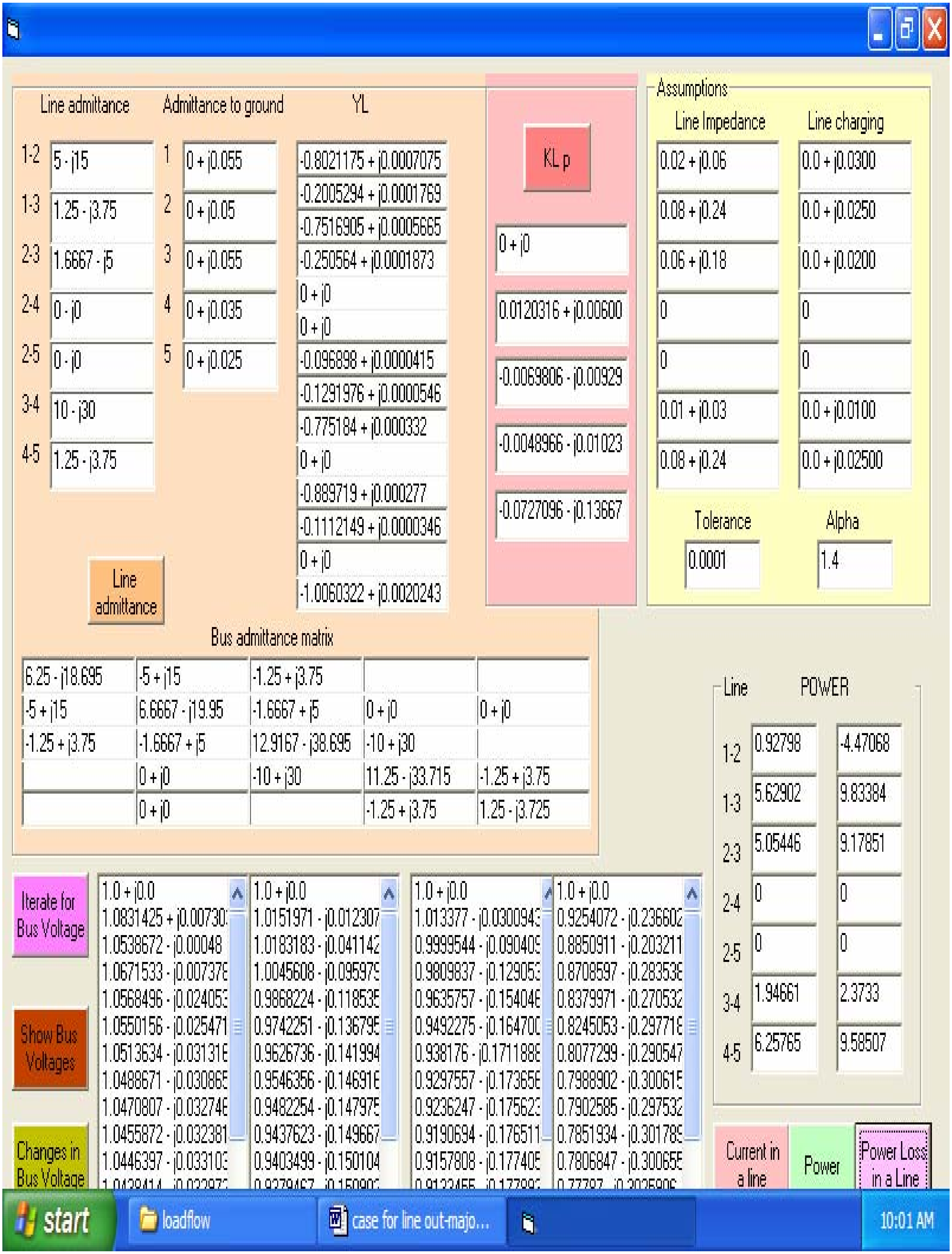
Case-20(When Lines 2-3 & 3-4 are out)



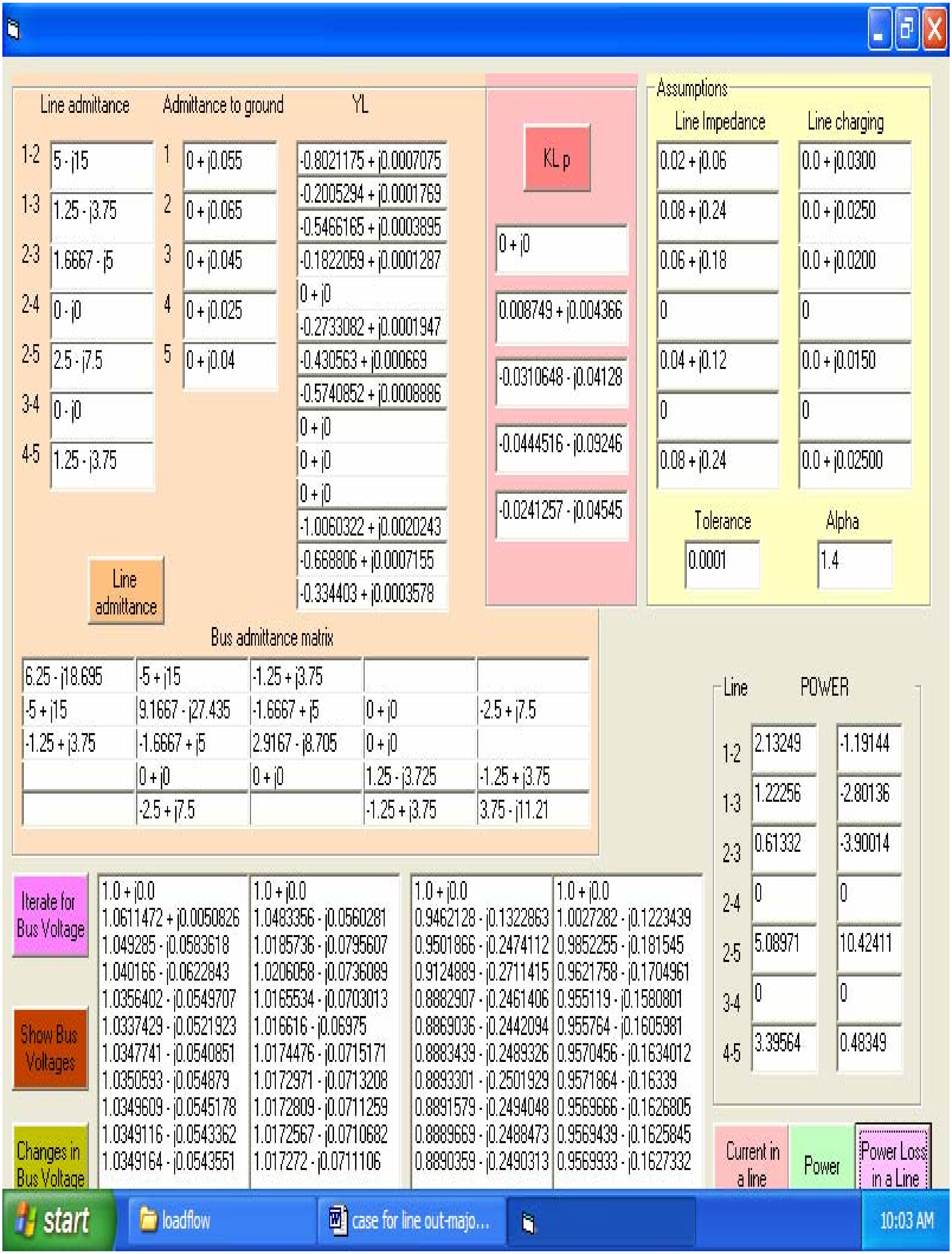
Case-21(When Lines 2-3 & 4-5 are out)



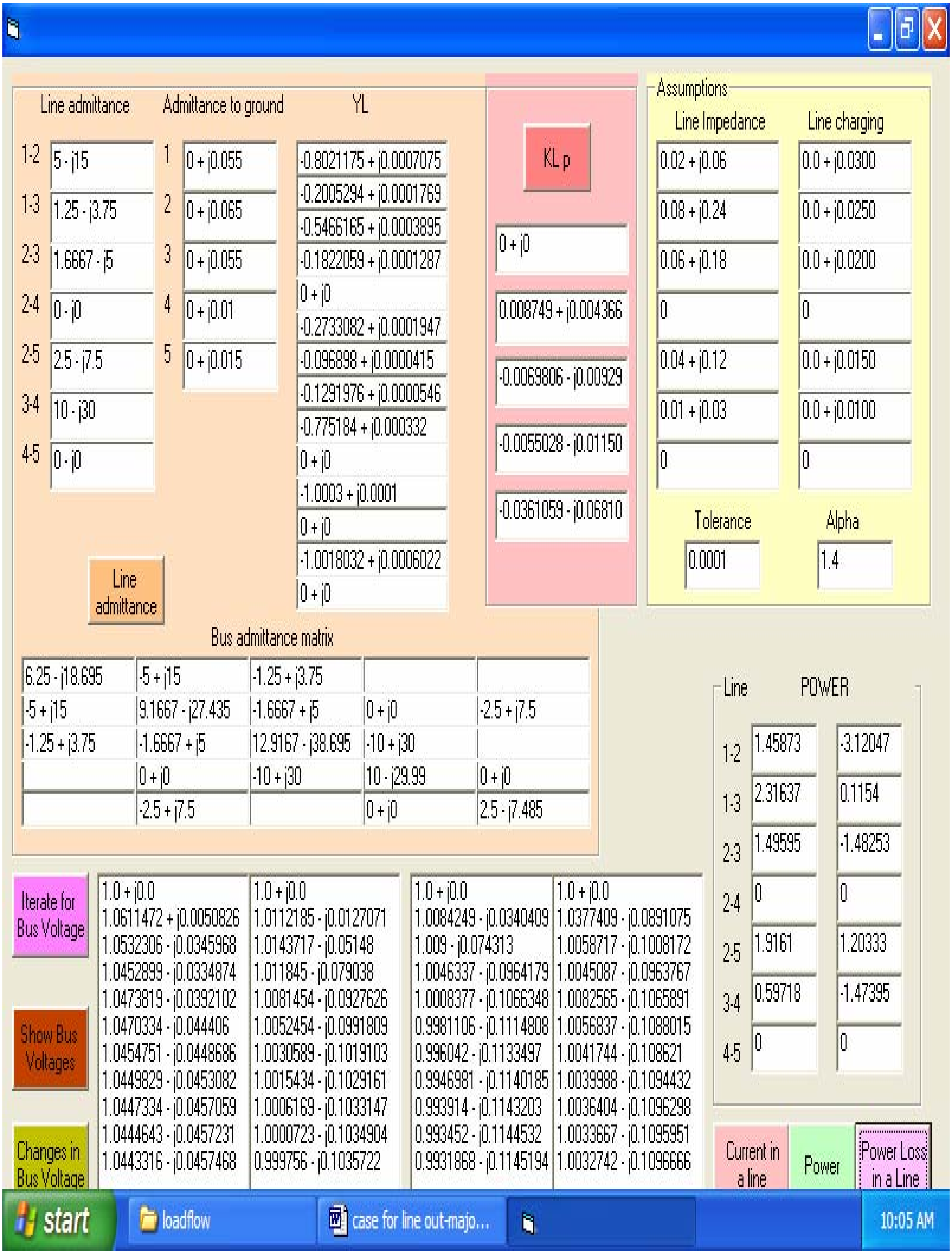
Case-22(When Lines 2-4 & 2-5 are out)



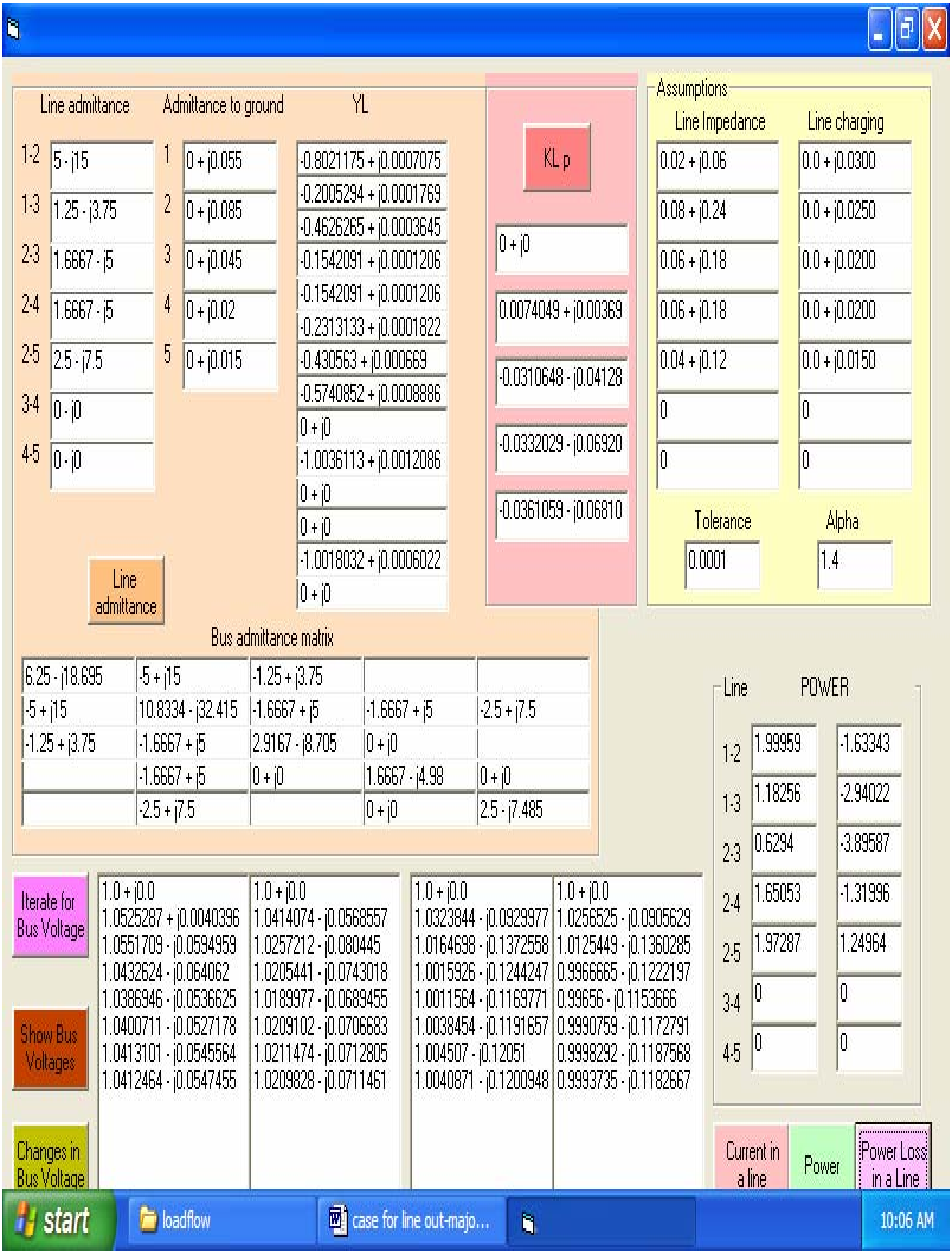
Case-23(When Lines 2-4 & 3-4 are out)



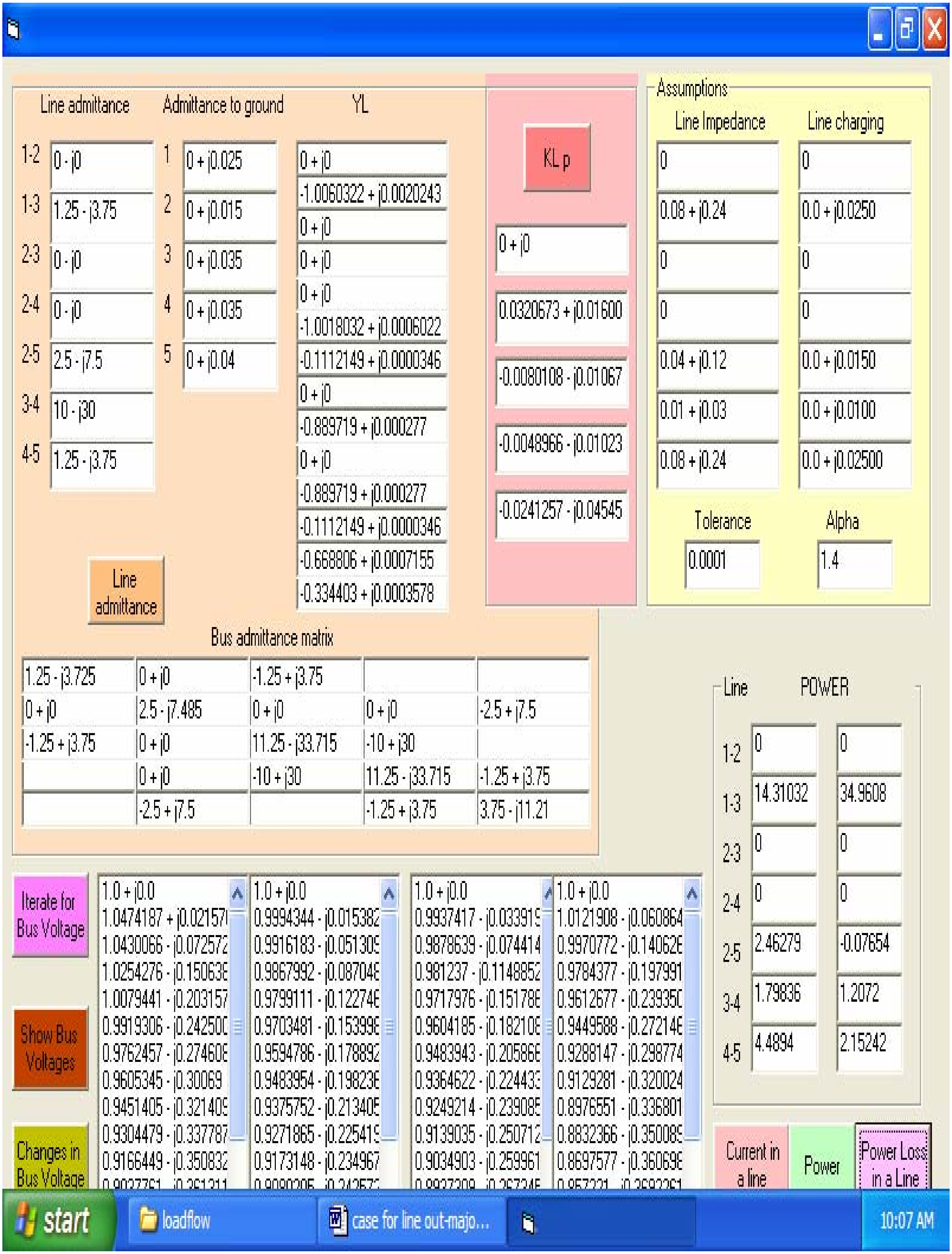
Case-24(When Lines 2-4 & 4-5 are out)



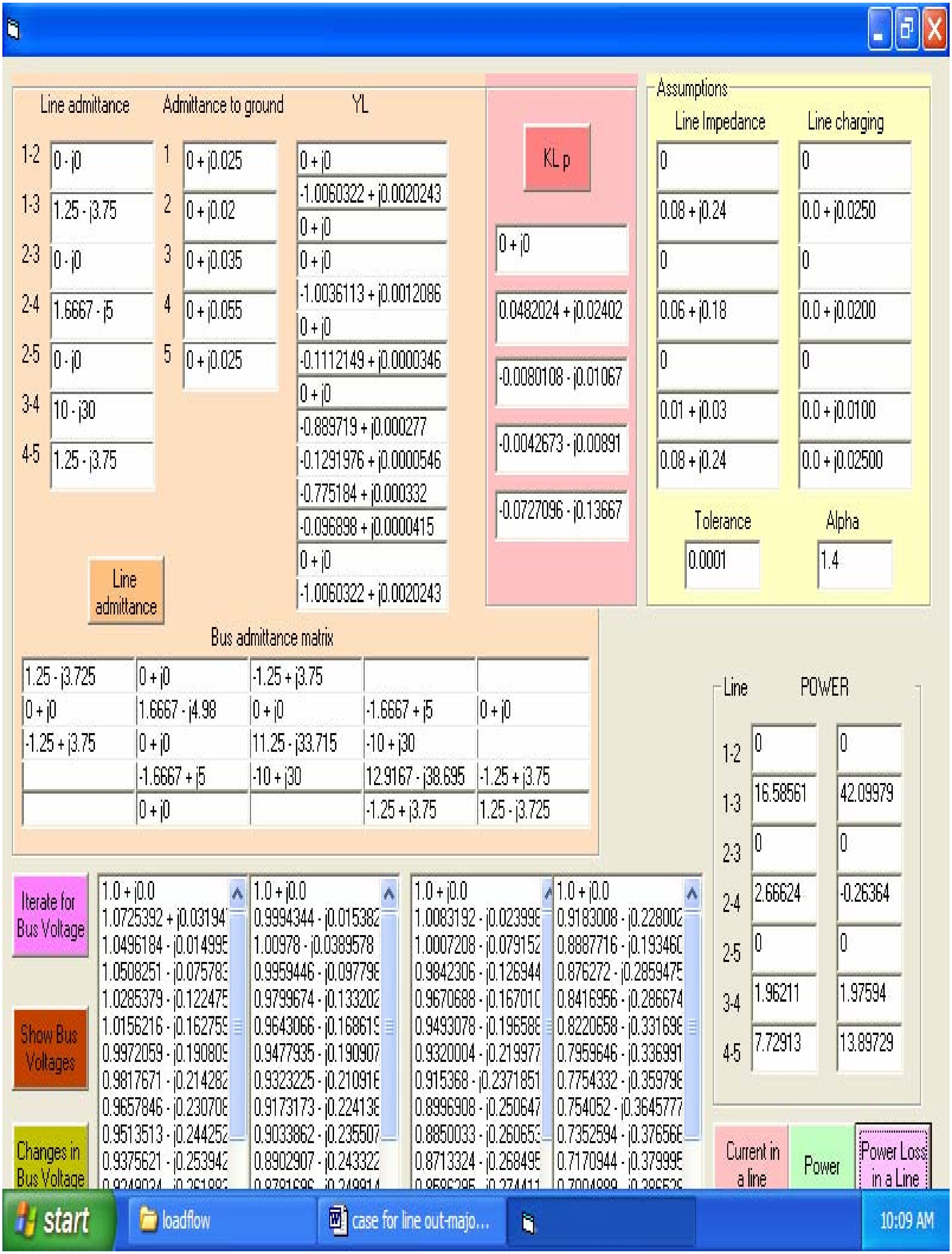
Case-25(When Lines 3-4 & 4-5 are out)



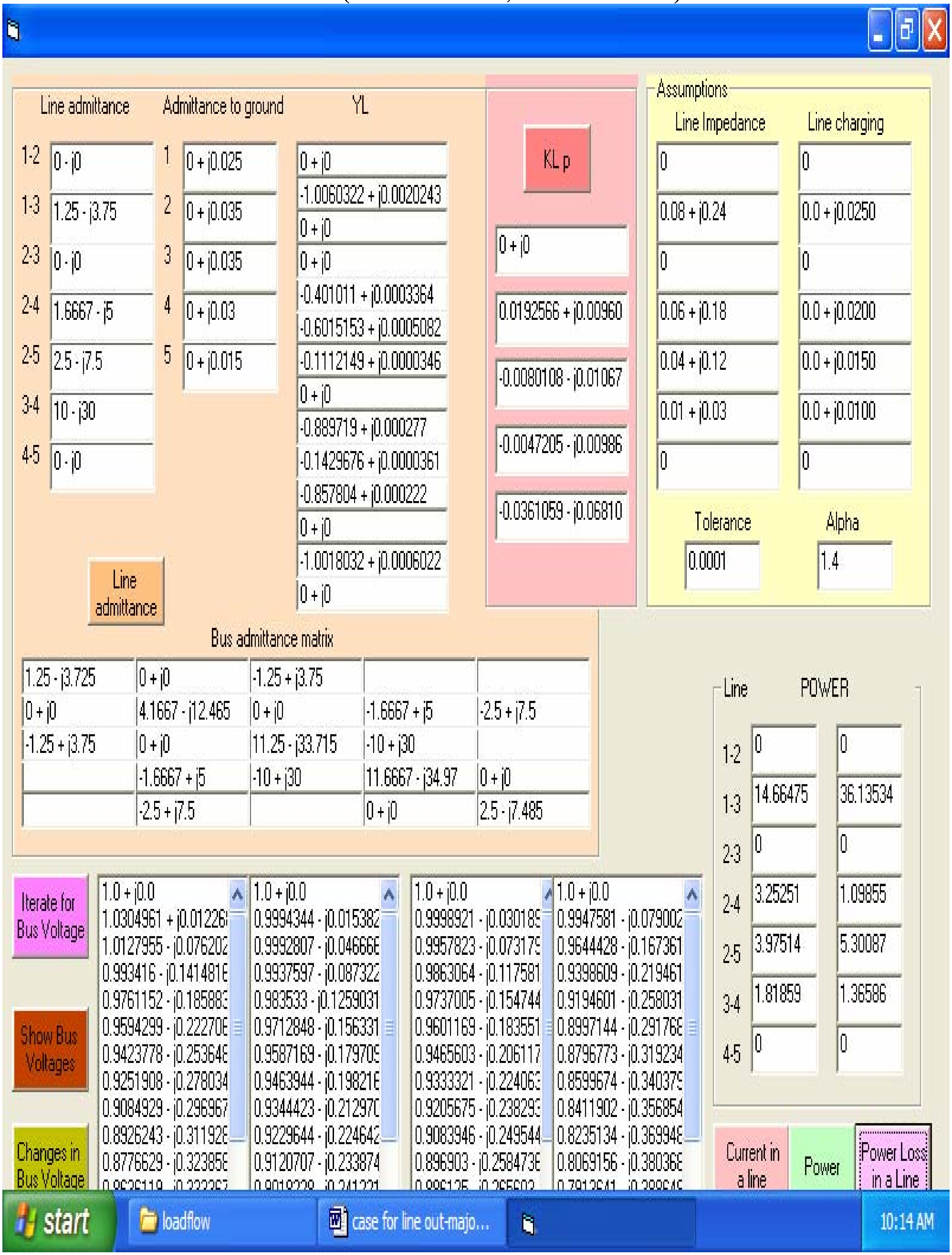
Case-26(When Lines 1-2 ,2-3 & 2-4 are out)



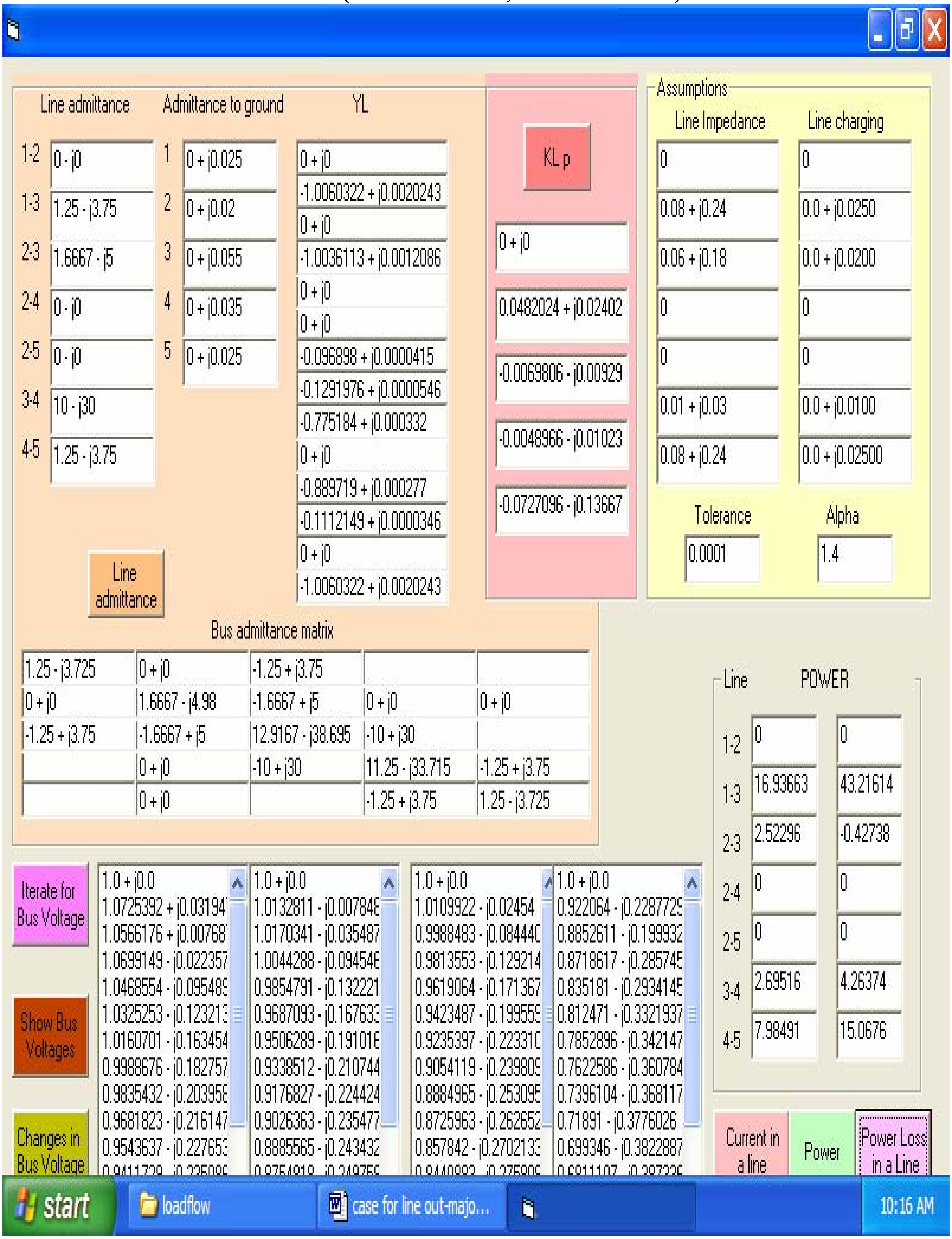
Case-27(When Lines 1-2 ,2-3 & 2-5 are out)


Line admittance
Admittance to ground
YL
1-2 0 - j0
1-3 1.25 - j3.75
2-3 0 - j0
2-4 1.6667 - j5
2-5 0 - j0
3-4 10 - j30
4-5 1.25 - j3.75
1 0 + j0.025
2 0 + j0.02
3 0 + j0.035
4 0 + j0.055
5 0 + j0.025
0 + j0
-1.0060322 + j0.0020243
0 + j0
0 + j0
-1.0036113 + j0.0012086
0 + j0
-0.1112149 + j0.0000346
0 + j0
-0.889719 + j0.000277
-0.1291976 + j0.0000546
-0.775184 + j0.000332
-0.096898 + j0.0000415
0 + j0
-1.0060322 + j0.0020243
Line admittance
KL p
0 + j0
0.0482024 + j0.02402
-0.0080108 - j0.01067
-0.0042673 - j0.00891
-0.0727096 - j0.13667
Assumptions
Line Impedance
Line charging
0
0
0.08 + j0.24
0.0 + j0.0250
0
0
0.06 + j0.18
0.0 + j0.0200
0
0
0.01 + j0.03
0.0 + j0.0100
0.08 + j0.24
0.0 + j0.02500
Tolerance
0.0001
Alpha
1.4
Bus admittance matrix
1.25 - j3.725 | 0 + j0 | -1.25 + j3.75 | |
0 + j0 | 1.6667 - j4.98 | 0 + j0 | -1.6667 + j5 | 0 + j0
-1.25 + j3.75 | 0 + j0 | 11.25 - j33.715 | -10 + j30 |
| -1.6667 + j5 | -10 + j30 | 12.9167 - j38.695 | -1.25 + j3.75
| 0 + j0 | | -1.25 + j3.75 | 1.25 - j3.725
Line
POWER
1-2 0 0
1-3 16.58561 42.09979
2-3 0 0
2-4 2.66624 -0.26364
2-5 0 0
3-4 1.96211 1.97594
4-5 7.72913 13.89729
Iterate for Bus Voltage
Show Bus Voltages
Changes in Bus Voltage
1.0 + j0.0
1.0725392 + j0.03194
1.0496184 - j0.014995
1.0508251 - j0.07578
1.0285379 - j0.122475
1.0156216 - j0.16275
0.9972059 - j0.19080
0.9817671 - j0.214282
0.9657846 - j0.23070
0.9513513 - j0.244252
0.9375621 - j0.253942
1.0 + j0.0
0.9994344 - j0.015382
1.00978 - j0.0389578
0.9959446 - j0.09779
0.9799674 - j0.13320
0.9643066 - j0.16861
0.9477935 - j0.190907
0.9323225 - j0.21091
0.9173173 - j0.22413
0.9033862 - j0.235507
0.8902907 - j0.243322
1.0 + j0.0
1.0083192 - j0.02399
1.0007208 - j0.079152
0.9842306 - j0.126944
0.9670688 - j0.16701
0.9493078 - j0.19658
0.9320004 - j0.21997
0.915368 - j0.2371851
0.8996908 - j0.250647
0.8850033 - j0.26065
0.8713324 - j0.26849
1.0 + j0.0
0.9183008 - j0.22800
0.8887716 - j0.19346
0.876272 - j0.285947
0.8416956 - j0.28667
0.8220658 - j0.33169
0.7959646 - j0.336991
0.7754332 - j0.35979
0.754052 - j0.364577
0.7352594 - j0.37656
0.7170944 - j0.37999
Current in a line
Power
Power Loss in a Line
start
loadflow
case for line out-majo...
10:09 AM

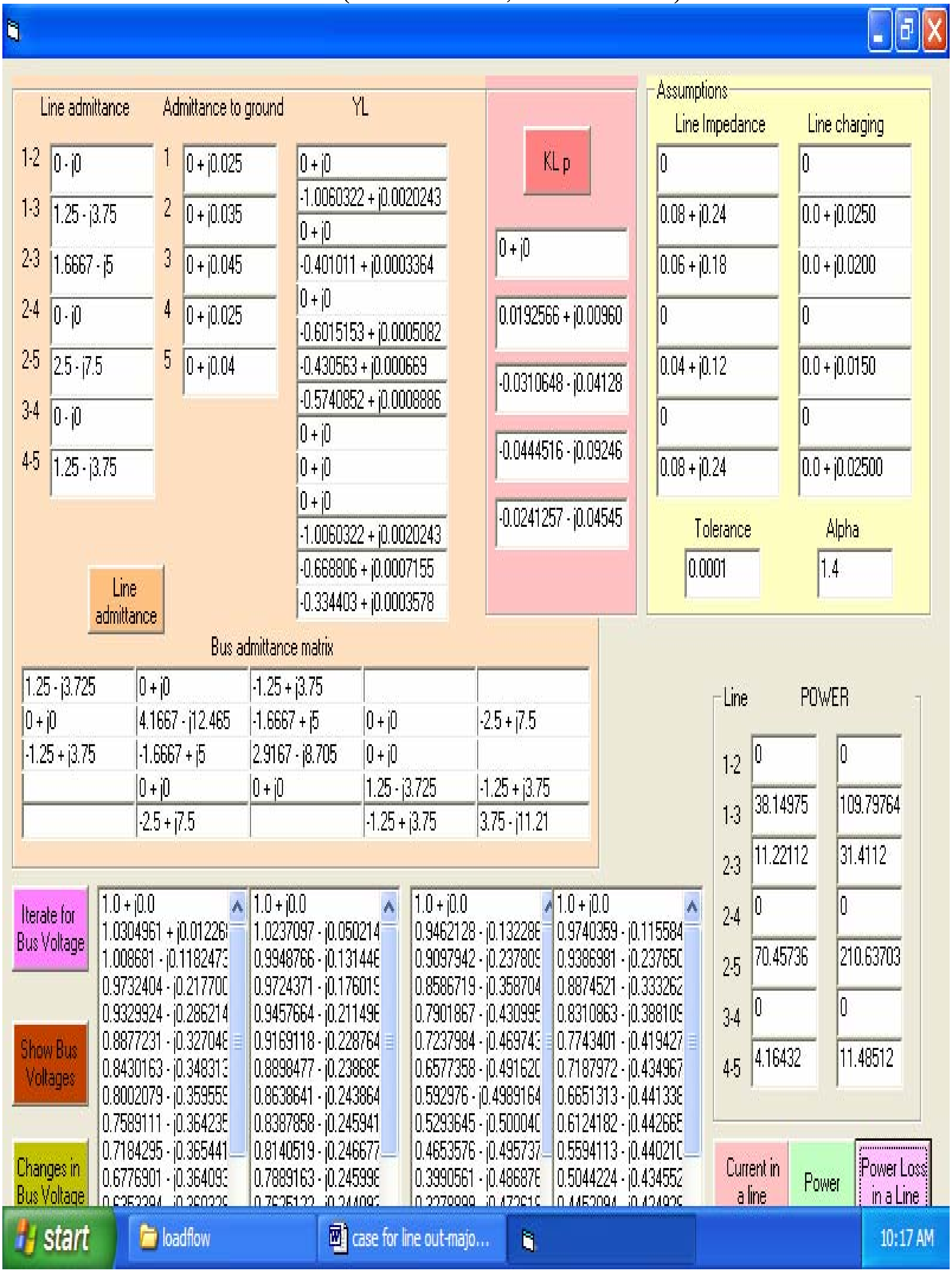
Case-29(When Lines 1-2 ,2-3 & 4-5 are out)



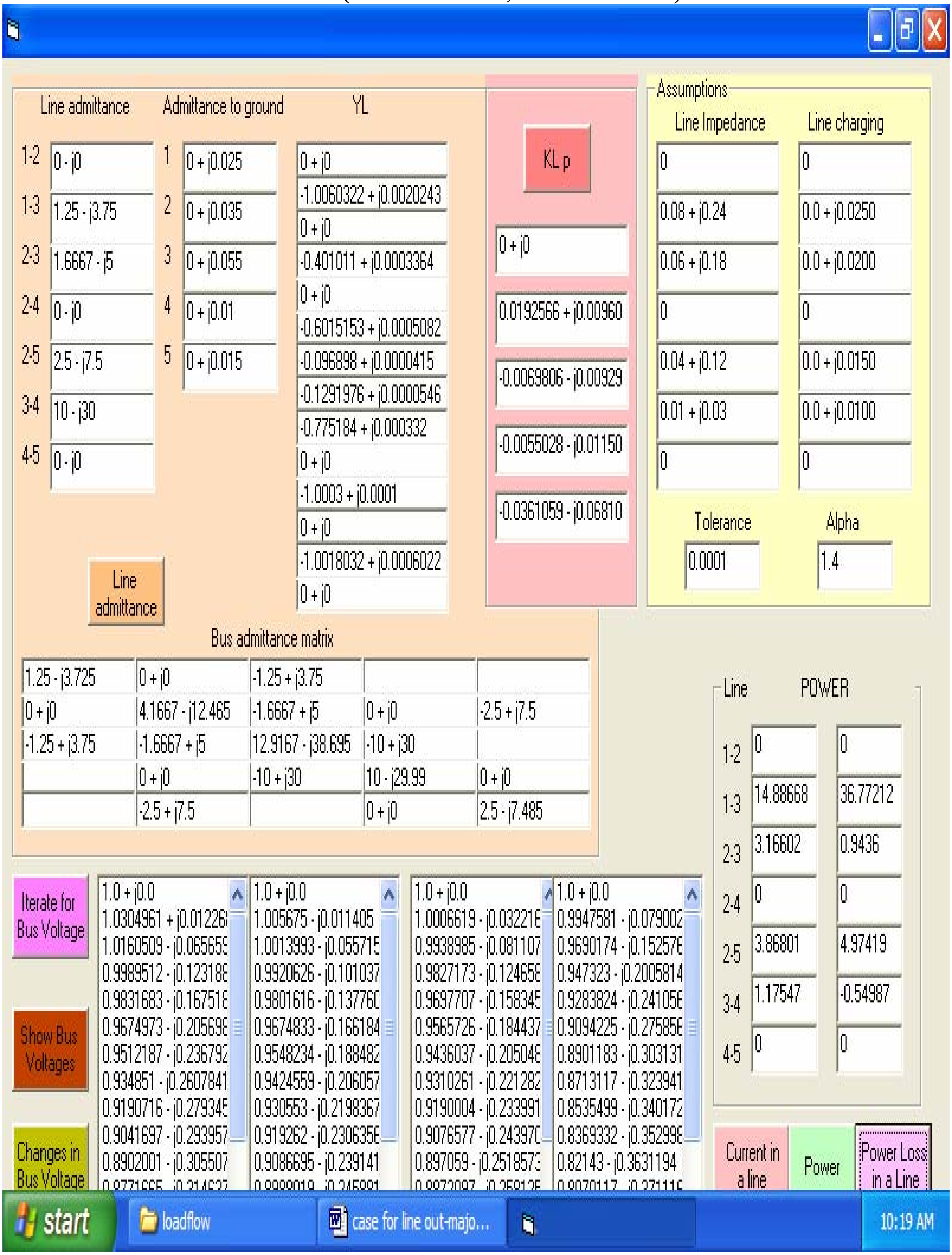
Case-30(When Lines 1-2 ,2-4 & 2-5 are out)



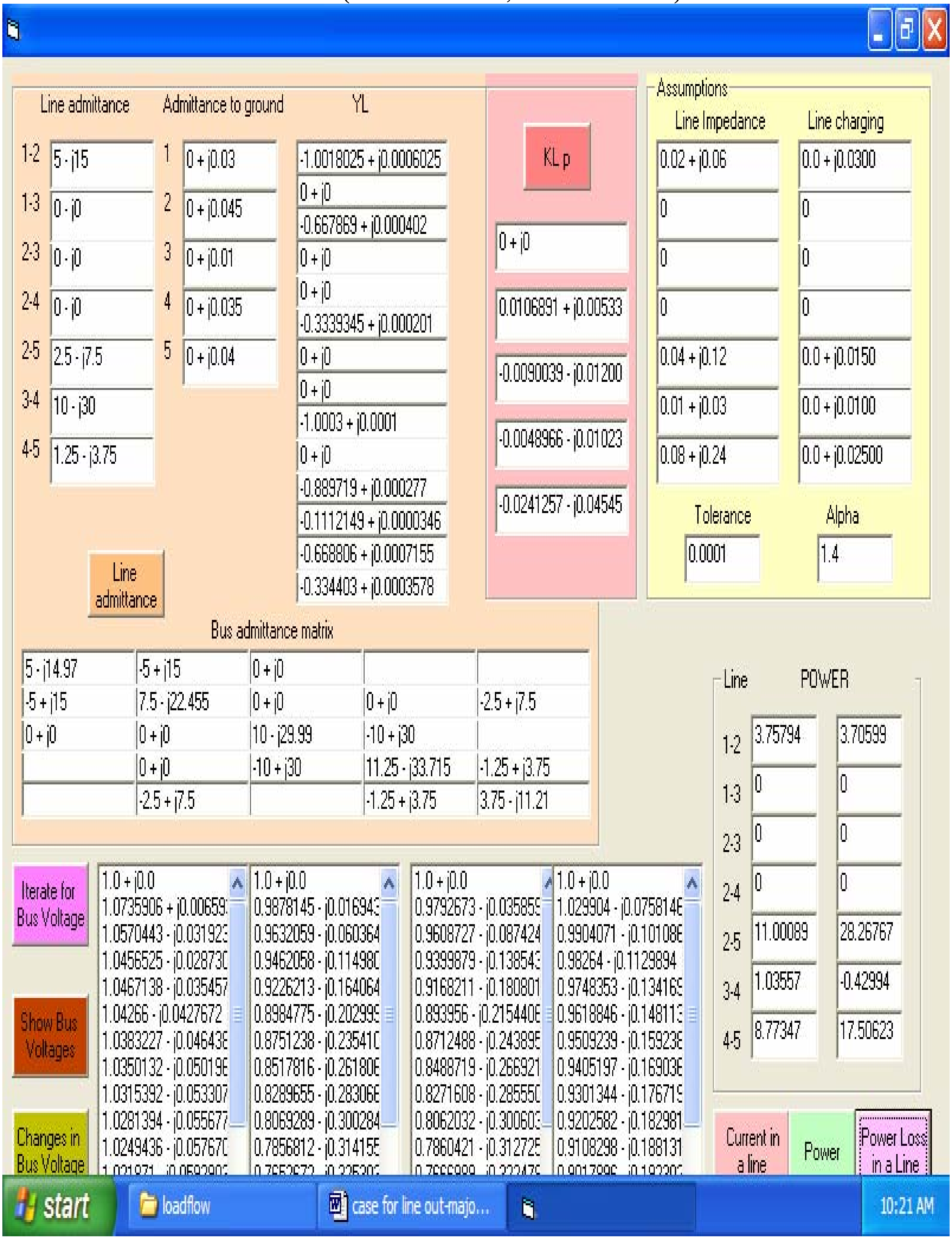
Case-31(When Lines 1-2 ,2-4 & 3-4 are out)



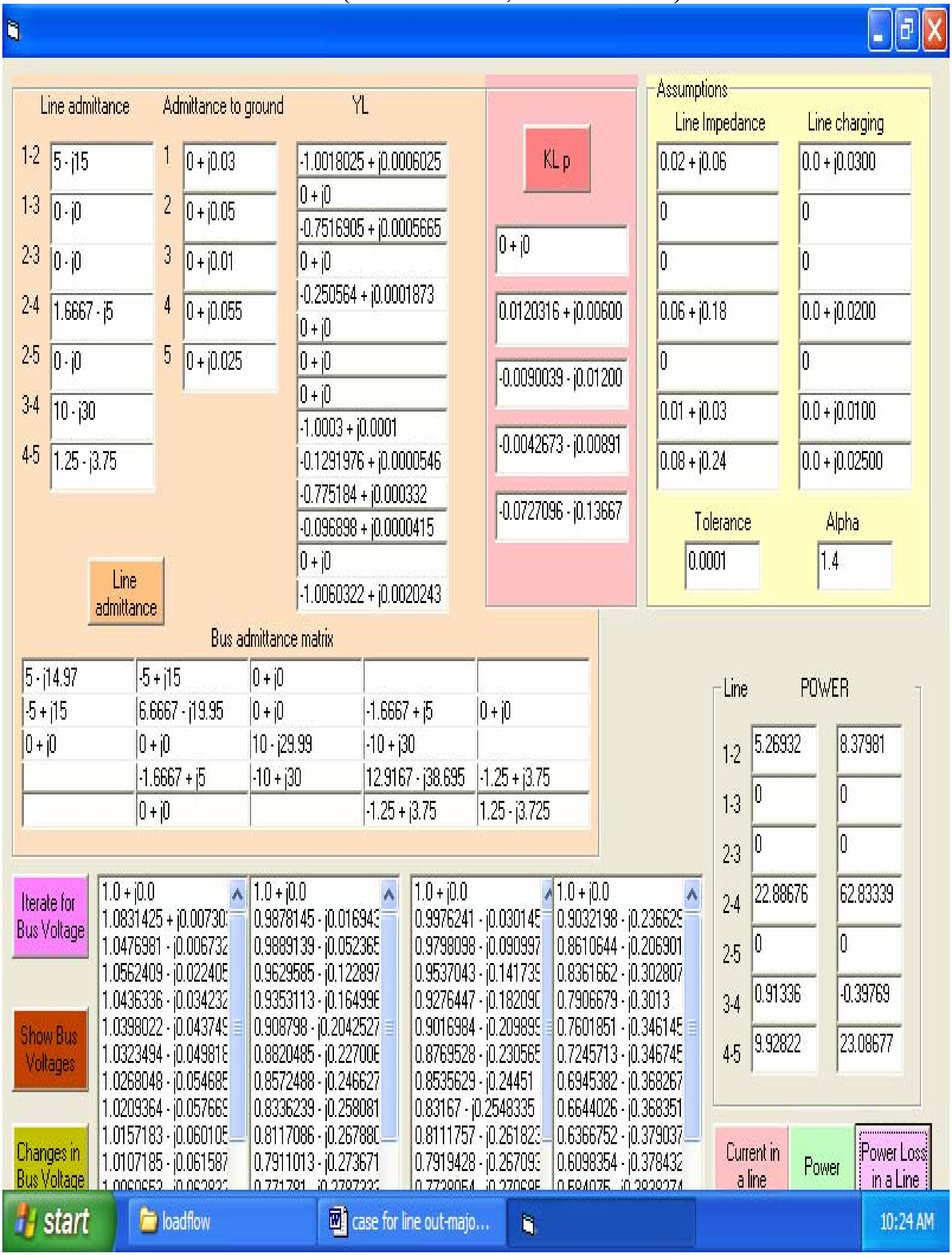
Case-32(When Lines 1-2 ,2-4 & 4-5 are out)



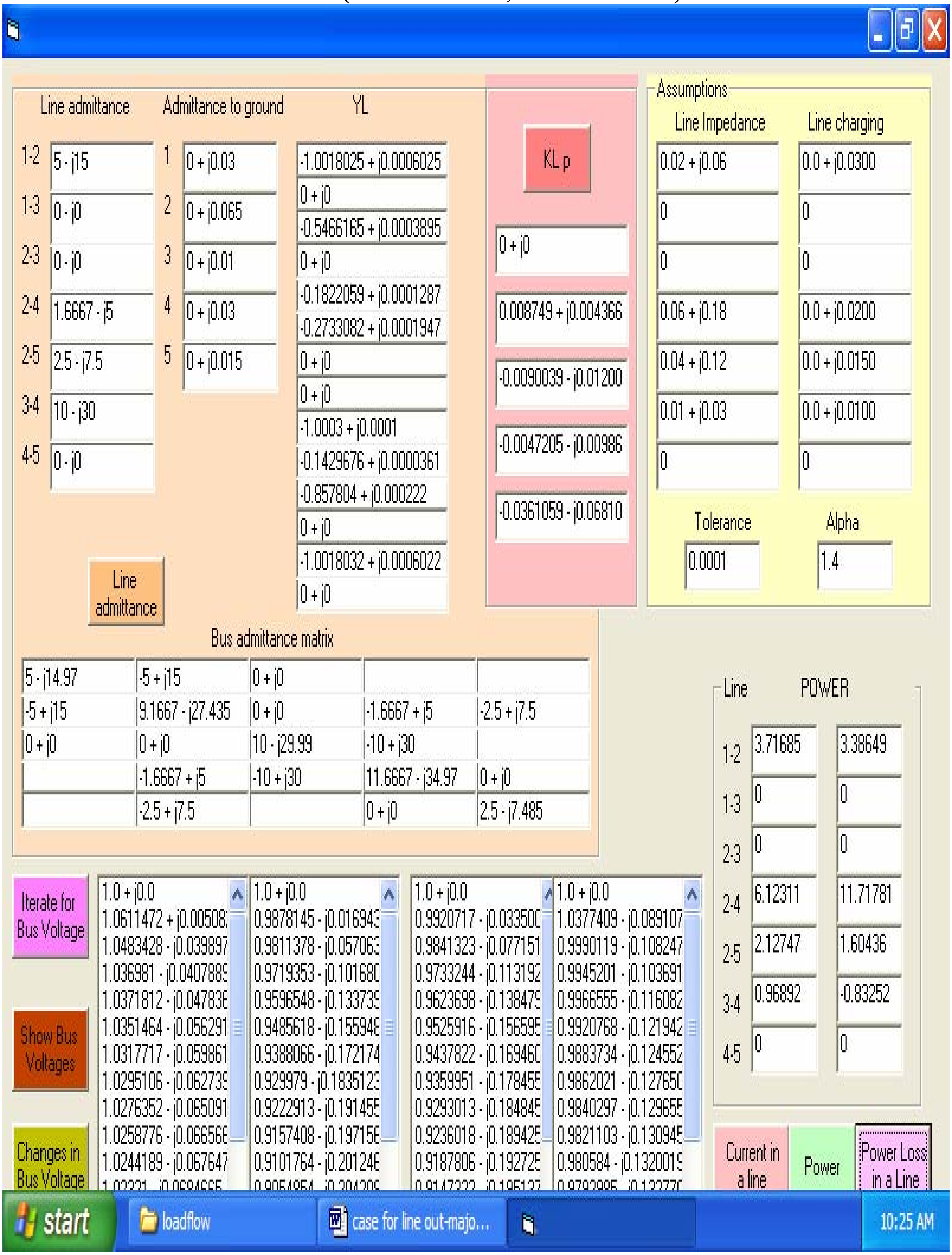
Case-33(When Lines 1-3 ,2-3 & 2-4 are out)



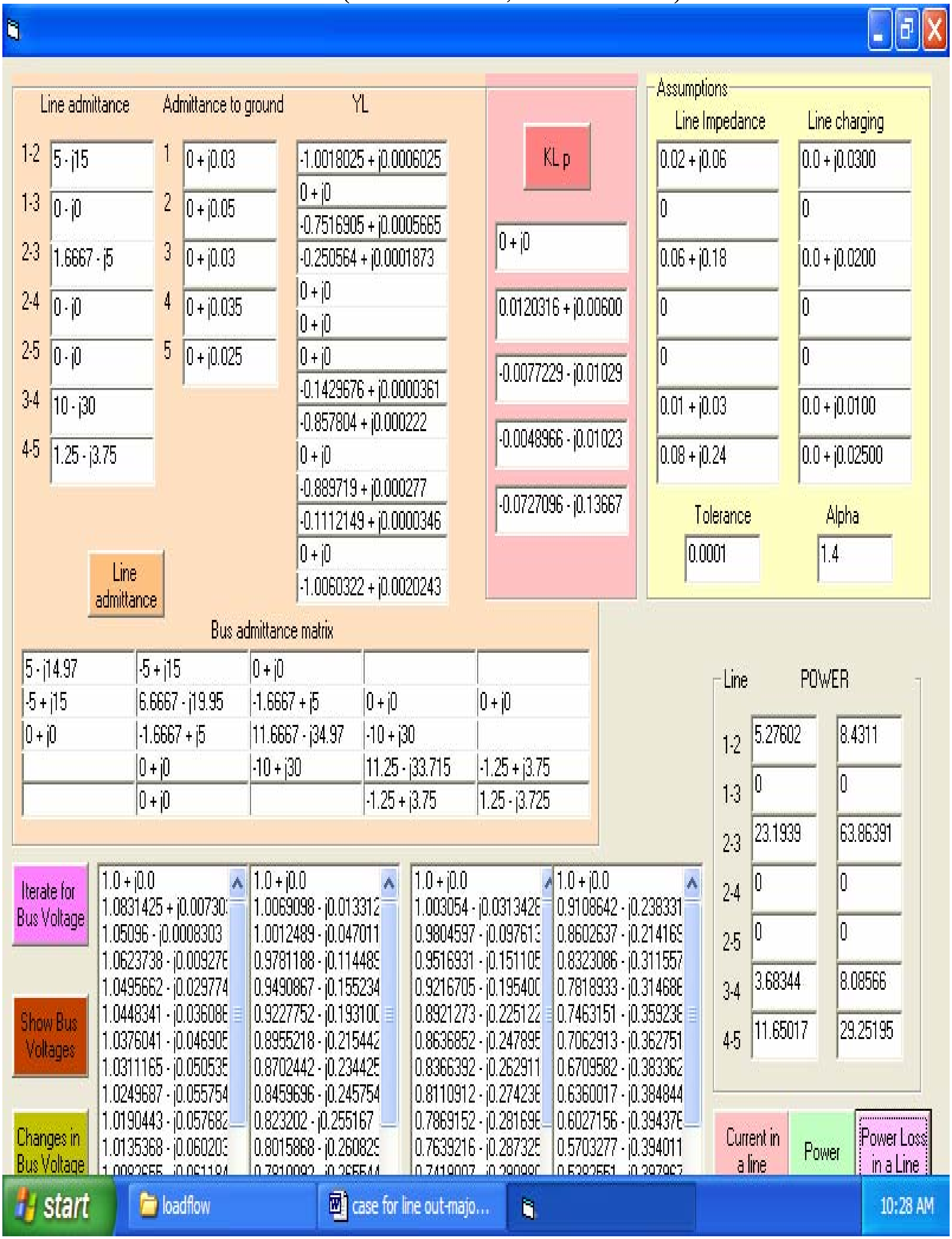
Case-34(When Lines 1-3 ,2-3 & 2-5 are out)



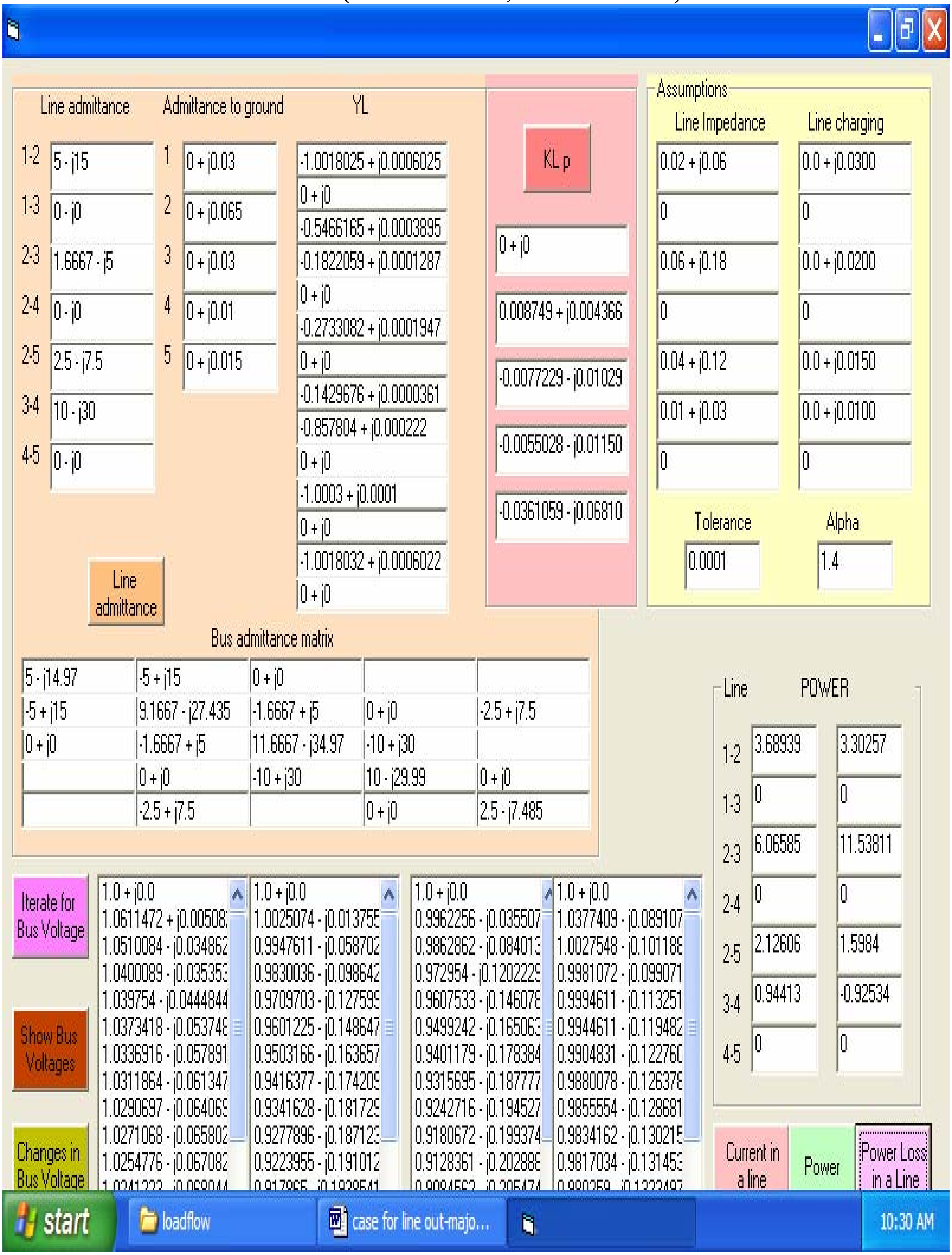
Case-35(When Lines 1-3 ,2-3 & 4-5 are out)



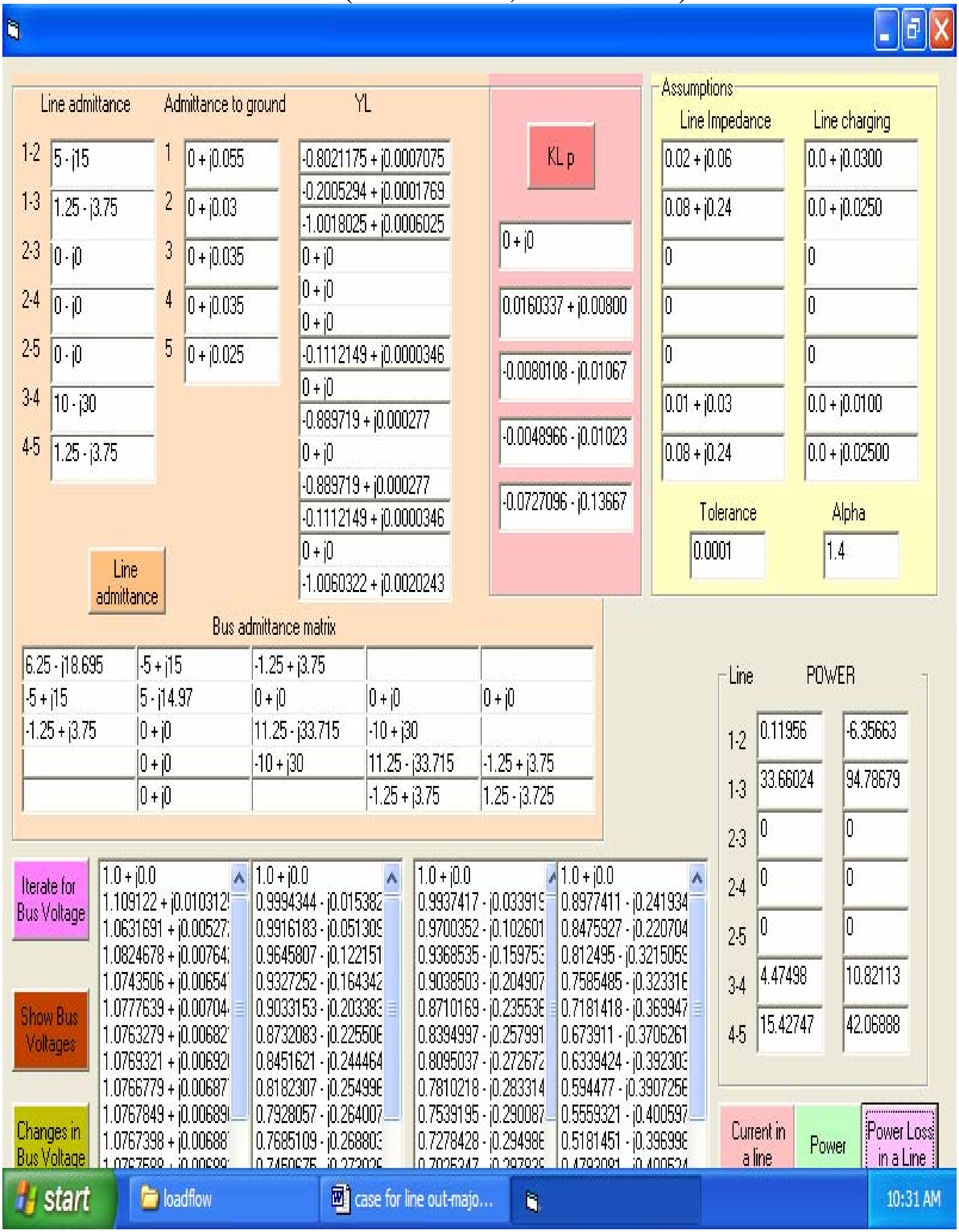
Case-36(When Lines 1-3 ,2-4 & 2-5 are out)



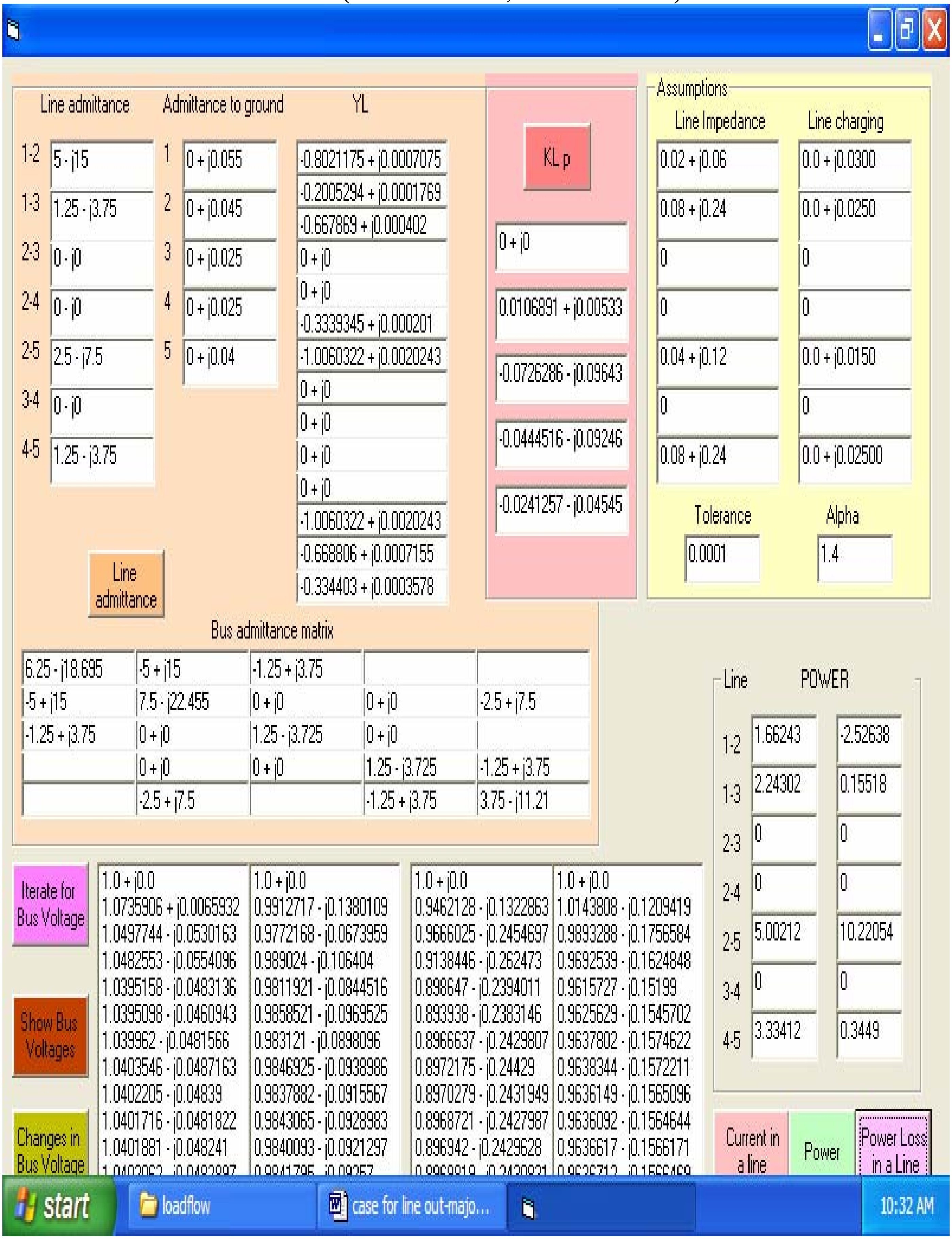
Case-37(When Lines 1-3 ,2-4 & 4-5 are out)



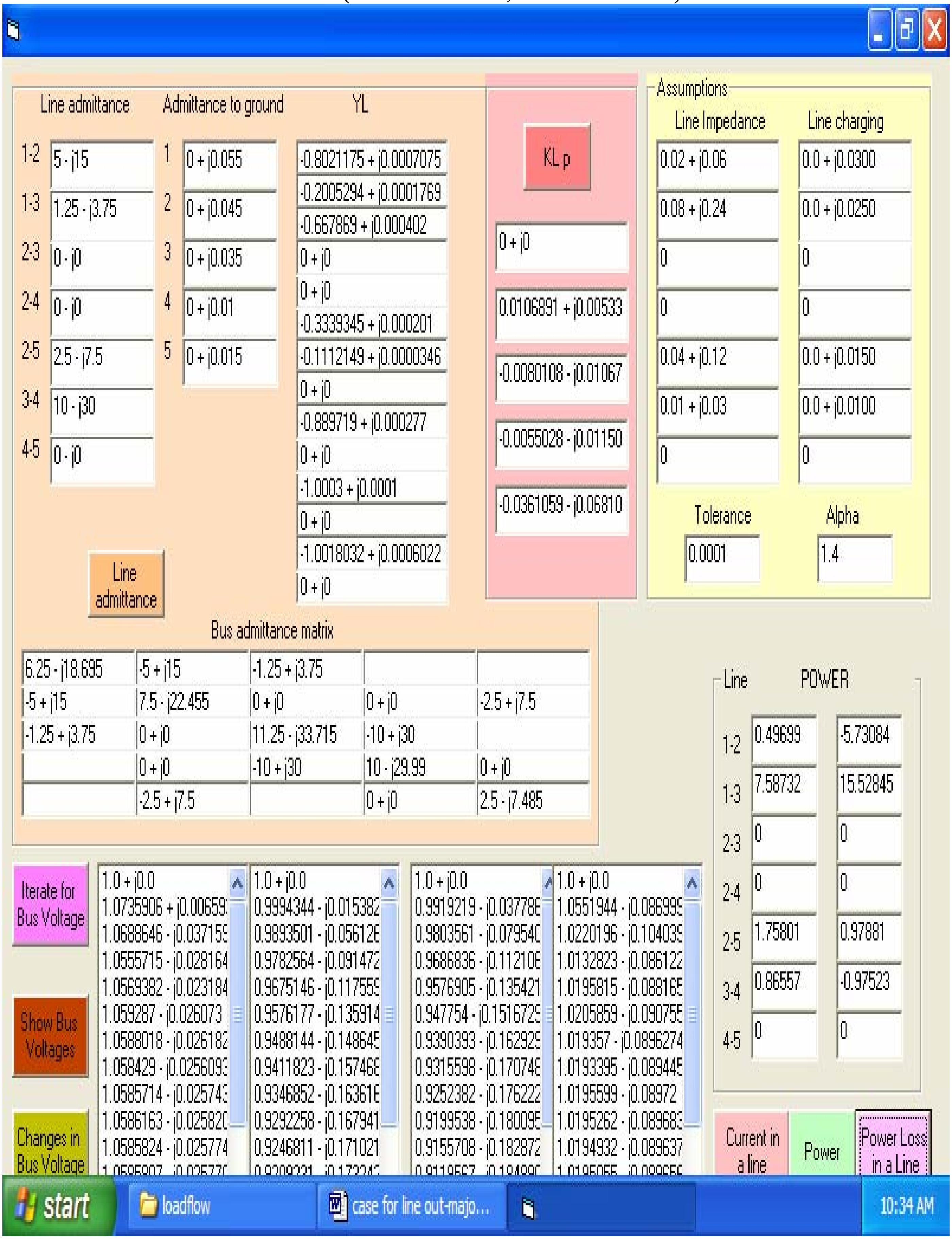
Case-38(When Lines 2-3 ,2-4 & 2-5 are out)



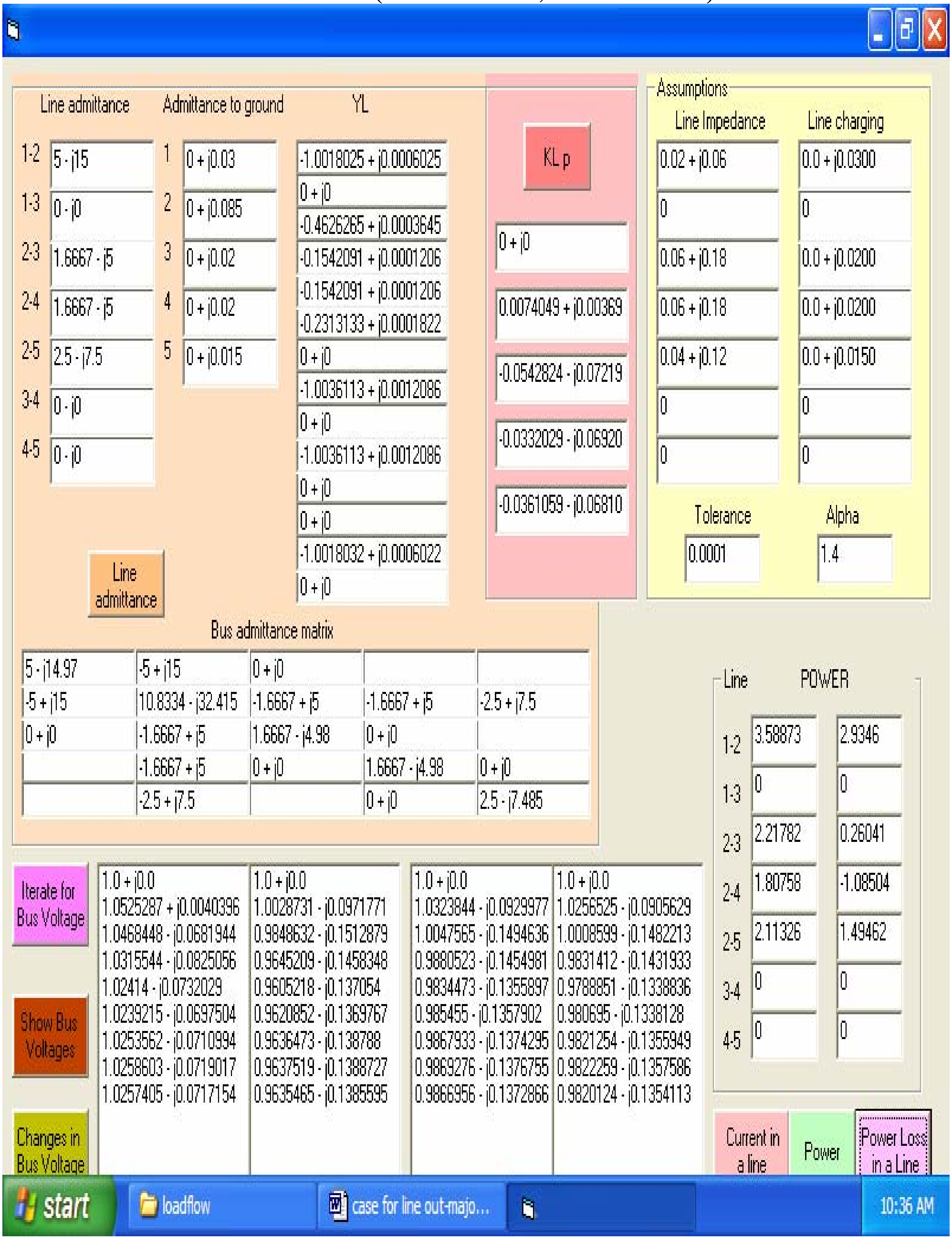
Case-39(When Lines 2-3 ,2-4 & 3-4 are out)



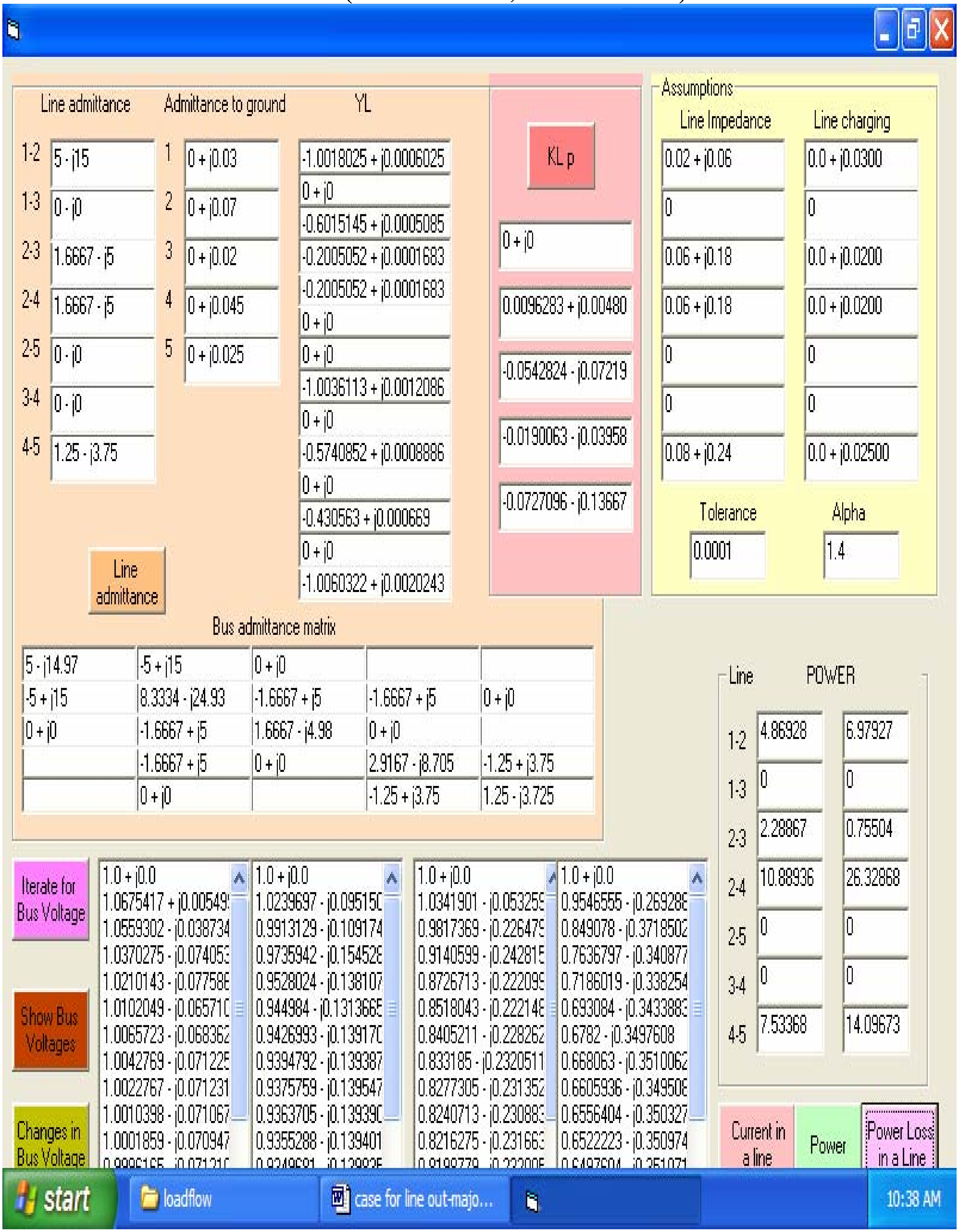
Case-40(When Lines 2-3 ,2-4 & 4-5 are out)



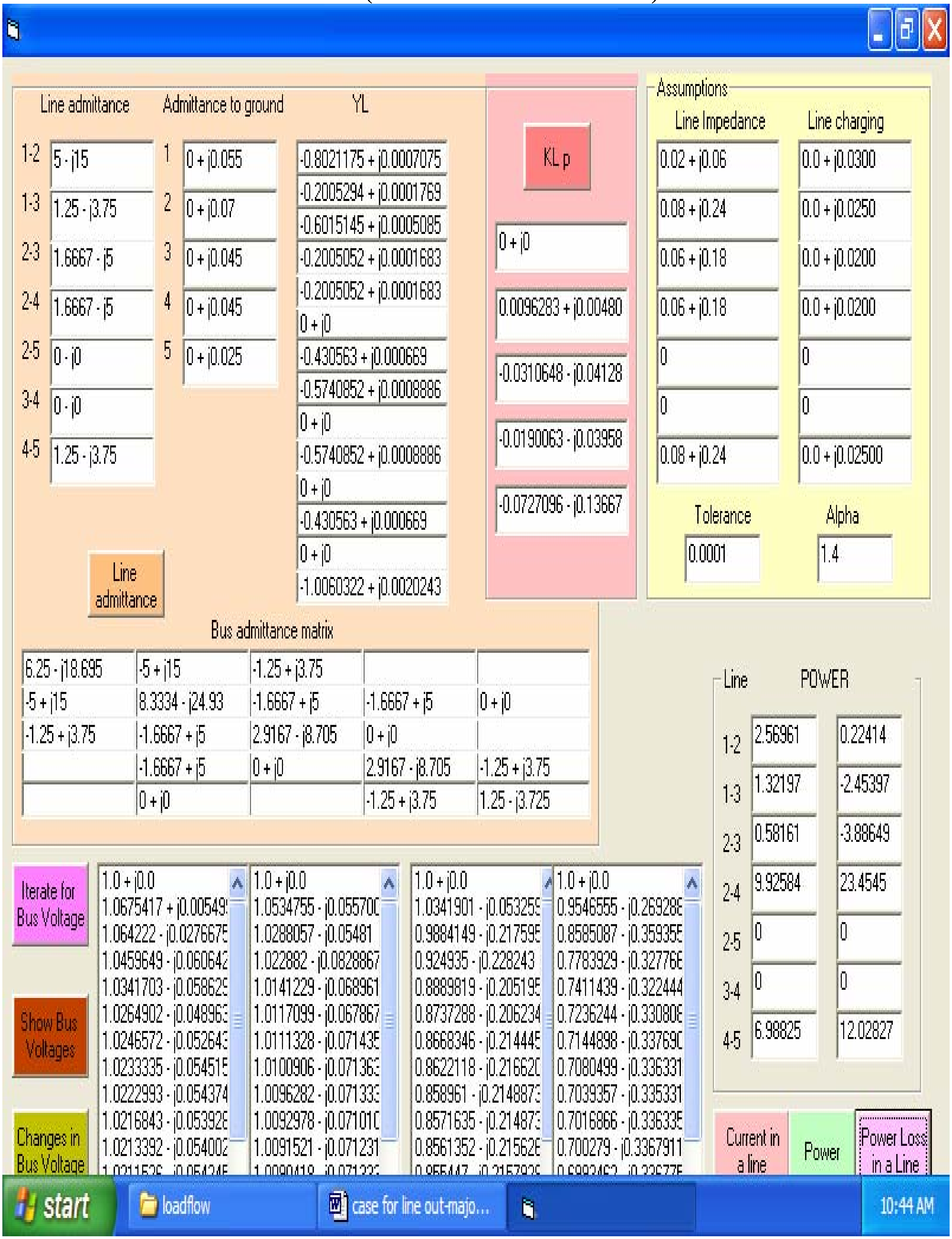
Case-41(When Lines 1-3 ,3-4 & 4-5 are out)



Case-28(When Lines 1-3 ,3-4 & 2-5 are out)



Case-42(When Lines 2-5 & 3-4 are out)



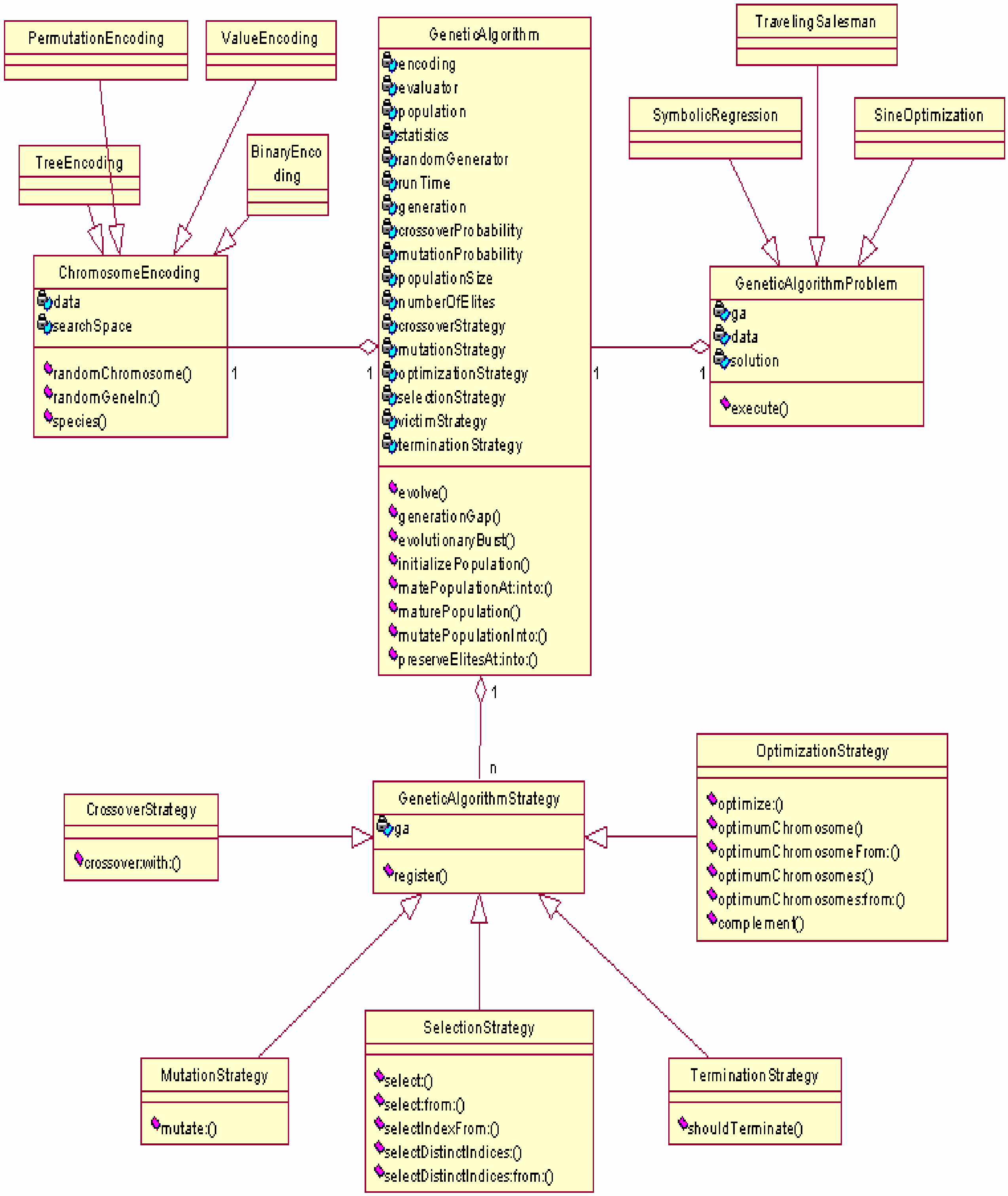
Annexure B: Transmission and distribution losses as a percentage of availability in state electricity departments: 1991/92 to 1999/2000

State	1991/92	1992/93	1993/94	1994/95	1995/96	1996/97	1997/98[a]	1998/99[b]	1999/00[c]
Andhra Pradesh	20.3	19.2	19.1	18.9	18.9	33.1	32.5	31.9	31.1
Arunachal Pradesh	28.2	34.9	31.6	31.0	36.0	32.6	31.0	31.1	31.5
Assam	22.7	21.0	20.8	24.9	26.2	26.0	30.1	23.0	30.0
Bihar	18.3	20.5	19.0	24.0	25.9	25.3	25.4	39.5	36.0
Daman and Diu	15.9	0	0	0	0	0	0	0	0
Goa	23.8	20.8	21.8	26.2	28.5	23.5	23.4	29.1	23.0
Gujarat	23.6	21.1	21.3	20.0	18.3	21.4	21.7	20.1	18.0
Haryana	26.8	25.4	25.5	28.5	31.4	32.8	33.4	29.6	29.5
Himachal Pradesh	19.2	18.5	17.3	17.4	17.5	18.4	19.2	18.5	18.1
Jammu and Kashmir	50.1	45.3	47.7	46.9	48.6	50.0	47.5	43.8	46.5
Karnataka	19.3	18.7	18.6	18.9	18.5	18.9	18.6	17	18.3
Kerala	22.5	21.0	20.2	20.1	20.1	21.4	17.9	17.5	17.0
Lakshadweep	17.4	0	0	0	0	0	0	0	0
Madhya Pradesh	25.8	22.2	20.2	20.1	19.5	20.6	19.7	17.8	18.6

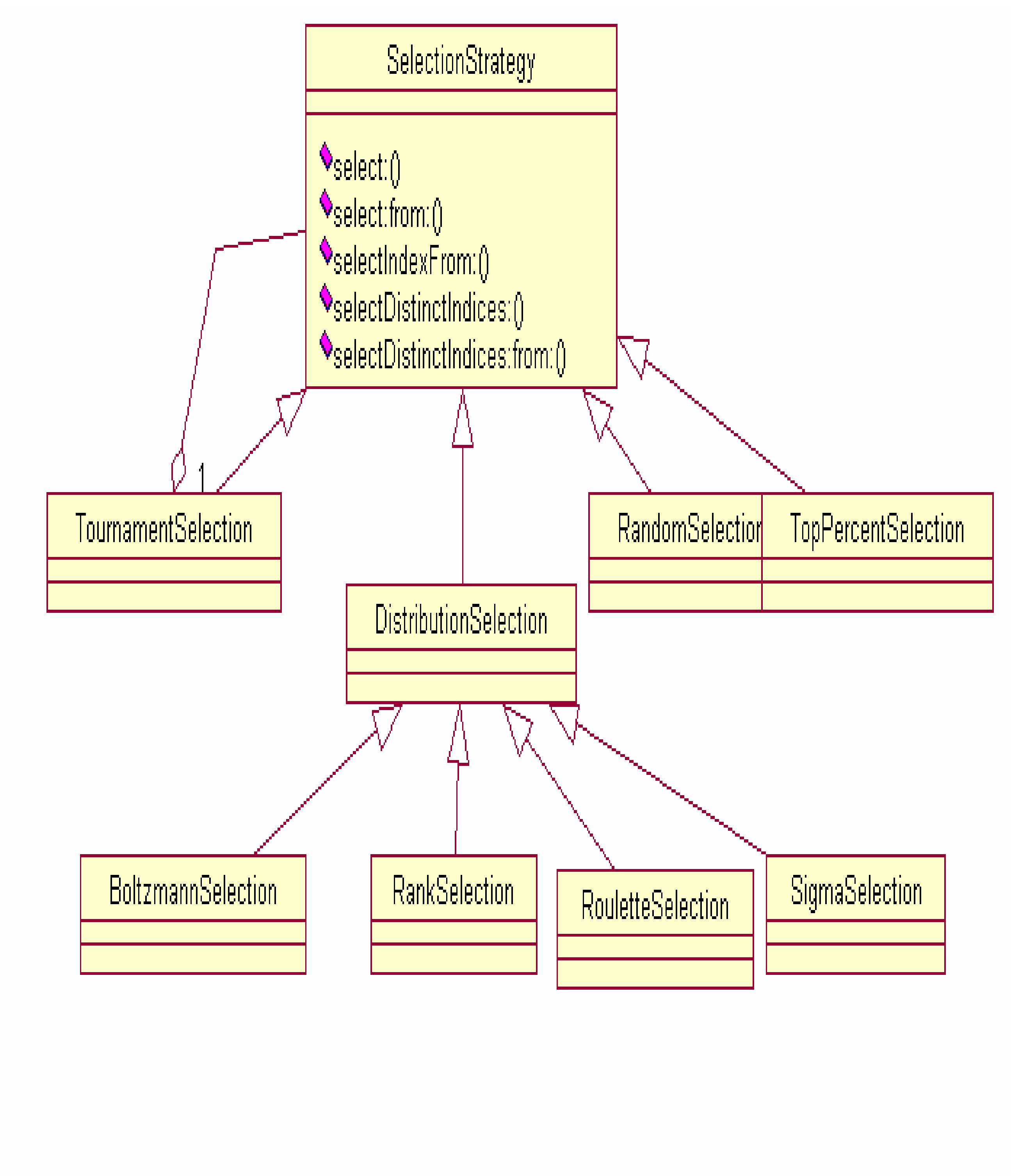
State	1991/92	1992/93	1993/94	1994/95	1995/96	1996/97	1997/98[a]	1998/99[b]	1999/00[c]
Maharashtra	18.6	16.4	15.8	15.3	15.4	17.7	17.1	17.3	17.0
Manipur	24.4	22.5	22.5	22.0	21.5	23.0	21.8	19.7	20.0
Meghalaya	11.7	12.2	10.7	18.7	17.8	19.5	17.9	18.9	19.0
Mizoram	34.9	28.1	28.0	28.0	27.0	34.4	25.7	42.0	43.0
Nagaland	23.1	32.4	31.6	30.8	30.0	26.8	29.5	29.0	28.5
Orissa	25.3	23.5	23.4	23.8	46.9	50.4	46.0	42.0	36.0
Punjab	21.8	18.7	18.5	18.3	18.2	18.9	17.8	17.1	17.7
Rajasthan	23.1	24.5	25.2	25.0	28.5	25.9	26.5	29.5	22.0
Sikkima	25.9	21.8	21.5	21.2	21.0	29.2	20.1	20.0	19.8
Tamil Nadu	18.4	17.5	17.3	16.9	17.0	17.2	16.8	16.6	16.5
Tripura	32.0	30.5	30.0	30.0	30.0	30.1	29.3	28.5	28.0
Uttar Pradesh	26.1	24.1	23.2	22.6	22.8	25.1	25.5	26.3	22.9
West Bengal	19.7	23.7	22.4	21.1	20.7	20.1	20.0	19.5	19.0
All-India (utilities)	22.8	19.8	20.2	20.3	22.2	24.5	23.9	23.2	22.0

[a] provisional; [b] revised ; [c] estimate

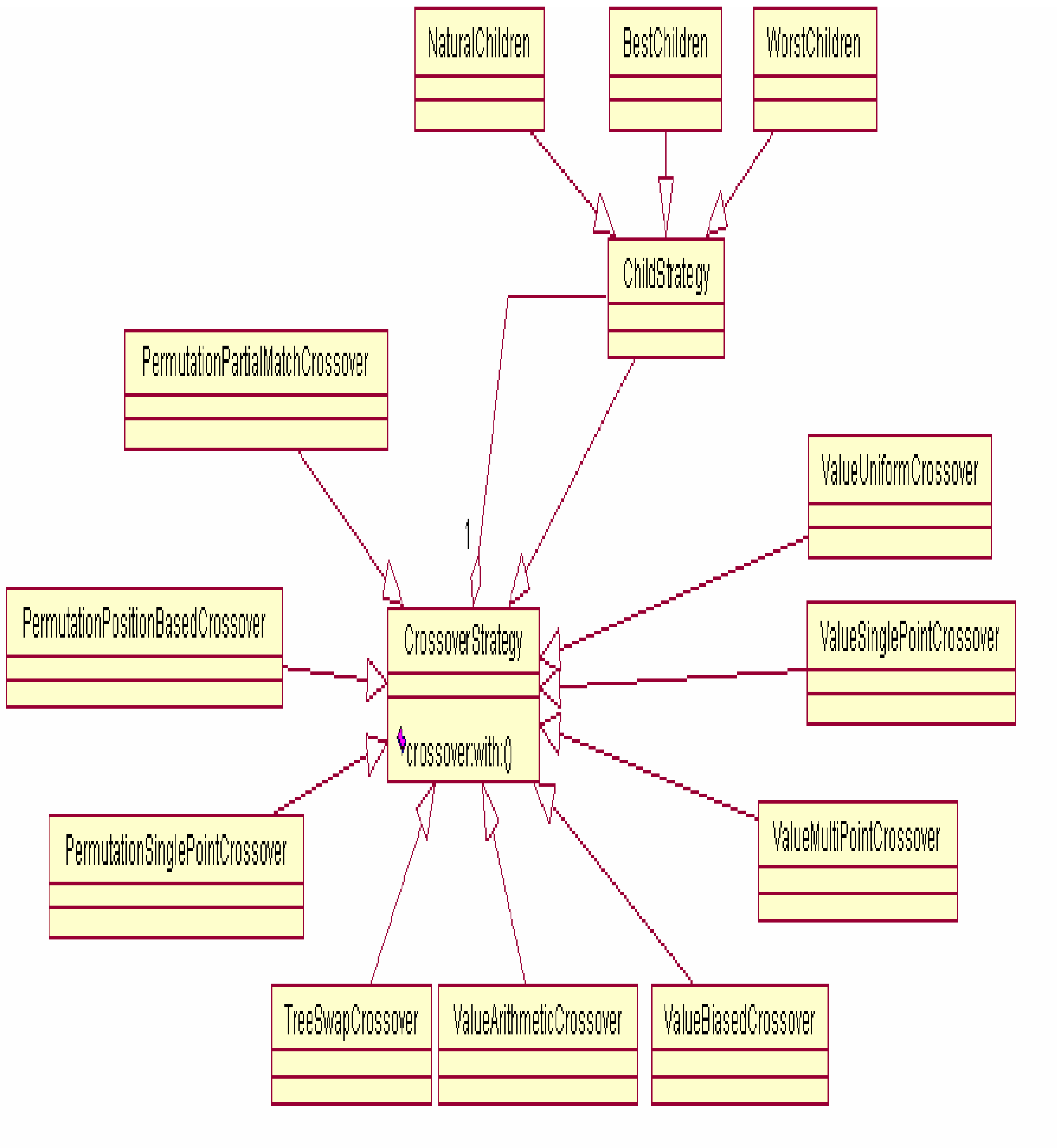
BLOCK DIAGRAM-GENERAL



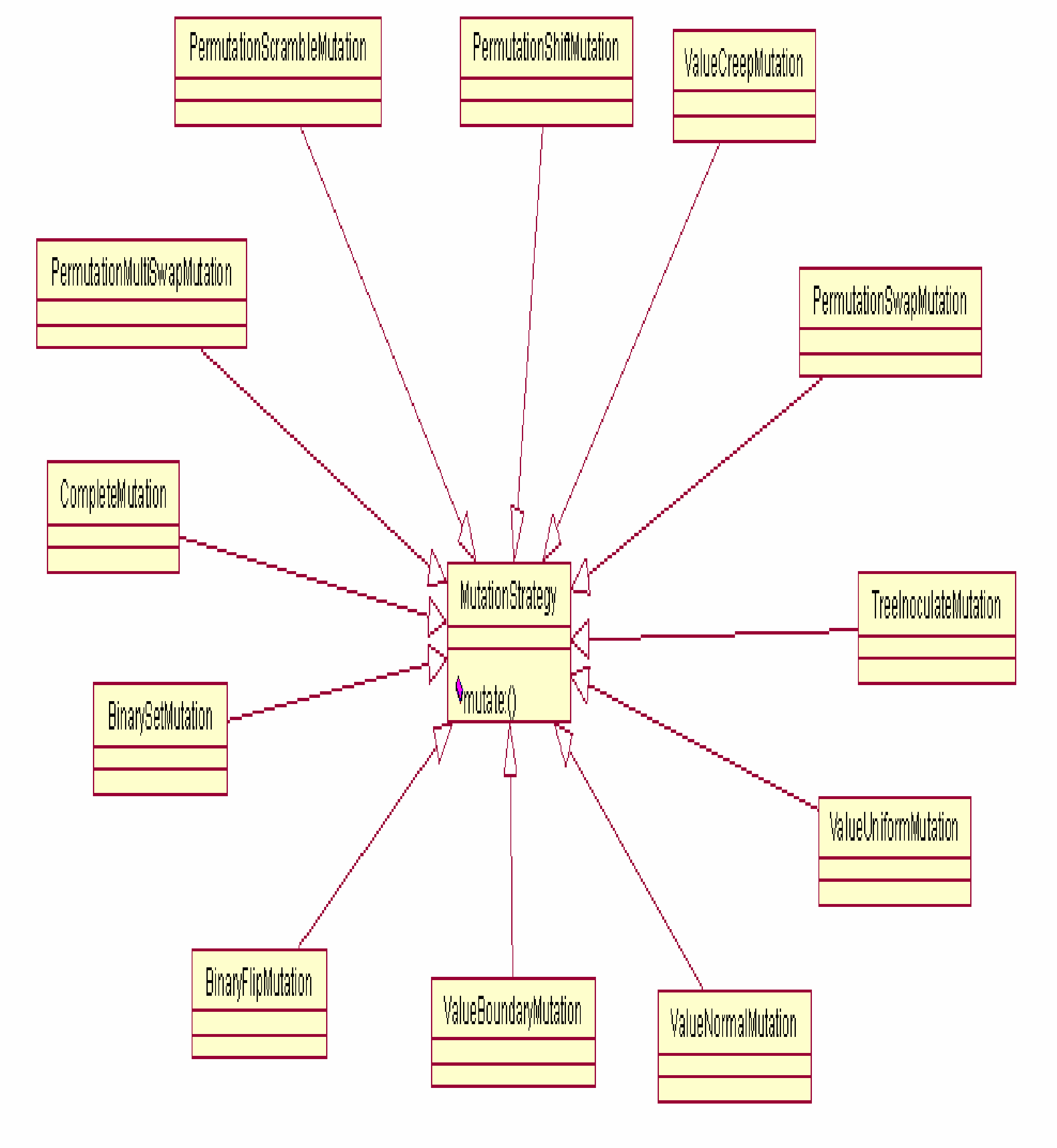
BLOCK DIAGRAM-SELECTION STRATEGY



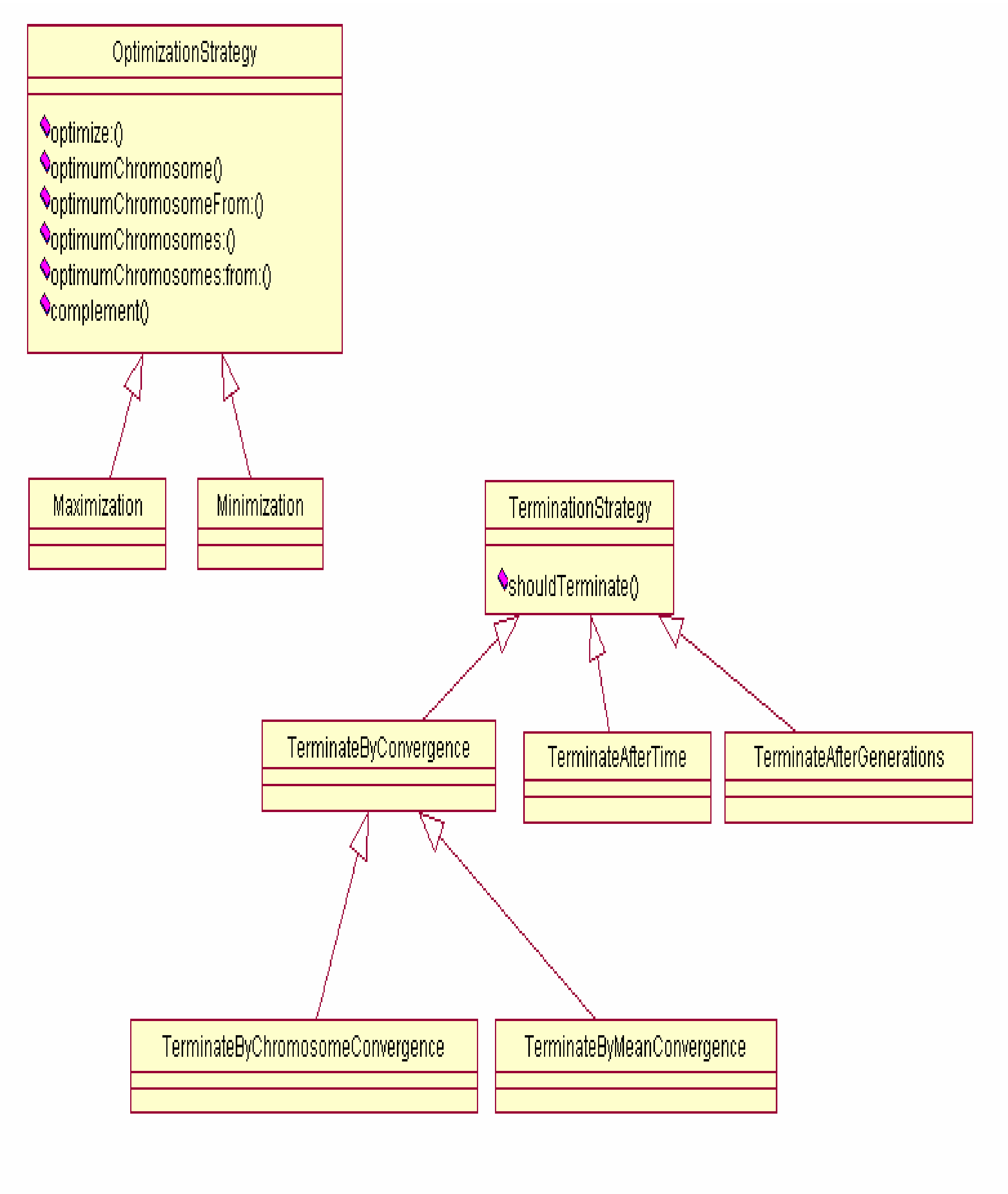
<u>**BLOCK DIAGRAM-CROSS OVER STRATEGY**</u>



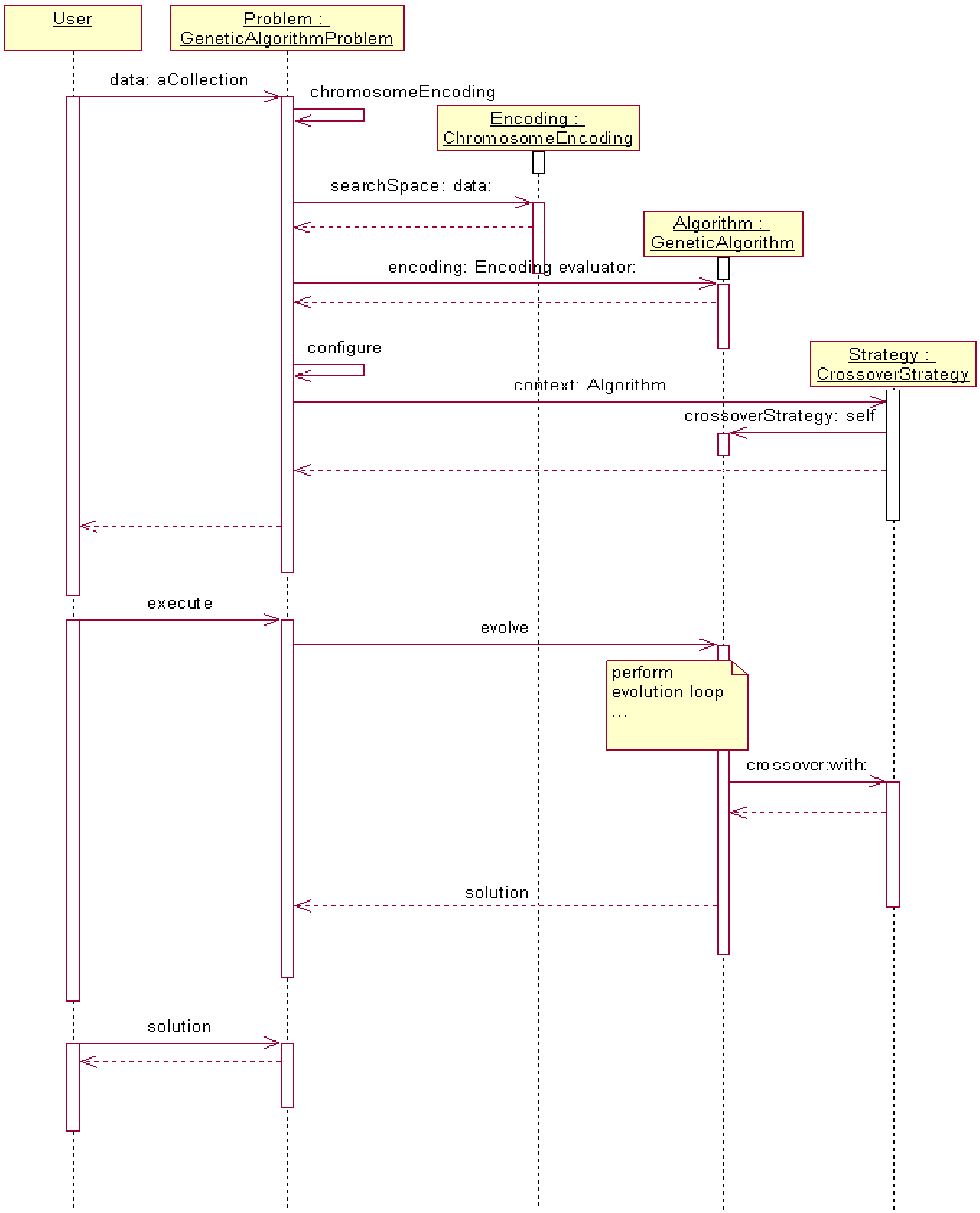
BLOCK DIAGRAM-MUTATION STRATEGY



BLOCK DIAGRAM-OPTIMIZATION STRATEGY



BLOCK DIAGRAM SHOWING COMPLETE PROCEDURE FOLLOWED IN G.A



SOFTWARE FOR LOAD FLOW STUDIES USING GUASS SEIDEL METHOD

```
'Defining the Variables used in the program

Dim jz12, jz13, jz23, jz24, jz25, jz34, jz45 As Long
Dim E1(20) As String, E2(20) As String, _
   E3(20) As String, E4(20) As String, E5(20) As String
Dim DE1(20) As String, DE2(20) As String, _
   DE3(20) As String, DE4(20) As String, DE5(20) As String
Dim E01, E02, E03, E04, E05 As String
Dim E1new, E2new, E3new, E4new, E5new As String
Dim L1, L2, L3, L4, L5 As String
Dim p1, p2 As Double
Dim tolerlim As Integer
Dim alpha As Double
Public Sub Form_Load()
txttol = 0.0001
txtalpha = 1.4

busno = 5
' impedance Zpq
Z12 = "-0.02 + j0.06"
Z13 = "0.08 + j0.24"
Z23 = "0.06 + j0.18"
Z24 = "0.06 + j0.18"
Z25 = "0.04 + j0.12"
Z34 = "0.01 + j0.03"
Z45 = "0.08 + j0.24"

' Line charging Ypq
ylc12 = "0.0 + j0.0300"
ylc13 = "0.0 + j0.0250"
ylc23 = "0.0 + j0.0200"
ylc24 = "0.0 + j0.0200"
ylc25 = "0.0 + j0.0150"
ylc34 = "0.0 + j0.0100"
ylc45 = "0.0 + j0.02500"


' Assumed bus voltage
E01 = "1.06 + j0.0"
E02 = "1.0 + j0.0"
E03 = "1.0 + j0.0"
E04 = "1.0 + j0.0"
E05 = "1.0 + j0.0"
E1(0) = E01
E2(0) = E02
E3(0) = E03
E4(0) = E04
E5(0) = E05


' Generation in MW an MVA and Load at the Bus
```

```
G1MW = 0
G1MV = 0
L1MW = 0
L1MV = 0
G2MW = 40 / 100
G2MV = 30 / 100
L2MW = 20 / 100
L2MV = 10 / 100
G3MW = 0
G3MV = 0
L3MW = 45 / 100
L3MV = 15 / 100
G4MW = 0
G4MV = 0
L4MW = 40 / 100
L4MV = 5 / 100
G5MW = 0
G5MV = 0
L5MW = 60 / 100
L5MV = 10 / 100

L1 = (G1MW - L1MW) & " - j" & (G1MV - L1MV)
If G1MV < L1MV Then L1 = (G1MW - L1MW) & " + j" & (L1MV - G1MV)
L2 = (G2MW - L2MW) & " - j" & (G2MV - L2MV)
If G2MV < L2MV Then L2 = (G2MW - L2MW) & " + j" & (L2MV - G2MV)
L3 = (G3MW - L3MW) & " - j" & (G3MV - L3MV)
If G3MV < L3MV Then L3 = (G3MW - L3MW) & " + j" & (L3MV - G3MV)
L4 = (G4MW - L4MW) & " - j" & (G4MV - L4MV)
If G4MV < L4MV Then L4 = (G4MW - L4MW) & " + j" & (L4MV - G4MV)
L5 = (G5MW - L5MW) & " - j" & (G5MV - L5MV)
If G5MV < L5MV Then L5 = (G5MW - L5MW) & " + j" & (L5MV - G5MV)



End Sub
```

```
Private Sub Cmdbusvolt_Click()
lbe2.Clear
lbe3.Clear
lbe4.Clear
lbe5.Clear

For i = 0 To 15
If i > tolerlim Then Exit For
If Len(Trim(E2(i))) <> 0 And Len(Trim(E3(i))) <> 0 And Len(Trim(E4(i))) <> 0 And Len(Trim(E5(i))) <>
0 Then
  lbe2.AddItem E2(i)
  lbe3.AddItem E3(i)
  lbe4.AddItem E4(i)
  lbe5.AddItem E5(i)
End If
Next i

End Sub
```

```
Private Sub Cmdchbusv_Click()
lbe2.Clear
lbe3.Clear
lbe4.Clear
lbe5.Clear

For i = 1 To 15
   If i > tolerlim Then Exit For
        dn = E1(0)
        dnt = minuscal(CStr(dn))
        DE1(i) = addcal(CStr(E1(0)), CStr(dnt))
        dn = E2(i - 1)
        dnt = minuscal(CStr(dn))
        DE2(i) = addcal(CStr(E2(i)), CStr(dnt))
        dn = E3(i - 1)
        dnt = minuscal(CStr(dn))
        DE3(i) = addcal(CStr(E3(i)), CStr(dnt))
        dn = E4(i - 1)
        dnt = minuscal(CStr(dn))
        DE4(i) = addcal(CStr(E4(i)), CStr(dnt))
        dn = E5(i - 1)
        dnt = minuscal(CStr(dn))
        DE5(i) = addcal(CStr(E5(i)), CStr(dnt))

If Len(Trim(DE2(i))) <> 0 And Len(Trim(DE3(i))) <> 0 And Len(Trim(DE4(i))) <> 0 And
Len(Trim(DE5(i))) <> 0 Then
    lbe2.AddItem DE2(i)
    lbe3.AddItem DE3(i)
    lbe4.AddItem DE4(i)
    lbe5.AddItem DE5(i)
End If


Next i


End Sub
```

```
Private Sub cmdcurrent_Click()
frameasmp.Visible = True
FrameW.Visible = False
framepower.Visible = False
If Len(Trim(E1new)) = 0 And Len(Trim(E2new)) = 0 Then
If Len(Trim(E3new)) = 0 And Len(Trim(E4new)) = 0 And Len(Trim(E5new)) = 0 Then
   MsgBox "Please calculate the Bus Voltage first"
   Cmditerate.SetFocus
   Exit Sub
End If
End If

i0 = addcal(CStr(E1new), minuscal(CStr(E2new)))
i1 = mcal(CStr(i0), CStr(y12))
i2 = mcal(CStr(E1new), CStr(ylc12))
i12 = addcal(CStr(i1), CStr(i2))

i0 = addcal(CStr(E1new), minuscal(CStr(E3new)))
i1 = mcal(CStr(i0), CStr(y13))
i2 = mcal(CStr(E1new), CStr(ylc13))
i13 = addcal(CStr(i1), CStr(i2))

i0 = addcal(CStr(E2new), minuscal(CStr(E3new)))
i1 = mcal(CStr(i0), CStr(y23))
i2 = mcal(CStr(E2new), CStr(ylc23))
i23 = addcal(CStr(i1), CStr(i2))

i0 = addcal(CStr(E2new), minuscal(CStr(E4new)))
i1 = mcal(CStr(i0), CStr(y24))
i2 = mcal(CStr(E2new), CStr(ylc24))
i24 = addcal(CStr(i1), CStr(i2))

i0 = addcal(CStr(E2new), minuscal(CStr(E5new)))
i1 = mcal(CStr(i0), CStr(y25))
i2 = mcal(CStr(E2new), CStr(ylc25))
i25 = addcal(CStr(i1), CStr(i2))

i0 = addcal(CStr(E3new), minuscal(CStr(E4new)))
i1 = mcal(CStr(i0), CStr(y34))
i2 = mcal(CStr(E3new), CStr(ylc34))
i34 = addcal(CStr(i1), CStr(i2))

i0 = addcal(CStr(E4new), minuscal(CStr(E5new)))
i1 = mcal(CStr(i0), CStr(y45))
i2 = mcal(CStr(E4new), CStr(ylc45))
i45 = addcal(CStr(i1), CStr(i2))
framecurrent.Visible = True
End Sub
```

```
Private Sub CmdKLP_Click()
If Len(Trim(YB11)) = 0 And Len(Trim(YB22)) = 0 And Len(Trim(YB33)) = 0 Then
If Len(Trim(YB44)) = 0 And Len(Trim(YB55)) = 0 Then
   MsgBox "Please calculate the Line admittances"
   Cmdlinead.SetFocus
   Exit Sub
End If
End If

framecurrent.Visible = False
tt = divcal(YB11)
KL1 = mcal(CStr(L1), CStr(tt))
tt = divcal(YB22)
KL2 = mcal(CStr(L2), CStr(tt))
tt = divcal(YB33)
KL3 = mcal(CStr(L3), CStr(tt))
tt = divcal(YB44)
KL4 = mcal(CStr(L4), CStr(tt))
tt = divcal(YB55)
KL5 = mcal(CStr(L5), CStr(tt))


End Sub

Private Sub CMDMWMVA_Click()
cmdpower_Click
frameasmp.Visible = False
framecurrent.Visible = False
framepower.Visible = False
FrameW.Visible = True

End Sub
```

```
Private Sub cmdpower_Click()

If Len(Trim(E1new)) = 0 And Len(Trim(E2new)) = 0 Then
If Len(Trim(E3new)) = 0 And Len(Trim(E4new)) = 0 And Len(Trim(E5new)) = 0 Then
    MsgBox "Please calculate the Bus Voltage first"
    Cmditerate.SetFocus
    Exit Sub
End If
End If

frameasmp.Visible = True
framecurrent.Visible = False
framepower.Visible = True
FrameW.Visible = False

'1-2
i1 = addcal(CStr(E1new), minuscal(CStr(E2new)))
i2 = mcal(CStr(i1), CStr(y12))
i3 = mcal(conjcal(CStr(E1new)), CStr(i2))
i4 = mcal(CStr(E1new), CStr(ylc12))
i5 = mcal(CStr(E1new), CStr(i4))
p12 = addcal(CStr(i3), CStr(i5))

'1-3
i1 = addcal(CStr(E1new), minuscal(CStr(E3new)))
i2 = mcal(CStr(i1), CStr(y13))
i3 = mcal(conjcal(CStr(E1new)), CStr(i2))
i4 = mcal(CStr(E1new), CStr(ylc13))
i5 = mcal(CStr(E1new), CStr(i4))
p13 = addcal(CStr(i3), CStr(i5))


'2-1
i1 = addcal(CStr(E2new), minuscal(CStr(E1new)))
i2 = mcal(CStr(i1), CStr(y12))
i3 = mcal(conjcal(CStr(E2new)), CStr(i2))
i4 = mcal(CStr(E2new), CStr(ylc12))
i5 = mcal(CStr(E2new), CStr(i4))
p21 = addcal(CStr(i3), CStr(i5))

'2-3
i1 = addcal(CStr(E2new), minuscal(CStr(E3new)))
i2 = mcal(CStr(i1), CStr(y23))
i3 = mcal(conjcal(CStr(E2new)), CStr(i2))
i4 = mcal(CStr(E2new), CStr(ylc23))
i5 = mcal(CStr(E2new), CStr(i4))
p23 = addcal(CStr(i3), CStr(i5))

pl13 = addcal(CStr(p13), CStr(p31))
'2-4
i1 = addcal(CStr(E2new), minuscal(CStr(E4new)))
i2 = mcal(CStr(i1), CStr(y24))
i3 = mcal(conjcal(CStr(E2new)), CStr(i2))
```

```
i4 = mcal(CStr(E2new), CStr(ylc24))
i5 = mcal(CStr(E2new), CStr(i4))
p24 = addcal(CStr(i3), CStr(i5))

'2-5
i1 = addcal(CStr(E2new), minuscal(CStr(E5new)))
i2 = mcal(CStr(i1), CStr(y25))
i3 = mcal(conjcal(CStr(E2new)), CStr(i2))
i4 = mcal(CStr(E2new), CStr(ylc25))
i5 = mcal(CStr(E2new), CStr(i4))
p25 = addcal(CStr(i3), CStr(i5))

'3-1
i1 = addcal(CStr(E3new), minuscal(CStr(E1new)))
i2 = mcal(CStr(i1), CStr(y13))
i3 = mcal(conjcal(CStr(E3new)), CStr(i2))
i4 = mcal(CStr(E3new), CStr(ylc13))
i5 = mcal(CStr(E3new), CStr(i4))
p31 = addcal(CStr(i3), CStr(i5))
'3-2
i1 = addcal(CStr(E3new), minuscal(CStr(E2new)))
i2 = mcal(CStr(i1), CStr(y23))
i3 = mcal(conjcal(CStr(E3new)), CStr(i2))
i4 = mcal(CStr(E3new), CStr(ylc23))
i5 = mcal(CStr(E3new), CStr(i4))
p32 = addcal(CStr(i3), CStr(i5))

'3-4
i1 = addcal(CStr(E3new), minuscal(CStr(E4new)))
i2 = mcal(CStr(i1), CStr(y34))
i3 = mcal(conjcal(CStr(E3new)), CStr(i2))
i4 = mcal(CStr(E3new), CStr(ylc34))
i5 = mcal(CStr(E3new), CStr(i4))
p34 = addcal(CStr(i3), CStr(i5))

'4-2
i1 = addcal(CStr(E4new), minuscal(CStr(E2new)))
i2 = mcal(CStr(i1), CStr(y24))
i3 = mcal(conjcal(CStr(E4new)), CStr(i2))
i4 = mcal(CStr(E4new), CStr(ylc24))
i5 = mcal(CStr(E4new), CStr(i4))
p42 = addcal(CStr(i3), CStr(i5))

'4-3
i1 = addcal(CStr(E4new), minuscal(CStr(E3new)))
i2 = mcal(CStr(i1), CStr(y34))
i3 = mcal(conjcal(CStr(E4new)), CStr(i2))
i4 = mcal(CStr(E4new), CStr(ylc34))
i5 = mcal(CStr(E4new), CStr(i4))
p43 = addcal(CStr(i3), CStr(i5))

'4-5
i1 = addcal(CStr(E4new), minuscal(CStr(E5new)))
```

```
i2 = mcal(CStr(i1), CStr(y45))
i3 = mcal(conjcal(CStr(E4new)), CStr(i2))
i4 = mcal(CStr(E4new), CStr(ylc45))
i5 = mcal(CStr(E4new), CStr(i4))
p45 = addcal(CStr(i3), CStr(i5))


'5-2
i1 = addcal(CStr(E5new), minuscal(CStr(E2new)))
i2 = mcal(CStr(i1), CStr(y25))
i3 = mcal(conjcal(CStr(E5new)), CStr(i2))
i4 = mcal(CStr(E5new), CStr(ylc25))
i5 = mcal(CStr(E5new), CStr(i4))
p52 = addcal(CStr(i3), CStr(i5))

'5-4
i1 = addcal(CStr(E5new), minuscal(CStr(E4new)))
i2 = mcal(CStr(i1), CStr(y45))
i3 = mcal(conjcal(CStr(E5new)), CStr(i2))
i4 = mcal(CStr(E5new), CStr(ylc45))
i5 = mcal(CStr(E5new), CStr(i4))
p54 = addcal(CStr(i3), CStr(i5))

pl12 = addcal(CStr(p12), CStr(p21))
pl13 = addcal(CStr(p13), CStr(p31))
pl23 = addcal(CStr(p23), CStr(p32))
pl34 = addcal(CStr(p34), CStr(p43))
pl24 = addcal(CStr(p24), CStr(p42))
pl25 = addcal(CStr(p25), CStr(p52))
pl45 = addcal(CStr(p45), CStr(p54))

plmw12 = cal(CStr(pl12)) * 100
plmva12 = calmj(CStr(pl12)) * 100
plmw13 = cal(CStr(pl13)) * 100
plmva13 = calmj(CStr(pl13)) * 100
plmw23 = cal(CStr(pl23)) * 100
plmva23 = calmj(CStr(pl23)) * 100
plmw24 = cal(CStr(pl24)) * 100
plmva24 = calmj(CStr(pl24)) * 100
plmw25 = cal(CStr(pl25)) * 100
plmva25 = calmj(CStr(pl25)) * 100
plmw34 = cal(CStr(pl34)) * 100
plmva34 = calmj(CStr(pl34)) * 100
plmw45 = cal(CStr(pl45)) * 100
plmva45 = calmj(CStr(pl45)) * 100


MW12 = cal(CStr(p12)) * 100
MVA12 = calmj(CStr(p12)) * 100
MW13 = cal(CStr(p13)) * 100
MVA13 = calmj(CStr(p13)) * 100
mw21 = cal(CStr(p21)) * 100
mva21 = calmj(CStr(p21)) * 100
```

```
MW23 = cal(CStr(p23)) * 100
MVA23 = calmj(CStr(p23)) * 100
MW24 = cal(CStr(p24)) * 100
MVA24 = calmj(CStr(p24)) * 100
MW25 = cal(CStr(p25)) * 100
MVA25 = calmj(CStr(p25)) * 100


mw31 = cal(CStr(p31)) * 100
mva31 = calmj(CStr(p31)) * 100
mw32 = cal(CStr(p32)) * 100
mva32 = calmj(CStr(p32)) * 100
MW34 = cal(CStr(p34)) * 100
MVA34 = calmj(CStr(p34)) * 100
mw42 = cal(CStr(p42)) * 100
mva42 = calmj(CStr(p42)) * 100
mw43 = cal(CStr(p43)) * 100
mva43 = calmj(CStr(p43)) * 100
MW45 = cal(CStr(p45)) * 100
MVA45 = calmj(CStr(p45)) * 100
mw52 = cal(CStr(p52)) * 100
mva52 = calmj(CStr(p52)) * 100
mw54 = cal(CStr(p54)) * 100
mva54 = calmj(CStr(p54)) * 100

End Sub
```

```
Public Sub CmdLinead_Click()


' LINE ADMITTANCES
y12.Text = odivcal(CStr(Z12))
y13.Text = odivcal(CStr(Z13))
y23.Text = odivcal(CStr(Z23))
y24.Text = odivcal(CStr(Z24))
y25.Text = odivcal(CStr(Z25))
y34.Text = odivcal(CStr(Z34))
y45.Text = odivcal(CStr(Z45))

y1.Text = addcal(CStr(ylc12), CStr(ylc13))
y2.Text = addcal(addcal(addcal(CStr(ylc23), CStr(ylc24)), CStr(ylc25)), CStr(ylc12))
y3.Text = addcal(addcal(CStr(ylc13), CStr(ylc23)), CStr(ylc34))
y4.Text = addcal(addcal(CStr(ylc24), CStr(ylc34)), CStr(ylc45))
y5.Text = addcal(CStr(ylc25), CStr(ylc45))




YB11.Text = addcal(addcal(CStr(y12), CStr(y13)), CStr(y1))
t22 = addcal(addcal(CStr(y12), CStr(y23)), addcal(CStr(y24), CStr(y25)))
YB22.Text = addcal(CStr(t22), CStr(y2))
YB33.Text = addcal(addcal(CStr(y13), CStr(y23)), addcal(CStr(y34), CStr(y3)))
YB44.Text = addcal(addcal(CStr(y24), CStr(y34)), addcal(CStr(y45), CStr(y4)))
YB55.Text = addcal(addcal(CStr(y25), CStr(y45)), CStr(y5))

YB12.Text = minuscal(CStr(y12.Text))
YB21.Text = minuscal(CStr(y12.Text))
YB13.Text = minuscal(CStr(y13.Text))
YB31.Text = minuscal(CStr(y13.Text))
YB43.Text = minuscal(CStr(y34.Text))
YB34.Text = minuscal(CStr(y34.Text))
YB24.Text = minuscal(CStr(y24.Text))
YB42.Text = minuscal(CStr(y24.Text))
YB25.Text = minuscal(CStr(y25.Text))
YB52.Text = minuscal(CStr(y25.Text))
YB23.Text = minuscal(CStr(y23.Text))
YB32.Text = minuscal(CStr(y23.Text))
YB45.Text = minuscal(CStr(y45.Text))
YB54.Text = minuscal(CStr(y45.Text))
YB14.Text = Space(10)
YB41.Text = Space(10)
YB15.Text = Space(10)
YB51.Text = Space(10)
YB35.Text = Space(10)
YB53.Text = Space(10)

YL12.Text = mcal(CStr(YB12), divcal(CStr(YB11)))
YL13.Text = mcal(CStr(YB13), divcal(CStr(YB11)))
YL21.Text = mcal(CStr(YB21), divcal(CStr(YB22)))
YL23.Text = mcal(CStr(YB23), divcal(CStr(YB22)))
```

```
YL24.Text = mcal(CStr(YB24), divcal(CStr(YB22)))
YL25.Text = mcal(CStr(YB25), divcal(CStr(YB22)))
YL31.Text = mcal(CStr(YB31), divcal(CStr(YB33)))
YL32.Text = mcal(CStr(YB32), divcal(CStr(YB33)))
YL34.Text = mcal(CStr(YB34), divcal(CStr(YB33)))
YL42.Text = mcal(CStr(YB42), divcal(CStr(YB44)))
YL43.Text = mcal(CStr(YB43), divcal(CStr(YB44)))
YL45.Text = mcal(CStr(YB45), divcal(CStr(YB44)))
YL52.Text = mcal(CStr(YB52), divcal(CStr(YB55)))
YL54.Text = mcal(CStr(YB54), divcal(CStr(YB55)))


End Sub
```

```
Private Sub Cmditerate_Click()

    Dim i As Integer

If Len(Trim(KL2)) = 0 And Len(Trim(KL3)) = 0 Then
If Len(Trim(KL4)) = 0 And Len(Trim(KL5)) = 0 Then
    MsgBox "Please calculate the Line admittances and KLp's first"
    Cmdlinead.SetFocus
    Exit Sub
End If
End If
lbe2.Clear
lbe3.Clear
lbe4.Clear
lbe5.Clear


    For i = 0 To 15
' BUS 2
        ss = conjcal(E2(i))
        tt = divcal(CStr(ss))
        ntt = mcal(CStr(KL2), CStr(tt))
        ntt2 = mcal(CStr(YL21), CStr(E1(0)))
        ntt3 = mcal(CStr(YL23), CStr(E3(i)))
        ntt4 = mcal(CStr(YL24), CStr(E4(i)))
        ntt5 = mcal(CStr(YL25), CStr(E5(i)))

        nt2 = addcal(CStr(ntt2), CStr(ntt3))
        nt4 = addcal(CStr(ntt4), CStr(ntt5))
        nt = addcal(CStr(nt2), CStr(nt4))
        ntm = minuscal(CStr(nt))
        ntt = addcal(CStr(ntt), CStr(ntm))
' NEW VOLTAGE
        E2(i + 1) = ntt
'CHANGE IN VOLTAGE
        dn = E2(i)
        dnt = minuscal(CStr(dn))
        DE2(i + 1) = addcal(CStr(ntt), CStr(dnt))
 ' accelerated value of bus voltage
        aa = mcal(CStr(txtalpha), CStr(DE2(i + 1)))
        E2(i + 1) = addcal(CStr(E2(i)), CStr(aa))


' BUS 3
        tt = divcal(conjcal(E3(i)))
        ntt = mcal(CStr(KL3), CStr(tt))
        ntt2 = mcal(CStr(YL31), CStr(E1(0)))
        ntt3 = mcal(CStr(YL32), CStr(E2(i + 1)))
        ntt4 = mcal(CStr(YL34), CStr(E4(i)))

        nt2 = addcal(CStr(ntt2), CStr(ntt3))
        nt = addcal(CStr(nt2), CStr(ntt4))
        ntm = minuscal(CStr(nt))
```

```
        ntt = addcal(CStr(ntt), CStr(ntm))
' NEW VOLTAGE
        E3(i + 1) = ntt
'CHANGE IN VOLTAGE
        dn = E3(i)
        dnt = minuscal(CStr(dn))
        DE3(i + 1) = addcal(CStr(ntt), CStr(dnt))
 ' accelerated value of bus voltage
        aa = mcal(CStr(txtalpha), CStr(DE3(i + 1)))
        E3(i + 1) = addcal(CStr(E3(i)), CStr(aa))


' BUS 4
        tt = divcal(conjcal(E4(i)))
        ntt = mcal(CStr(KL4), CStr(tt))
        ntt2 = mcal(CStr(YL42), CStr(E2(i + 1)))
        ntt3 = mcal(CStr(YL43), CStr(E3(i + 1)))
        ntt4 = mcal(CStr(YL45), CStr(E5(i)))

        nt2 = addcal(CStr(ntt2), CStr(ntt3))
        nt = addcal(CStr(nt2), CStr(ntt4))
        ntm = minuscal(CStr(nt))
        ntt = addcal(CStr(ntt), CStr(ntm))
' NEW VOLTAGE
        E4(i + 1) = ntt
'CHANGE IN VOLTAGE
        dn = E4(i)
        dnt = minuscal(CStr(dn))
        DE4(i + 1) = addcal(CStr(ntt), CStr(dnt))
 ' accelerated value of bus voltage
        aa = mcal(CStr(txtalpha), CStr(DE4(i + 1)))
        E4(i + 1) = addcal(CStr(E4(i)), CStr(aa))


' BUS 5
        tt = divcal(conjcal(E5(i)))
        ntt = mcal(CStr(KL5), CStr(tt))
        ntt2 = mcal(CStr(YL52), CStr(E2(i + 1)))
        ntt3 = mcal(CStr(YL54), CStr(E4(i + 1)))

        nt = addcal(CStr(ntt2), CStr(ntt3))
        ntm = minuscal(CStr(nt))
        ntt = addcal(CStr(ntt), CStr(ntm))
' NEW VOLTAGE
        E5(i + 1) = ntt
'CHANGE IN VOLTAGE
        dn = E5(i)
        dnt = minuscal(CStr(dn))
        DE5(i + 1) = addcal(CStr(ntt), CStr(dnt))

 ' accelerated value of bus voltage
        aa = mcal(CStr(txtalpha), CStr(DE5(i + 1)))
        E5(i + 1) = addcal(CStr(E5(i)), CStr(aa))
```

```
 ' Variations

       dn = E1(i)
       dnt = minuscal(CStr(dn))
       DE1(i + 1) = addcal(CStr(E1(i + 1)), CStr(dnt))
       dn = E2(i)
       dnt = minuscal(CStr(dn))
       DE2(i + 1) = addcal(CStr(E2(i + 1)), CStr(dnt))
       dn = E3(i)
       dnt = minuscal(CStr(dn))
       DE3(i + 1) = addcal(CStr(E3(i + 1)), CStr(dnt))
       dn = E4(i)
       dnt = minuscal(CStr(dn))
       DE4(i + 1) = addcal(CStr(E4(i + 1)), CStr(dnt))
       dn = E5(i)
       dnt = minuscal(CStr(dn))
       DE5(i + 1) = addcal(CStr(E5(i + 1)), CStr(dnt))

If i > 1 Then
If Abs(cal(DE1(i + 1))) < Val(txttol.Text) And Abs(calj(DE1(i + 1))) < Val(txttol.Text) Then
If Abs(cal(DE2(i + 1))) < Val(txttol.Text) And Abs(calj(DE2(i + 1))) < Val(txttol.Text) Then
If Abs(cal(DE3(i + 1))) < Val(txttol.Text) And Abs(calj(DE3(i + 1))) < Val(txttol.Text) Then
If Abs(cal(DE4(i + 1))) < Val(txttol.Text) And Abs(calj(DE4(i + 1))) < Val(txttol.Text) Then
If Abs(cal(DE5(i + 1))) < Val(txttol.Text) And Abs(calj(DE5(i + 1))) < Val(txttol.Text) Then
tolerlim = i
MsgBox "Tolerance limit reached at Iteration  " & i
  Exit For
End If
End If
End If
End If
End If
End If

lbe2.AddItem E2(i)
lbe3.AddItem E3(i)
lbe4.AddItem E4(i)
lbe5.AddItem E5(i)


   Next
E1new = E1(0)
E2new = E2(i)
E3new = E3(i)
E4new = E4(i)
E5new = E5(i)


End Sub
```

```
Private Function cal(zvalue) As Double
    cal = Val(zvalue)
End Function
Public Function calj(zvalue) As Double
    calj = 0
    If InStr(zvalue, "j") > 0 Then
      calj = Val(Mid(zvalue, InStr(zvalue, "j") + 1, Len(zvalue) - InStr(zvalue, "j")))
    End If
    If InStr(Mid(zvalue & Space(20), 3, 30), "-") > 0 Then
      calj = Val(-1 * Val(calj))
    End If
End Function
Public Function calmj(zvalue) As Double
    calmj = 0
    If InStr(zvalue, "j") > 0 Then
      calmj = Val(Mid(zvalue, InStr(zvalue, "j") + 1, Len(zvalue) - InStr(zvalue, "j")))
    End If
    If InStr(Mid(zvalue & Space(20), 3, 30), "-") > 0 Then
      calmj = Val(-1 * Val(calmj))
    End If
      calmj = Val(-1 * Val(calmj))

End Function
```

```
Public Function divcal(zvalue As String) As String
Dim p1, p2, p1new, p2new As Double

    p1 = cal(zvalue)
    p1new = p1
    If p1 < 0 Then p1 = Val(-1 * Val(p1))
    p2 = calj(zvalue)
    p2new = Val(-1 * Val(p2))
    If p2 < 0 Then p2 = Val(-1 * Val(p2))
    DD = Round(Val((p1 * p1) + (p2 * p2)), 7)

    If DD <> 0 Then
      p1 = Round(Val(p1new / DD), 7)
      p2 = Round(Val(p2new / DD), 7)
    End If
    divcal = p1 & " + j" & p2
    If p2 < 0 Then
      p2 = Val(-1 * Val(p2))
      divcal = p1 & " - j" & p2
    End If

End Function
```

```
Public Function odivcal(zvalue As String) As String
    p1 = cal(zvalue)
    If p1 < 0 Then p1 = Val(-1 * Val(p1))
    p2 = calj(zvalue)
    If p2 < 0 Then p2 = Val(-1 * Val(p2))
    DD = Val((p1 * p1) + (p2 * p2))
    If DD <> 0 Then
      p1 = Round(Val(p1 / DD), 4)
      p2 = Val(p2 / DD)
    End If
    odivcal = p1 & " - j" & p2
End Function
```

```
Public Function addcal(zvalue As String, yvalue As String) As String
   p1 = cal(zvalue)
   p2 = calj(zvalue)
   q1 = cal(yvalue)
   q2 = calj(yvalue)
   v1 = Round(Val(p1 + q1), 7)
   v2 = Round(Val(p2 + q2), 7)
   addcal = v1 & " + j" & v2
   If v2 < 0 Then
     v2 = Val(-1 * Val(v2))
     addcal = v1 & " - j" & v2
   End If
End Function
```

```
Public Function minuscal(zvalue As String) As String
   p1 = Val(-1 * cal(zvalue))
   p2 = Val(-1 * calj(zvalue))
   minuscal = p1 & " + j" & p2
   If p2 < 0 Then
     p2 = Val(-1 * Val(p2))
     minuscal = p1 & " - j" & p2
   End If
End Function
```

```
Public Function conjcal(zvalue As String) As String
   p1 = Val(cal(zvalue))
   p2 = Val(-1 * calj(zvalue))
   conjcal = p1 & " + j" & p2
   If p2 < 0 Then
     p2 = Val(-1 * Val(p2))
     conjcal = p1 & " - j" & p2
   End If
End Function
```

```
Public Function mcal(zvalue As String, yvalue As String) As String
   p1 = cal(zvalue)
   p2 = calj(zvalue)
   q1 = cal(yvalue)
   q2 = calj(yvalue)
   v1 = Round(Val(p1 * q1) - Val(p2 * q2), 7)
   v2 = Round(Val(Val(p1 * q2) + Val(p2 * q1)), 7)
   mcal = v1 & " + j" & v2
   If v2 < 0 Then
     v2 = Val(-1 * Val(v2))
     mcal = v1 & " - j" & v2
   End If
End Function
```